What is learning?

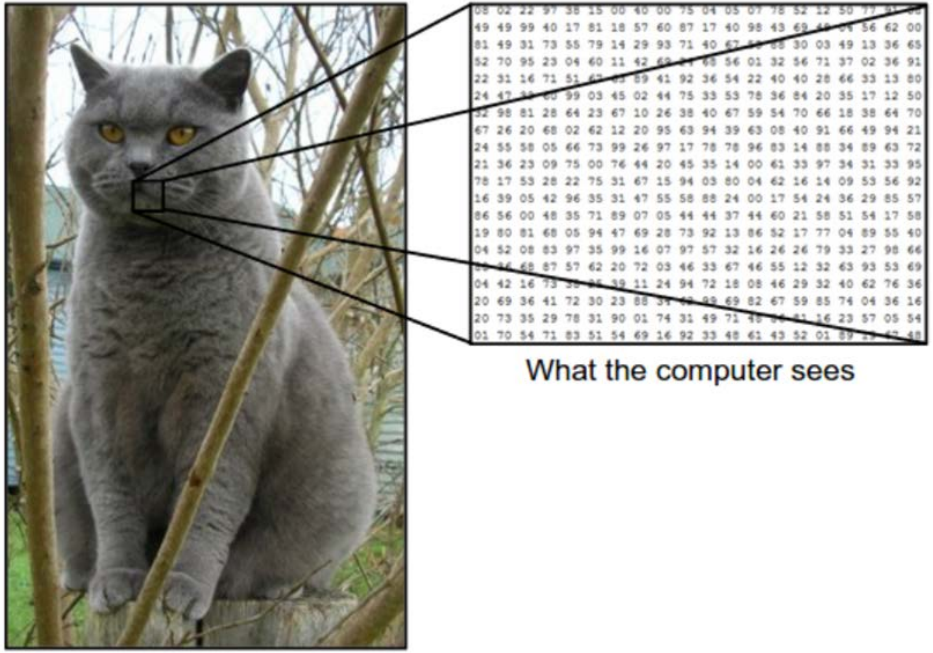
Image Source: Internet

Can You Recognize these Pictures ?

- If Yes, How do you Recognize it?

Image Source: Internet

Descriptors/ Feature Vectors



What the computer sees

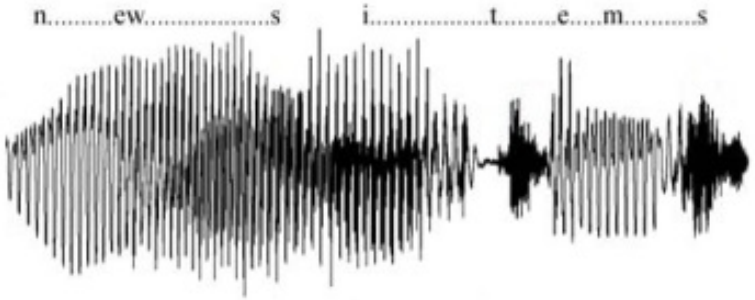
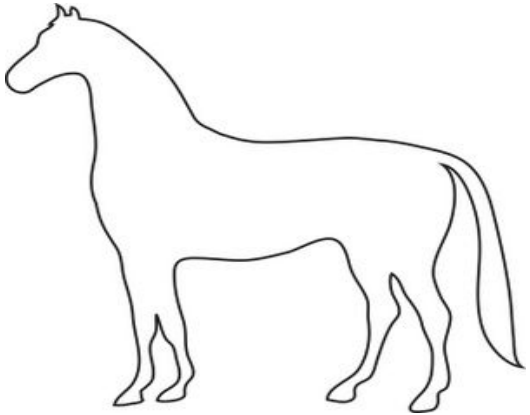


Image Source: Internet

Descriptors/ Feature Vectors

Image Source: Internet

Descriptors/ Feature Vectors



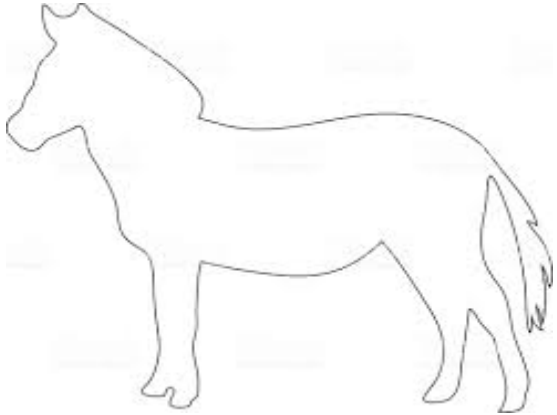


Image Source: Internet

Descriptors/ Feature Vectors

Image Source: Internet

Challenges

Image Source: Internet

Viewing Angle

Image Source: Internet

Pose

Image Source: Internet

Illumination

Image Source: Internet

Intraclass Variation






Image Source: Internet

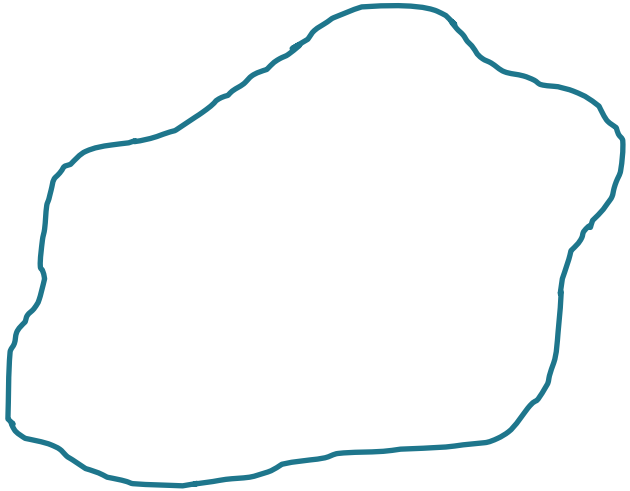
Distortion and Occlusion

Image Source: Internet

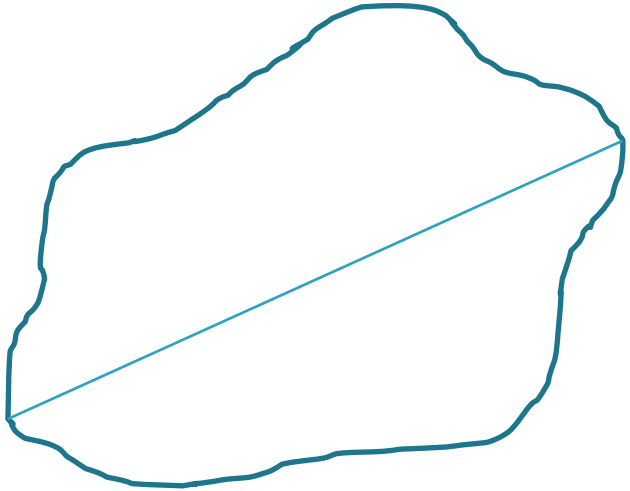
Power of Deep Learning

Boundary Descriptors

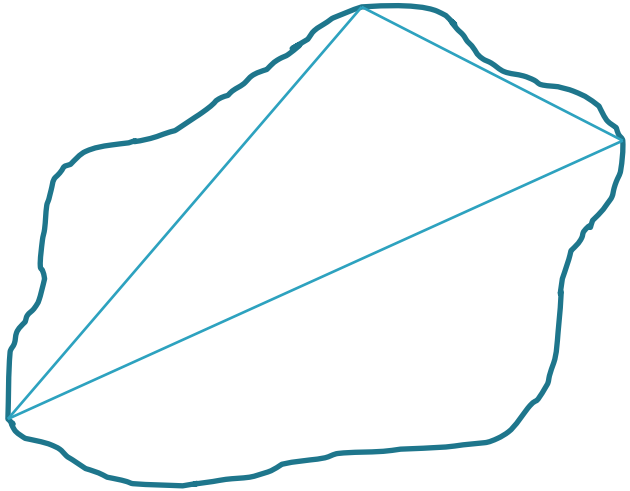
Boundary Descriptors



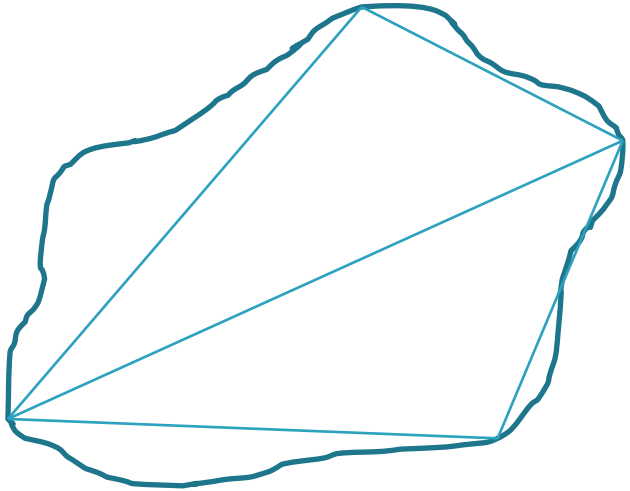
Boundary Descriptors



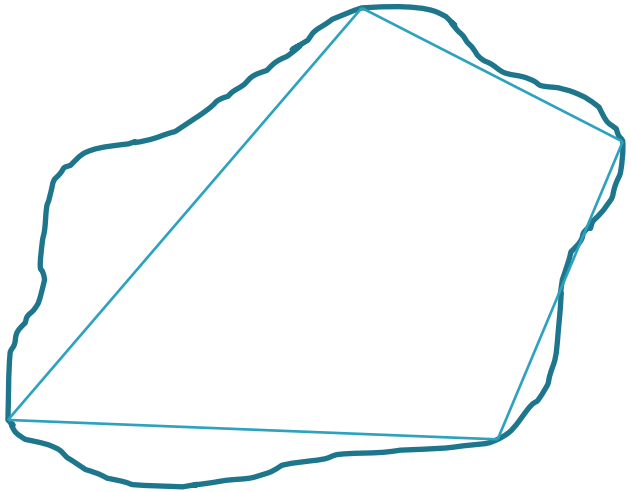
Boundary Descriptors

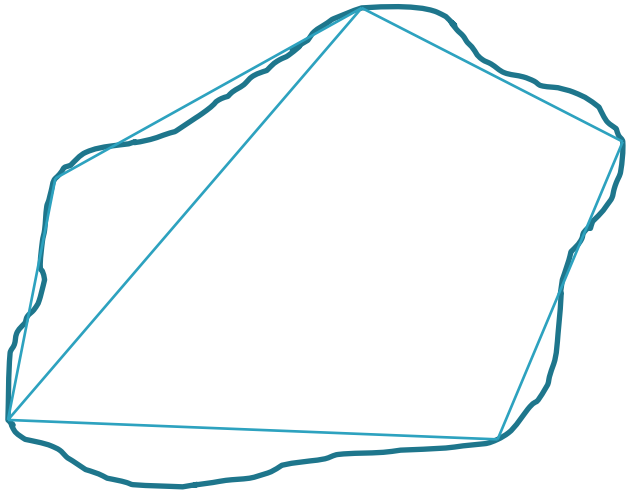


Boundary Descriptors

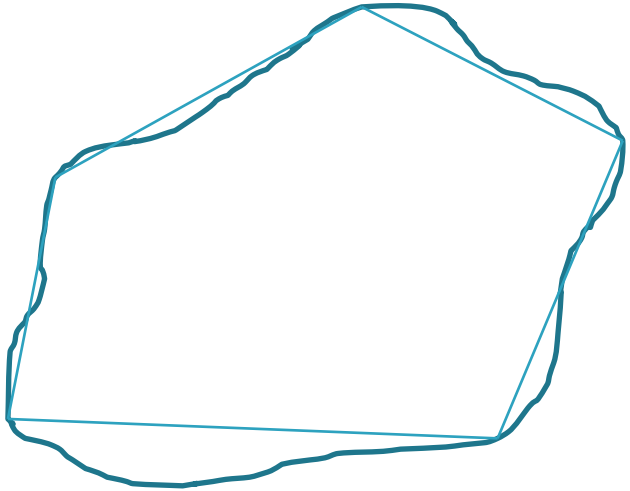


Boundary Descriptors

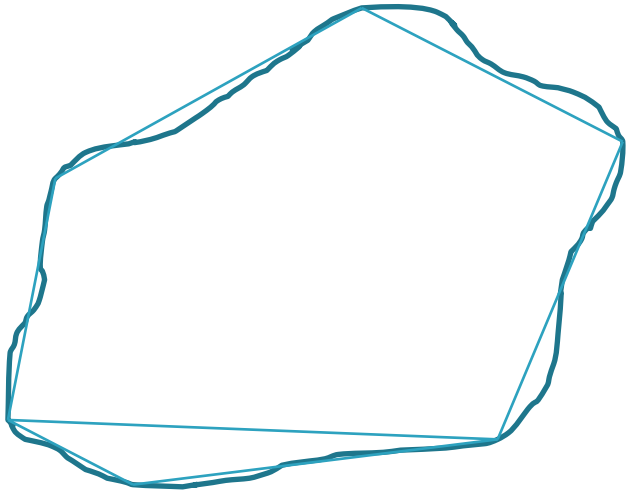




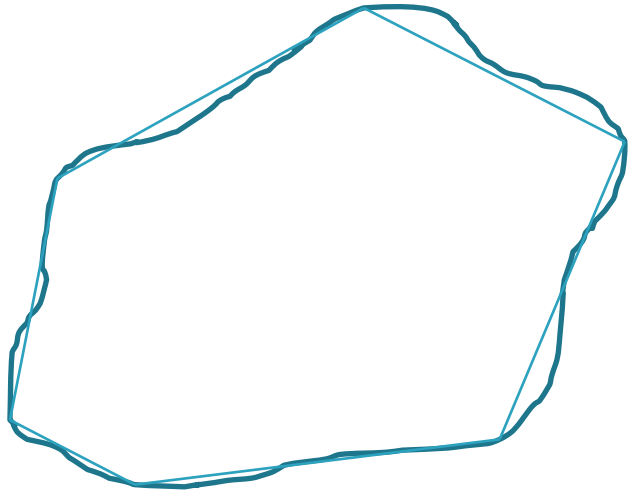
Boundary Descriptors



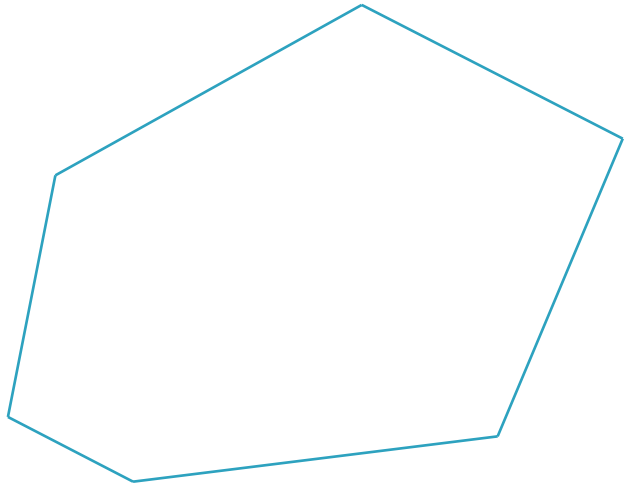
Boundary Descriptors



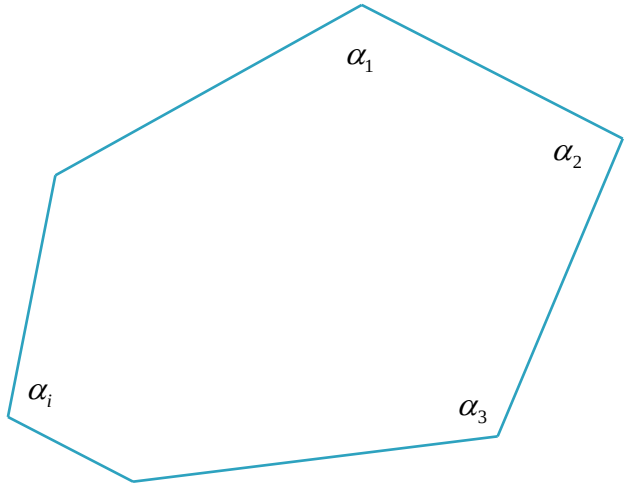
Boundary Descriptors



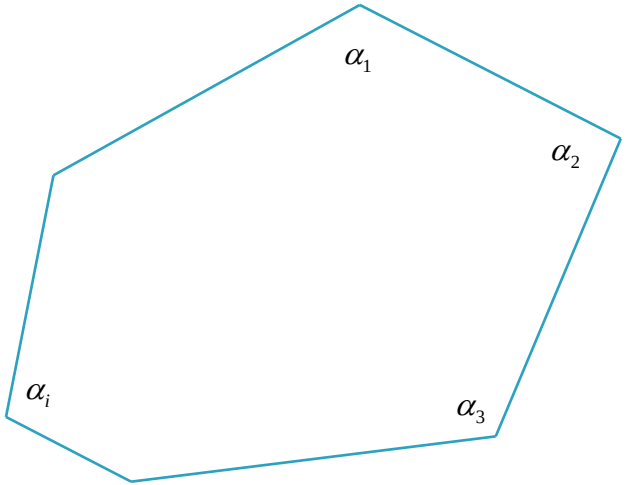
Boundary Descriptors



Boundary Descriptors



Boundary Descriptors



$$\alpha_1\ \alpha_2\ \alpha_3 \ldots \alpha_{i-k}\ \ldots \alpha_i\ \alpha_{i+1}\ \ldots$$

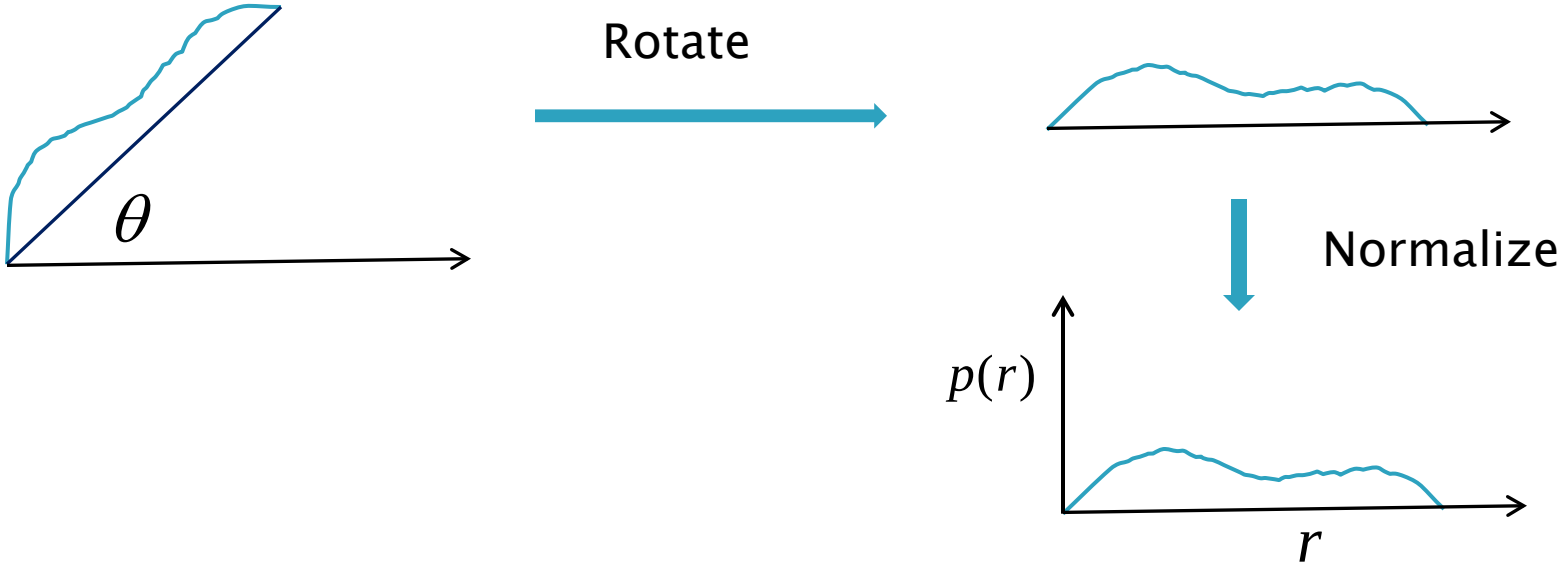
$$\alpha_i = \sum_{k=1}^{p} \beta_k \alpha_{i-k}$$

$$\beta_1\ \beta_2\ \beta_3\ \ldots \beta_p$$

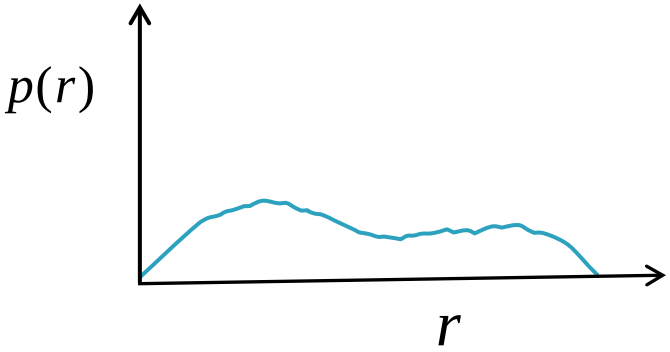
Descriptors (features) of polygonal shape

Boundary Descriptors

Boundary Descriptors



Boundary Descriptors



Statistical Moments

$$\sum_{i=1}^{k}(r_i-\hat{r})^2\,p(r_i)$$ Spread

$$\sum_{i=1}^{k}(r_i-\hat{r})^3\,p(r_i)$$ Skewness

$$\sum_{i=1}^{k}(r_i-\hat{r})^4\,p(r_i)$$ Kurtosis

Region Descriptors

Intensity Descriptor

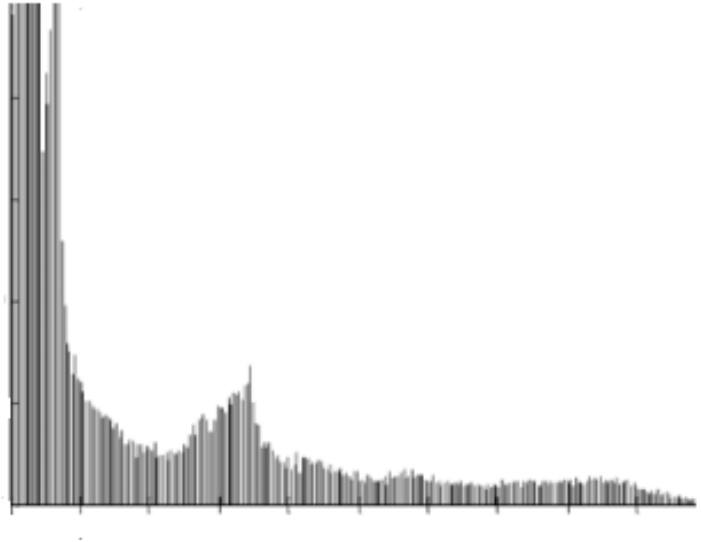


Image Source: Internet

Intensity Descriptor

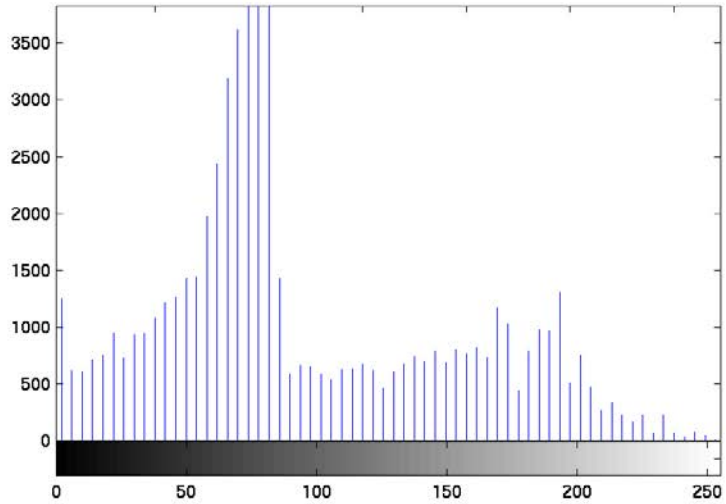


Image Source: Internet

Colour Feature

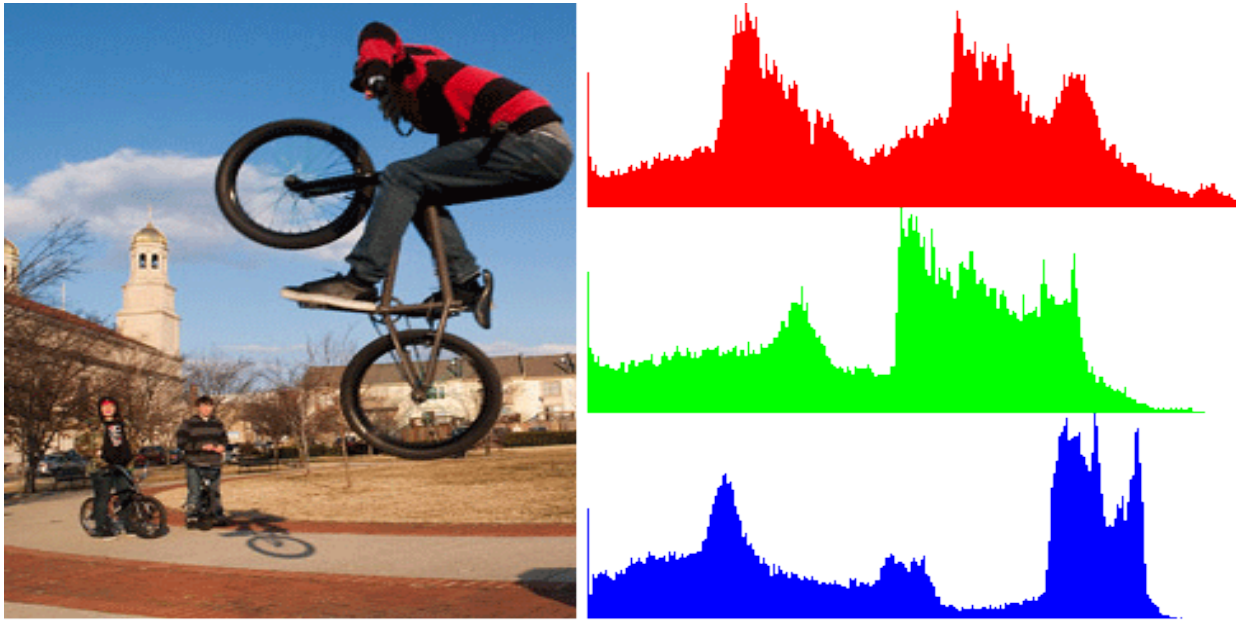
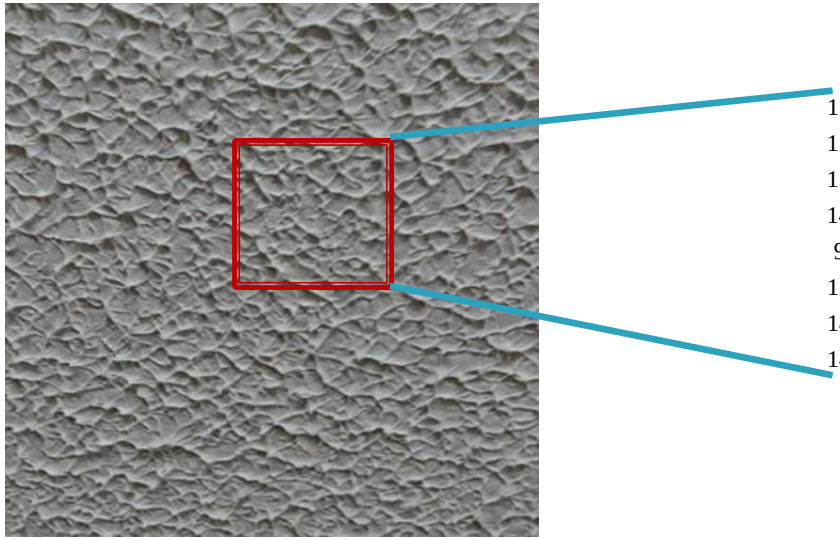


Image Source: https://billmill.org/the_histogram.html

Texture Descriptors

Pixel Domain/ Co-occurrence Matrix



150	100	115	109	112	100	145	140
110	112	120	135	125	120	132	133
152	99	129	130	122	135	98	100
147	138	142	95	108	136	110	125
99	127	149	138	138	129	108	129
128	125	139	115	120	145	137	131
146	159	150	130	147	139	143	127
140	120	128	98	100	106	115	119

Pixel Domain/ Co-occurrence Matrix

10	9	7	9	5	8	11	9
6	5	15	12	4	6	3	2
9	3	2	10	6	8	4	5
8	2	4	3	7	5	6	1
2	0	11	8	10	9	8	2
8	4	7	1	6	0	7	6
2	3	8	9	11	6	3	9
7	2	8	8	6	12	6	7

Pixel Domain/ Co-occurrence Matrix

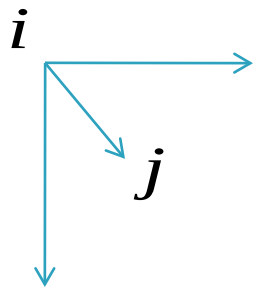
7	6	7	5	5	6	4	6
6	5	1	4	4	2	3	2
7	3	2	5	6	7	4	5
5	2	4	3	7	5	6	1
2	0	4	7	3	5	7	2
5	4	7	1	6	0	7	6
2	3	4	5	6	6	3	5
7	2	3	5	6	3	6	7

$$A_{l,\theta}(i, j)$$

Pixel Domain/ Co-occurrence Matrix

7	6	7	5	5	6	4	6
6	5	1	4	4	2	3	2
7	3	2	5	6	7	4	5
5	2	4	3	7	5	6	1
2	0	4	7	3	5	7	2
5	4	7	1	6	0	7	6
2	3	4	5	6	6	3	5
7	2	3	5	6	3	6	7

$$A_{1,45^0}(i, j)$$



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

Co-occurrence matrix based descriptors

Maximum Probability $\max_{i,j}(c_{ij})$

Element Difference Moment $\sum_i \sum_j (i-j)^k C_{i,j}$

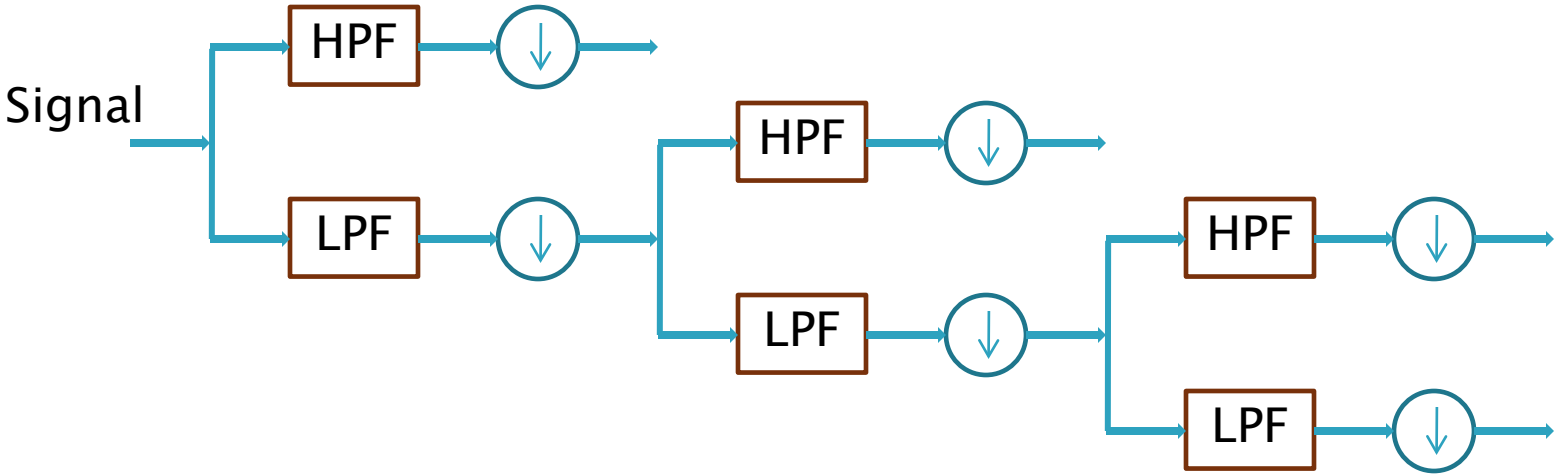
Inverse Element Difference Moment $\sum_i \sum_j C_{i,j} / (i-j)^k \qquad i \neq j$

Uniformity $\sum_i \sum_i C_{ij}^2$

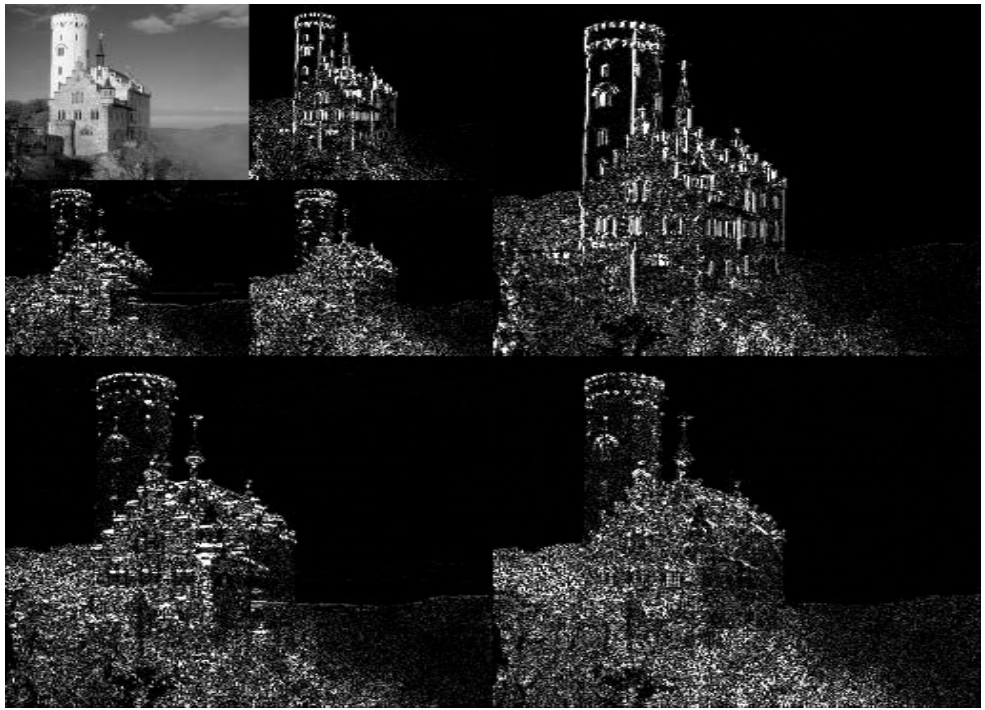
Entropy $-\sum_i \sum_j c_{ij} \log_2 C_{ij}$

Transform Domain Features

Wavelet Transform



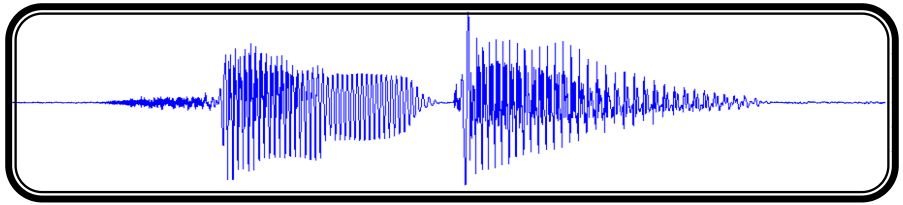
Wavelet Transform



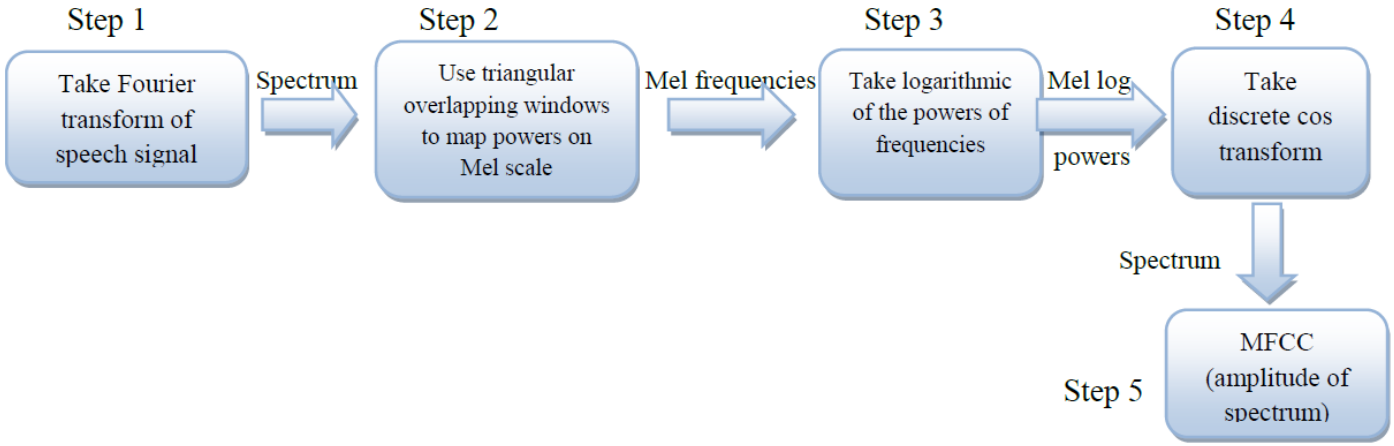
Source: Wikipedia

Audio

Time Domain Feature – LPC



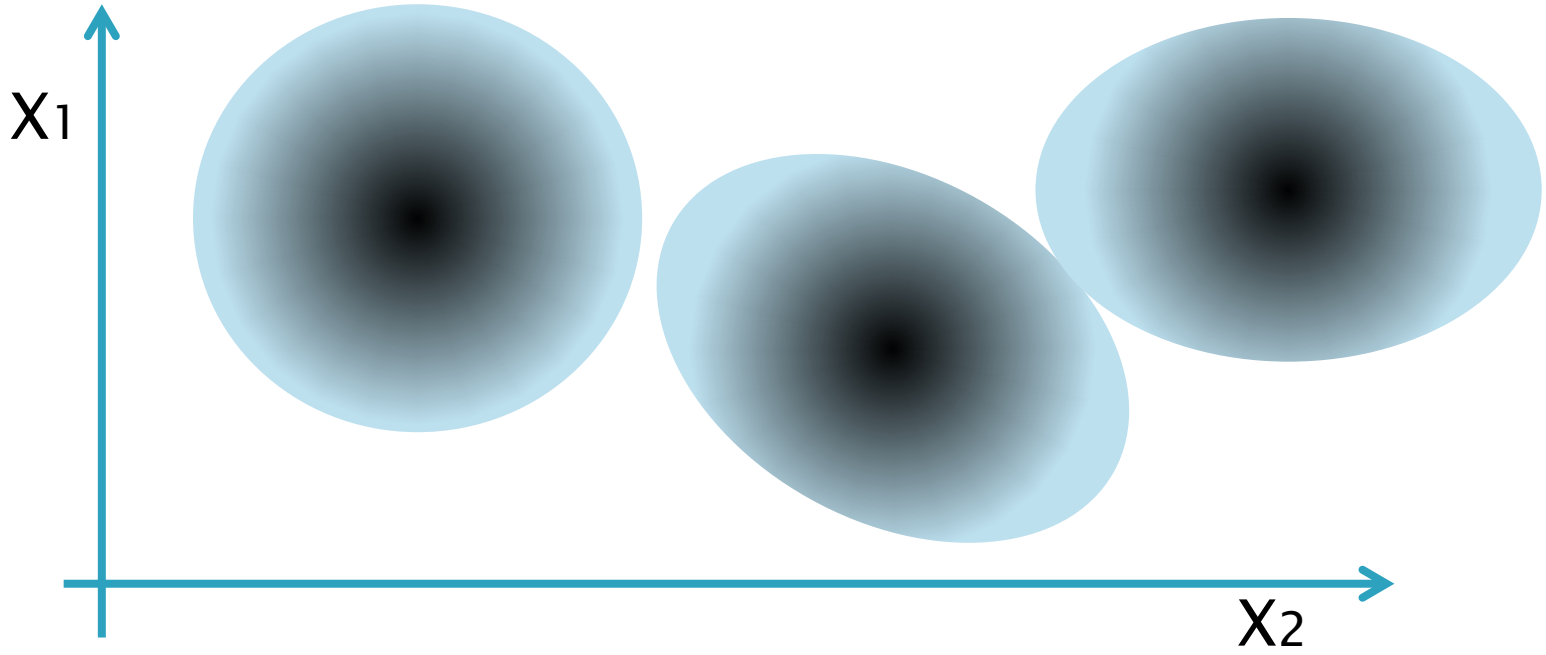
Spectral Domain- MFCC



Feature Space Representation

Image Source: Internet

Feature Space Representation



Feature Space Representation

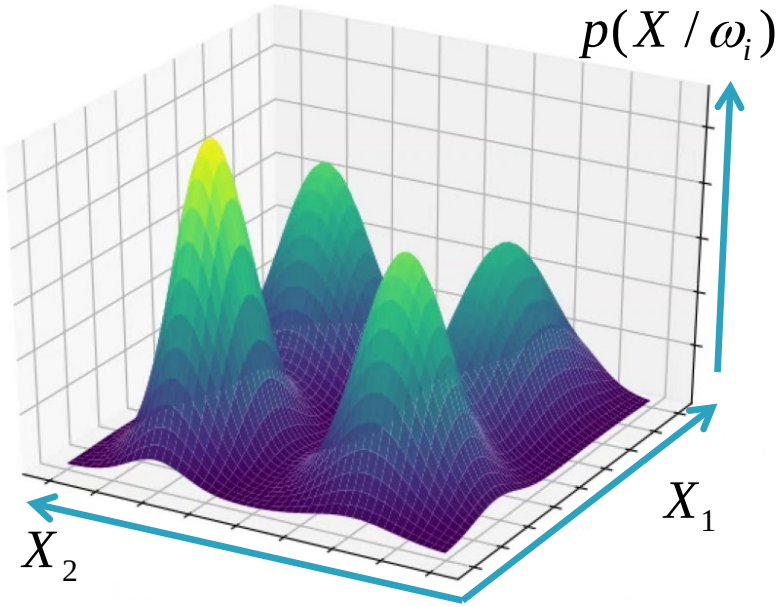


Image Source: Internet

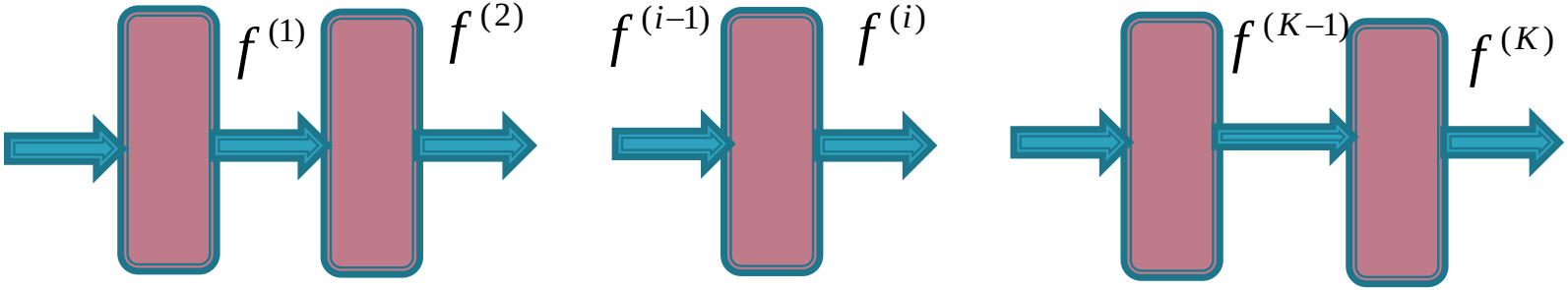
Bayesian Learning

$$\textit{Class conditional density} \Rightarrow p(X / \omega_i)$$

$$\textit{For classification} \Rightarrow p(\omega_i / X)$$

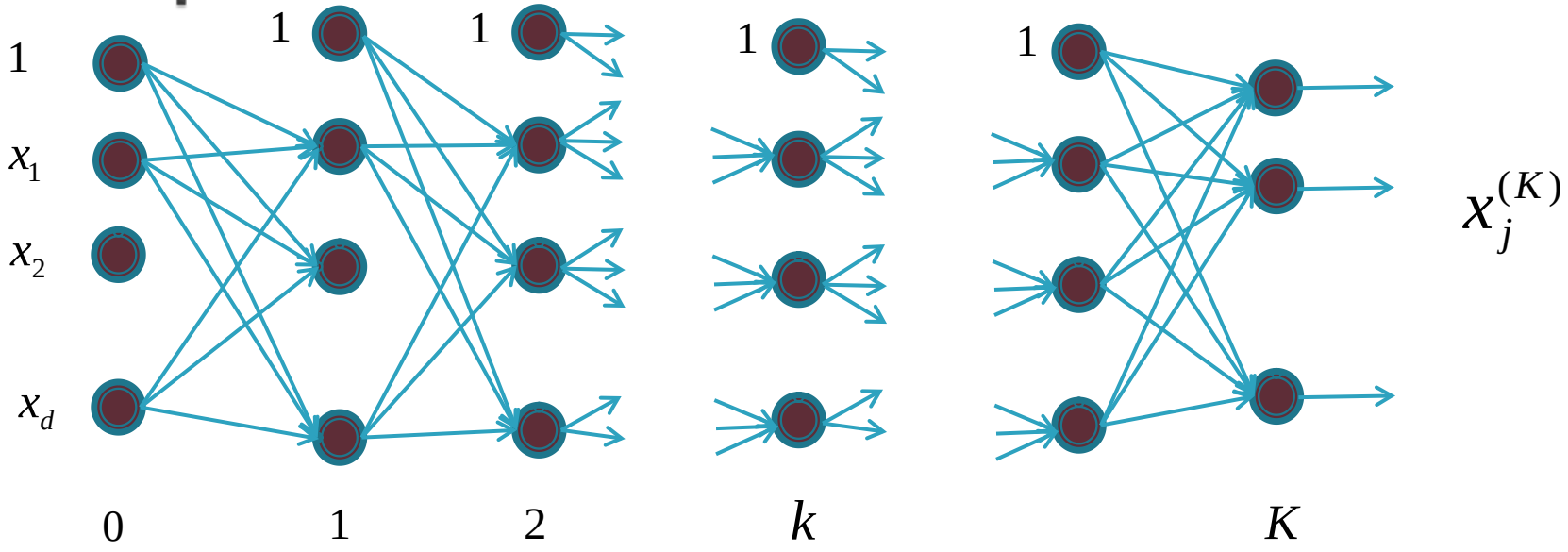
$$p(\omega_i / X) = \frac{p(X / \omega_i) P(\omega_i)}{p(X)}$$

Neural Network Function



$$f^{(K)}(f^{(K-1)}.....(f^{(i)}....(f^{(2)}(f^{(1)}(X)))))$$

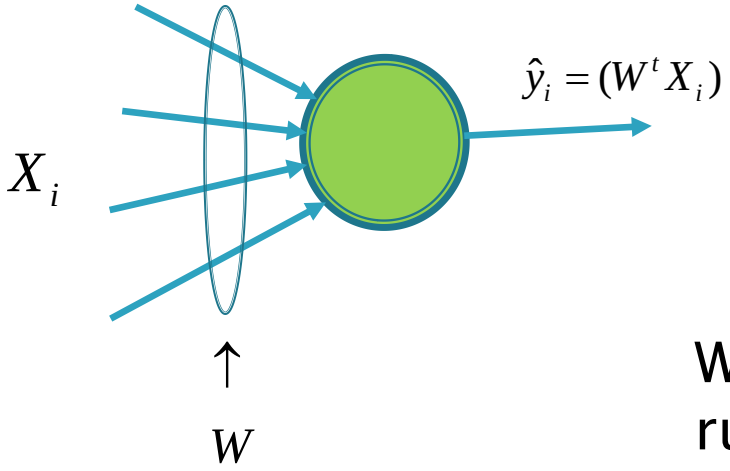
Multilayer Perceptron



$M_k \rightarrow$ No. of nodes in k^{th} layer

Back Propagation Learning

Single Layer Network– Single Output without nonlinearity



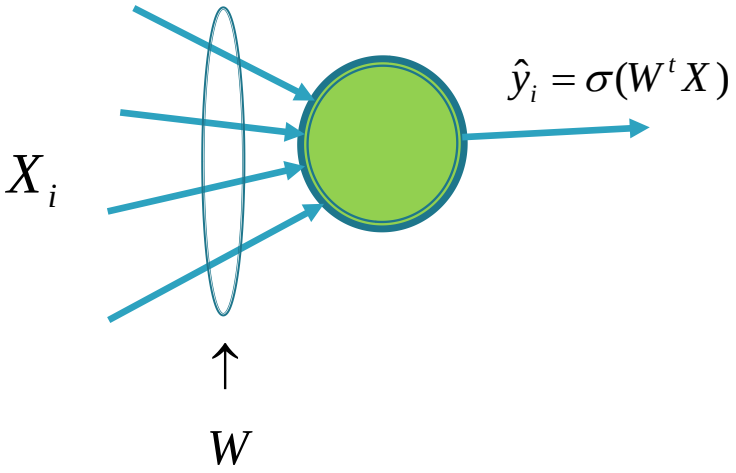
$$E = \frac{1}{2}\sum_{i=1}^{N}(W^t X_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

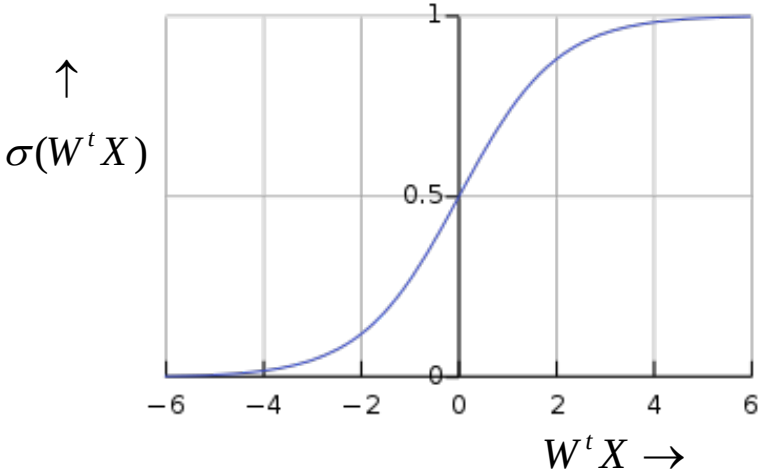
$$\nabla_W E = \sum_{i=1}^{N}(\hat{y}_i - y_i)X_i$$

Weight updation rule

$$W \leftarrow W - \eta\sum_{i=1}^{N}(\hat{y}_i - y_i)X_i$$

Single Layer Network– Single Output with nonlinearity

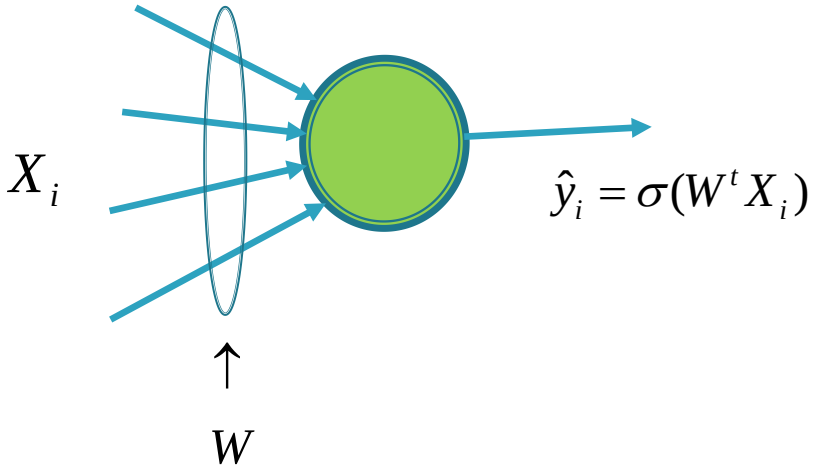




Single Layer Network– Single Output with nonlinearity

$$E = \frac{1}{2}(\hat{y}_i - y_i)^2 = \frac{1}{2}(\sigma(W^t X_i) - y_i)^2$$

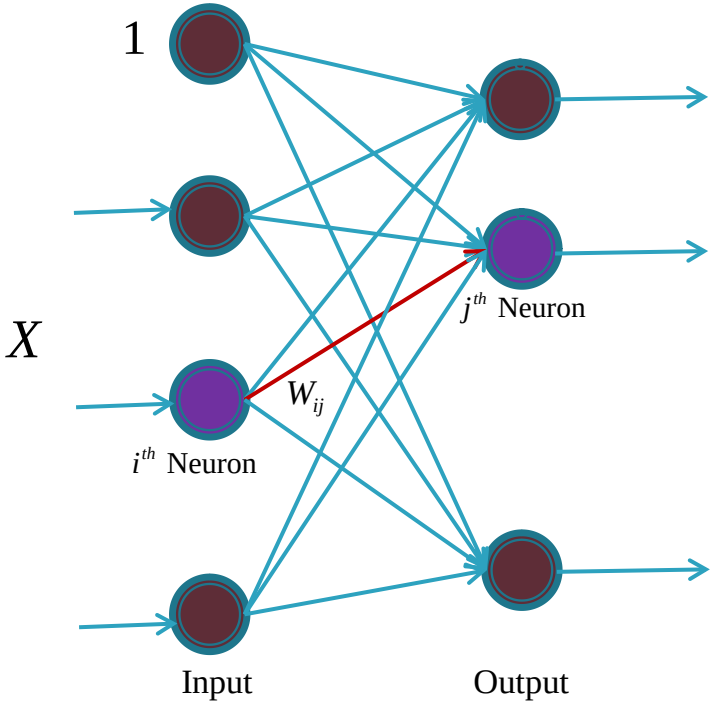
$$\frac{\partial E}{\partial W} = \hat{y}_i(1-\hat{y}_i)(\hat{y}_i - y_i)X_i$$



Weight updation rule $\Rightarrow$

$$W \leftarrow W - \eta\hat{y}_i(1-\hat{y}_i)(\hat{y}_i - y_i)X_i$$

Back Propagation Learning:- Single Layer Multiple Output



$$o_j = \frac{1}{1+e^{-\theta_j}} \qquad \theta_j = \sum_{i=1}^{D} W_{ij} x_i$$

$$E = \frac{1}{2}\sum_{j=1}^{M}\left(o_j - t_j\right)^2$$

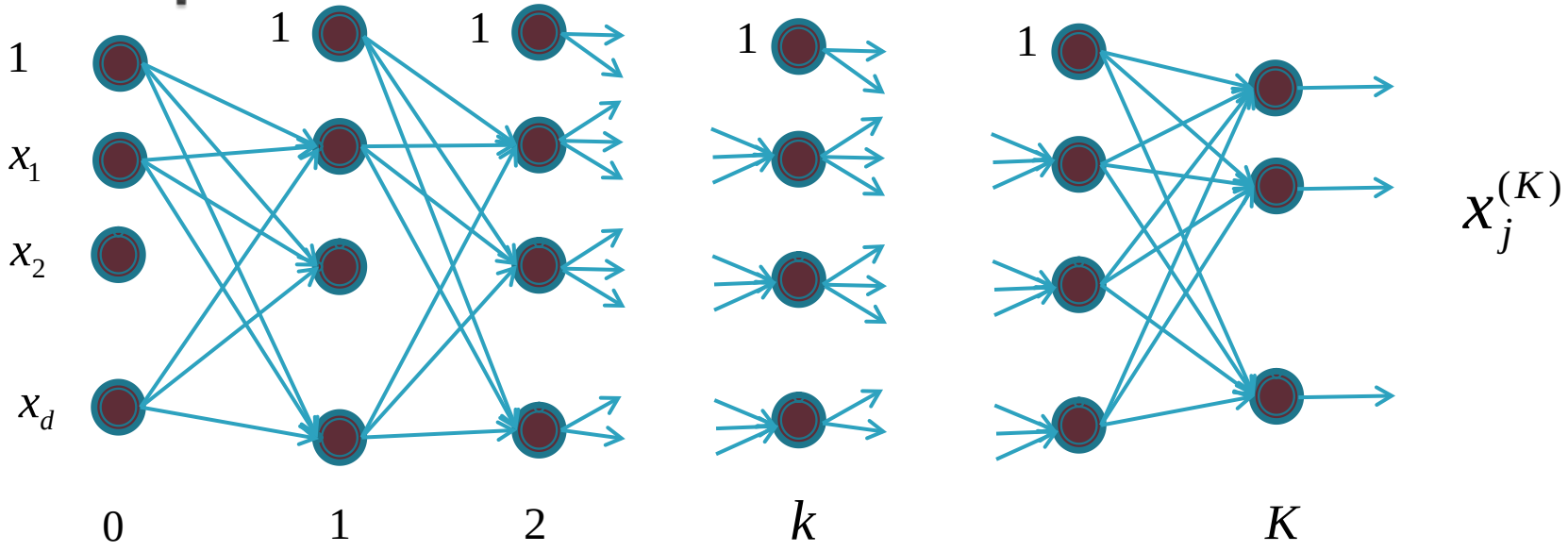
$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial \theta_j} \cdot \frac{\partial \theta_j}{\partial W_{ij}}$$

$$= (o_j - t_j) o_j (1 - o_j) x_i$$

Weight updation rule $\Rightarrow$

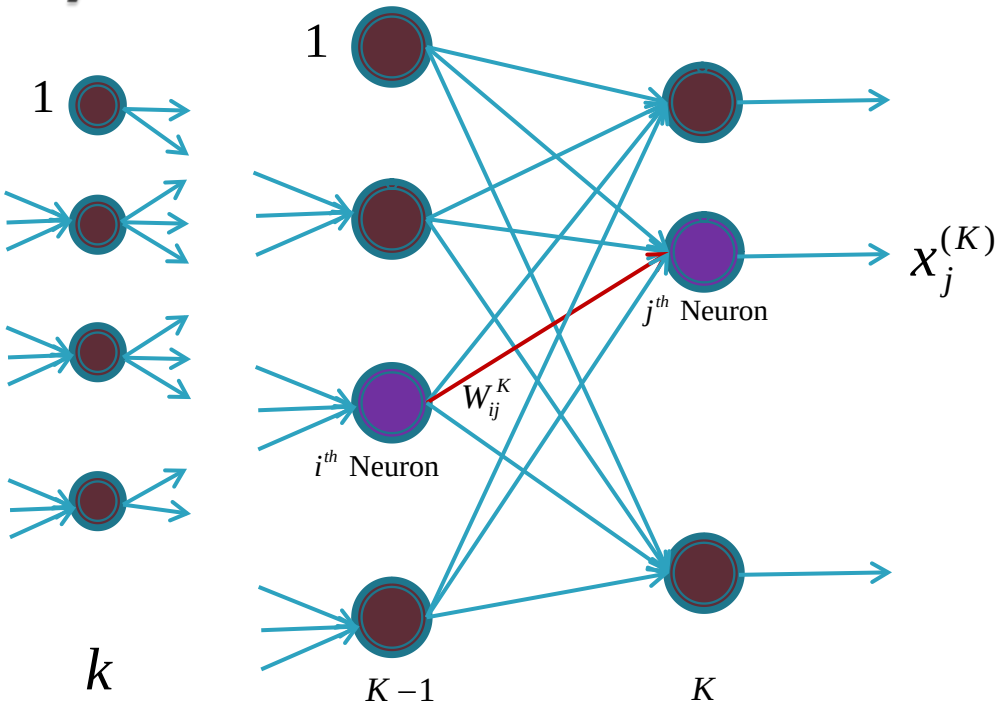
$$W_{ij} \leftarrow W_{ij} - \eta (o_j - t_j) o_j (1 - o_j) x_i$$

Multilayer Perceptron



$M_k \rightarrow$ No. of nodes in k^{th} layer

Back Propagation Learning:- Output Layer



$$x_j^K = \frac{1}{1+e^{-\theta_j^K}} \qquad \theta_j^K = \sum_{i=1}^{M_{K-1}} W_{ij}^K x_i^{K-1}$$

$$E = \frac{1}{2}\sum_{j=1}^{M_K}\left(x_j^K - t_j\right)^2$$

Back Propagation Learning:- Output Layer

Find W_{ij}^{K} that minimizes $E = \frac{1}{2}\sum_{j=1}^{M_K}\left(x_j^K - t_j\right)^2$

Gradient Descent $\nabla_{W_{ij}^{K}} E$

Back Propagation Learning:- Output Layer

$$\frac{\partial E}{\partial W_{ij}^K} = \frac{\partial E}{\partial x_j^K} \cdot \frac{\partial x_j^K}{\partial \theta_j^K} \cdot \frac{\partial \theta_j^K}{\partial W_{ij}^K}$$

$$= (x_j^K - t_j) x_j^K (1 - x_j^K) x_i^{K-1}$$

Let $\delta_j^K = x_j^K (1 - x_j^K)(x_j^K - t_j)$

$$\Rightarrow \frac{\partial E}{\partial W_{ij}^K} = \delta_j^K x_i^{K-1}$$

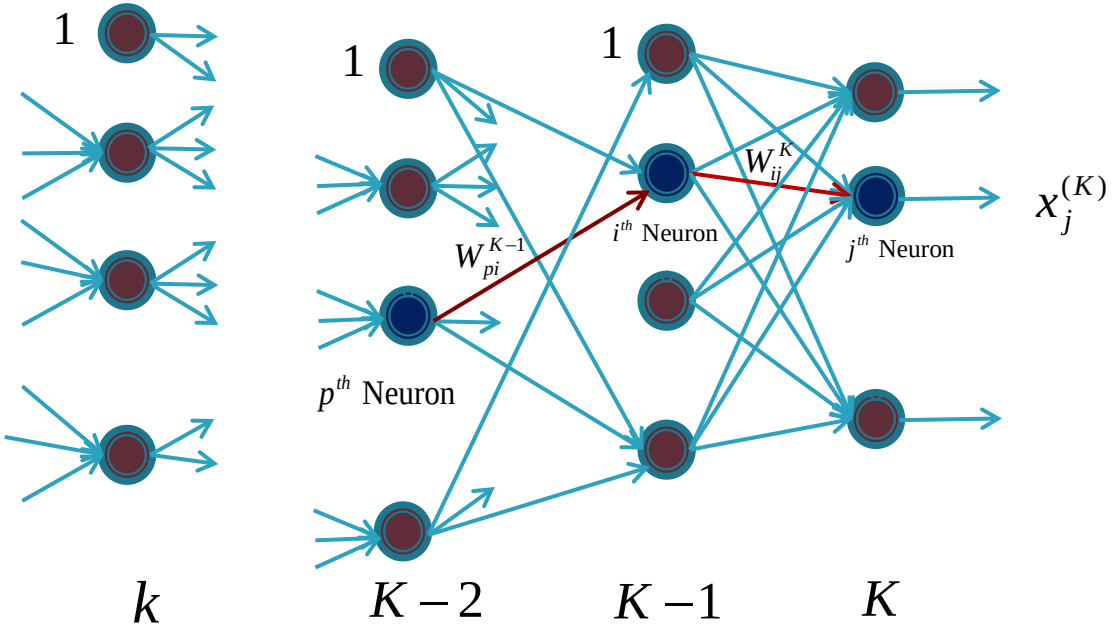
$$x_j^K = \frac{1}{1 + e^{-\theta_j^K}} \qquad \theta_j^K = \sum_{i=1}^{M_{K-1}} W_{ij}^K x_i^{K-1}$$

Weight updation rule

Output Layer

$$W_{ij}^K \leftarrow W_{ij}^K - \eta \delta_j^K x_i^{K-1}$$

Back Propagation Learning:- Hidden Layer



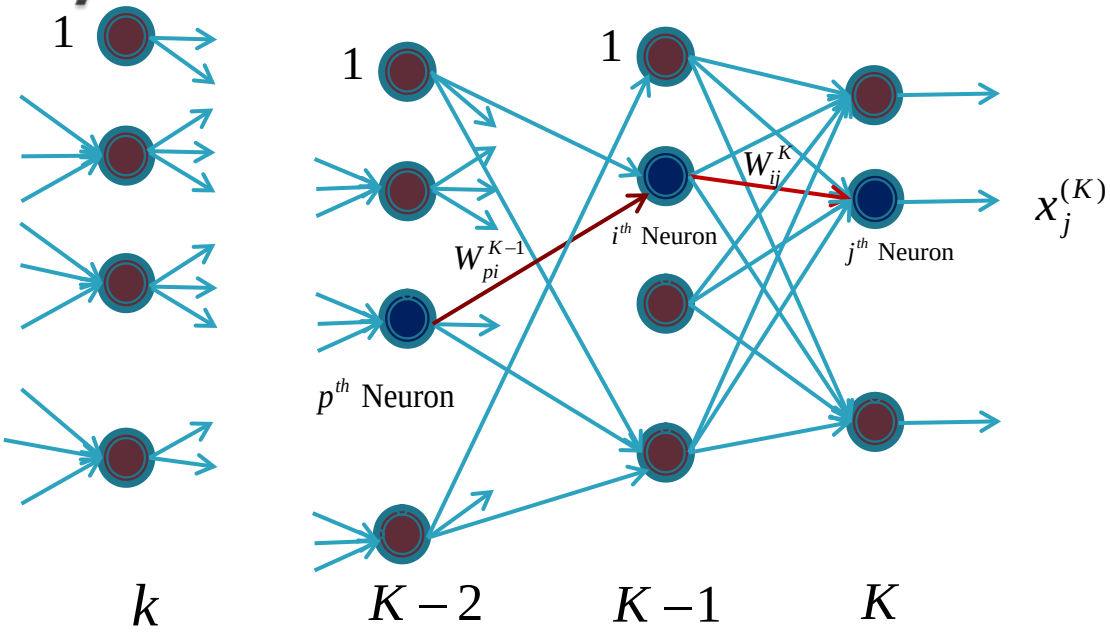
$$E = \frac{1}{2}\sum_{j=1}^{M_K}\left(x_j^K - t_j\right)^2$$

Back Propagation Learning:- Hidden Layer

Find W_{pi}^{K-1} that minimizes $E = \frac{1}{2}\sum_{j=1}^{M_K}\left(x_j^K - t_j\right)^2$

Gradient Descent $\Rightarrow \nabla_{W_{pi}^{K-1}} E$

Back Propagation Learning:- Hidden Layer



$$x_i^{K-1} = \frac{1}{1+e^{-\theta_i^{K-1}}}$$

$$\theta_i^{K-1} = \sum_{p=1}^{M_{K-2}} W_{pi}^{K-1} x_p^{K-2}$$

Back Propagation Learning:- Hidden Layer

$$\frac{\partial E}{\partial W_{pi}^{K-1}} = \frac{\partial E}{\partial x_i^{K-1}} \cdot \frac{\partial x_i^{K-1}}{\partial W_{pi}^{K-1}}$$

$$= \frac{\partial E}{\partial x_i^{K-1}} \cdot \frac{\partial x_i^{K-1}}{\partial \theta_i^{K-1}} \cdot \frac{\partial \theta_i^{K-1}}{\partial W_{pi}^{K-1}}$$

$$= \frac{\partial E}{\partial x_i^{K-1}} . x_i^{K-1}(1 - x_i^{K-1}) . x_p^{K-2}$$

$$x_i^{K-1} = \frac{1}{1 + e^{-\theta_i^{K-1}}}$$

$$\theta_i^{K-1} = \sum_{p=1}^{M_{K-2}} W_{pi}^{K-1} x_p^{K-2}$$

Back Propagation Learning:- Hidden Layer

$$\frac{\partial E}{\partial x_i^{K-1}} = \frac{\partial E}{\partial x_j^K} \cdot \frac{\partial x_j^K}{\partial \theta_j^K} \cdot \frac{\partial \theta_j^K}{\partial x_i^{K-1}}$$

$$= \sum_{j=1}^{M_K} (x_j^k - t_j) x_j^K (1 - x_j^K) W_{ij}^K$$

$$= \sum_{j=1}^{M_K} \partial_j^K W_{ij}^K$$

$$E = \frac{1}{2} \sum_{j=1}^{M_K} \left(x_j^K - t_j \right)^2$$

$$x_j^K = \frac{1}{1 + e^{-\theta_j^K}} \qquad \theta_j^K = \sum_{i=1}^{M_{K-1}} W_{ij}^K x_i^{K-1}$$

$$\delta_j^K = x_j^K (1 - x_j^K)(x_j^K - t_j)$$

Back Propagation Learning:- Hidden Layer

$$\frac{\partial E}{\partial W_{pi}^{K-1}} = \frac{\partial E}{\partial x_i^{K-1}} . x_i^{K-1}(1-x_i^{K-1}).x_p^{K-2} \quad = x_i^{K-1}(1-x_i^{K-1}).x_p^{K-2}\sum_{j=1}^{M_K}\partial_j^K W_{ij}^K$$

Putting $\delta_i^{K-1} = x_i^{K-1}(1-x_i^{K-1})\sum_{j=1}^{M_K}\partial_j^K W_{ij}^K$

Weight updation rule

Last but Output Layer

$$W_{pi}^{K-1} \leftarrow W_{pi}^{K-1} - \eta\delta_i^{K-1}x_p^{K-2}$$

Back Propagation Learning:- Hidden Layer

For any hidden layer weight W_{ij}^{k}

Putting $\delta_i^k = x_i^k(1-x_i^k)\sum_{j=1}^{M_{k+1}} \partial_j^{k+1} W_{ij}^{k+1}$

Weight updation rule

$$W_{ij}^{k} \leftarrow W_{ij}^{k} - \eta \delta_j^k x_i^{k-1}$$

Machine Learning
vs
Deep Learning

Convolution Kernel

1 D → 2A+1

$$y(n) = \sum_{p=-A}^{A} w(p)x(n-p)$$

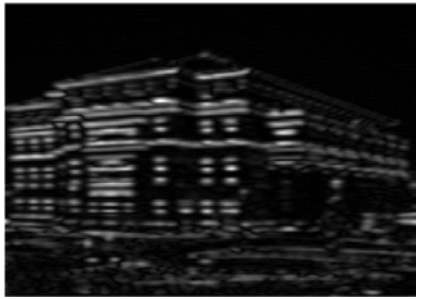
2 D →

(2A+1)x(2A+1)

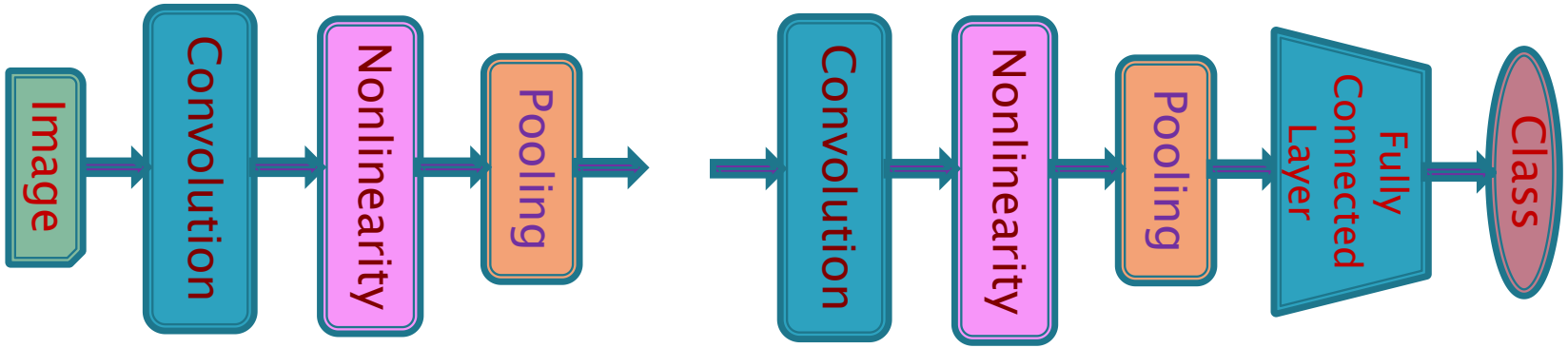
$$y(m,n) = \sum_{p=-A}^{A}\sum_{q=-A}^{A} w(p,q)x(m-p,n-q)$$

Finite Convolution Kernel

Feature at a point is local in nature

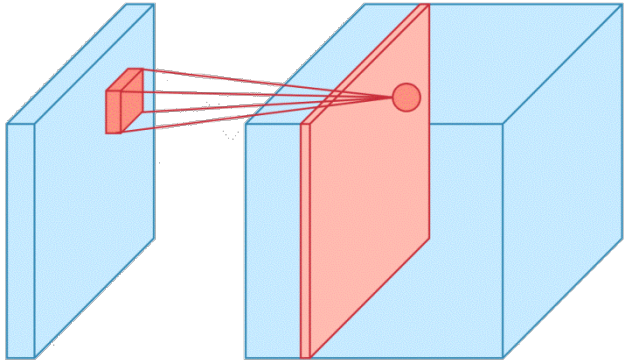


CNN Architecture



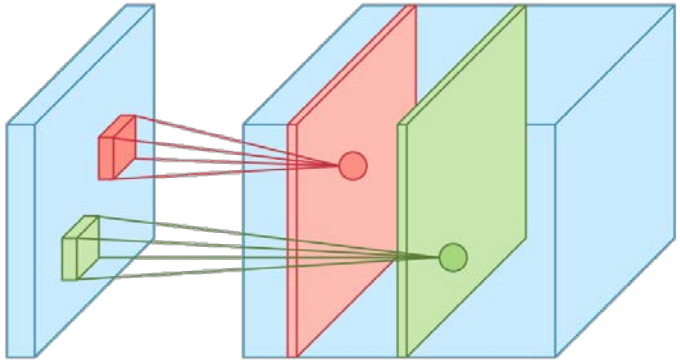
3 D Convolution- Visualization

- The kernel strides over the input Image.
- At each location $I(m,n)$ compute $f(m,n) = \sum\sum w(p,q)I(p-m,q-n)$ collect them in the feature map.
- The animation shows the sliding operation at 4 locations, but in reality it is performed over the entire input.



Animation:- Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

3 D Convolution- Visualization



- Red and green boxes are two different featured maps obtained by convolving the same input with two different kernels.
- The feature maps are stacked along the depth dimension as shown.

Figure: Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

3 D Convolution- Visualization

- An RGB Image of size 32X32X3
- 10 Kernels of size 5x5x3
- Output featuremap of size 32x32x10

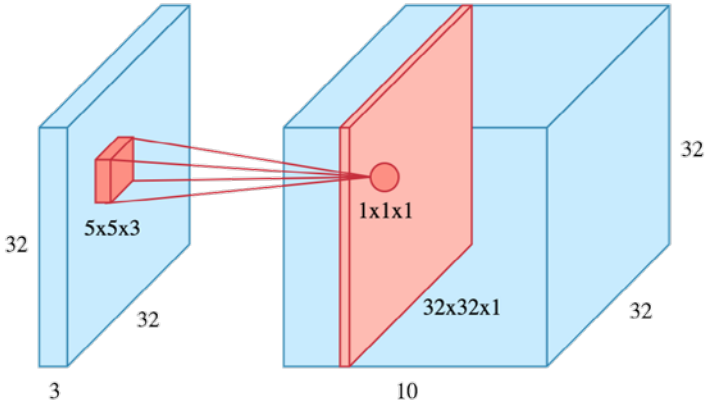


Figure: Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Nonlinearity

- ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero

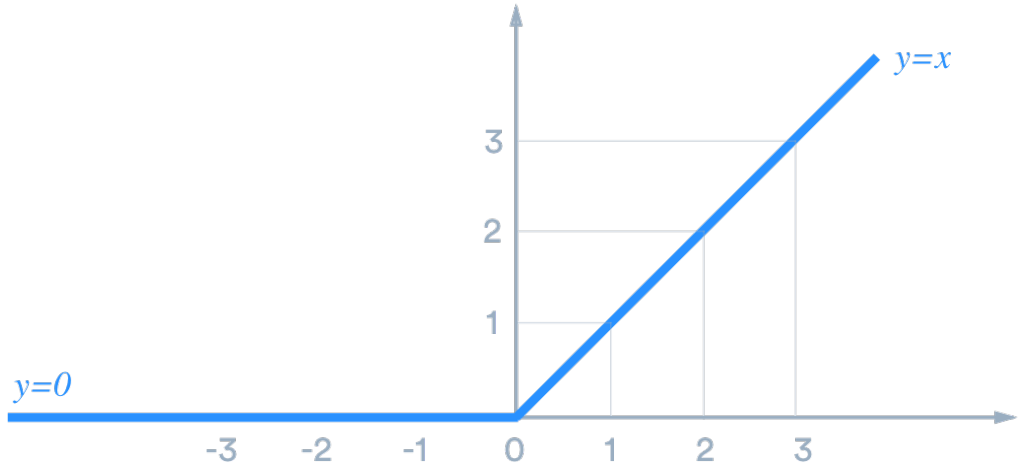


Figure: Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Pooling

- Replaces the output of a node at certain locations with a summary statistic of nearby locations.
- Spatial Pooling can be of different types: Max, Average, Sum etc.
- Max Pooling report the maximum output within a rectangular neighborhood.
- Pooling helps to make the output approximately invariant to small translation.
- Pooling layers down sample each feature map independently, reducing the height and width, keeping the depth intact.
- In pooling layer stride and window size needs to be specified

Pooling

- Figure below is the result of max pooling using a 2x2 window and stride 2. Each color denotes a different window. Since both the window size and stride are 2, the windows are not overlapping

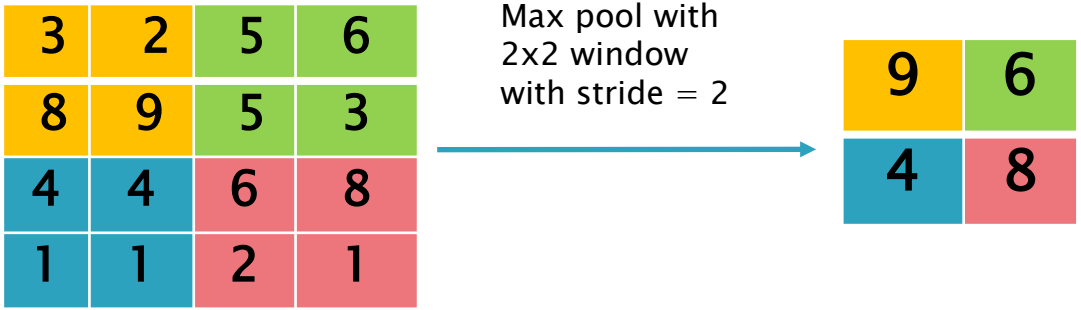


Figure: Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Pooling

- Pooling reduces the height and the width of the feature map, but the depth remains unchanged as shown in figure
- Pooling operation is independently carried out across each depth

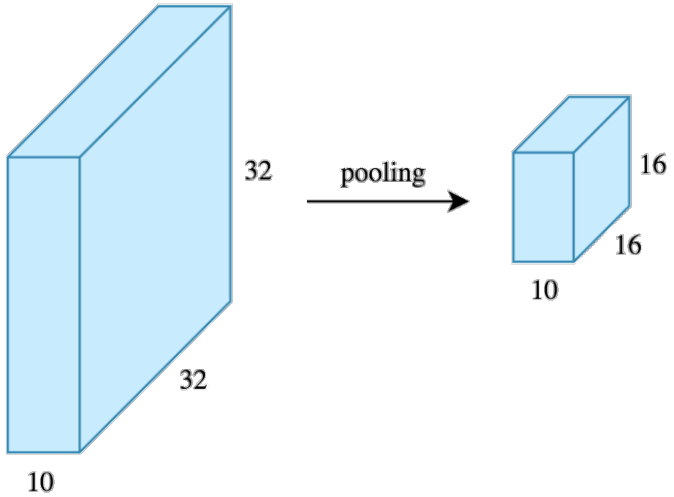
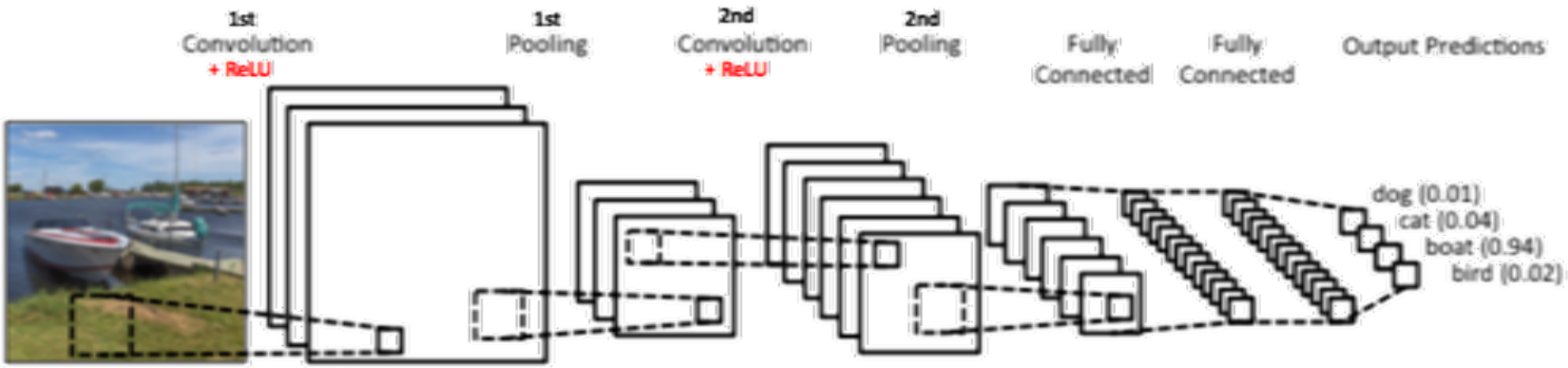
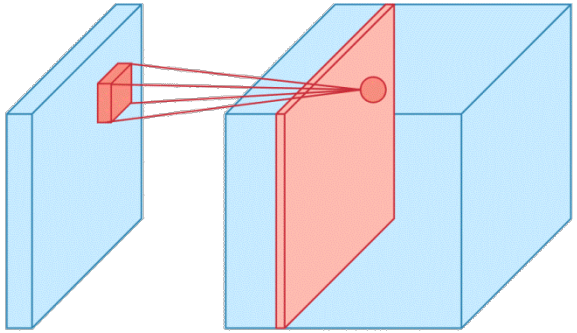


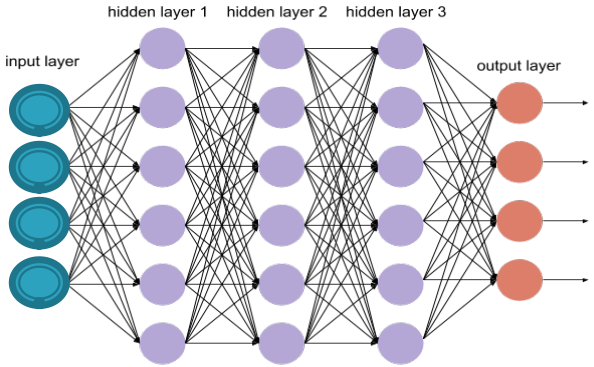
Figure: Arden Dertat https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Architecture



MLP vs CNN

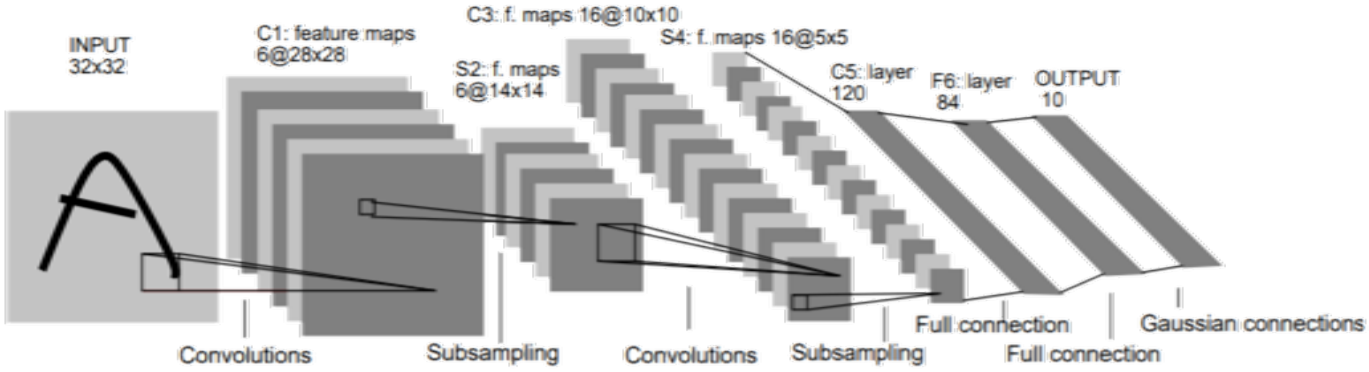




- Sparse Connectivity: Every node in the Convolution Layer receives input from a small number of nodes in the previous layer (Receptive Field), needing smaller number of parameters.
- Parameter Sharing: Each member of the Convolution Kernel is used at every position of the input, dramatically reducing the number of parameters.
- This makes CNN much more efficient than MLP.

LeNet 5

- Proposed by Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner for handwritten and machine-printed character recognition.
- Used by many Banks for recognition of hand written numbers on cheques.
- This architecture achieves an error rate as low as **0.95%** on test data



Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner, "Gradient -Based Learning Applied to Document Recognition", Proc. IEEE, Nov. 1998

IMAGENET Large Scale Visual Recognition Challenge (ILSVRC)

ILSVRC

- IMAGENET Large Scale Visual Recognition Challenge.
- Evaluates algorithms for Object Detection and Image Classification on large image database.
- Helps researchers to review state of the art Machine Learning techniques for object detection across a wider variety of objects.
- Monitor the progress of computer vision for large scale image indexing for retrieval and annotation.
- Database contains large number of Images from 1000 categories.
- More than 1000 images in every category.

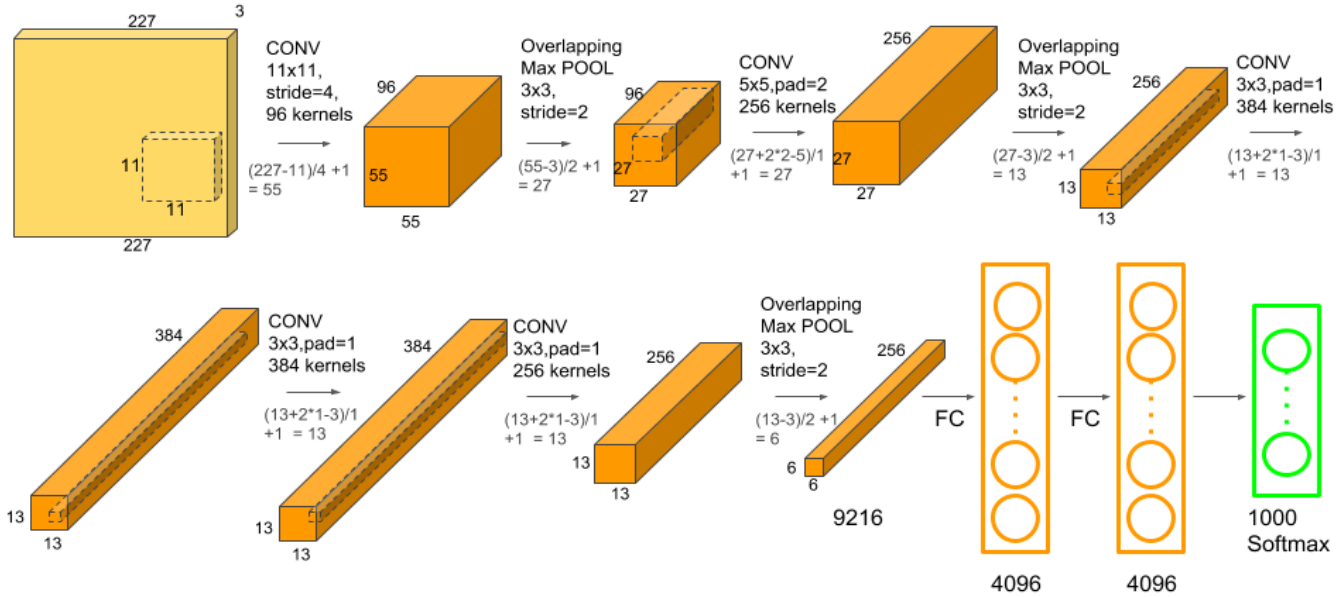
ILSVRC

- Every year of the challenge the forum also organizes a workshop at one of the premier computer vision conferences.
- The purpose of the workshop is to disseminate the new findings of the challenge.
- Contestants with the most successful and innovative techniques are invited to present their work.

AlexNet
ILSVRC 2012 Winer

Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, “Imagenet Classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, 2012

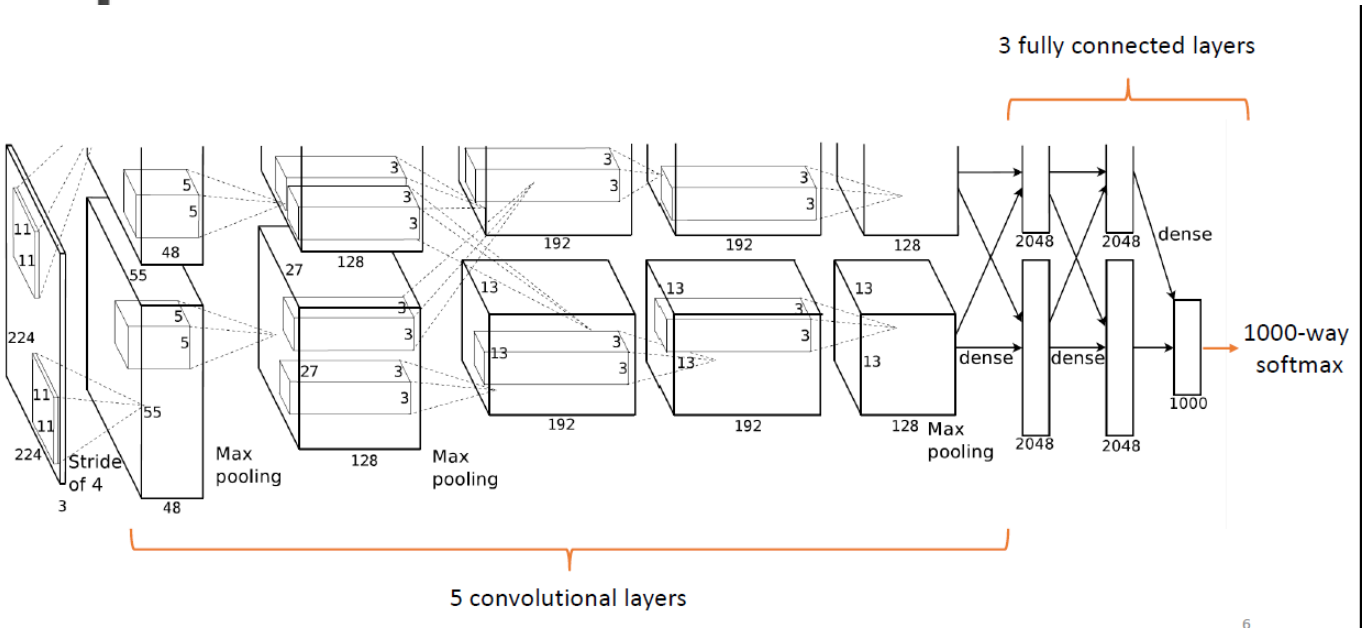
AlexNet



ILSVRC 2012
Winner

https://www.learnopencv.com/understanding-alexnet/

AlexNet



Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, “Imagenet Classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, 2012

AlexNet

- 60 Million parameters and 650000 neurons.
- The network is split into two pipelines and was trained on two GPU.
- Input Image size 256 x 256 RGB.
- Grey scale images to be replicated to obtain 3-Channel RGB
- Random crops of size 227 x 227 are fed to the input layer of AlexNet.
- Stochastic Gradient Descent with Momentum Optimizer.
- Top-5 error rate 15.3%.

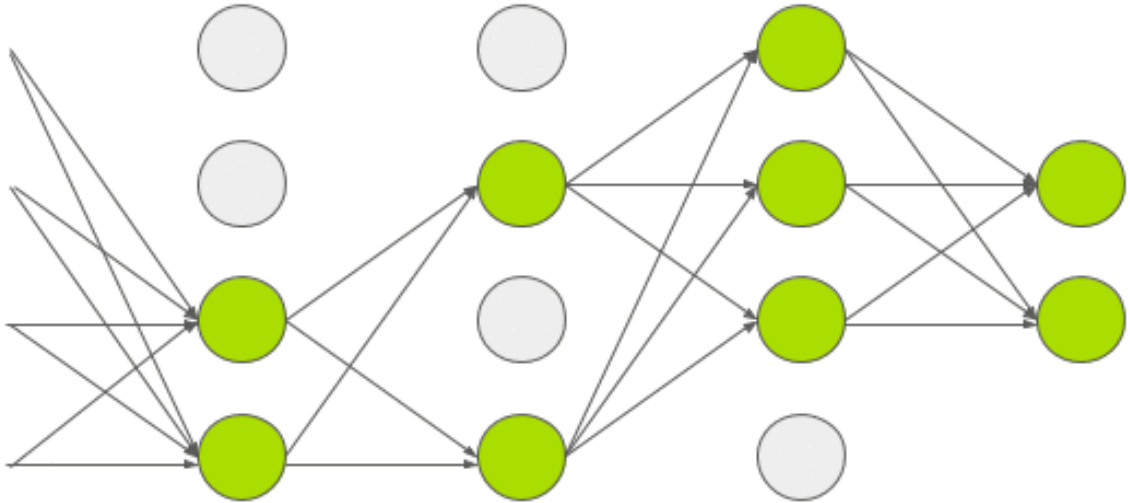
Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, "Imagenet Classification with deep convolutional neural networks", Advances in Neural Information Processing Systems, 2012

Reducing Overfitting

- Train the network with different variants of the same image helps avoiding overfitting
  - Generate additional data from existing data (Augmentation)
  - Data augmentation by mirroring
  - Data Augmentation by random crops
- Dropout Regularization

Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, “Imagenet Classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, 2012

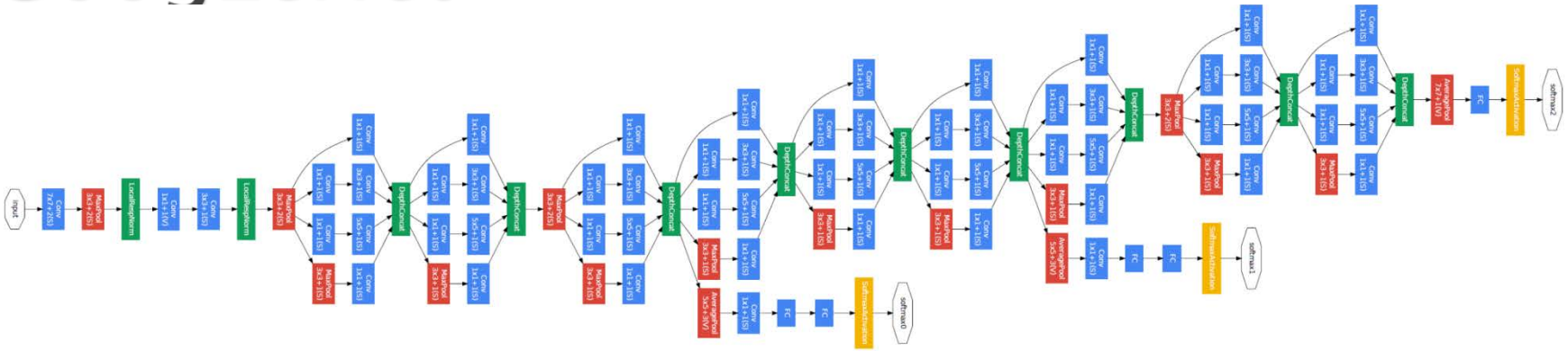
AlexNet



https://www.learnopencv.com/understanding-alexnet/

GoogLeNet

GoogLeNet

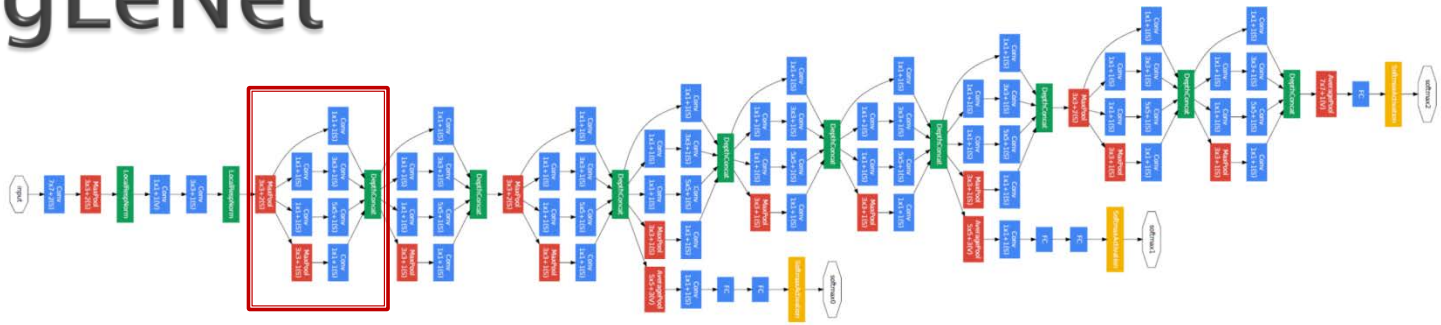


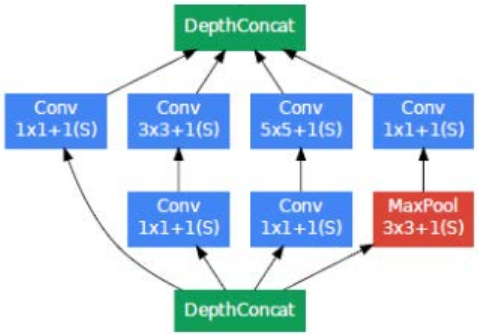
- 22 Layers with parameters
- 27 Layer including Maxpool layers

- Convolution Layer
- Maxpool Layer
- Feature Concatenation
- Softmax Layer

Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, "Imagenet Classification with deep convolutional neural networks", Advances in Neural Information Processing Systems, 2012

GoogLeNet





Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hilton, “Imagenet Classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, 2012

Discriminative vs. Generative Model

Image Source: Internet

Discriminative Model


Cat


Dog


Dog


Cat


Cat


Dog

$x_i \sim P(x)$

P (X)

Learning Algorithm (θ)

Error Backprop

Loss

Predicted P (Y | X)

Cat	Cat
Cat	Dog
Dog	Cat

Image Source: Internet

Generative Model

"What I can not create, I do not understand."

– Richard Feynman

- Collect a large amount of data in some domain
- Train a model to generate data like it.

Autoencoder

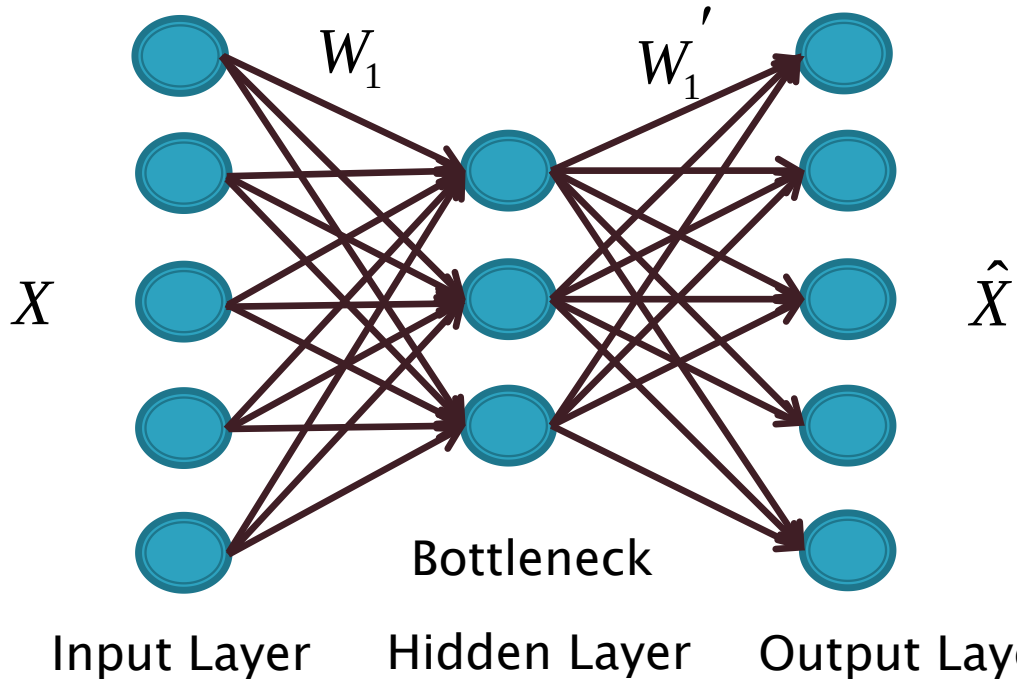
- Unsupervised Learning where Neural Networks are subject to the task of representation learning.
- Impose a bottleneck in the network
- The bottleneck forces a compressed knowledge representation of the input.

Autoencoder

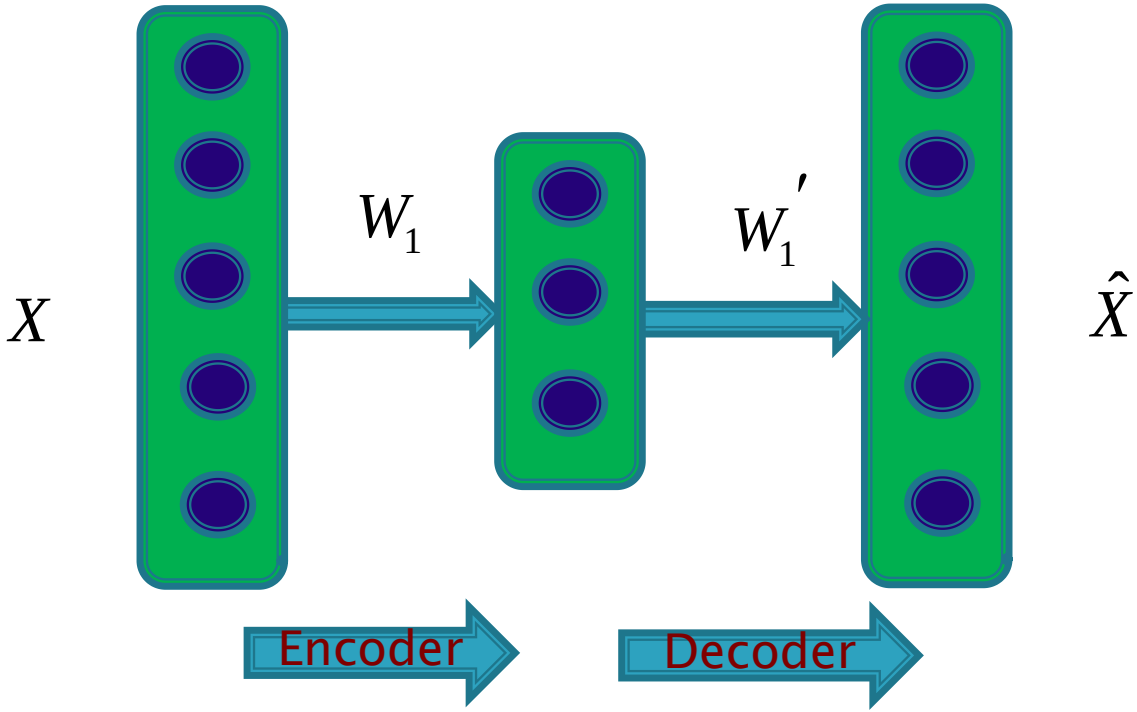
Assumption:

- High degree of correlation/structure exists in the data.
- For uncorrelated data (input features are independent), then compression and subsequent reconstruction would be difficult.

Autoencoder

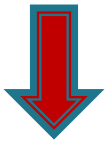


Autoencoder



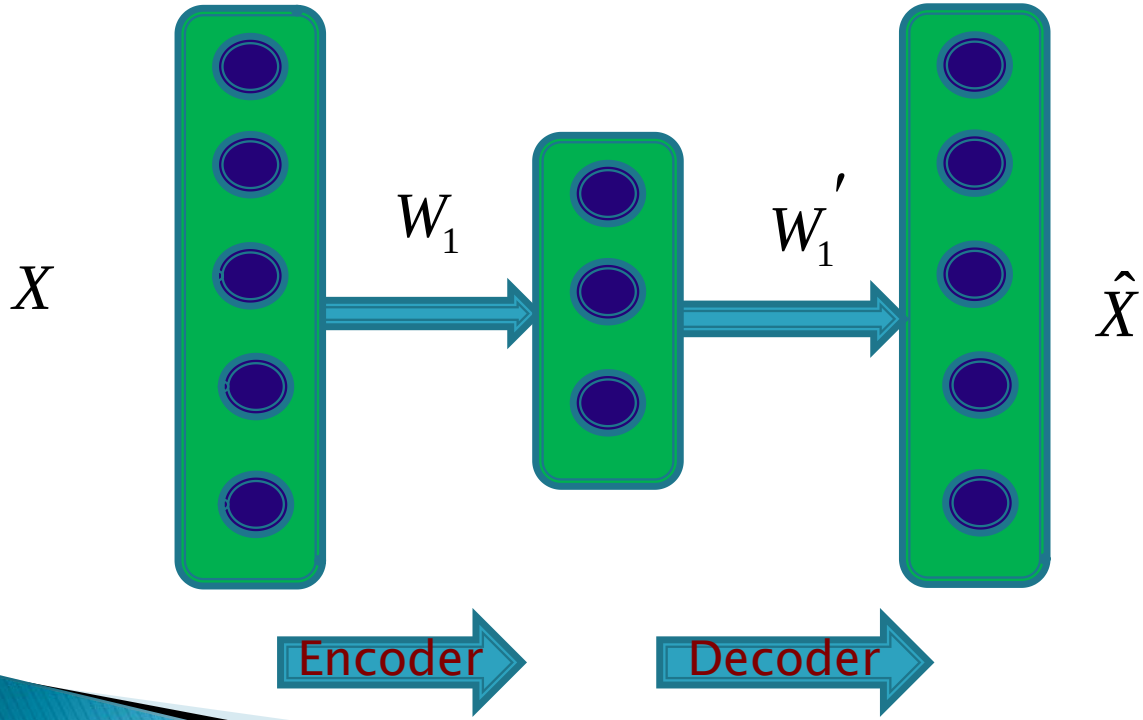
Expectation

- Sensitive enough to input for accurate reconstruction
- Insensitive enough that it does not memorize or overfit the training data



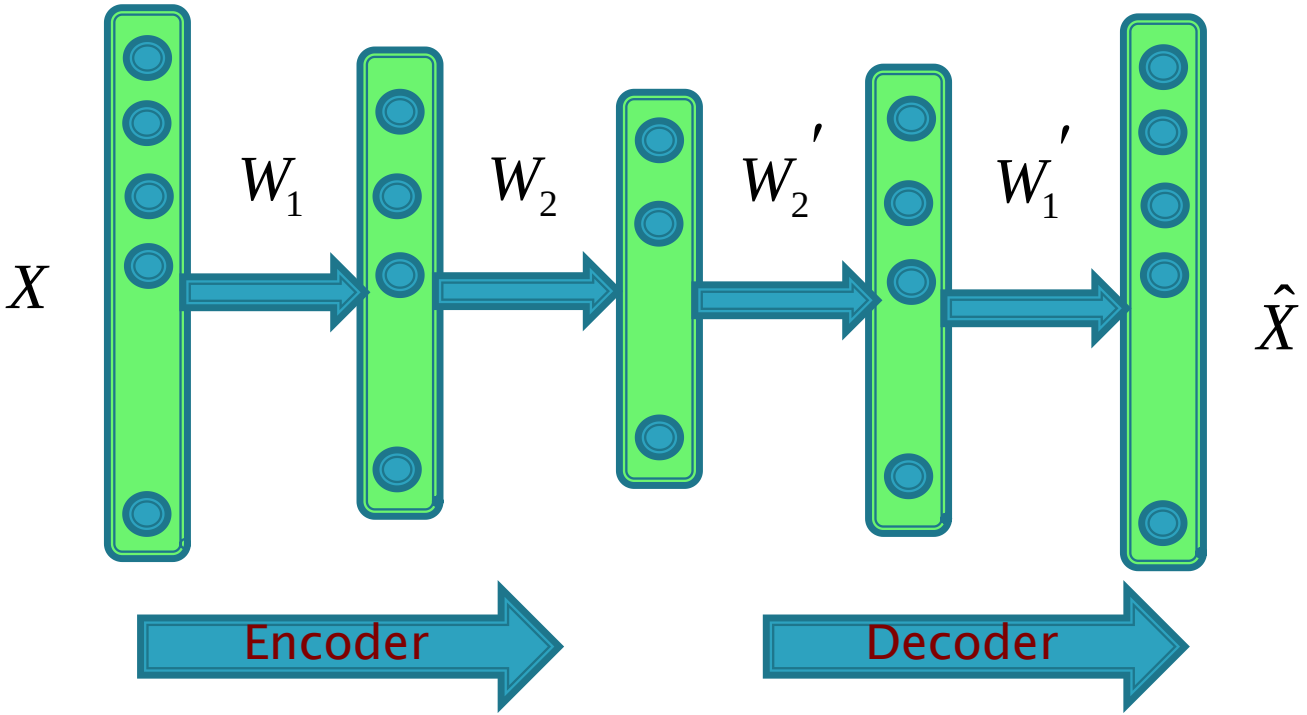
$$\text{Loss Function} \Rightarrow L(X, \hat{X}) + \text{Regularizer}$$

Undercomplete Autoencoder



$$L(X, \hat{X}) = \frac{1}{2} \sum_{N} \left\| X - \hat{X} \right\|^2$$

Stacked Autoencoder



$$L(X, \hat{X}) = \frac{1}{2} \sum_{N} \left\| X - \hat{X} \right\|^2$$