
ARTIFICIAL
NEURAL NETWORK

Introduction to DL

Unit 4

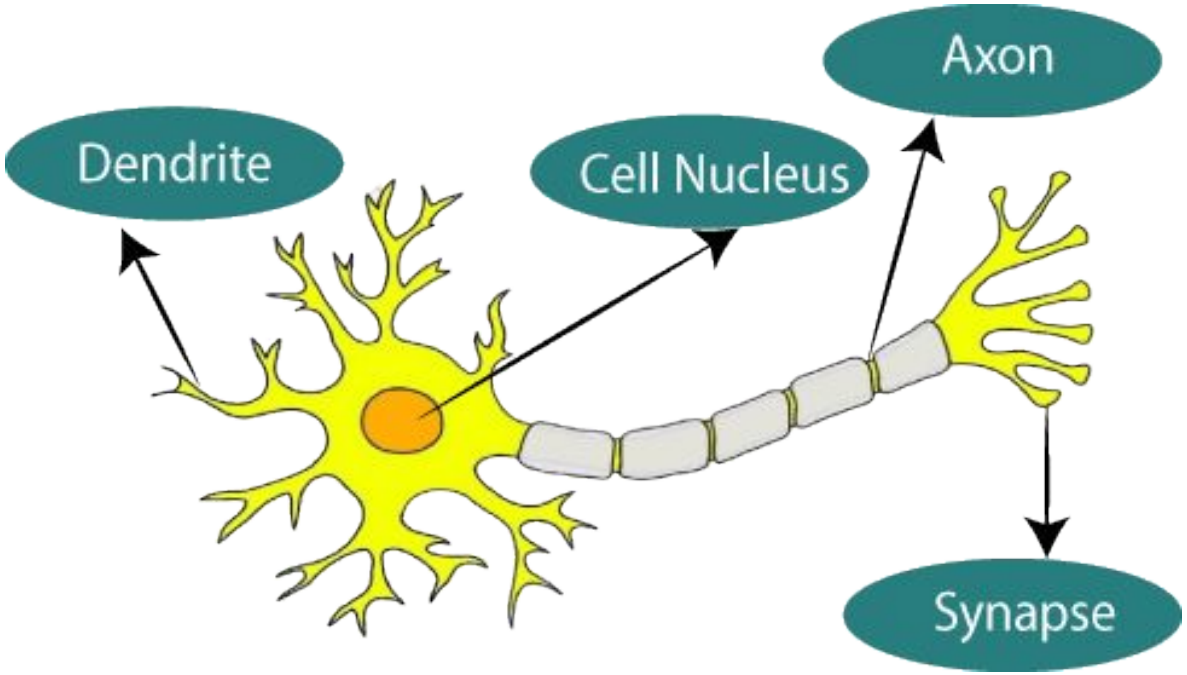
Syllabus

1. Introduction to Deep Learning, ANN, Machine Learning Vs Deep Learning, Working of Deep Learning;
2. Convolutional Neural Network: Introduction, Components of CNN Architecture, Properties of CNN, Architectures of CNN, Applications of CNN,
3. Recurrent Neural Network: Introduction, Simple RNN, LSTM Implementation, Deep RNN,
4. Autoencoder: Introduction, Features, Types, Applications of Deep Learning. Self-learning Topics: Restricted Boltzmann Machine (RBM).

DEEP LEARNING AND ARTIFICIAL NEURAL NETWORK

- Deep Learning is a type of machine learning that imitates the way humans gain certain type of knowledge.
- Deep Learning is a subset of Artificial Intelligence also an important element of data science, which includes statistics and predictive modelling.
- Deep Learning networks are mathematical models that are used to represent human brains in order to solve tasks with unstructured data. These mathematical models are created in form of neural network that consists of neurons.
- An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain.
- Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons (nodes) that are linked to each other in various layers of the networks.

BIOLOGICAL NEURAL NETWORK

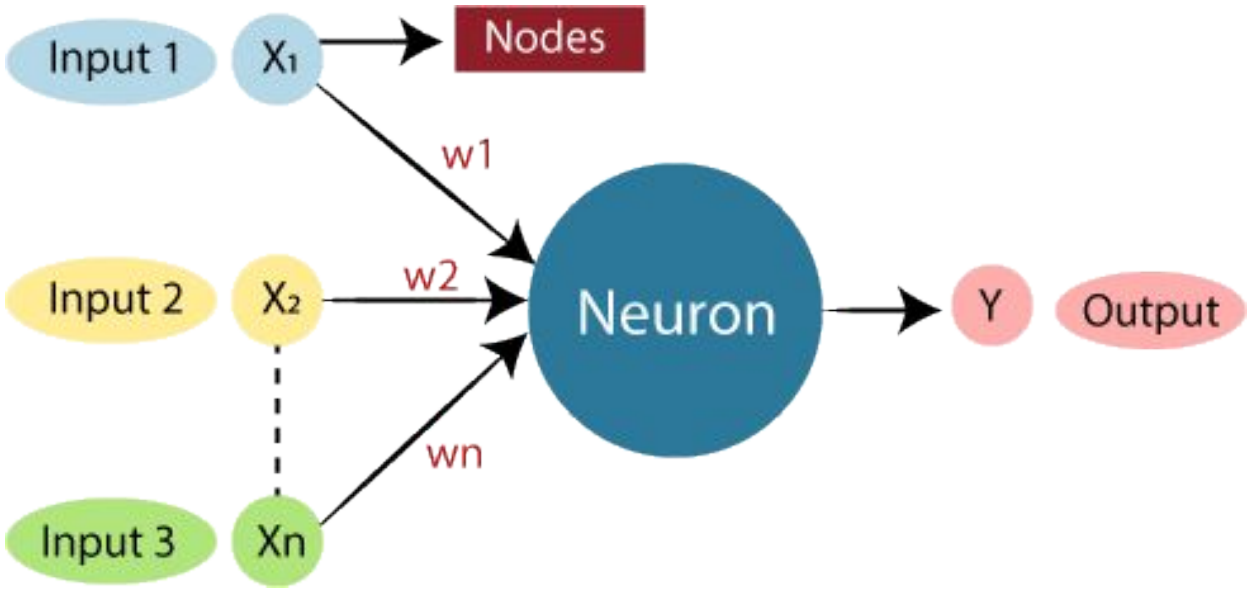


Soma or cell body: Here the cell nucleus is located.

Dendrites: Where the nerve is connected to the cell body.

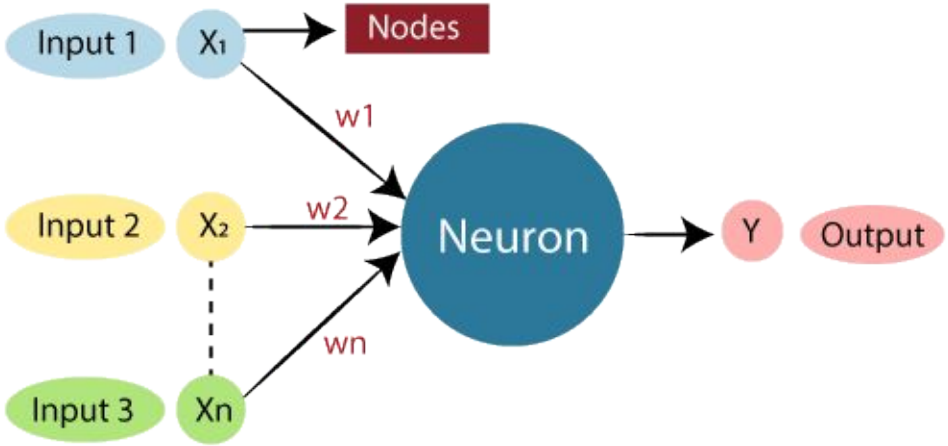
Axon: It carries the impulses of the neuron.

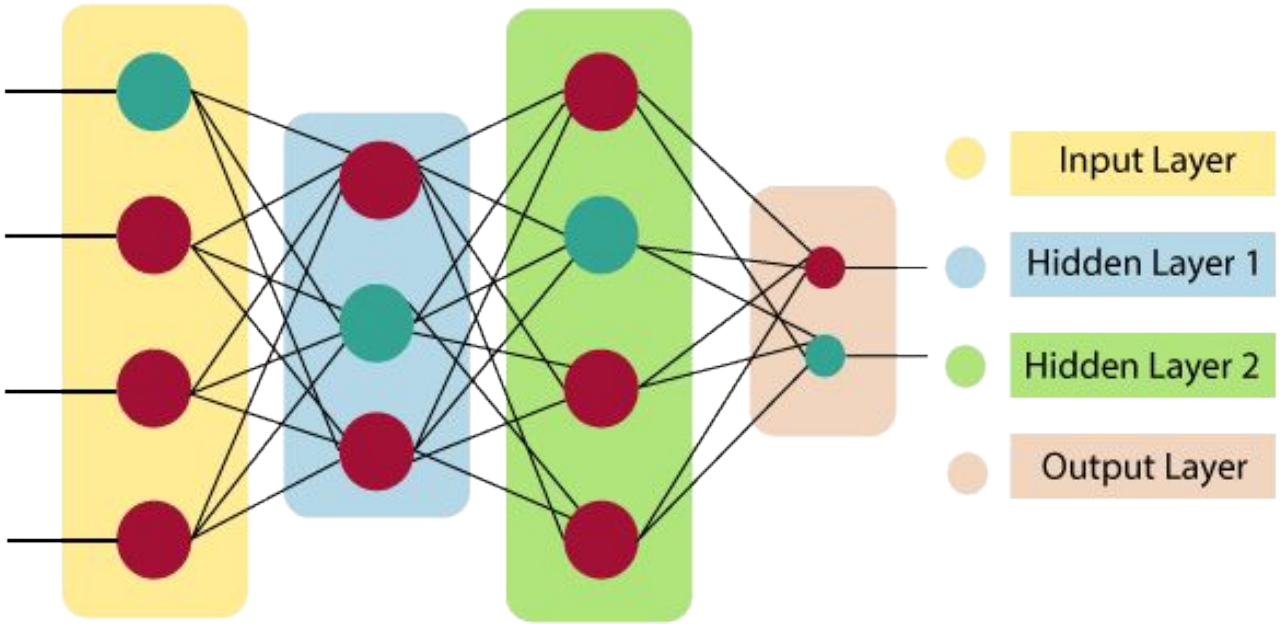
COMPARISON OF ANN AND BNN



Sr. No.	Biological neural network	Artificial neural network
1.	Cell nucleus	Neuron / Nodes
2.	Synapse	Weights or interconnections
3.	Dendrites	Inputs
4.	Axon	Output

ARCHITECTURE OF ANN



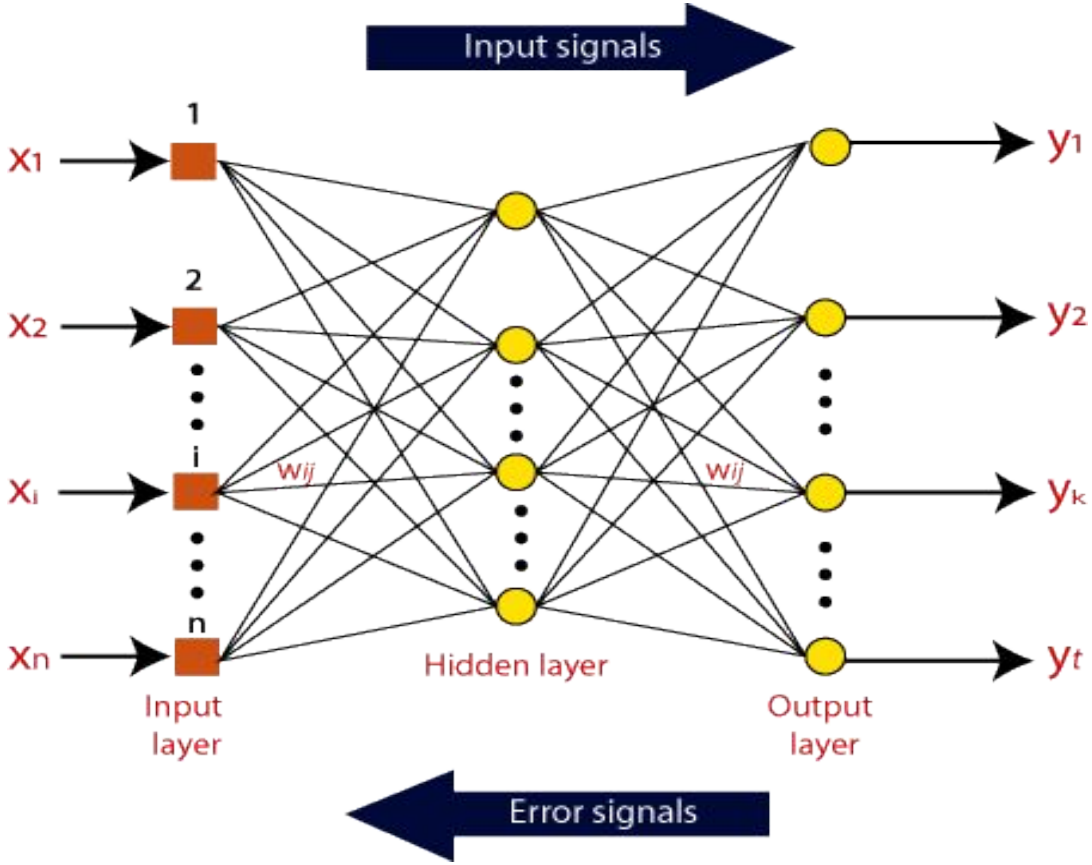


ARCHITECTURE OF ANN

1. **Input Layer**: Input layer accepts inputs in several formats provided by the programmer.
2. **Hidden Layer**: The hidden layer(s) is/are present between input and output layers. It performs all the calculations to find hidden features and patterns.
3. **Output Layer**: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

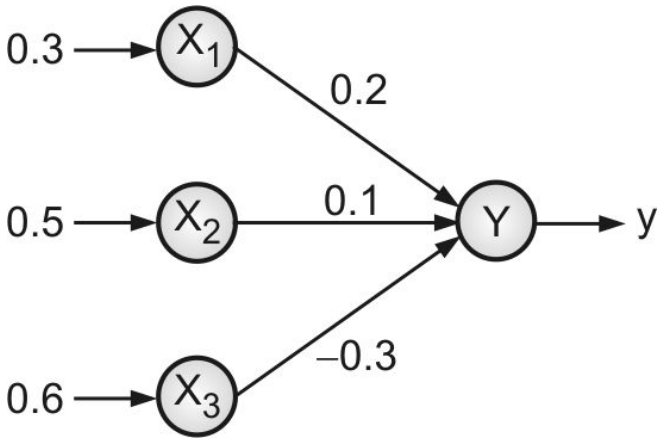
The artificial neural network takes input and computes the weighted sum of the inputs and includes a **bias**. This computation is represented in the form of a **transfer function**.

WORKING OF ANN



$$y = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n = x_i w_i$$

EXAMPLE 1

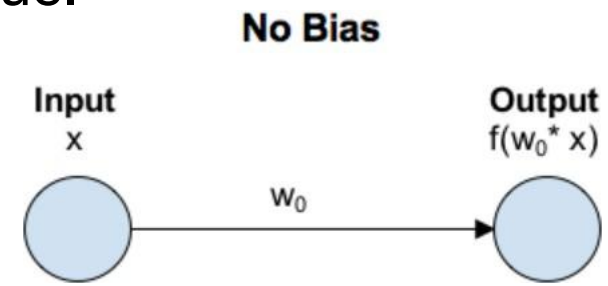


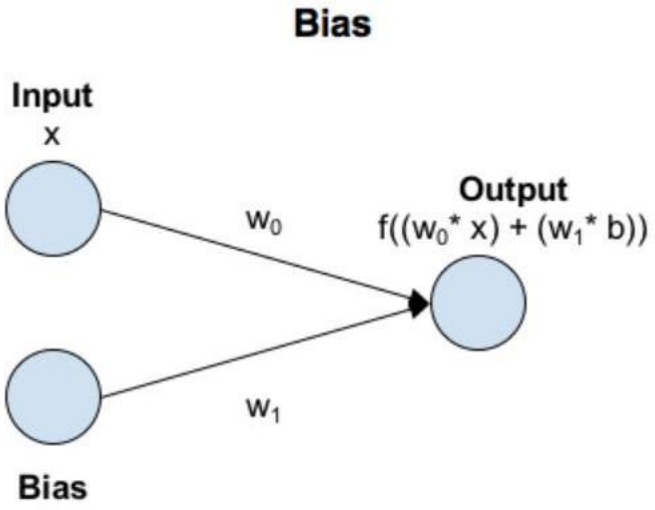
yin = x1 w1 + x2 w2 + x3 w3 = (0.3)(0.2) + (0.5)(0.1) + (0.6)(– 0.3)

yin = 0.06 + 0.05 – 0.18 = –
0.07

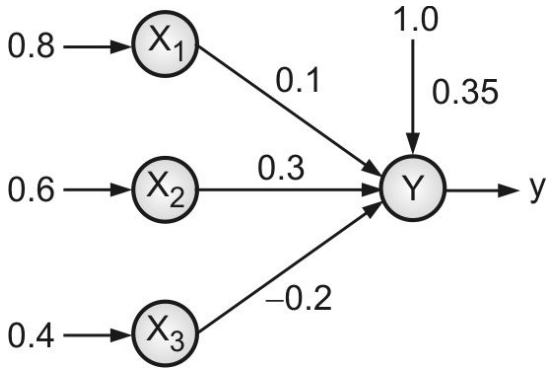
BIAS

- Bias in Neural Networks can be thought of as analogous to the role of a constant in a linear function, whereby the line is effectively transposed by the constant value.





EXAMPLE 2



- $y_{in} = b + x_i w_i$
  - $= b + x_1 w_1 + x_2 w_2 + x_3 w_3 = 0.35 + (0.8)(0.1) + (0.6)(0.3) + (0.4)(-0.2)$
  - $= 0.35 + 0.08 + 0.18 - 0.08$
  - $= 0.53$

ACTIVATION/THRESHOLD FUNCTIONS

- To obtain exact output, the activation function is applied over the net input of an ANN.
- Activation Function helps the neural network to useimportant information while suppressing irrelevant data points.
- The purpose of an activation function is to add non-linearity to the neural network.
- What if a neural network doesn't use activation function?

Linear Regression Model……Can't learn for complex tasks

ACTIVATION FUNCTIONS TYPES

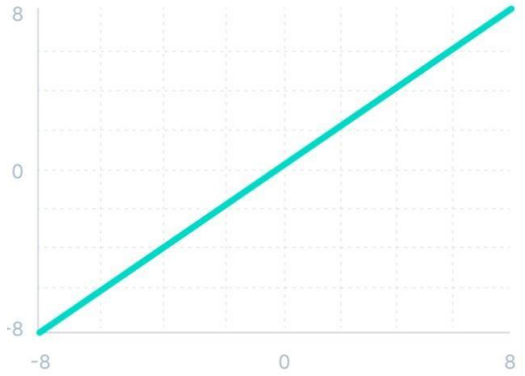
- Binary step function
- Linear Activation Function
- Sigmoid/Logistic Activation Function
- The derivative of the sigmoid Activation Function
- Tanh Function (Hyperbolic Tangent)
- Gradient of the Tanh Activation function
- ReLu Activation Function
- SoftMax

SOME LINEAR ACTIVATION FUNCTIONS EXPLAINED…

- **Linear function**: It is defined as

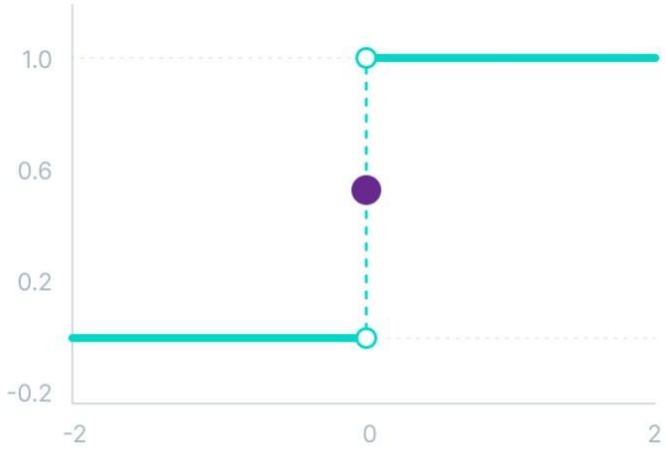
  f(x)=x for all x

  Here, input=output



- **Binary Step function**: The function is defined as :

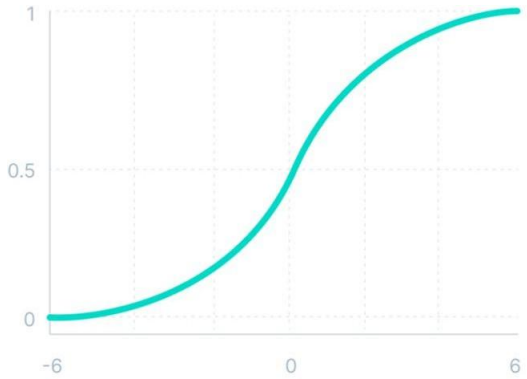
$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$



SOME LINEAR ACTIVATION FUNCTIONS EXPLAINED…..

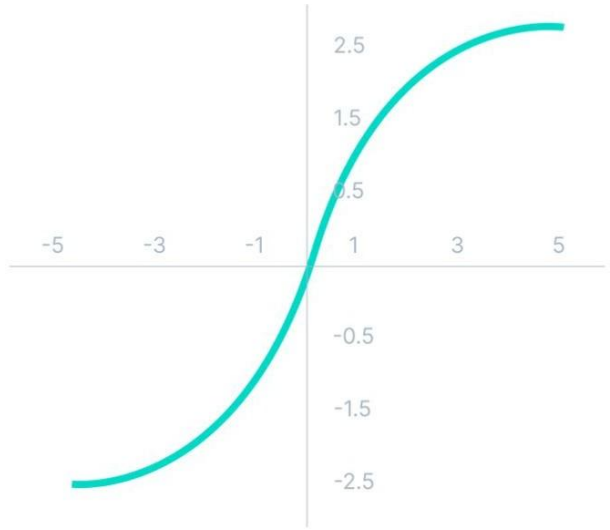
- **Sigmoid function**:

$$f(x) = \frac{1}{1 + e^{-x}}$$



- **Tanh function**:

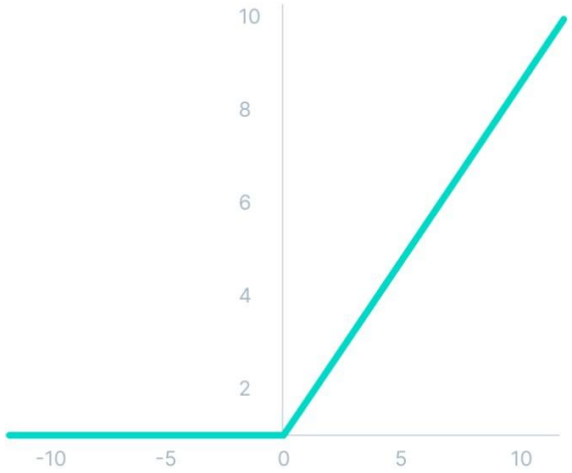
$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$



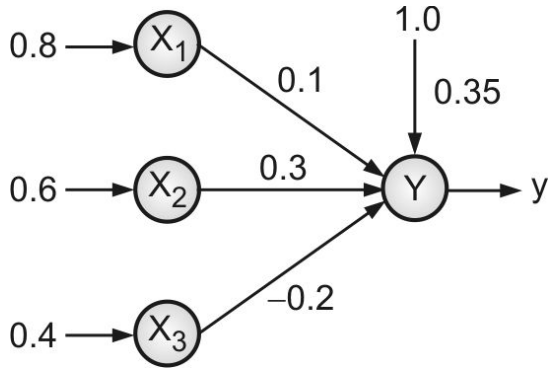
SOME LINEAR ACTIVATION FUNCTIONS EXPLAINED…..

ReLU:

$$f(x) = max\ (0, x)$$



EXAMPLE 3



- $y_{in} = b + x_i w_i$
  - $= b + x_1 w_1 + x_2 w_2 + x_3 w_3 = 0.35 + (0.8)(0.1) + (0.6)(0.3) + (0.4)(-0.2)$
  - $= 0.35 + 0.08 + 0.18 - 0.08$
  - $= 0.53$

Binary sigmoidal activation function :

$$y = f(y_{in}) = \left(\frac{1}{1 + e^{-y_{in}}}\right) = \frac{1}{1 + e^{-0.53}} = 0.625$$

Bipolar sigmoidal activation function :

$$y = f(y_{in}) = \left(\frac{2}{1 + e^{-y_{in}}}\right) - 1 = \left(\frac{2}{1 + e^{-0.53}}\right) - 1 = 0.259$$

VANISHING GRADIENTS

- Like the sigmoid function, certain activation functions squish an ample input space into a small output space between 0 and 1.
- Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small.
- However, when more layers are used, it can cause the gradient to be too small

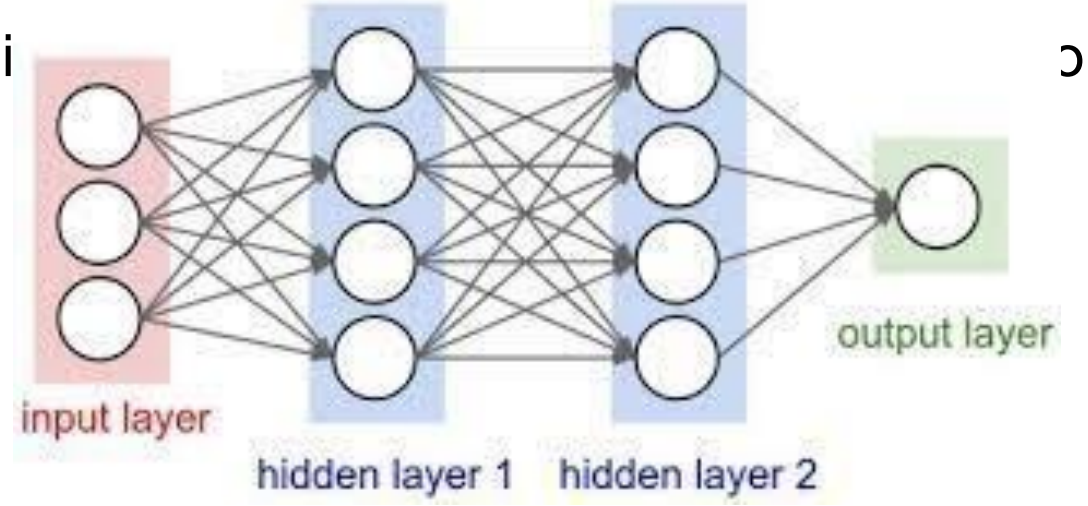
for training to work effectively.

EXPLODING GRADIENTS

- Exploding gradients are problems where significant error gradients accumulate and result in very large updates to neural network model weights during training.
- An unstable network can result when there are exploding gradients, and the learning cannot be completed.
- The values of the weights can also become so large as to overflow and result in something called NaN values.

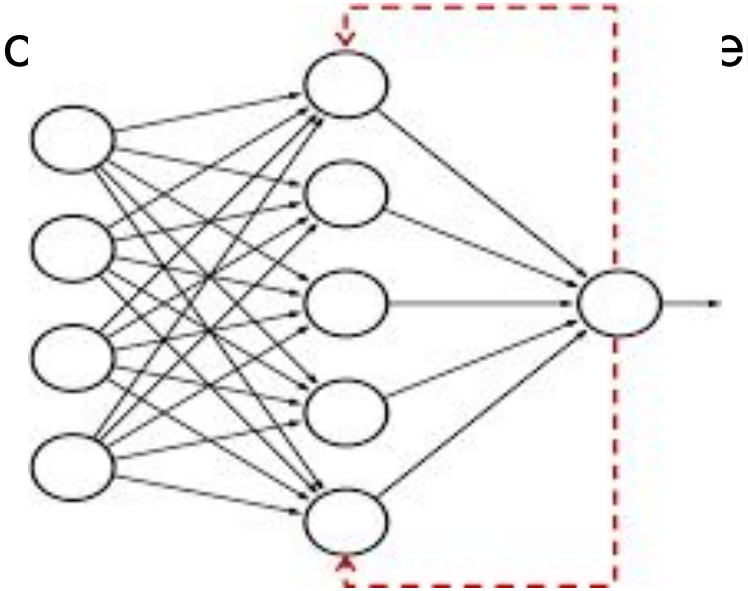
FEED FORWARD ANN

- A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron.
- Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided.
- The primary advantage of this network is that it figures out how to evaluate and recogni ɔ not form a loop.

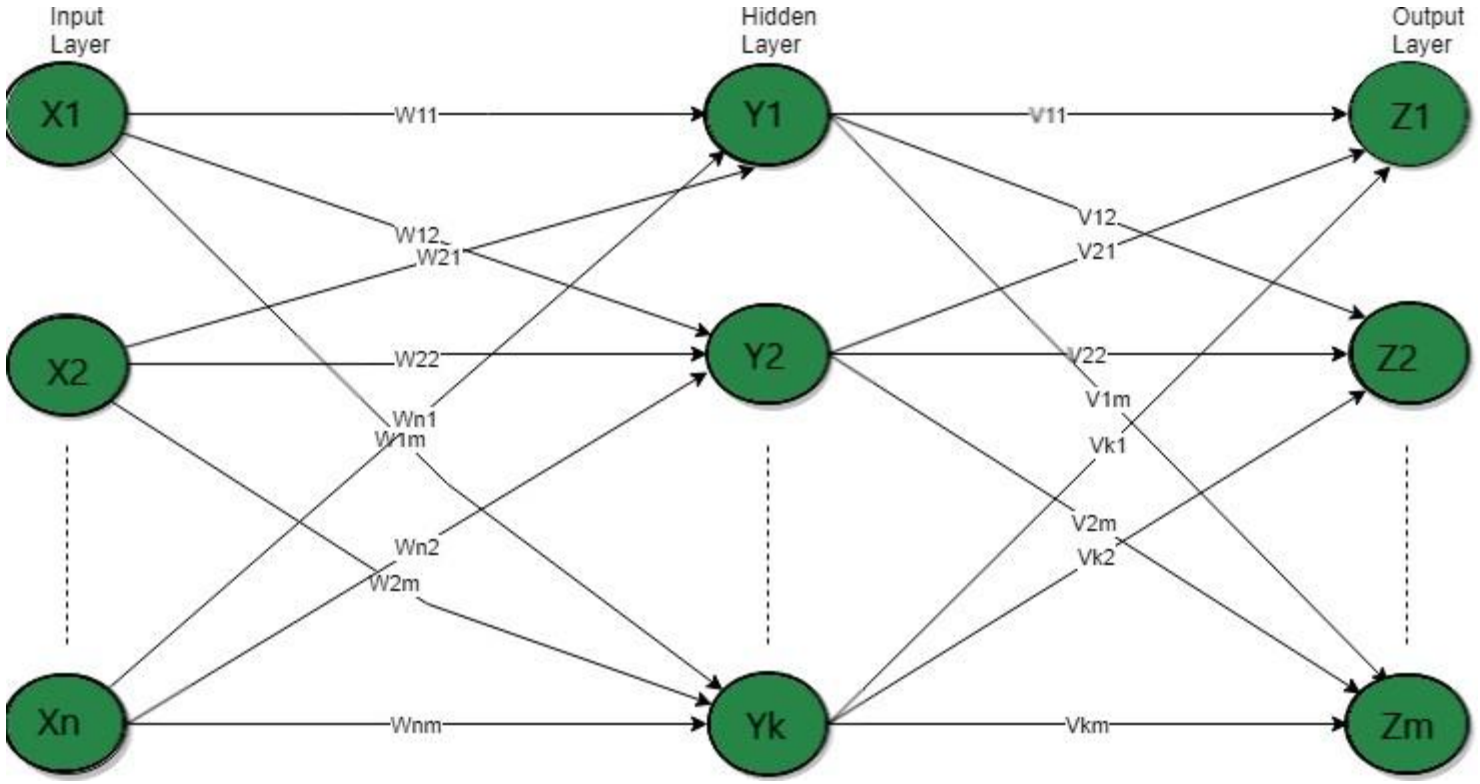


FEEDBACK ANN

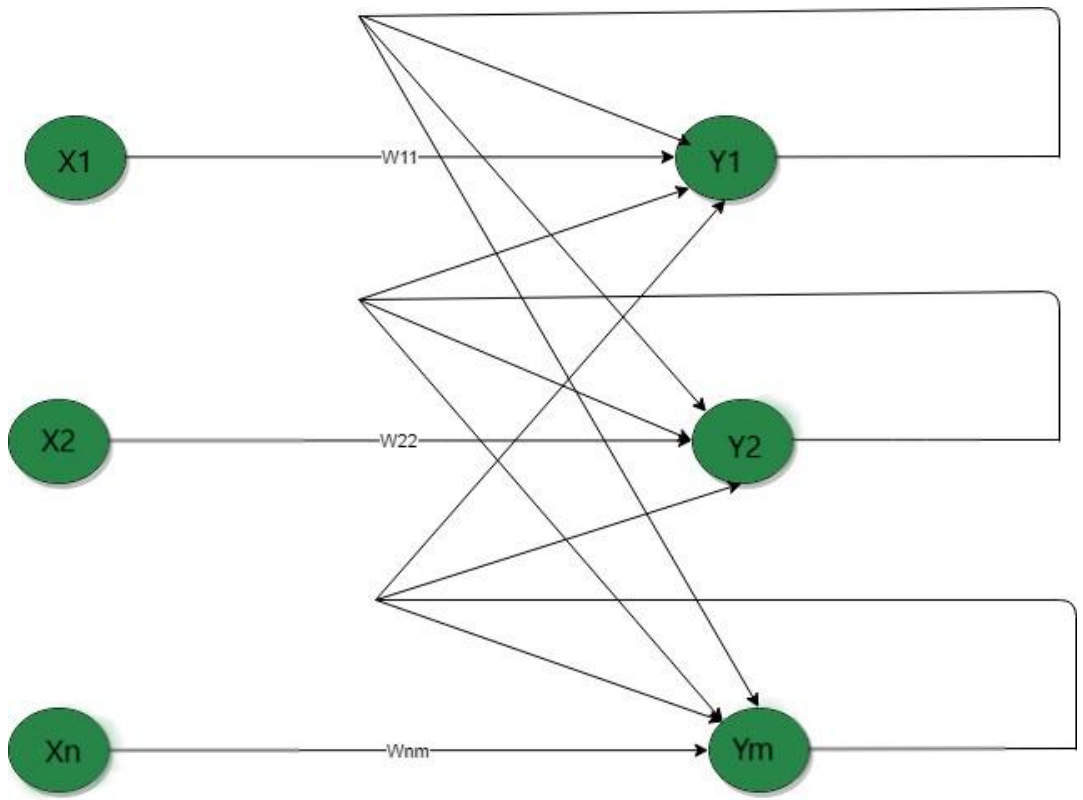
- In thistype of ANN, the output returns into the network to accomplish the best-evolved results internally.
- The feedback networks feed information back into itself and are well suited to solve optimizatic em error corrections utilize feedback ANNs.



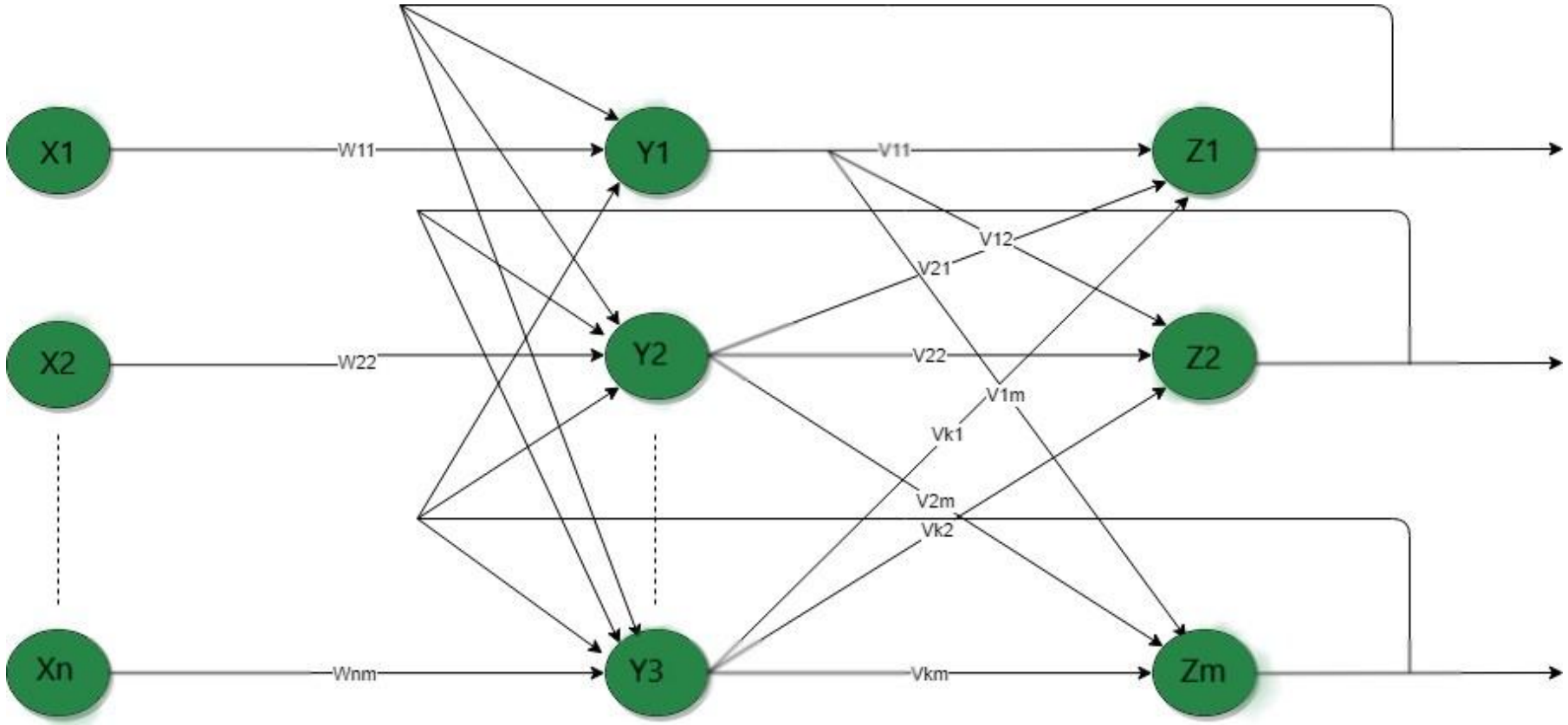
MULTILAYER FEED-FORWARD NETWORK



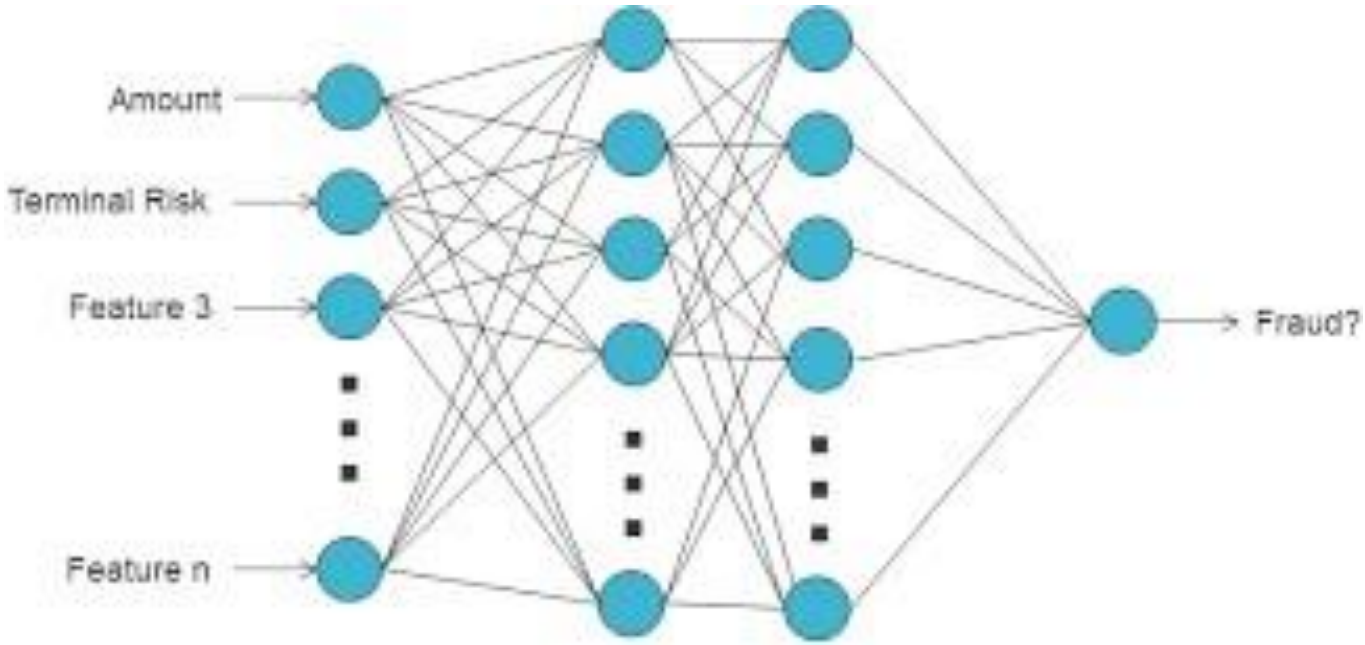
SINGLE LAYER RECURRENT NETWORK



MULTI-LAYE RECURRENT NETWORK



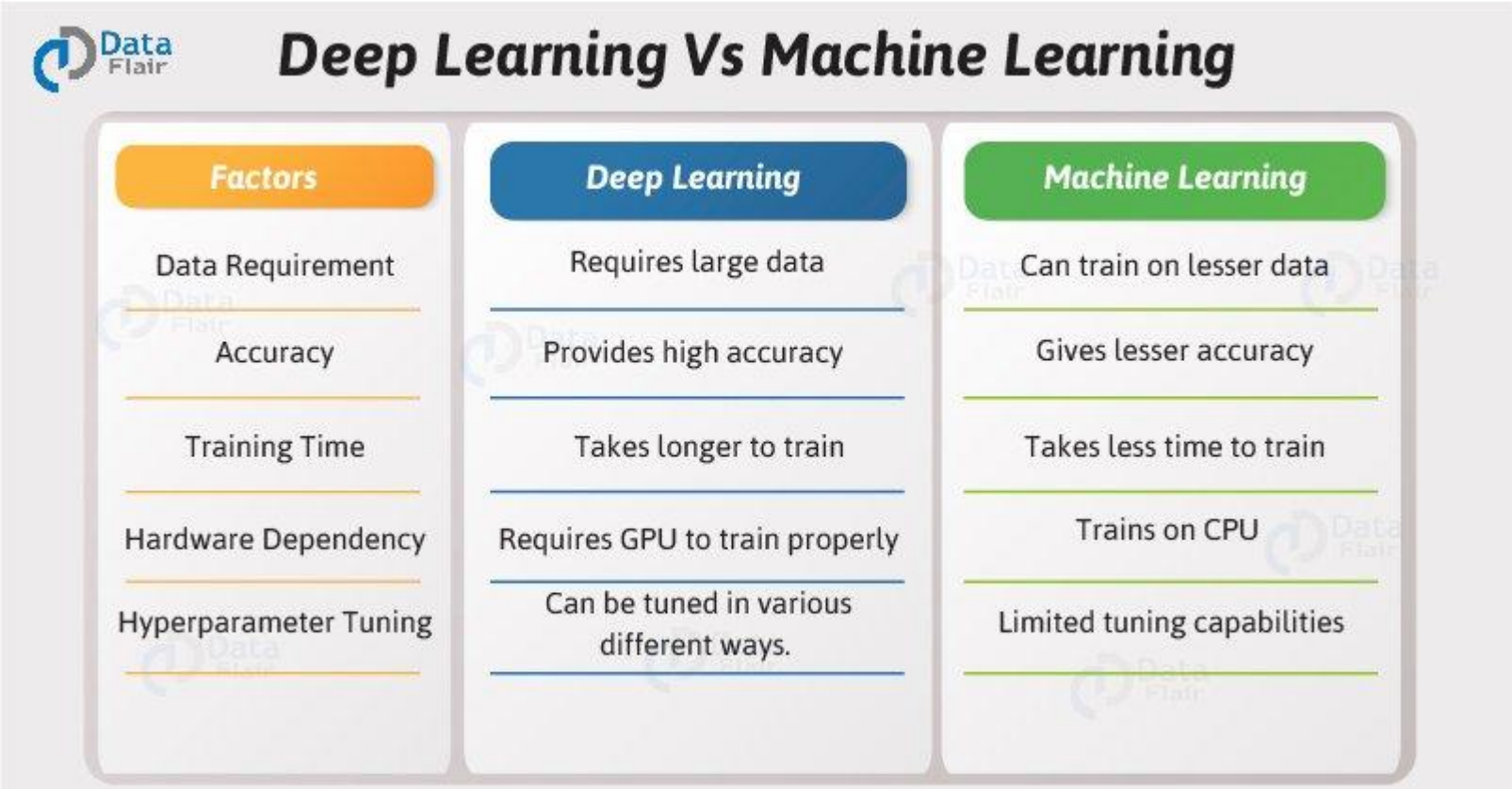
CREDIT CARD FRAUD DETECTION USING ANN



APPLICATIONS OF NEURAL NETWORKS

- Image Processing – Face Recognition, Signature verification, Handwriting Analysis
- Stock Market Prediction
- Social Media
- Aerospace
- Pattern Recognition
- Defence
- Healthcare
- Optimization/Constraint Satisfactions
- Forecasting and Risk Assessment
- Control system

Machine Learning Vs Deep Learning



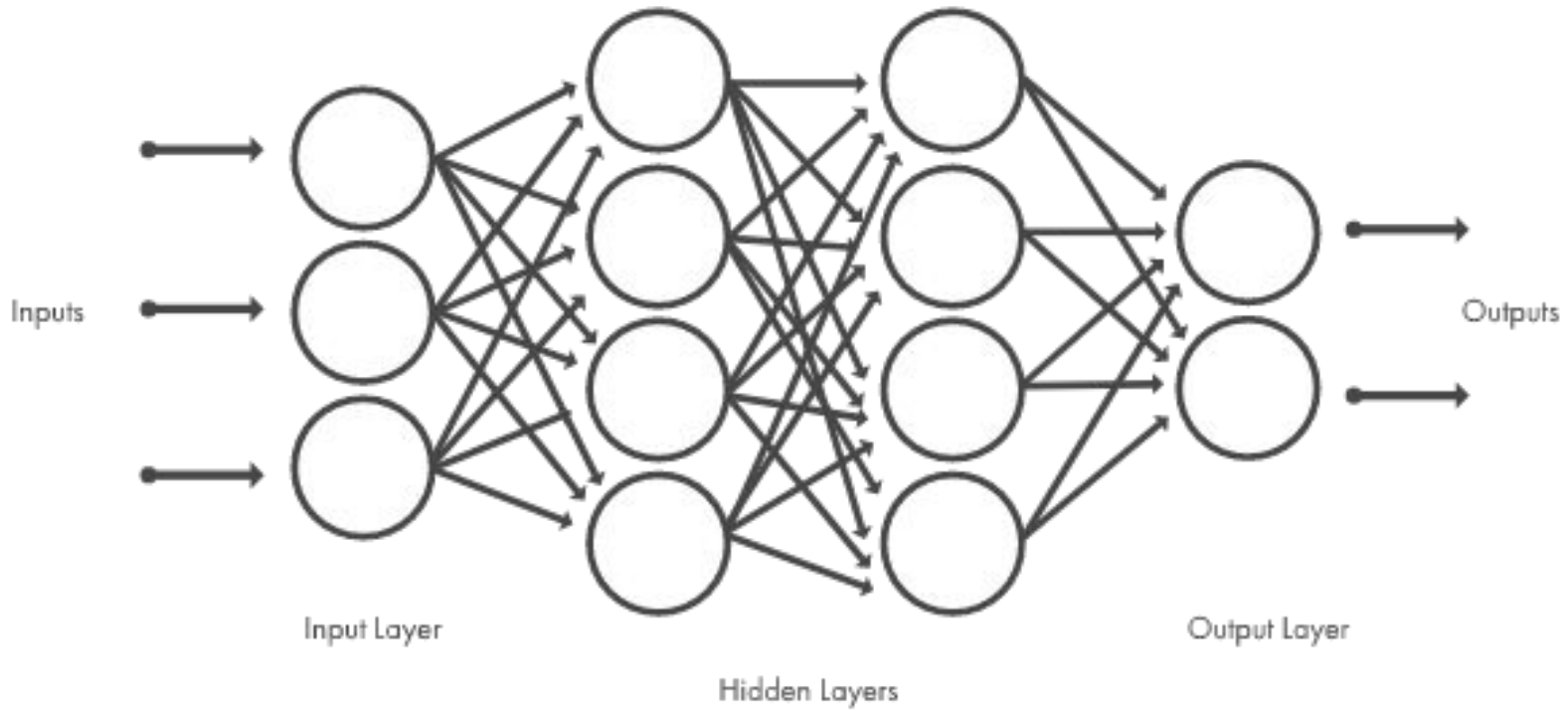
Machine Learning

- Small data set with less feature
  - Feature Extraction
  - Feature Selection
- Manual feature engineering is tough & time consuming

Artificial Neural Network

- Auto Feature Engineering
- Activation function , bias, weight can be added
- Capable of learning any non linear function hence referred as universal function approximation

Working of Deep Learning



Working of Deep Learning

- Most deep learning methods use **neural network** architectures, which is why deep learning models are often referred to as **deep neural networks**.
- The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.
- Neurons are grouped into three different types of layers:
  - **a) Input layer**
  - **b) Hidden layer**
  - **c) Output layer**
- **Application**
  - **Image recognition**
  - **Speech Recognition**
  - **Huge Input Parameter**

Deep Learning Layers

- 1. Input Layer: It receives the input data from the observation

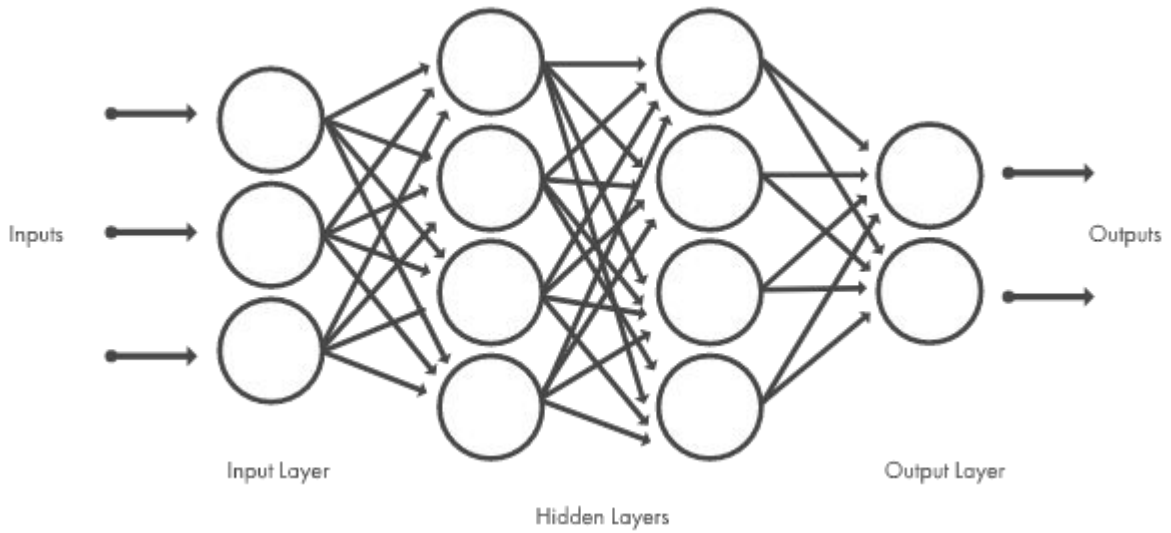
- 2. Hidden Layer: It performs mathematical computations on input data. To decide the number of hidden layers and the number of neurons in each layer is challenging. It forms the hierarchy concepts from the learning. In the hierarchy, each level grasps to transform the input data into a more and more abstract and composite representation.
- Output Layer:It gives the desired result.

Applications that limits the ANN but RNN can do

- Time series analysis or use case
- Share market
- Profit forecast for quarters
- Language Translation
- Image caption
- NLP use case
- Cant handle sequential data
- Does not have memory / considers current input

- https://data-flair.training/blogs/deep-learning-vs-machine-learning/
- https://www.mathworks.com/discovery/deep-learning.html

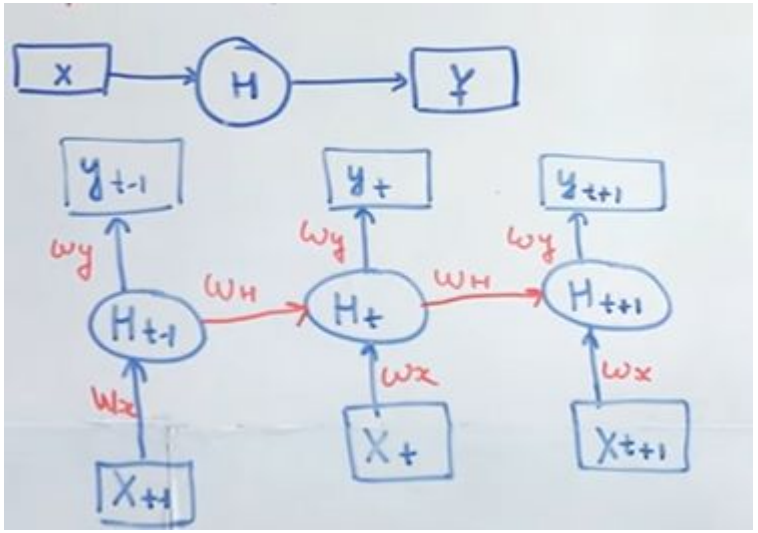
Recurrent Neural Network: Introduction, Simple RNN



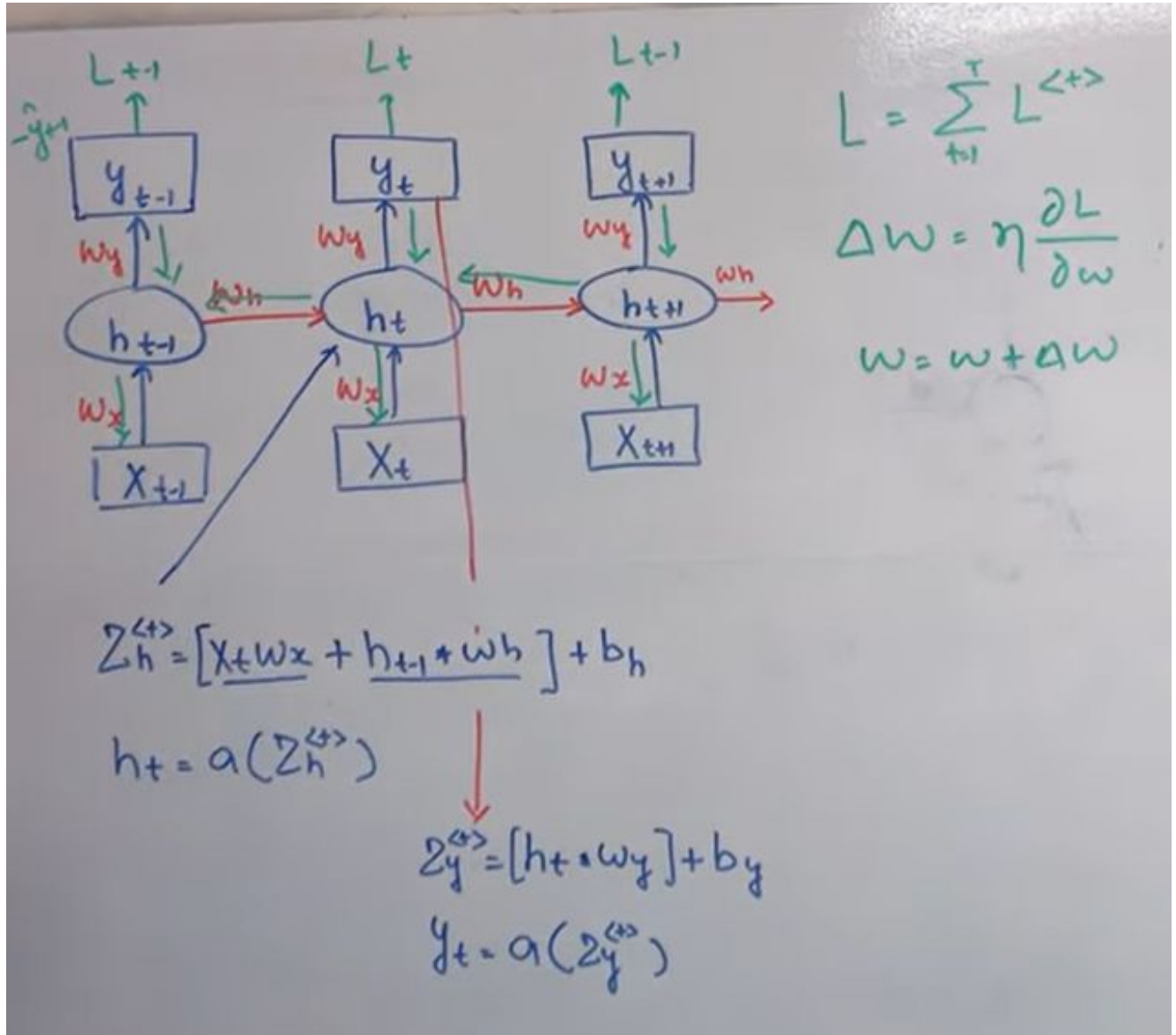
RNN

- RNNs are a powerful and robust type of neural network, and belong to the most promising algorithms in use because it is the only one with an internal memory.
- NNs and feed-forward neural networks get their names from the way they channel information.
- Backpropagation is used for calculating the gradient of an error function with respect to a neural network's weights. The algorithm works its way backwards through the various layers of gradients to find the partial derivative of the errors with respect to the weights. Backprop then uses these weights to decrease error margins when training.
- Applications
  - Image Recognition
  - Speech Recognition

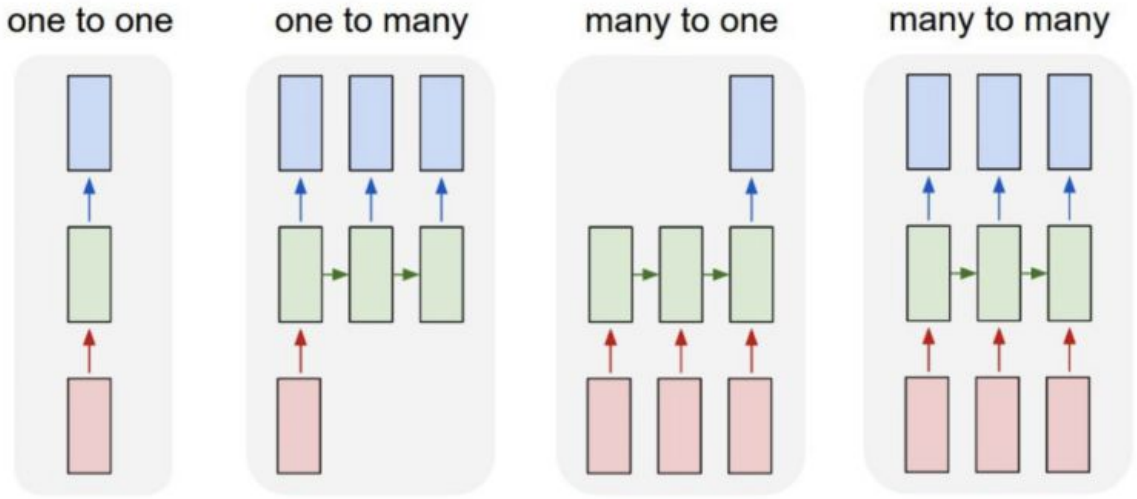
- **TYPES OF RNNS**
  - One to One
  - One to Many
  - Many to One
  - Many to Many

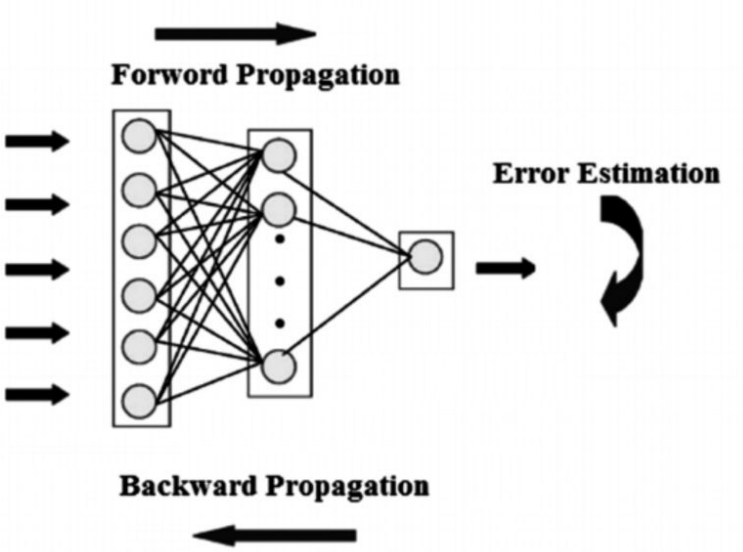


RNN

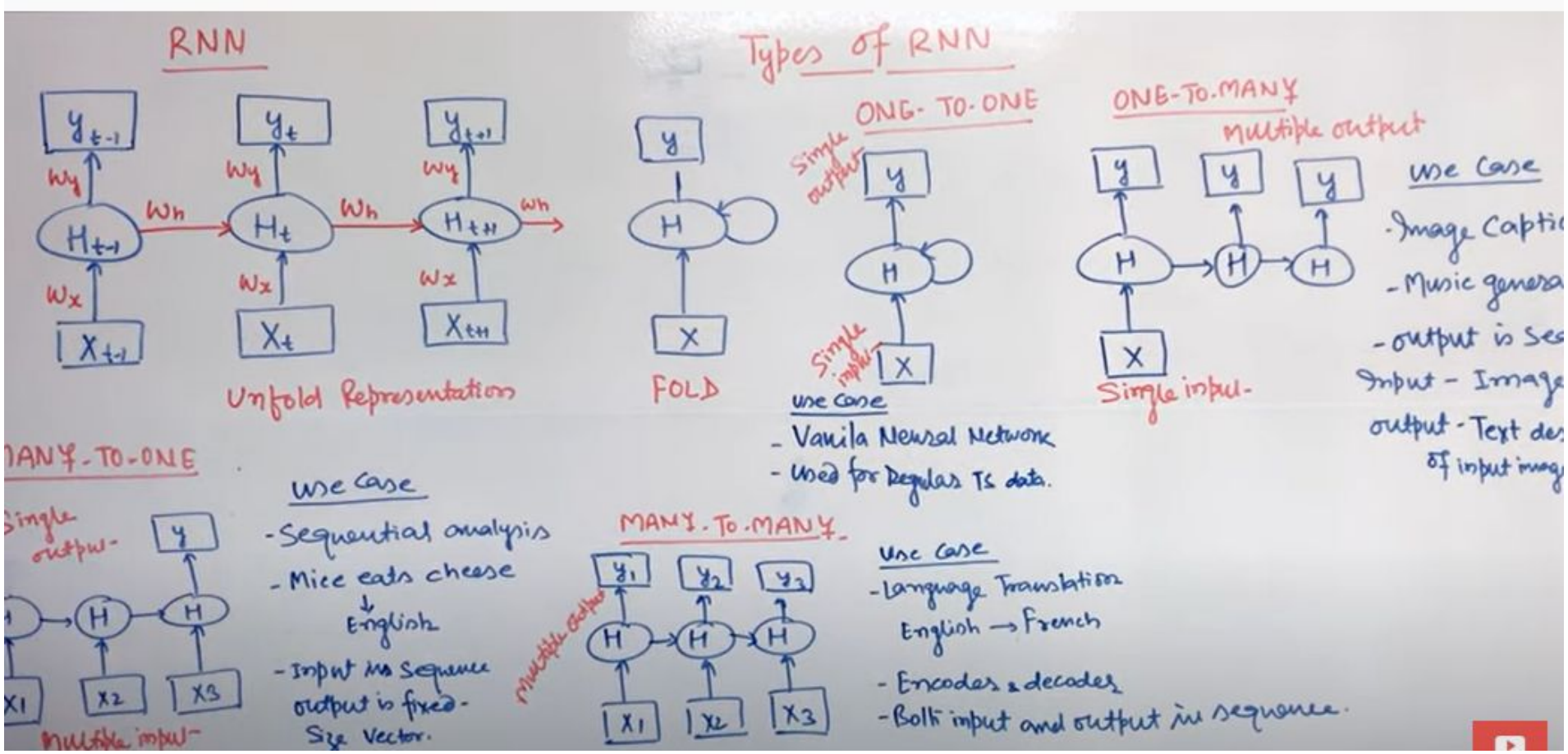

L_{t-1}
L_t
L_{t+1}
$\hat{y}_{t-1}$
y_{t-1}
y_t
y_{t+1}
w_y
w_h
h_{t-1}
h_t
h_{t+1}
w_x
X_{t-1}
X_t
X_{t+1}
$L = \sum_{t=1}^{T} L^{<t>}$
$\Delta w = \eta \frac{\partial L}{\partial w}$
$w = w + \Delta w$
$Z_h^{<t>} = [X_t w_x + h_{t-1} * w_h] + b_h$
$h_t = a(Z_h^{<t>})$
$Z_y^{<t>} = [h_t * w_y] + b_y$
$y_t = a(Z_y^{<t>})$

Types of Recurrent NN & Backpropagation NN





Types of RNN with application



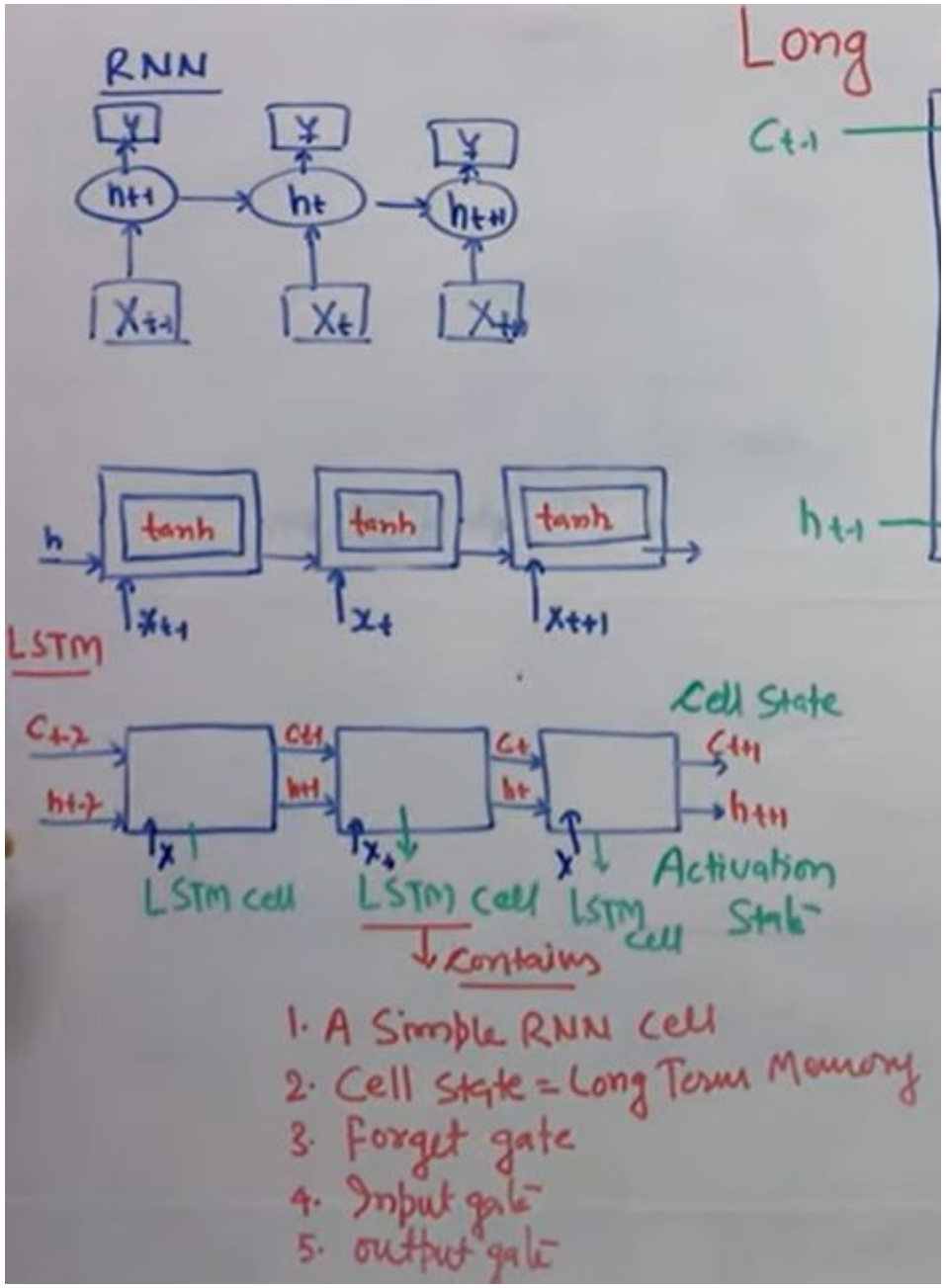
LSTM Implementation

- Long short-term memory (LSTM) networks are an extension of RNN that extend the memory. LSTM are used as the building blocks for the layers of a RNN. LSTMs assign data “weights” which helps RNNs to either let new information in, forget information or give it importance enough to impact the output.
- The units of an LSTM are used as building units for the layers of a RNN, often called an LSTM network.
- This memory can be seen as a gated cell, with gated meaning the cell decides whether or not to store or delete information

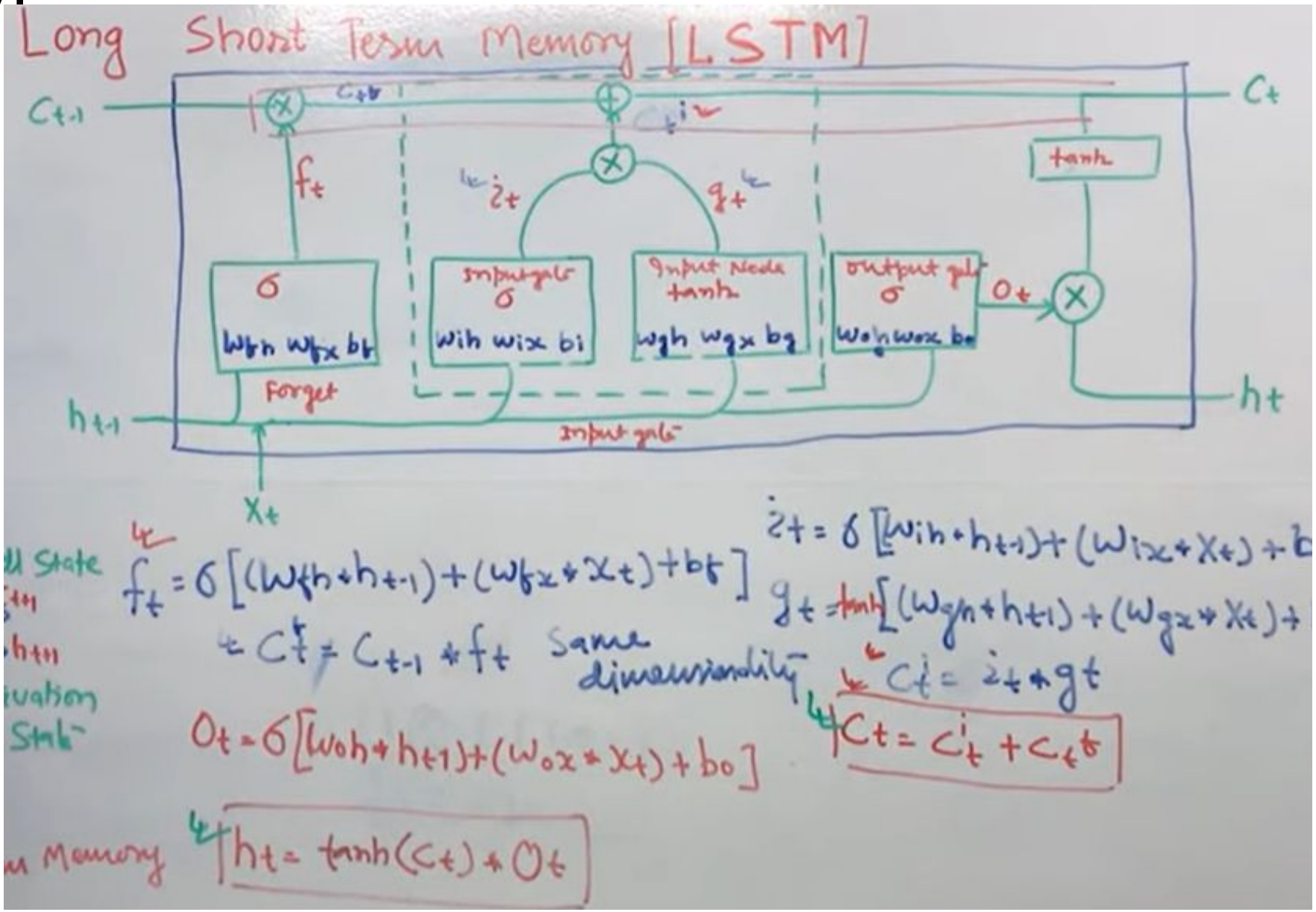
LSTM

- The units of an LSTM are used as building units for the layers of a RNN, often called an LSTM network.
- LSTMs enable RNNs to remember inputs over a long period of time. This is because LSTMs contain information in a memory, much like the memory of a computer. The LSTM can read, write and delete information from its memory.
- This memory can be seen as a gated cell, with gated meaning the cell decides whether or not to store or delete information.
- This simply means that it learns over time what information is important and what is not.
- In a long short-term memory cell you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn’t important (forget gate), or let it impact the output at the current timestep (output gate)
- The problematic issue of vanishing gradients is solved through LSTM because it keeps the gradients steep enough, which keeps the training relatively short and the accuracy high.

LSTM


RNN
Long
tanh
tanh
tanh
LSTM
Cell State
LSTM cell
LSTM cell
LSTM cell
Activation
Contains
1. A Simple RNN cell
2. Cell state = Long Term Memory
3. Forget gate
4. Input gate
5. output gate

LSTM


Long Short Term Memory [LSTM]
C_{t-1}
C_t
f_t
i_t
g_t
o_t
h_{t-1}
h_t
x_t
tanh
σ
Wfh Wfx bf
Forget
Wih Wix bi
input gate
tanh
Wgh Wgx bg
input Node
σ
Woh Wox bo
output gate
input gate
$f_t = \sigma[(W_{fh} * h_{t-1}) + (W_{fx} * X_t) + b_f]$
$i_t = \sigma[(W_{ih} * h_{t-1}) + (W_{ix} * X_t) + b$
$g_t = \tanh[(W_{gh} * h_{t-1}) + (W_{gx} * X_t) +$
$C_t^i = C_{t-1} * f_t$
Same dimensionality
$C_t^i = i_t * g_t$
$O_t = \sigma[(W_{oh} * h_{t-1}) + (W_{ox} * X_t) + b_o]$
$C_t = C_t^i + C_t^b$
$h_t = \tanh(C_t) * O_t$

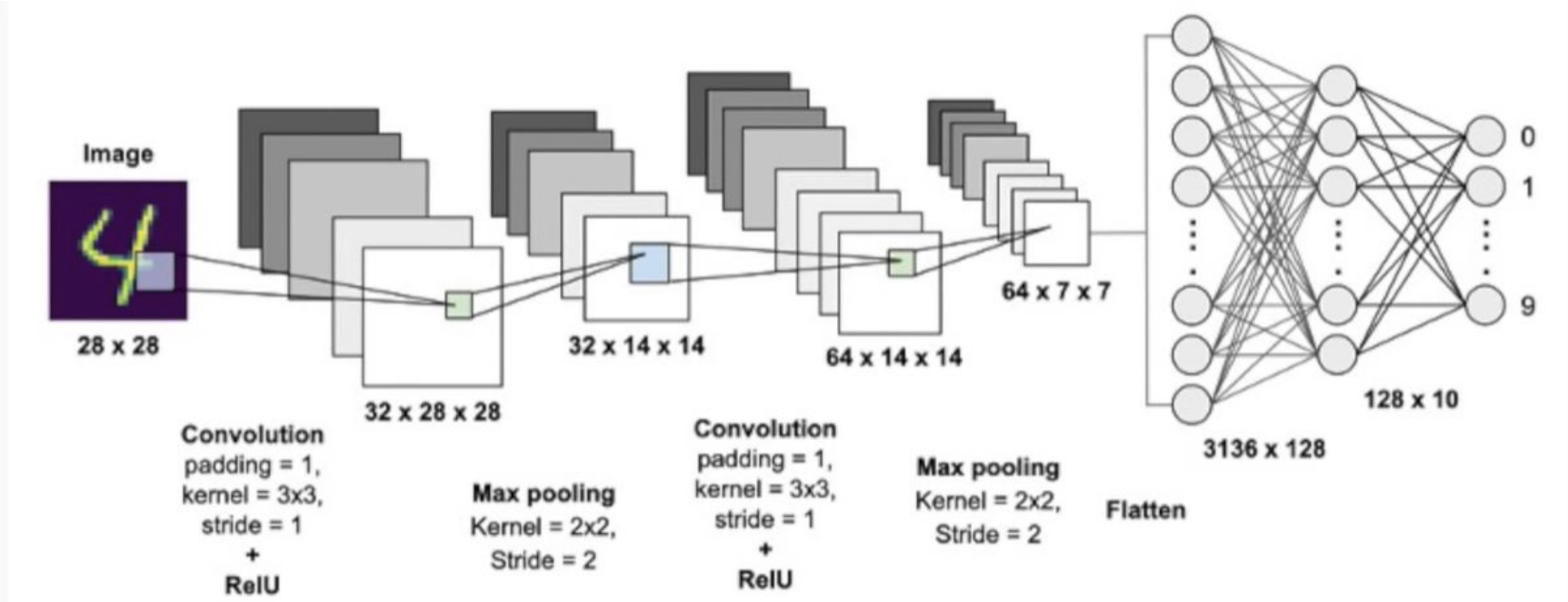
Reference

- https://www.youtube.com/watch?v=Opj2AT0iYCw&t=416s

Deep RNN

- Fortunately, by learning from the multitude of already existing compositions, a neural network does have the ability to generate a new kind of music. Generating music using computers is an exciting application of what a neural network can do.
- In DNN inputs are independent of each other
- All inputs are given at same time to the network
- Use Different parameters at any layers of the network
- Application but not useful for sequential information
- Is to use sequential information for predicting next word in sentence
- Handwriting recognition
- speech recognition
- machine translation

Convolutional Neural Network: Introduction



Advantage of CNN

Image size = 1920 x 1080 X 3

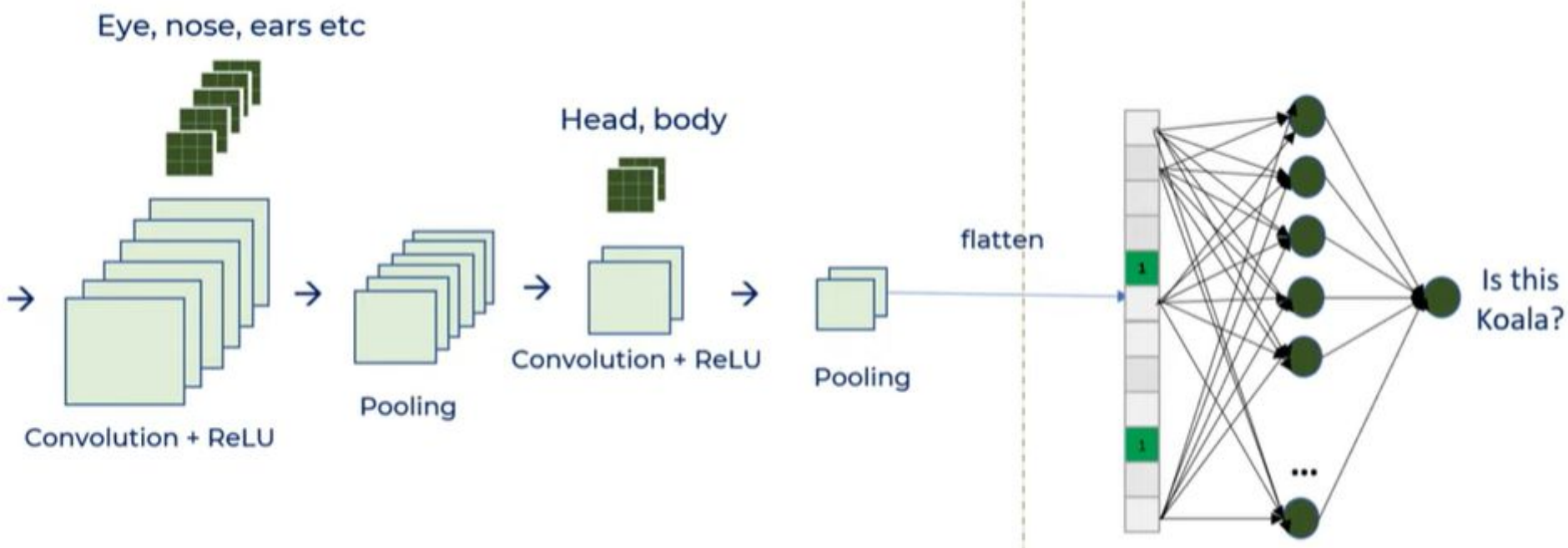
First layer neurons = 1920 x 1080 X 3 ~ 6 million

Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

Feature Extraction

Classification



Rotation

Thickness

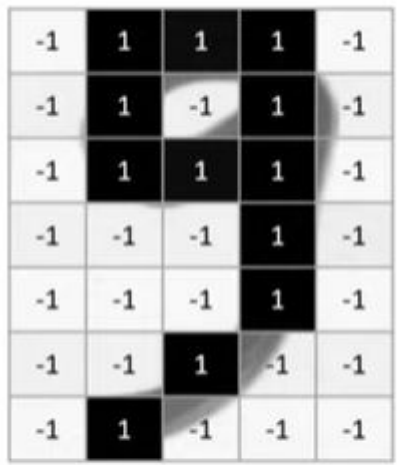
CNN by itself doesn't take care of rotation and scale

- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples

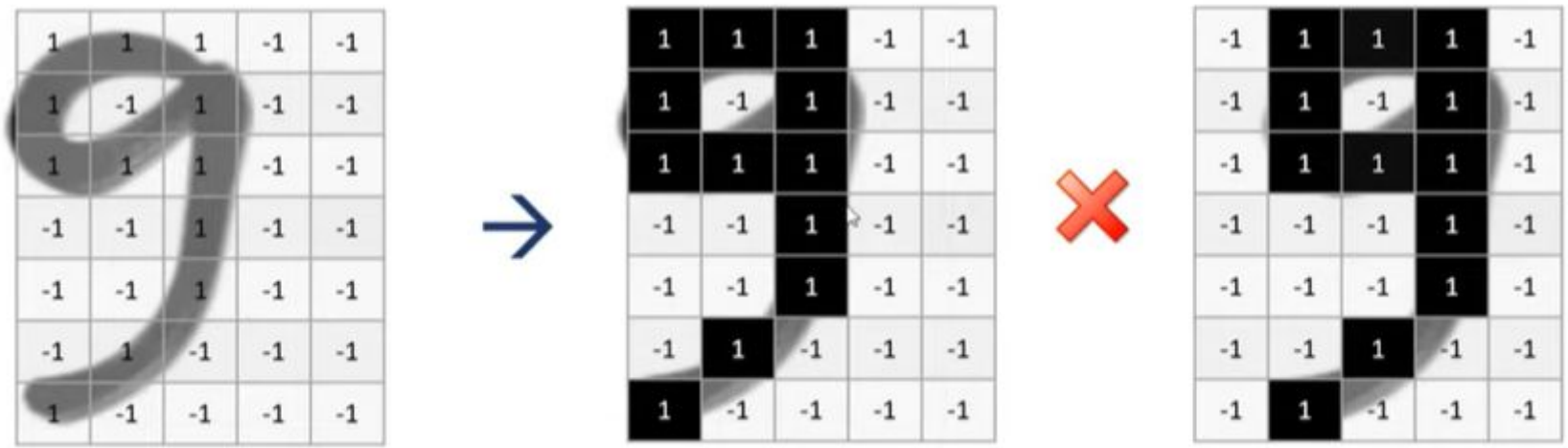
Input image to recognize handwriting from images

hardcoded

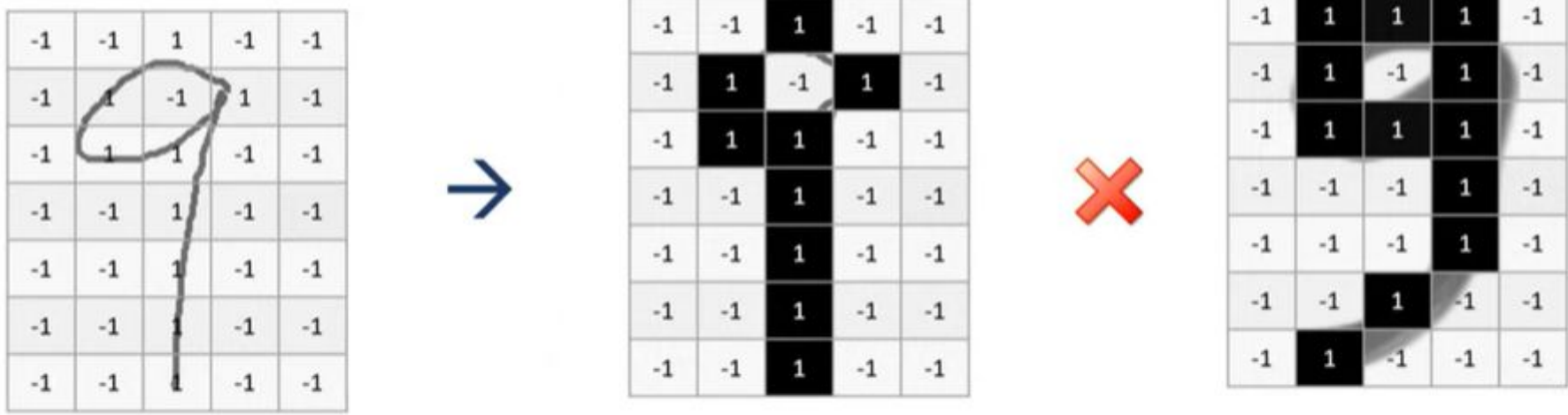
-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1



Location shifted

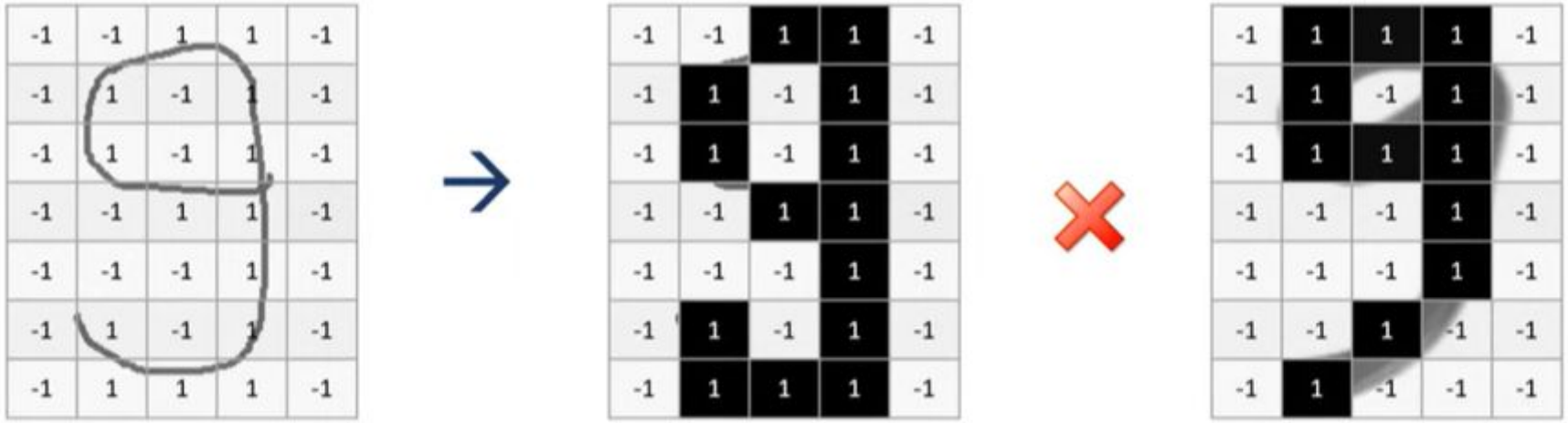


Variation 1


-1 -1 1 -1 -1
-1 1 -1 1 -1
-1 1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1

Representation of numbers

Variation 2

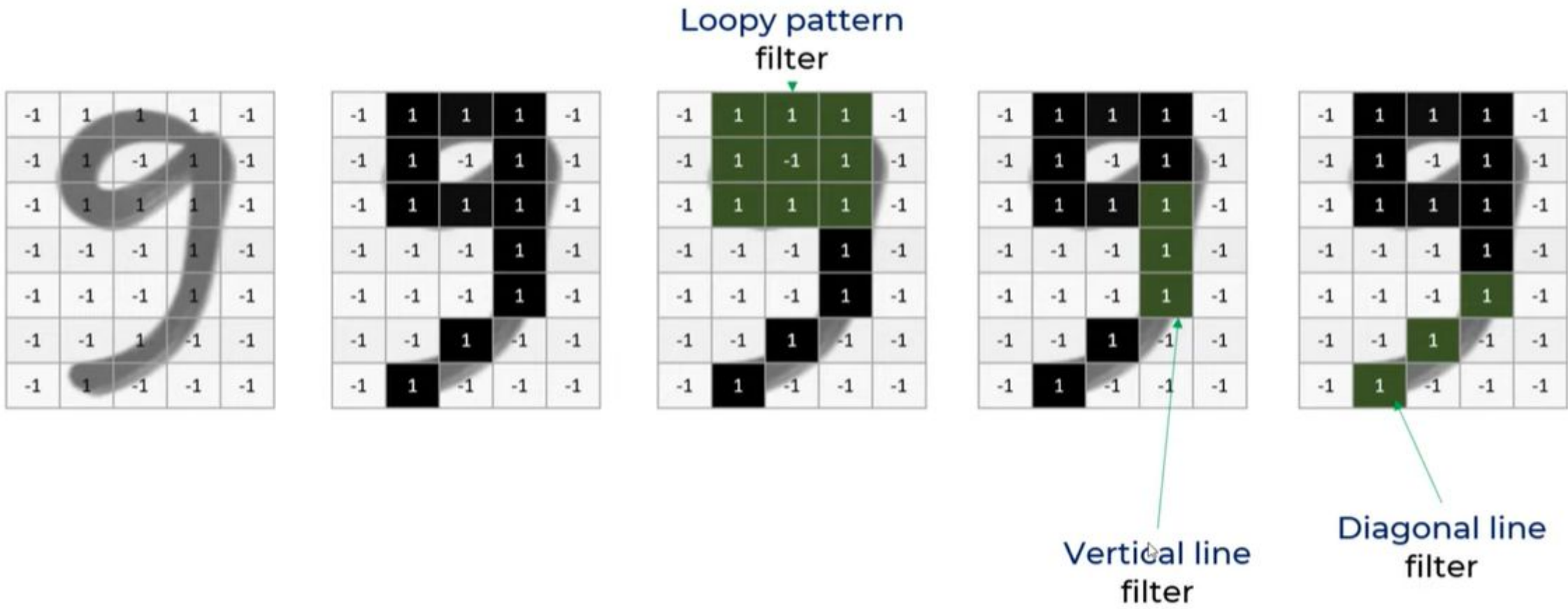


To handle variety in digits we can use simple artificial neural network (ANN)

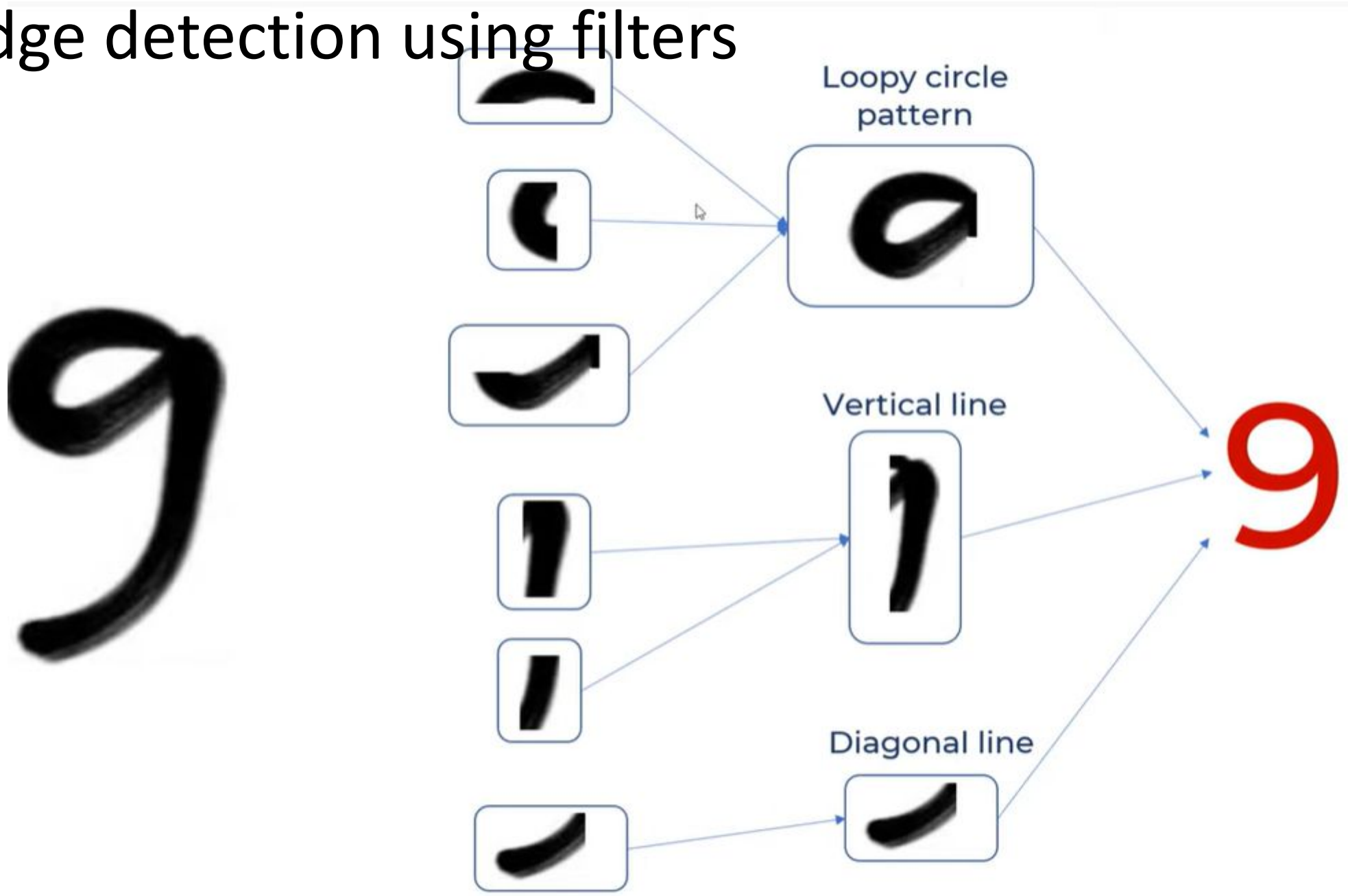
Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image


HOW CAN WE MAKE COMPUTERS
RECOGNIZE THESE TINY FEATURES?
imgflip.com


Loopy pattern filter
Vertical line filter
Diagonal line filter

Edge detection using filters



-1+1+1-1-1-1-1+1+1 = -1 → -1/9 = -0.11

-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-0.11		

-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

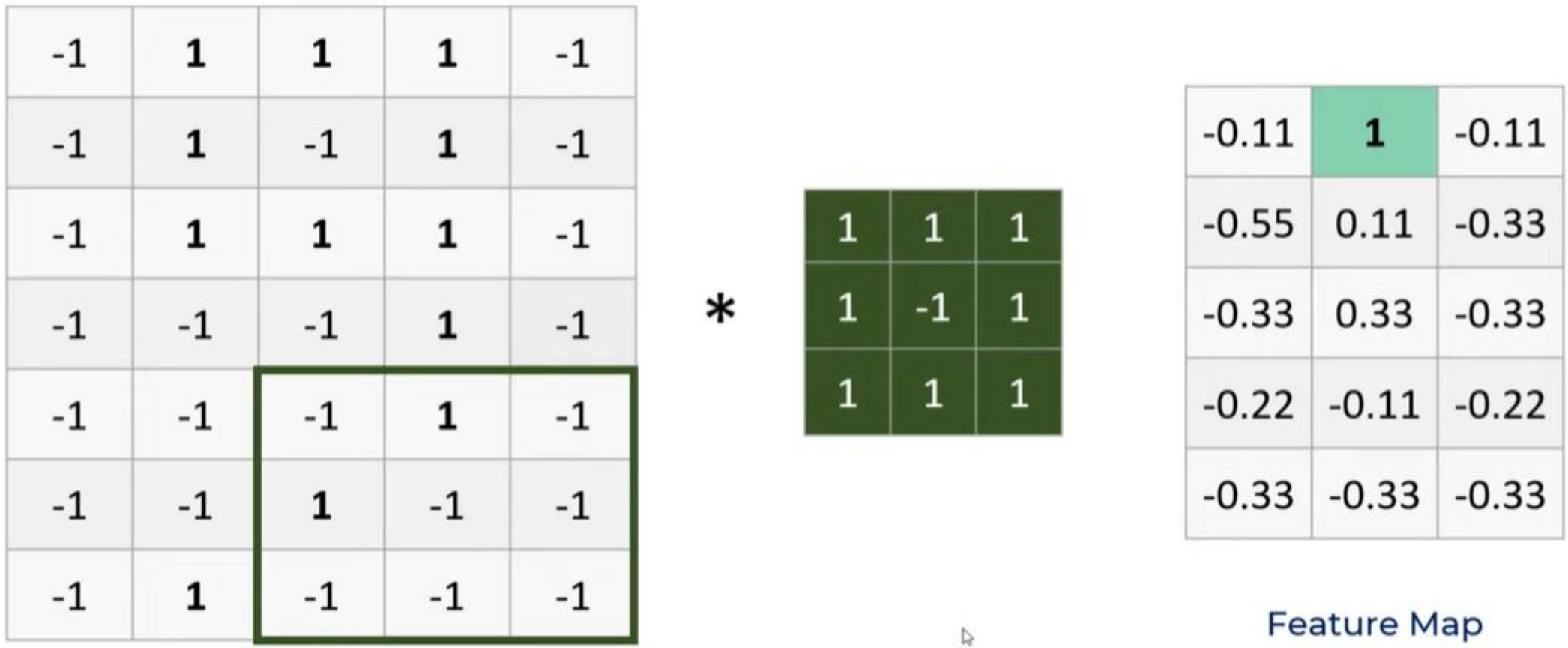
-0.11	**1**	

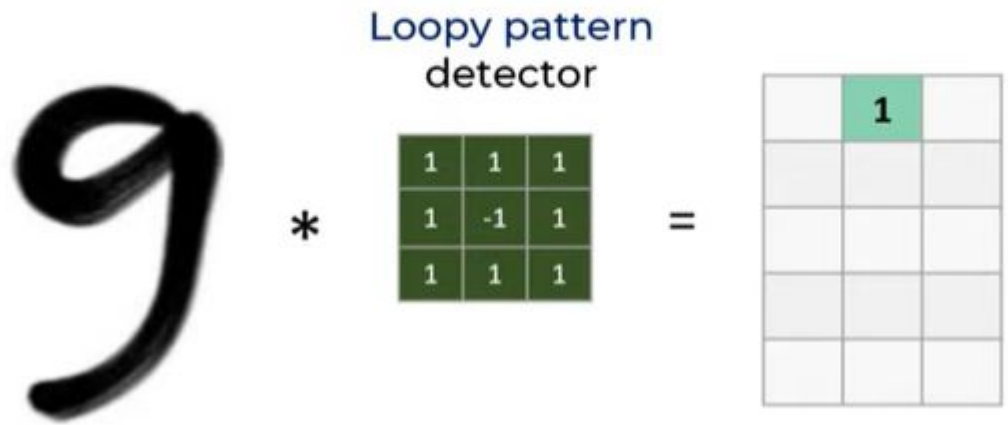
-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

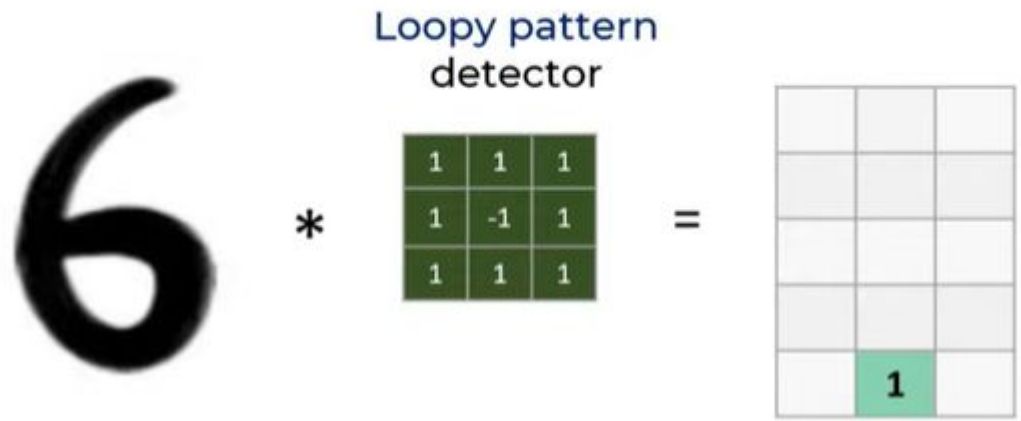
*

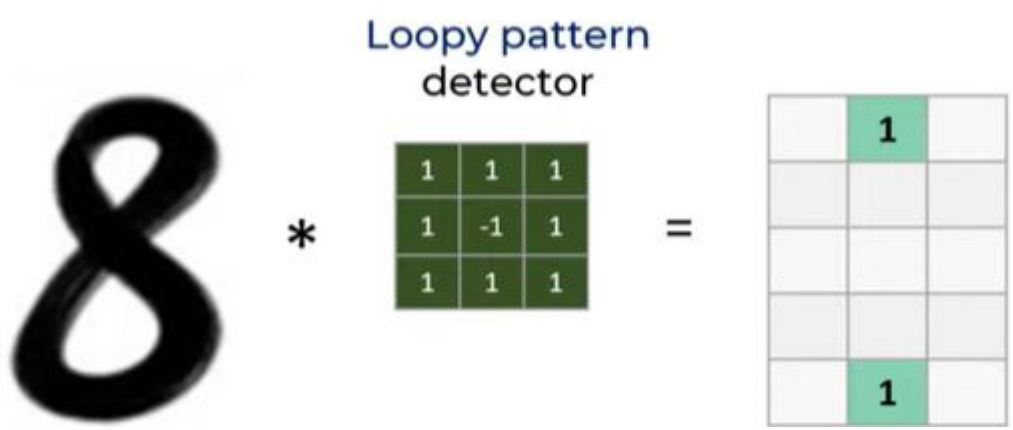
1	1	1
1	-1	1
1	1	1

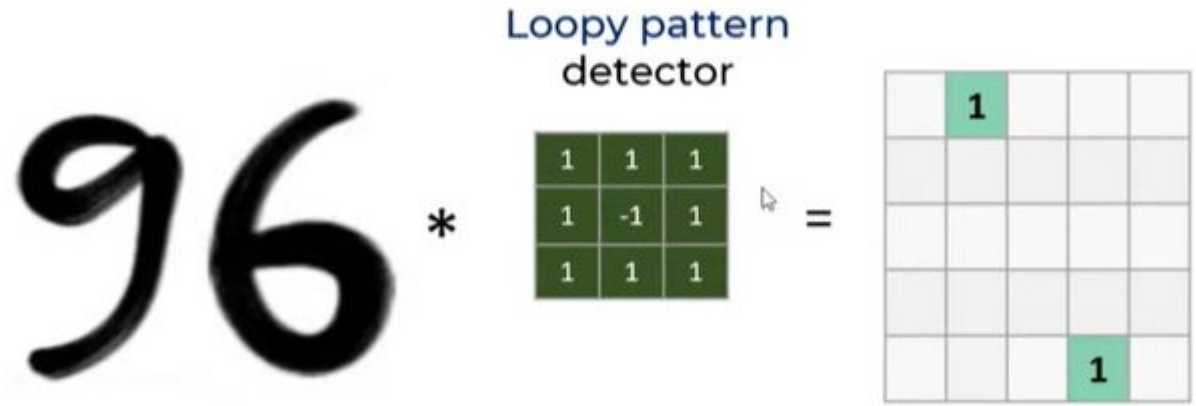
-0.11	1	-0.11


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
Feature Map


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1

Location invariant: It can detect eyes in any location of the image

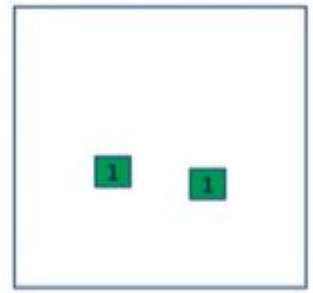
*

eye
detector

=

eye
detector

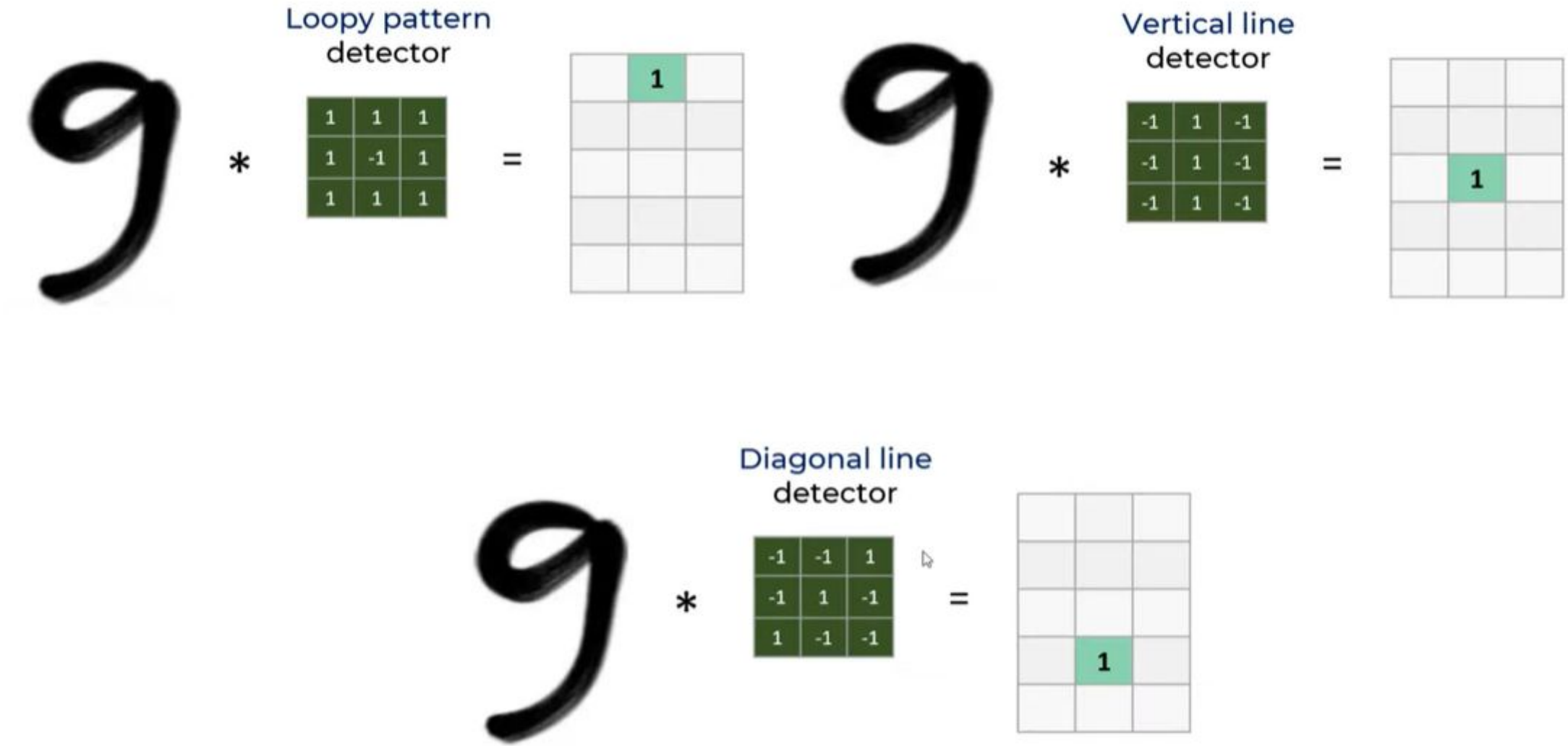
=



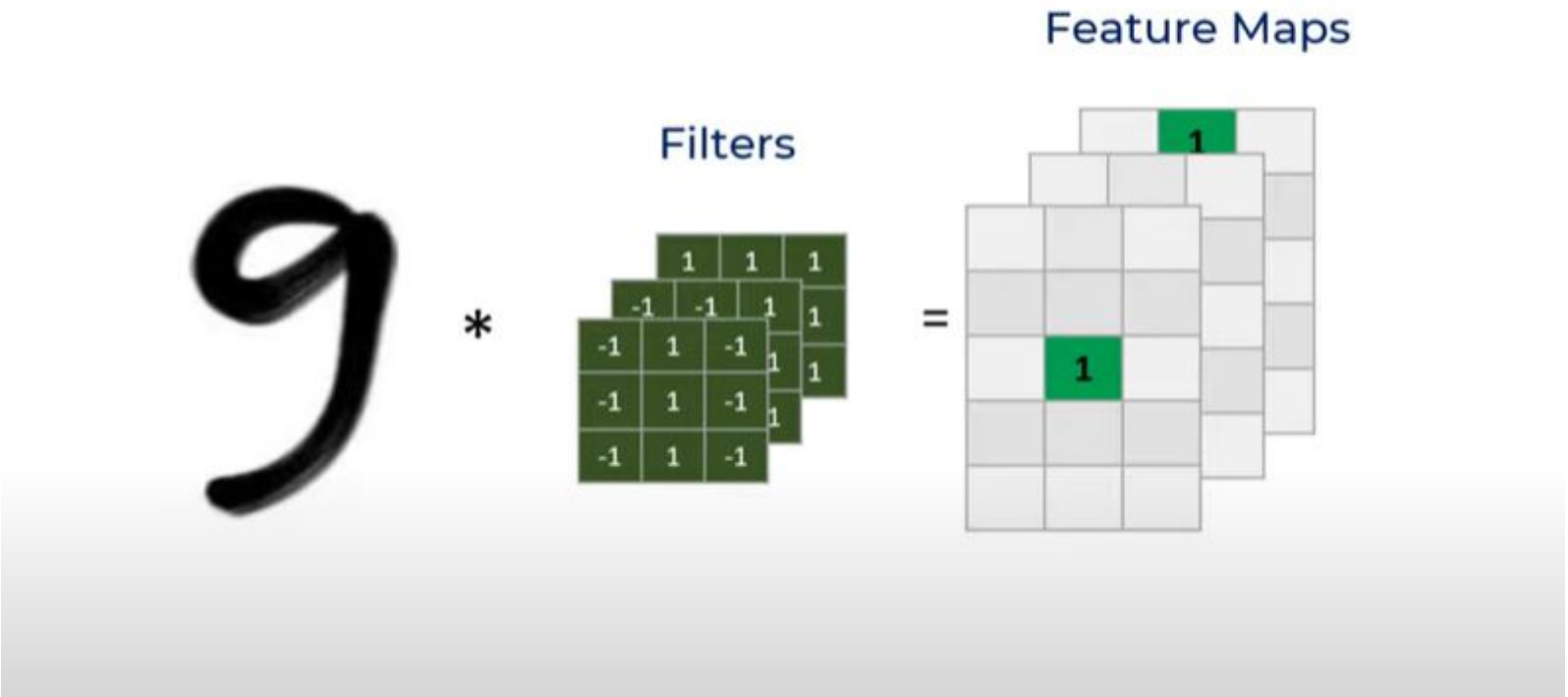
eye
detector

*

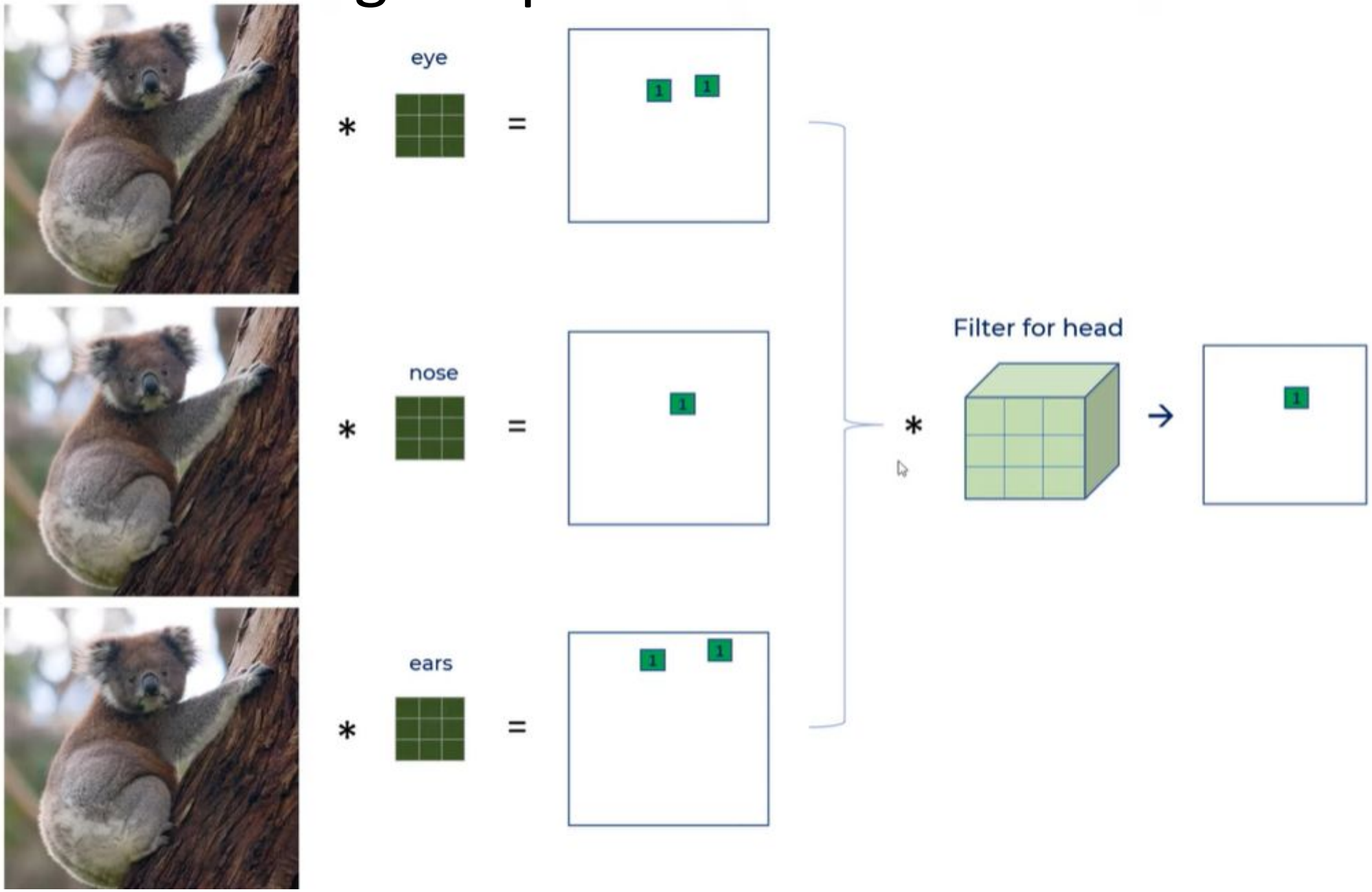
=


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Vertical line detector
-1 1 -1
-1 1 -1
-1 1 -1
1
Diagonal line detector
-1 -1 1
-1 1 -1
1 -1 -1
1

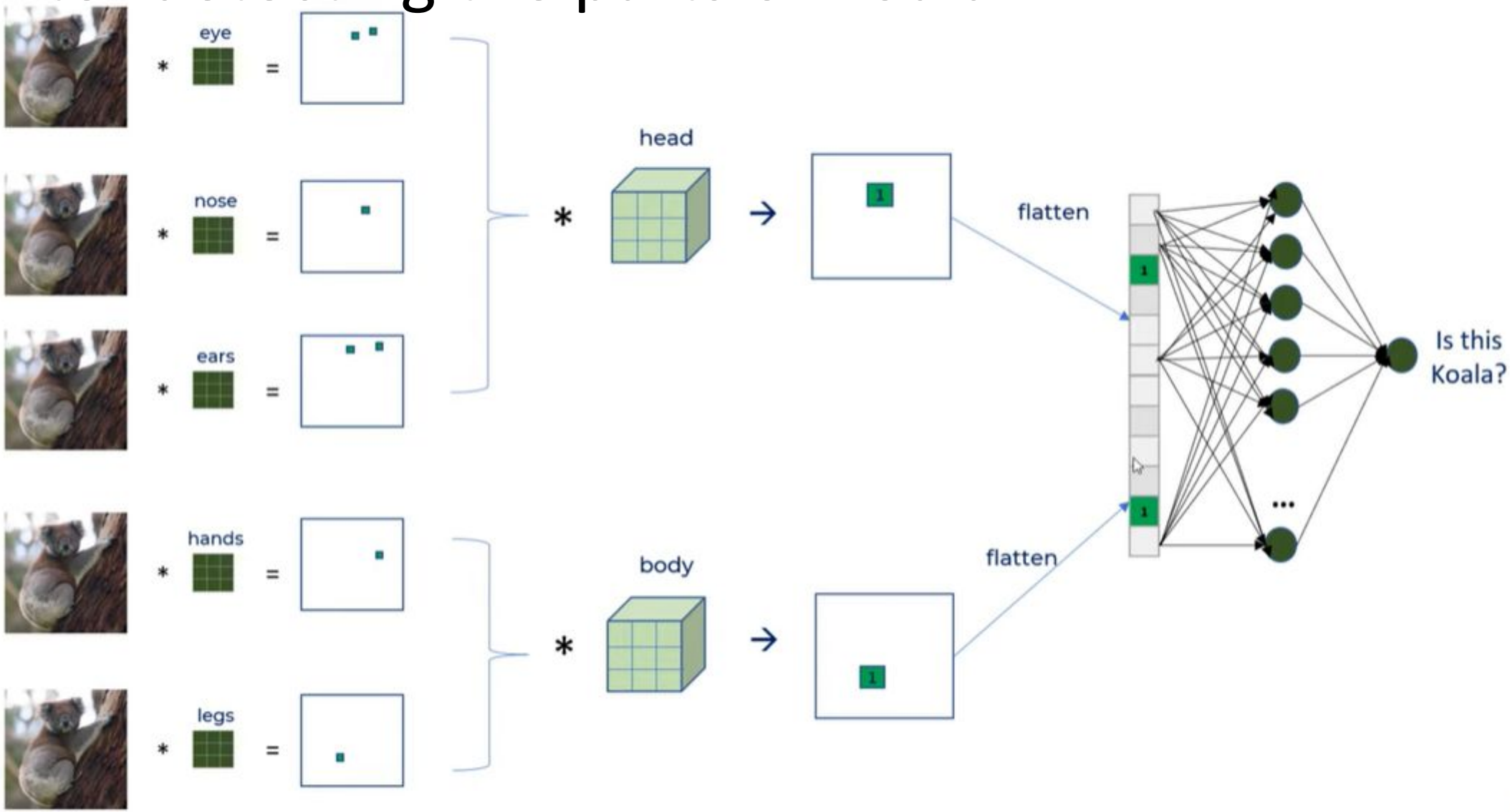
3 Feature Maps – volume



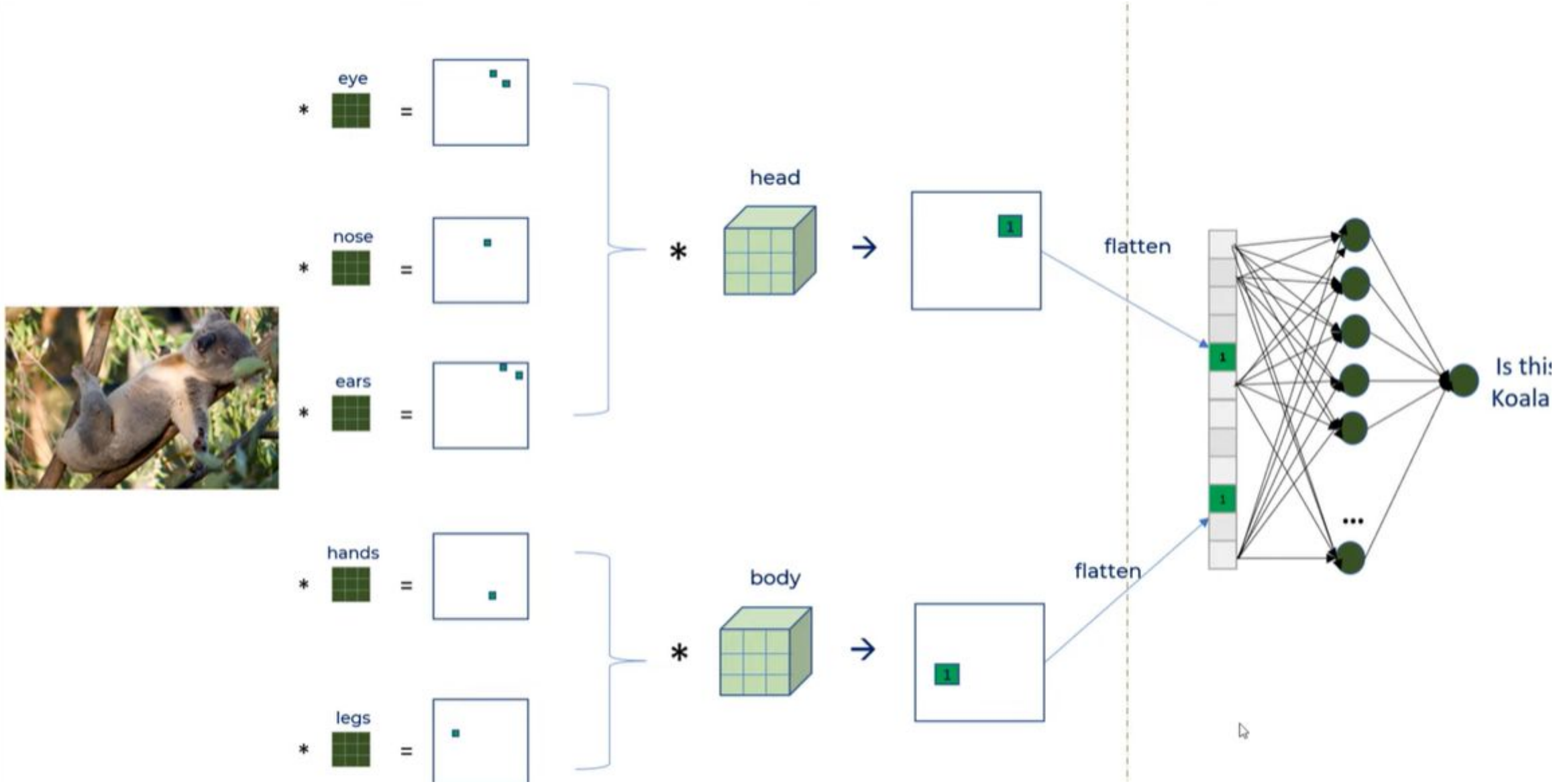
Filter detecting the parts of Koala



Filter detecting the parts of Koala



Activation Function – Relu


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this
Koala

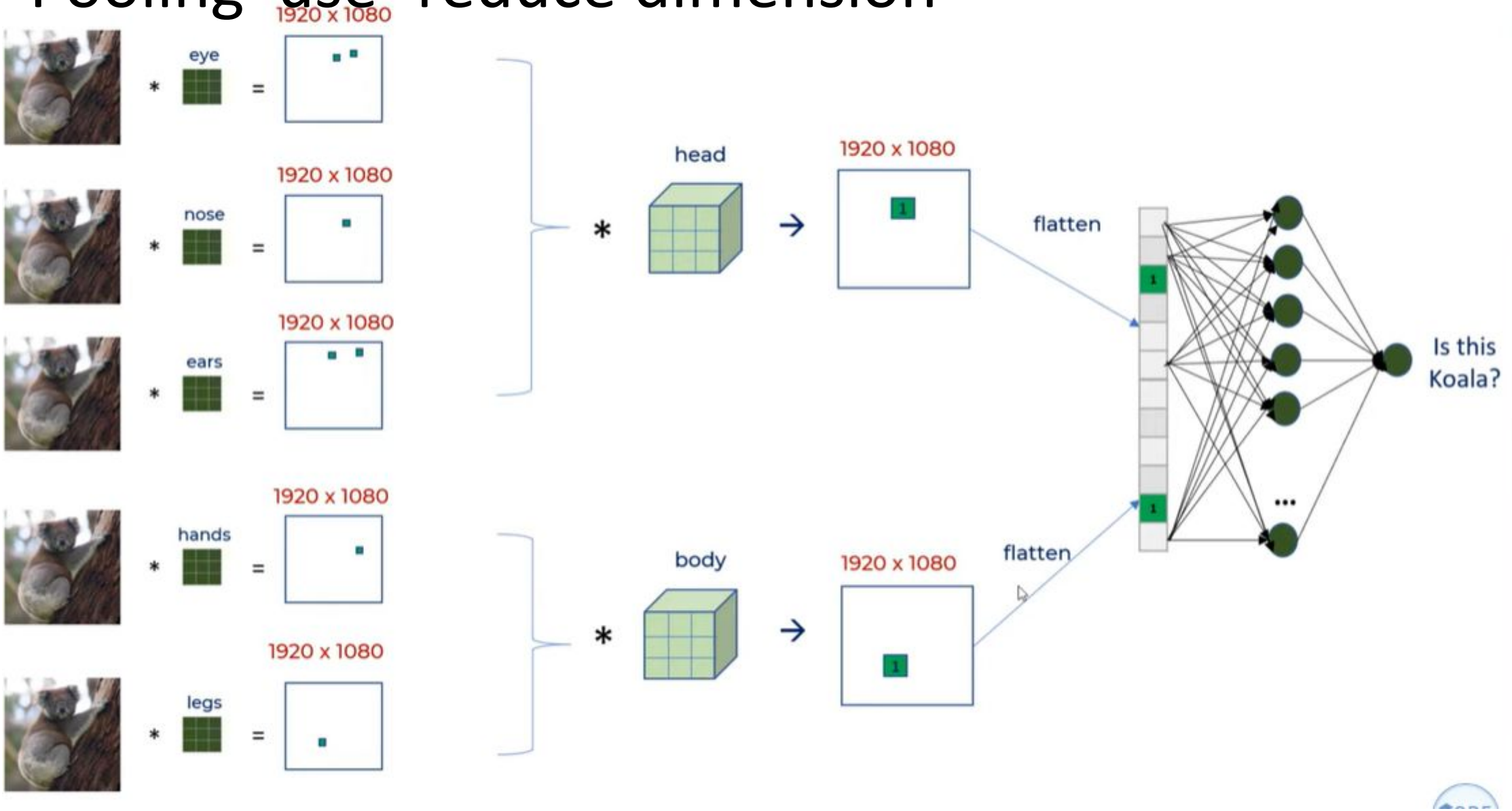
Stride = 2

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

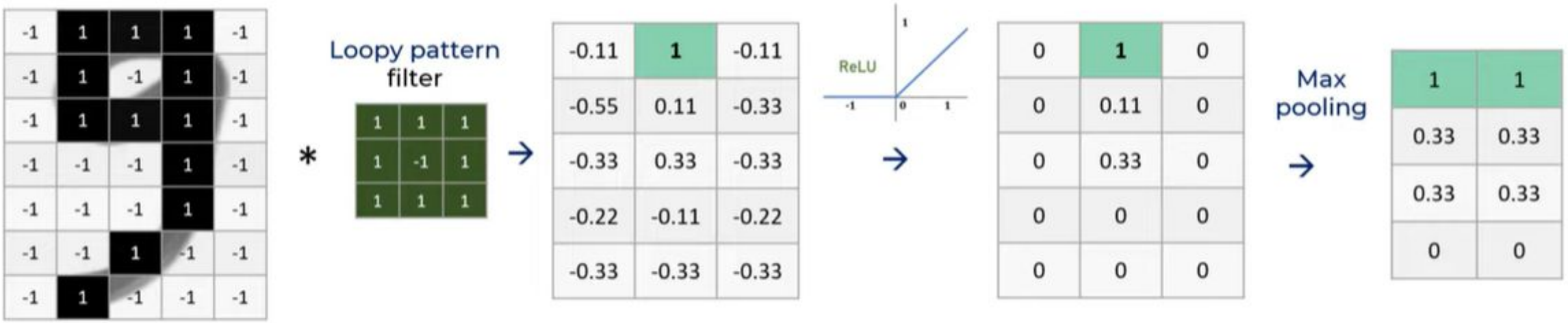
8	**9**
3	2

2 by 2 filter with stride = 2

Pooling use- reduce dimension



Max Pooling


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
Loopy pattern filter
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
ReLU
1
-1
0
1
0 1 0
0 0.11 0
0 0.33 0
0 0 0
0 0 0
Max pooling
1 1
0.33 0.33
0.33 0.33
0 0

There is average pooling also...

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

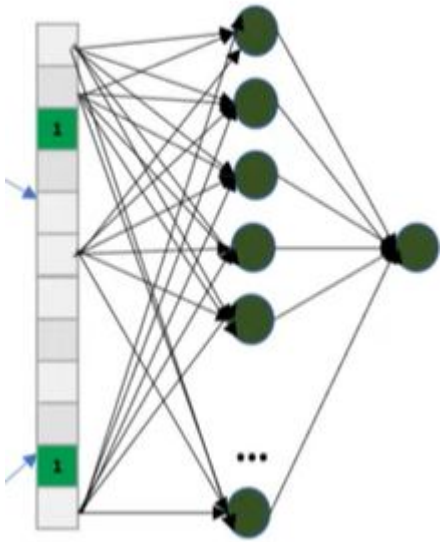
4	4.5
2	0.75

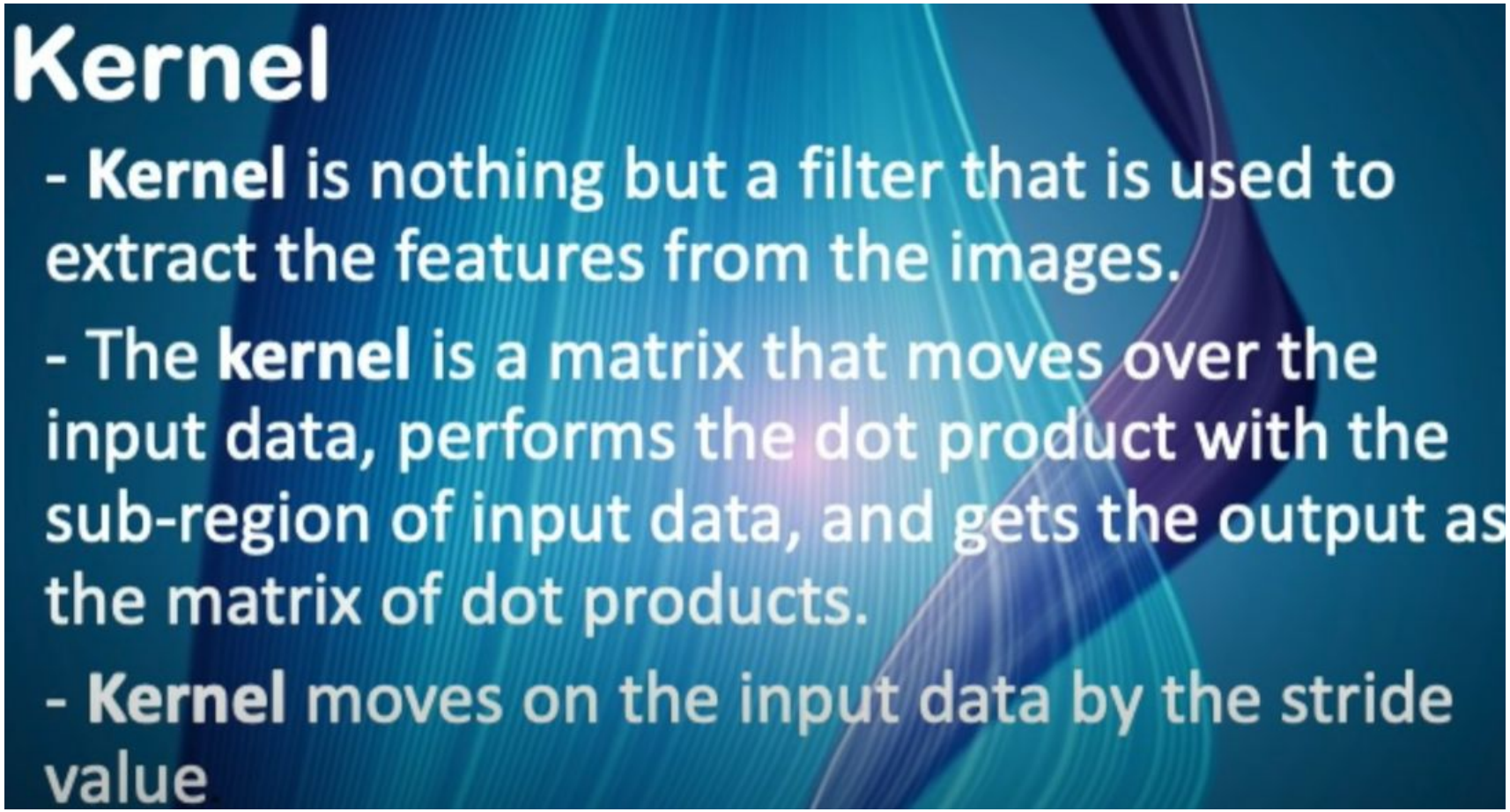
Lets get Serious

CNN

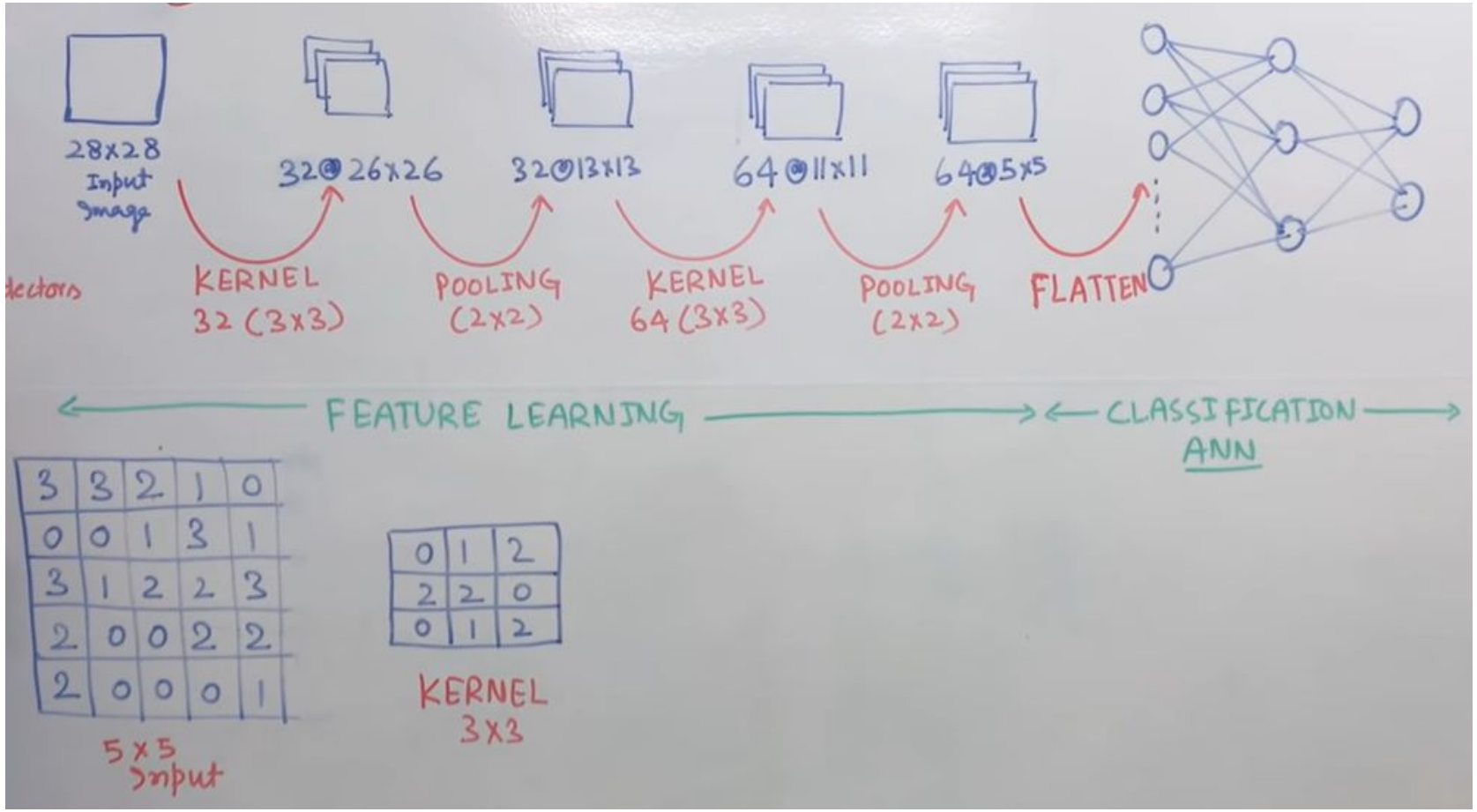
Feature Extraction ⇔ Convolution Layer

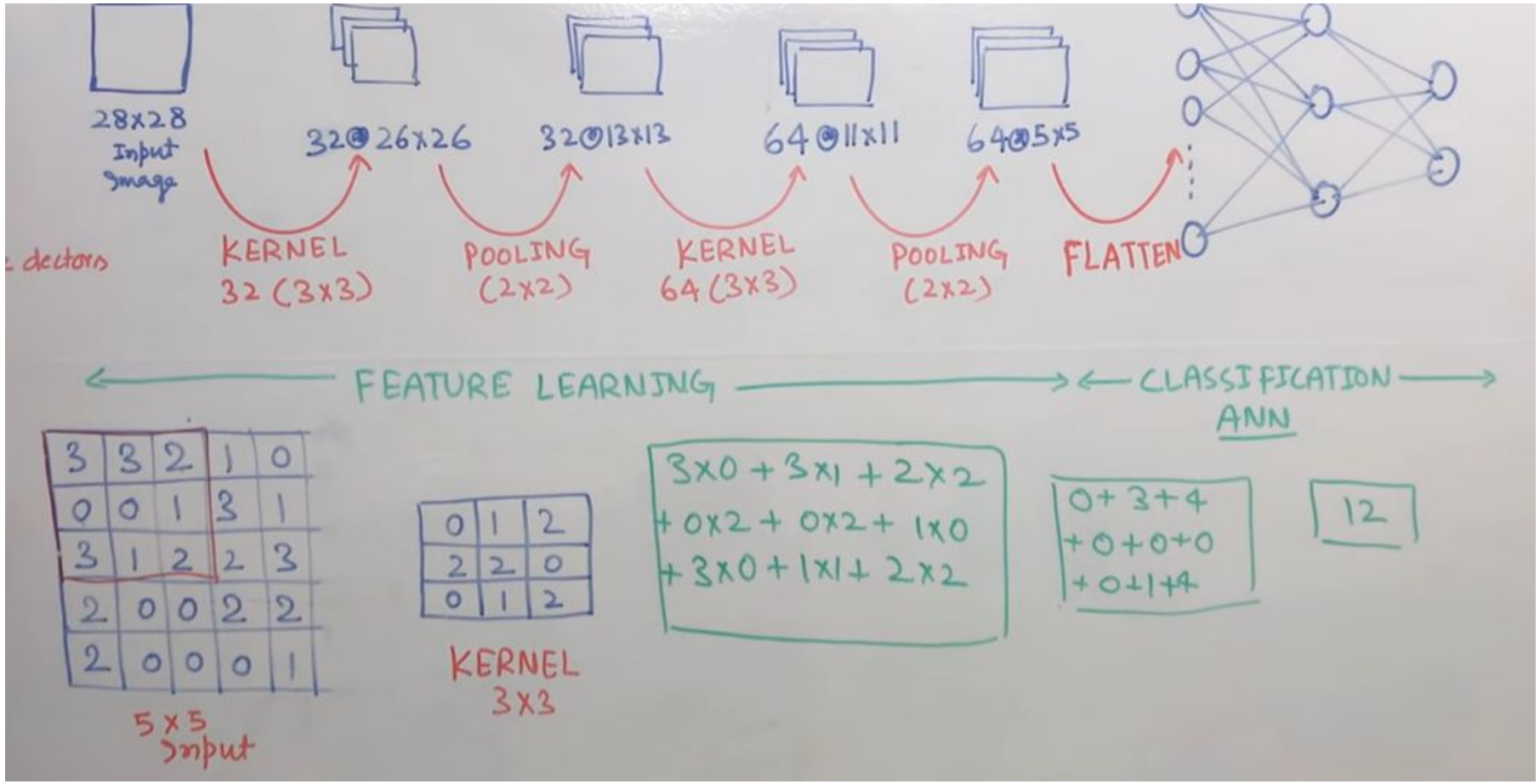
Dense layers- Classification

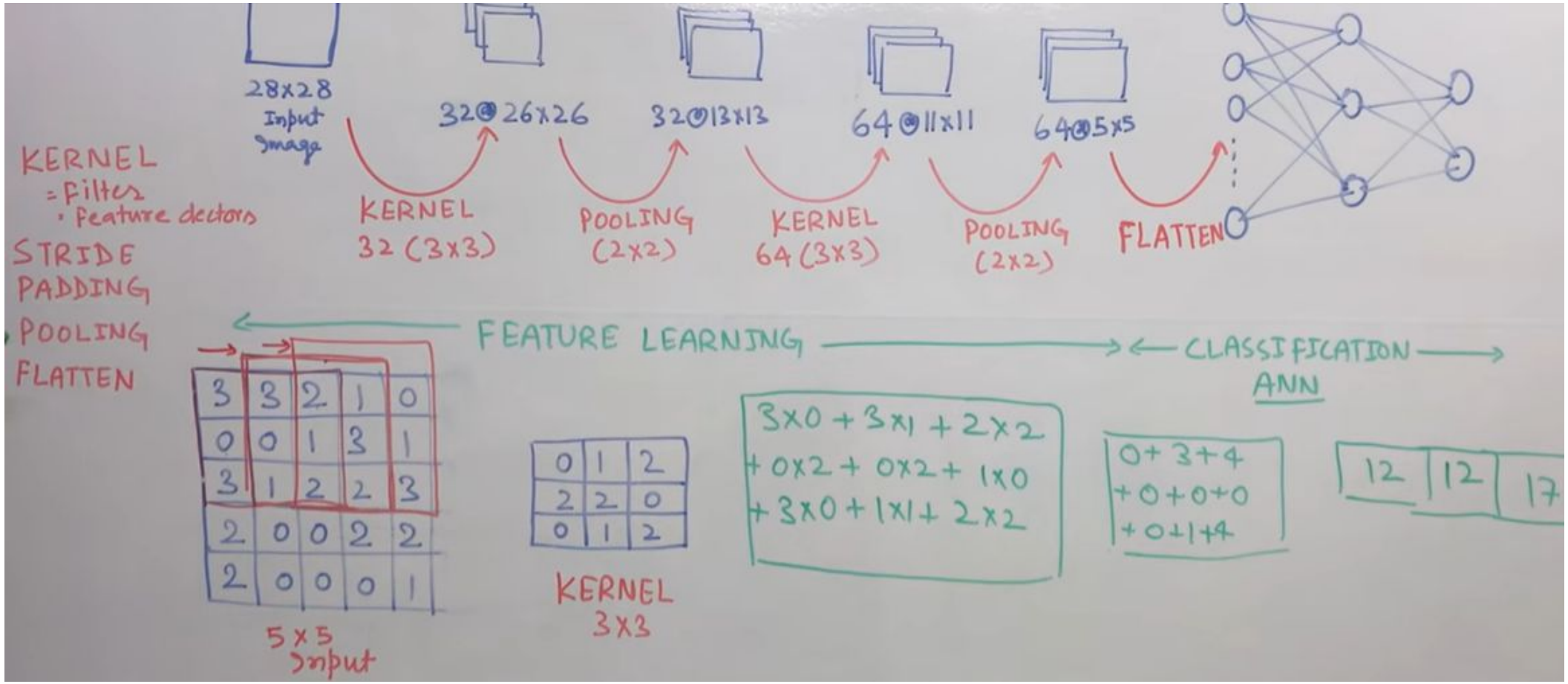


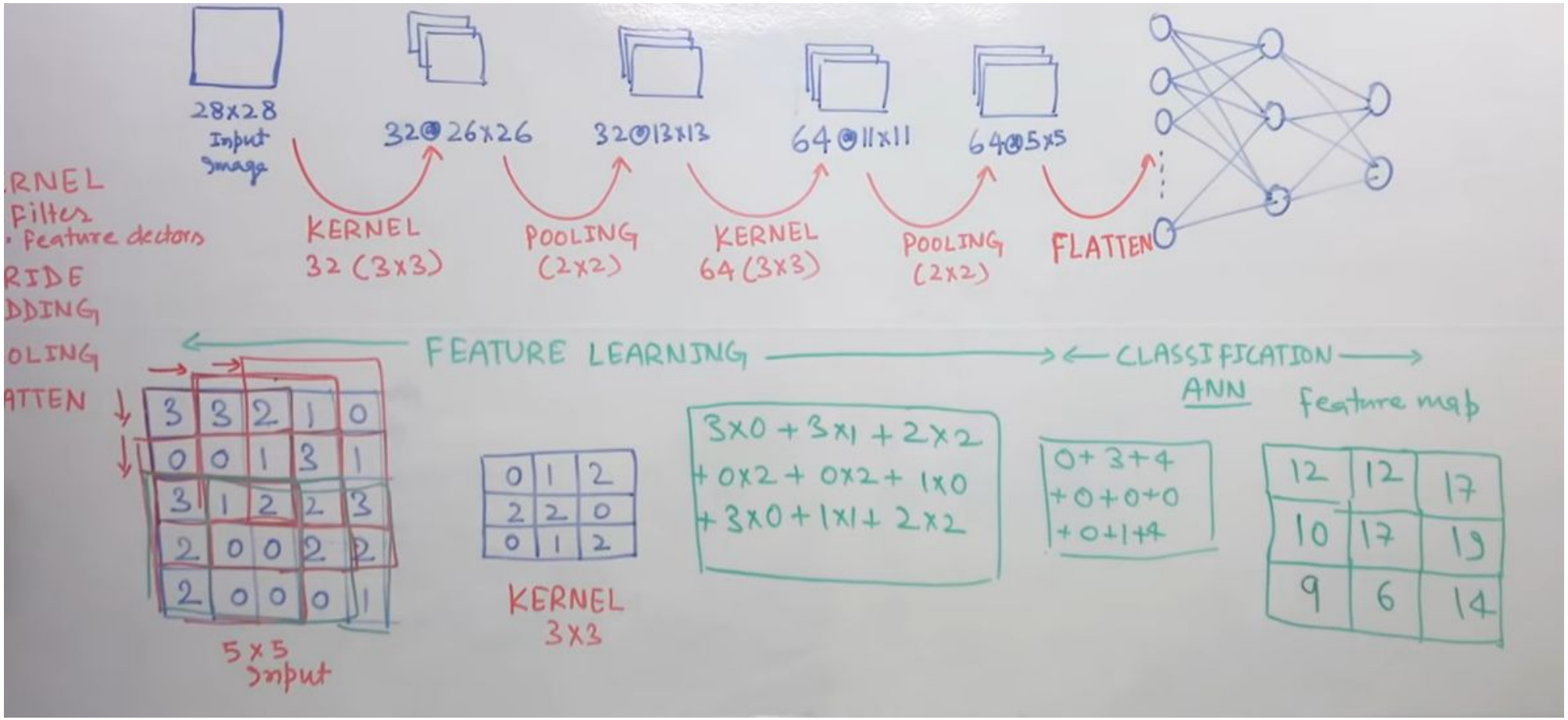

Kernel
- Kernel is nothing but a filter that is used to extract the features from the images.
- The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products.
- Kernel moves on the input data by the stride value

Kernel-> Filter+ Stride




28x28
Input
Image
32@26x26
32@13x13
64@11x11
64@5x5
KERNEL
32 (3x3)
POOLING
(2x2)
KERNEL
64 (3x3)
POOLING
(2x2)
FLATTEN
FEATURE LEARNING
CLASSIFICATION
ANN
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
5x5
Input
0 1 2
2 2 0
0 1 2
KERNEL
3x3
3x0 + 3x1 + 2x2
+ 0x2 + 0x2 + 1x0
+ 3x0 + 1x1 + 2x2
0+3+4
+0+0+0
+0+1+4
12

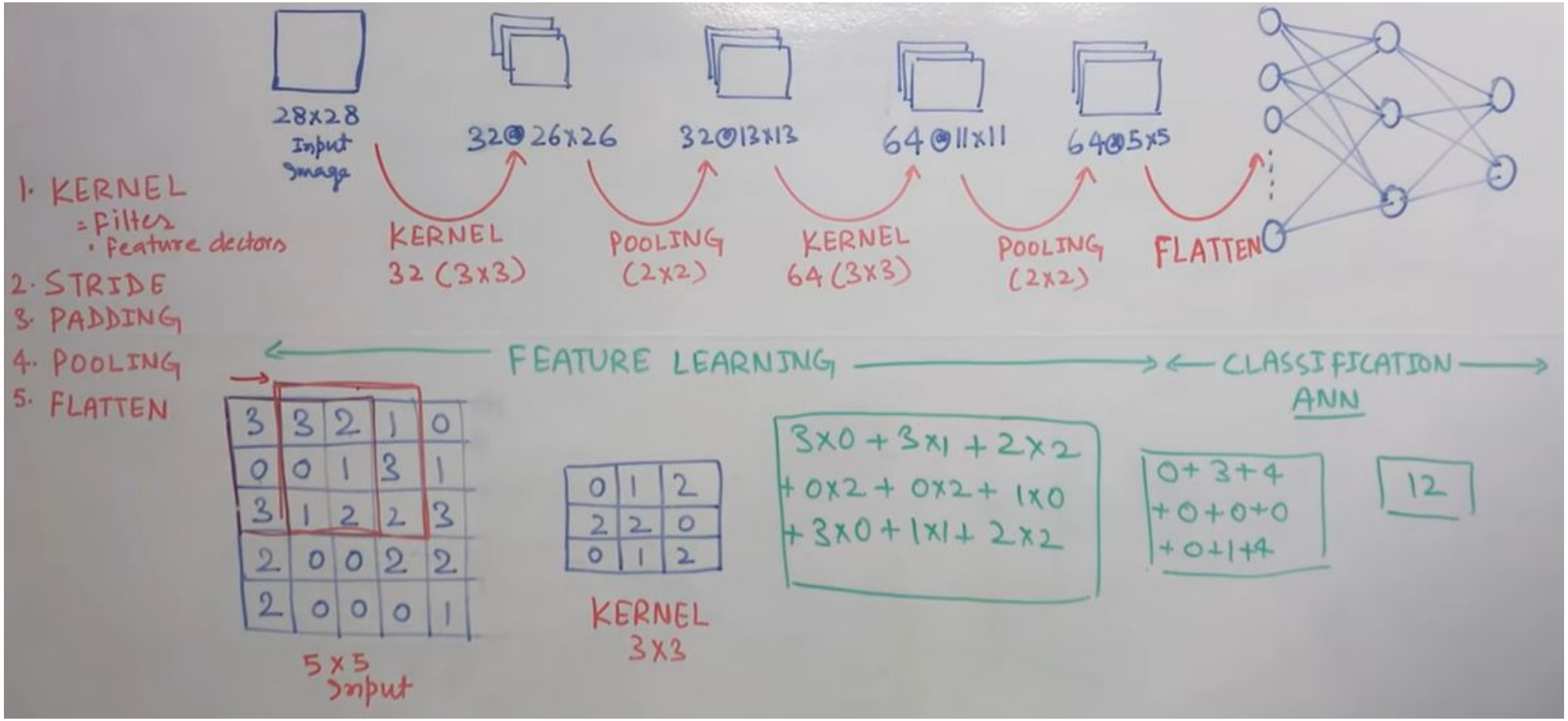

28x28
Input
Image
32@26x26
32@13x13
64@11x11
64@5x5
KERNEL
= Filter
• Feature dectors
STRIDE
PADDING
POOLING
FLATTEN
KERNEL
32 (3x3)
POOLING
(2x2)
KERNEL
64 (3x3)
POOLING
(2x2)
FLATTEN
FEATURE LEARNING
CLASSIFICATION
ANN
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
5x5
Input
0 1 2
2 2 0
0 1 2
KERNEL
3x3
3x0 + 3x1 + 2x2
+ 0x2 + 0x2 + 1x0
+ 3x0 + 1x1 + 2x2
0+3+4
+0+0+0
+0+1+4
12
12
17


28x28
Input
Image
32@26x26
32@13x13
64@11x11
64@5x5
KERNEL
32 (3x3)
POOLING
(2x2)
KERNEL
64 (3x3)
POOLING
(2x2)
FLATTEN
RNEL
Filters
Feature dectors
RIDE
DDING
OLING
ATTEN
FEATURE LEARNING
CLASSIFICATION
ANN
Feature map
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
5x5
Input
0 1 2
2 2 0
0 1 2
KERNEL
3x3
3x0 + 3x1 + 2x2
+ 0x2 + 0x2 + 1x0
+ 3x0 + 1x1 + 2x2
0+3+4
+0+0+0
+0+1+4
12 12 17
10 17 19
9 6 14

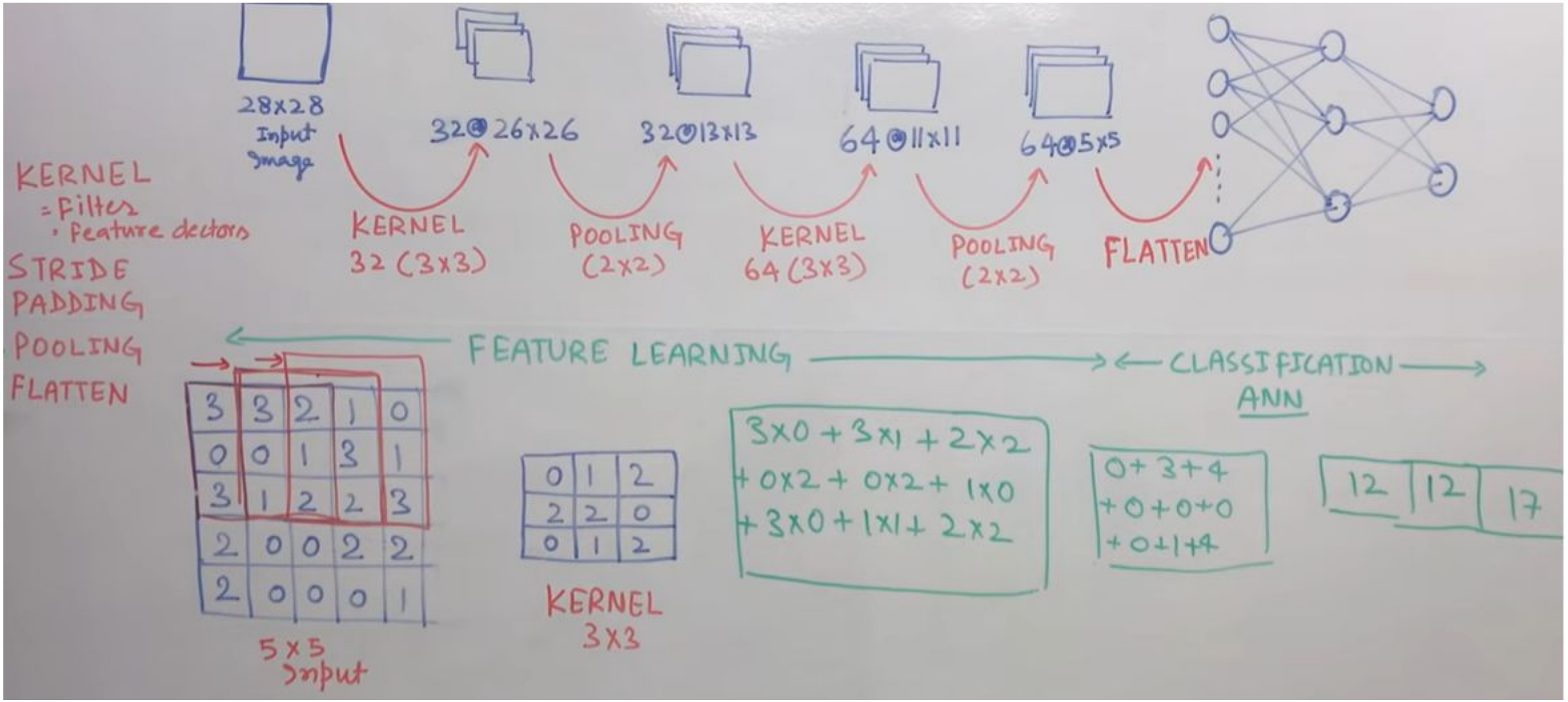
Stride

- The filter is moved across the image left to right, top to bottom, with a one-pixel column change on the horizontal movements, then a one-pixel row change on the vertical movements.
- The amount of movement between applications of the filter to the input image is referred to as the stride, and it is almost always symmetrical in height and width dimensions.
- The default stride or strides in two dimensions is (1,1) for the height and the width movement.

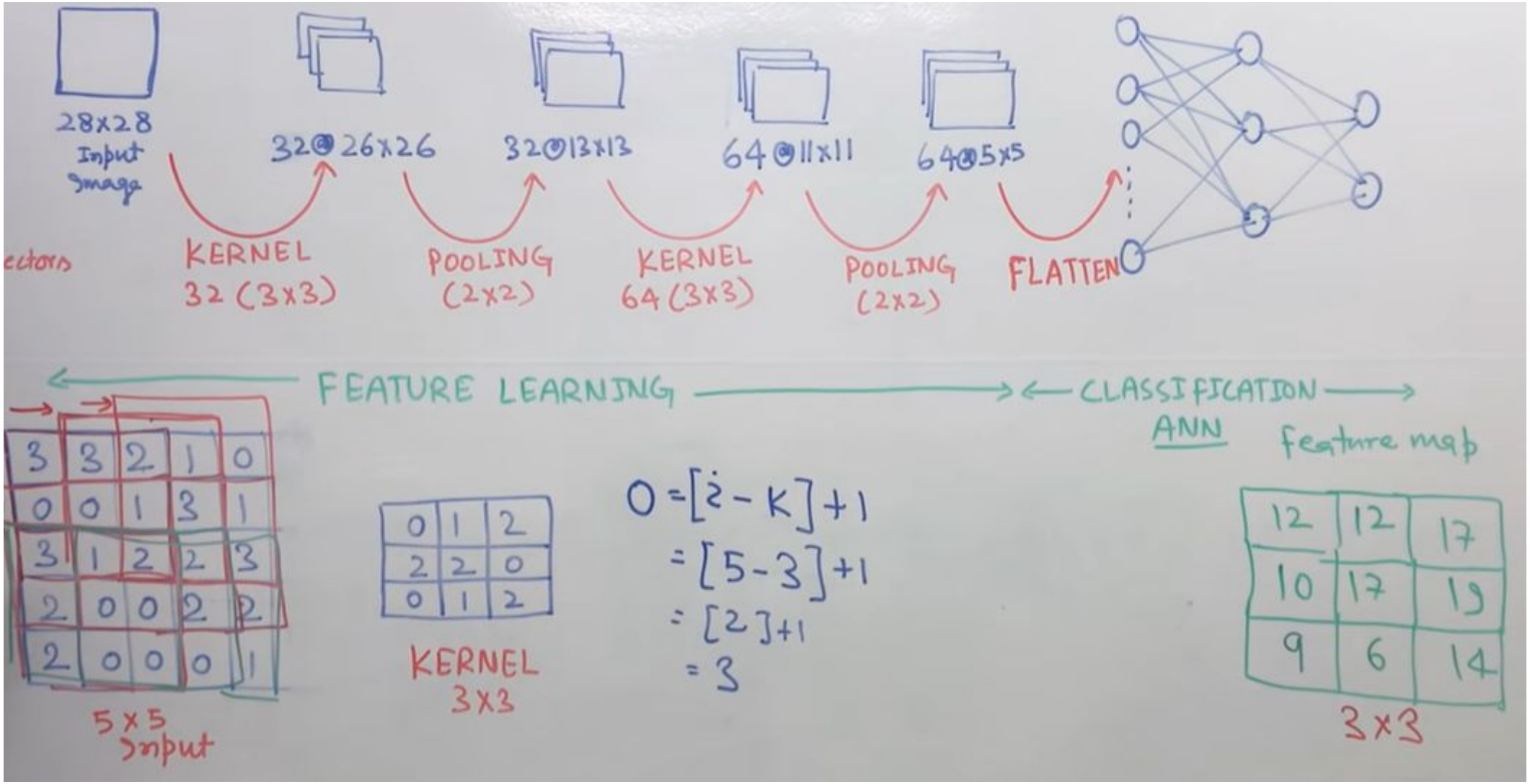
Stride=1


28x28 Input Image
32@26x26
32@13x13
64@11x11
64@5x5
KERNEL 32 (3x3)
POOLING (2x2)
KERNEL 64 (3x3)
POOLING (2x2)
FLATTEN
1. KERNEL = Filters • Feature dectors
2. STRIDE
3. PADDING
4. POOLING
5. FLATTEN
FEATURE LEARNING
CLASSIFICATION
ANN
5x5 Input
KERNEL 3x3
3x0 + 3x1 + 2x2 + 0x2 + 0x2 + 1x0 + 3x0 + 1x1 + 2x2
0+3+4 +0+0+0 +0+1+4
12

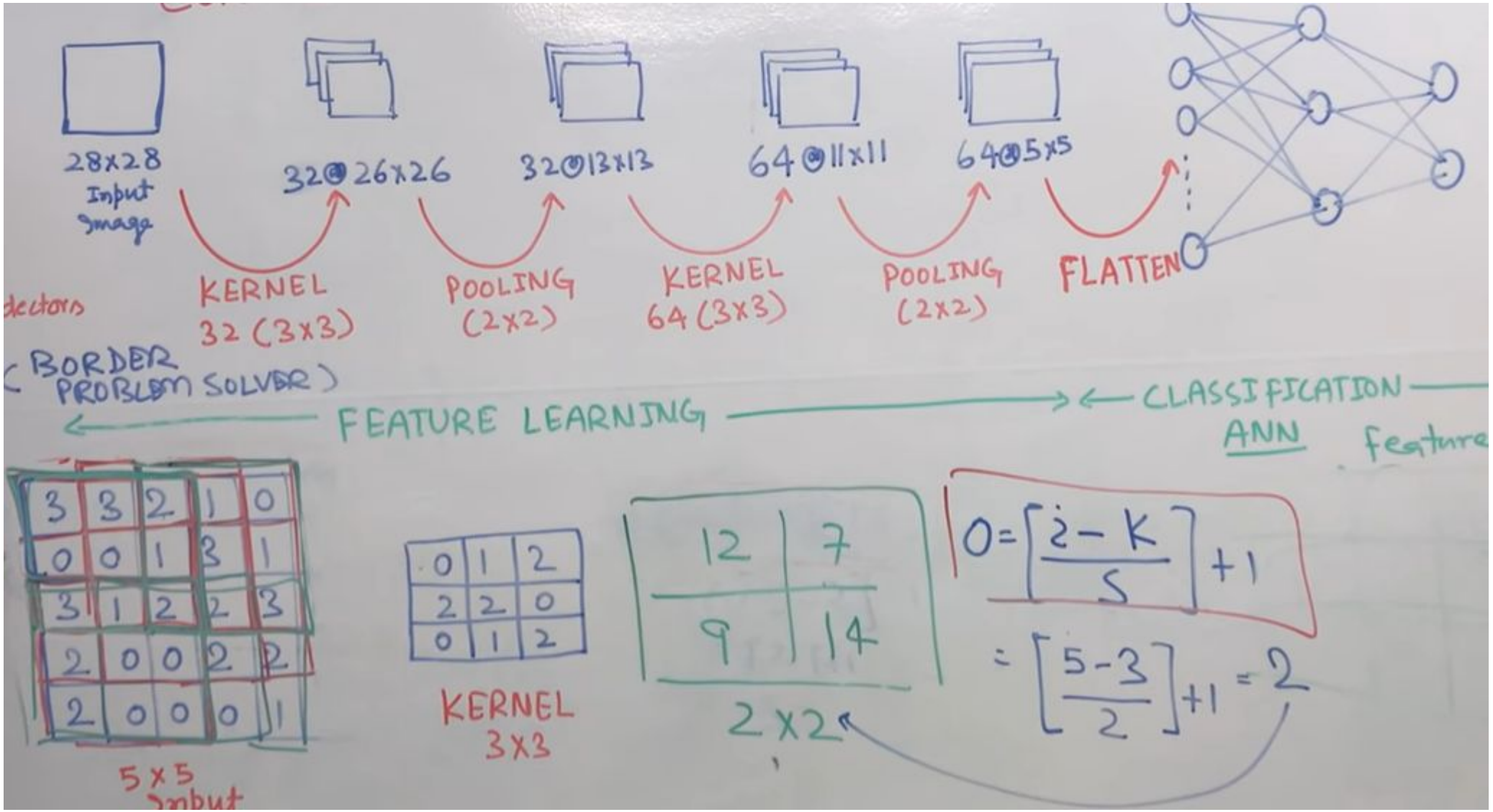
Stride=1


28x28
Input
Image
32@26x26
32@13x13
64@11x11
64@5x5
KERNEL
=Filter
• Feature dectors
STRIDE
PADDING
POOLING
FLATTEN
KERNEL
32 (3x3)
POOLING
(2x2)
KERNEL
64 (3x3)
POOLING
(2x2)
FLATTEN
FEATURE LEARNING
CLASSIFICATION
ANN
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
5x5
Input
0 1 2
2 2 0
0 1 2
KERNEL
3x3
3x0 + 3x1 + 2x2
+ 0x2 + 0x2 + 1x0
+ 3x0 + 1x1 + 2x2
0+3+4
+0+0+0
+0+1+4
12
12
17

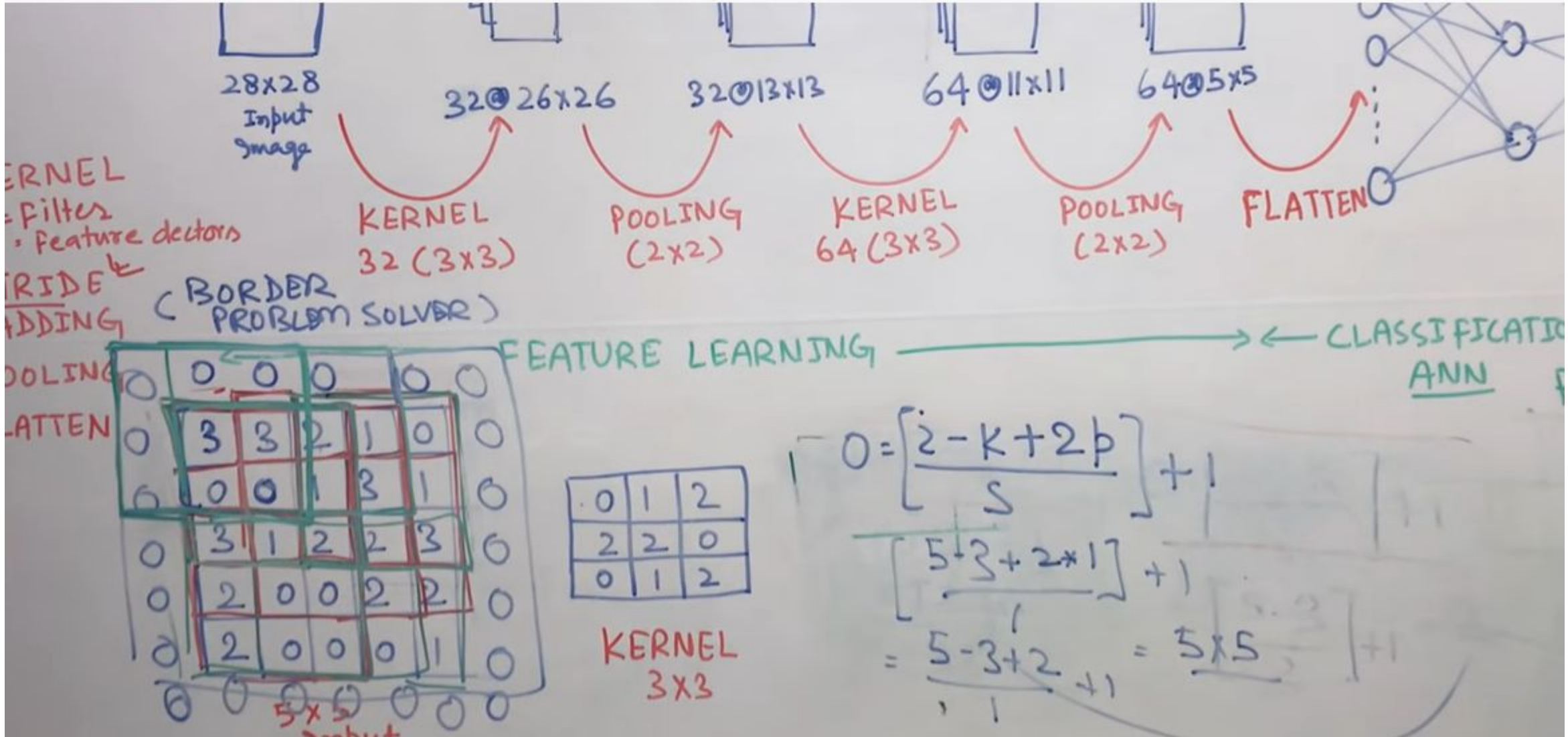
Formula to calculate resultant matrix size, if stride =1



Formula to calculate resultant matrix size, if stride =2



Padding formula for resultant Matrix

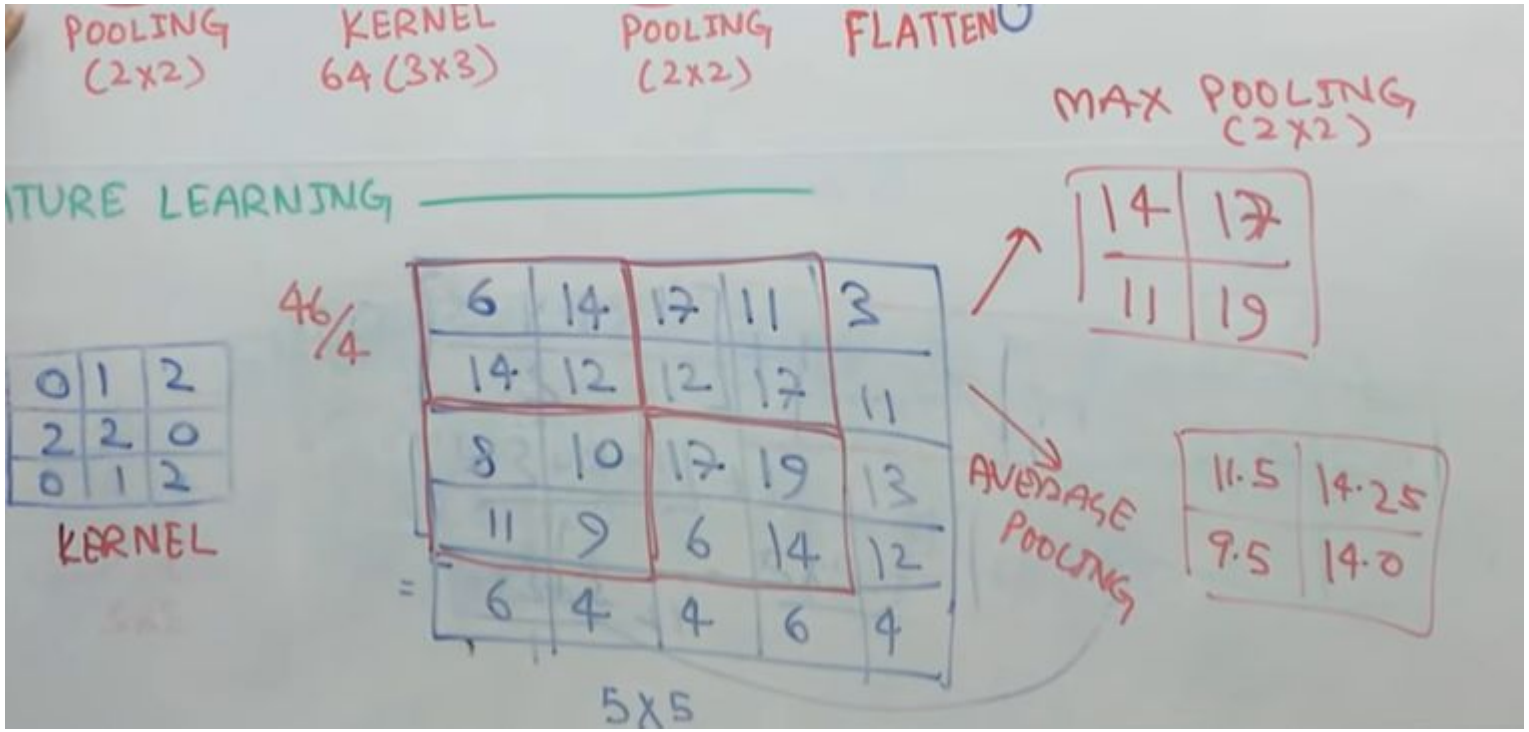


Pooling

- Pooling is required to down sample the detection of features in feature maps.

- Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively.

Pooling types



Benefits of pooling

Reduces dimensions & computation

Reduce overfitting as there are less parameters

Model is tolerant towards variations, distortions

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

ReLU

- Introduces nonlinearity
- Speeds up training, faster to compute

Pooling

- Reduces dimensions and computation
- Reduces overfitting
- Makes the model tolerant towards small distortion and variations

Flattening

- Once the pooled featured map is obtained, the next step is to flatten it.
- It involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

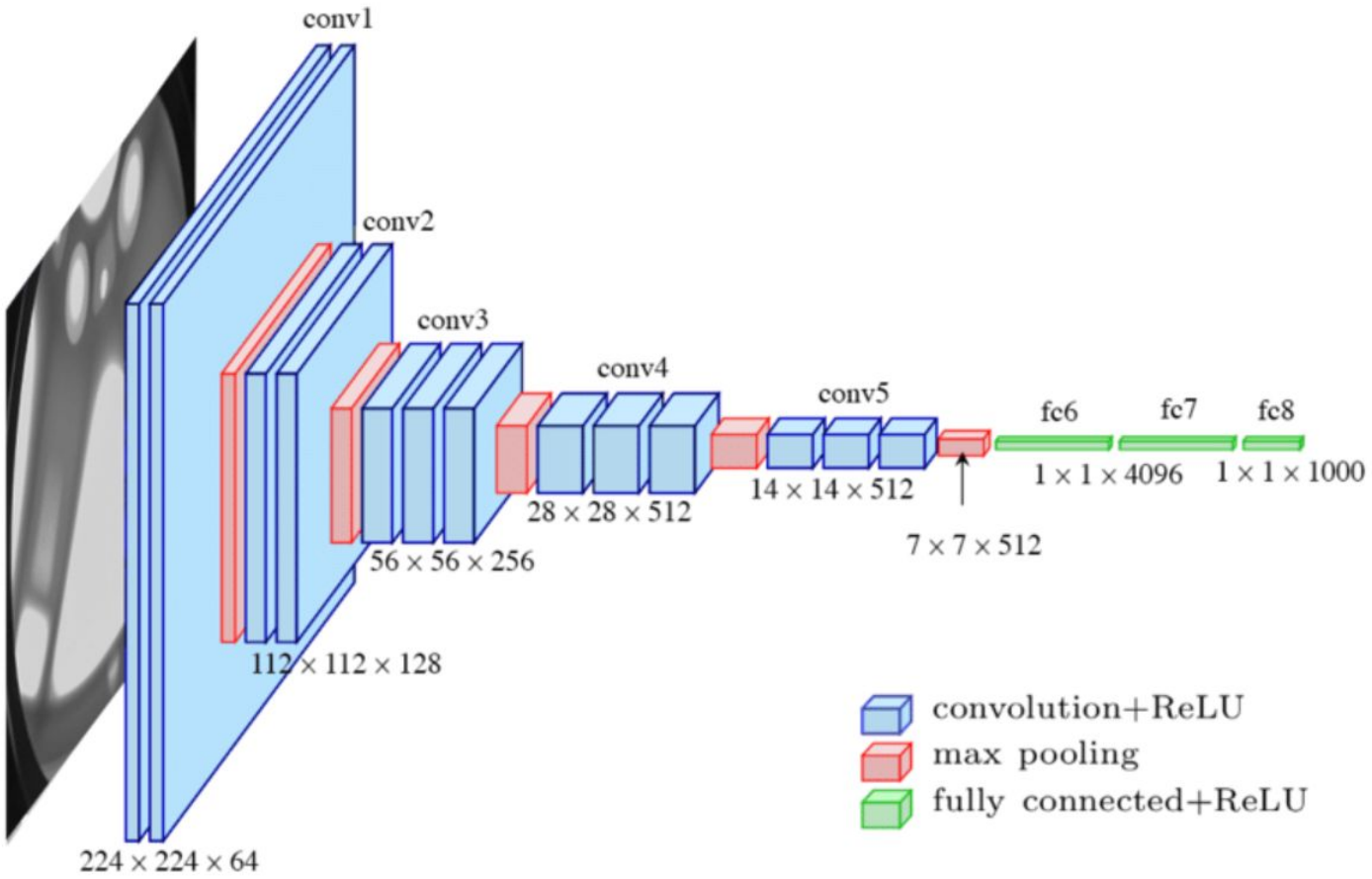
Convolutional Neural Network: Components of CNN Architecture,

- https://youtu.be/Y1qxI-Df4Lk
- https://youtu.be/zfiSAzpy9NM

Properties of CNN

	AlexNet	GoogLeNet	VGG-19
Model	Sequential	Inception modul	Sequential
Depth	8	22	19
Number of layers	25	144	47
Number of filters	96	64	64
Filter size	11x11	7x7	3x3
Stride	4x4	1x1	1x1
Dropout		50%	
Activation function		ReLU	
Output size		2	

Architectures of CNN



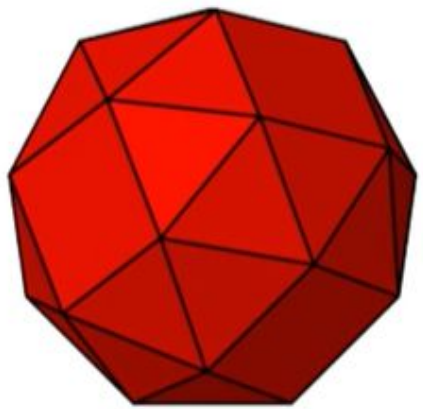
Applications of CNN

- Image recognition / Retina/ Face Recognition
- Image captioning
- Image based Search Engine
- Image quality improvement
- Object recognition
- Colour generation from BW pictures
- Tumour reorganization

Autoencoder: Introduction

- Data compression
- Unsupervised NN that uses ML for performing compression of data – dimensionality reduction
- Can learn non linear transformation

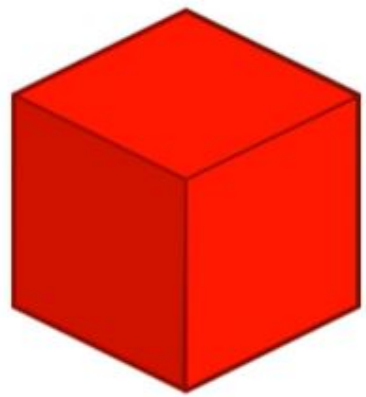
Multidimensional Data



Data represented best

Slow performance, High Precision

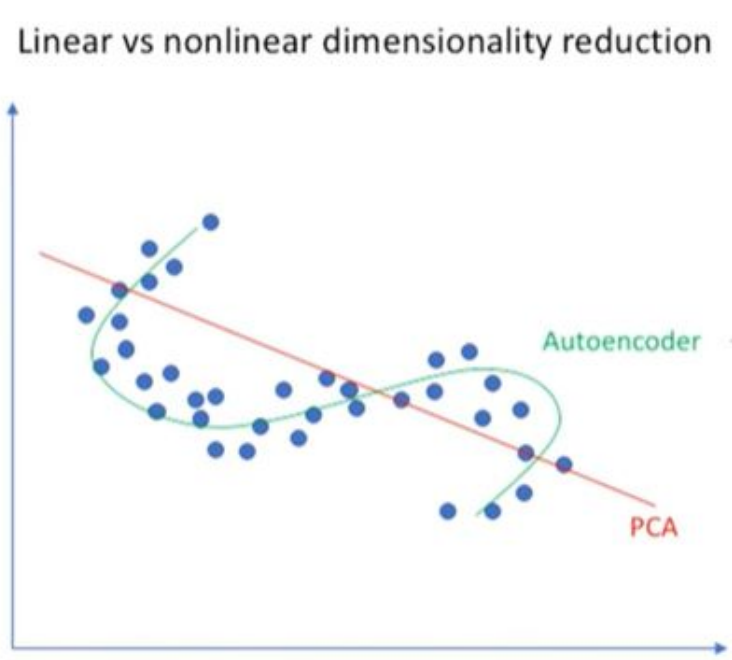
Low Dimensional Data



Reduced Precision

High Performance

Need of Autoencoders when Principle Component analysis is there?



Non-linear Transformations
Non-linear activation function and multiple layers

Convolutional Layers
An autoencoder doesn't have to learn dense layers

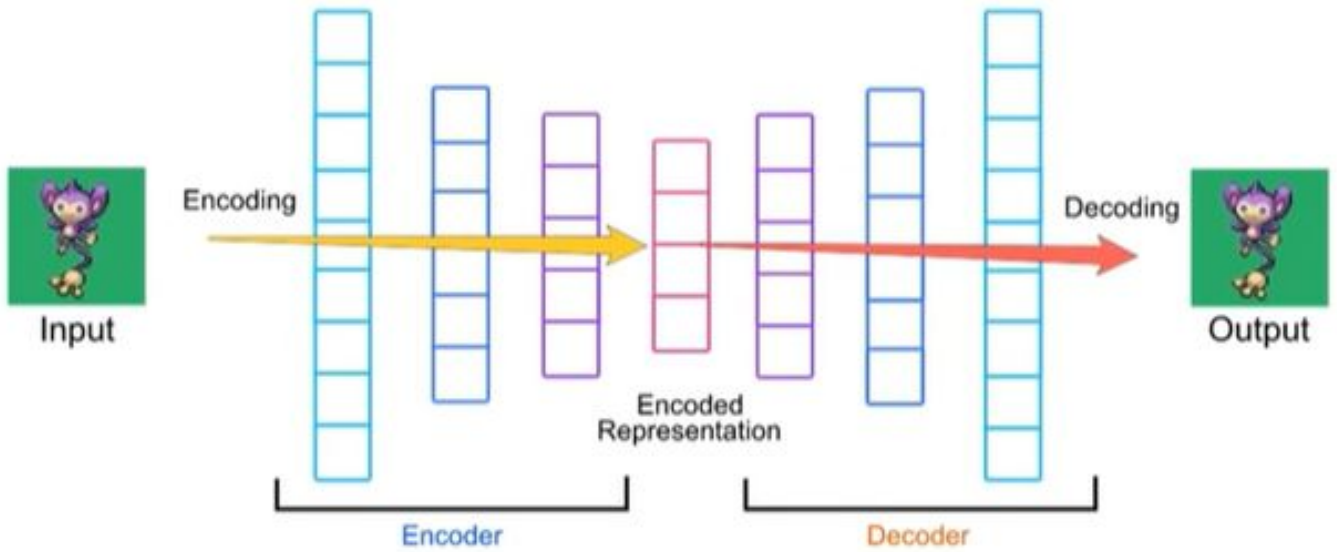
Higher Efficiency
More efficient to learn several layers with an autoencoder

Multiple Transformations
It gives a representation as the output of each layer

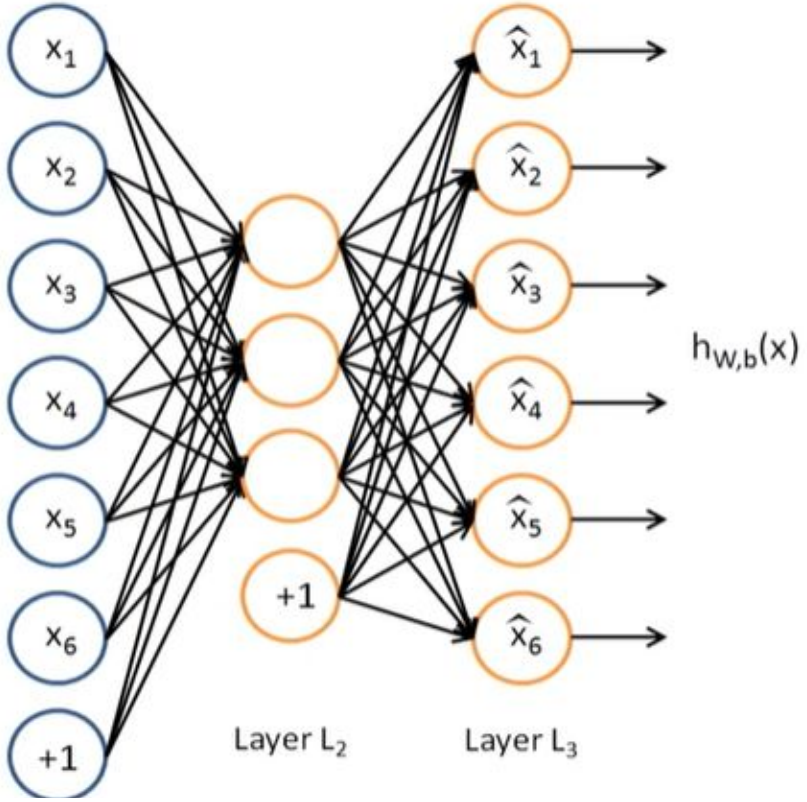
Need of Autoencoders – dimension reduction and data visualization are the application of AECD

Introduction to Autoencoders

An ***autoencoder*** neural network is an unsupervised Machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs.



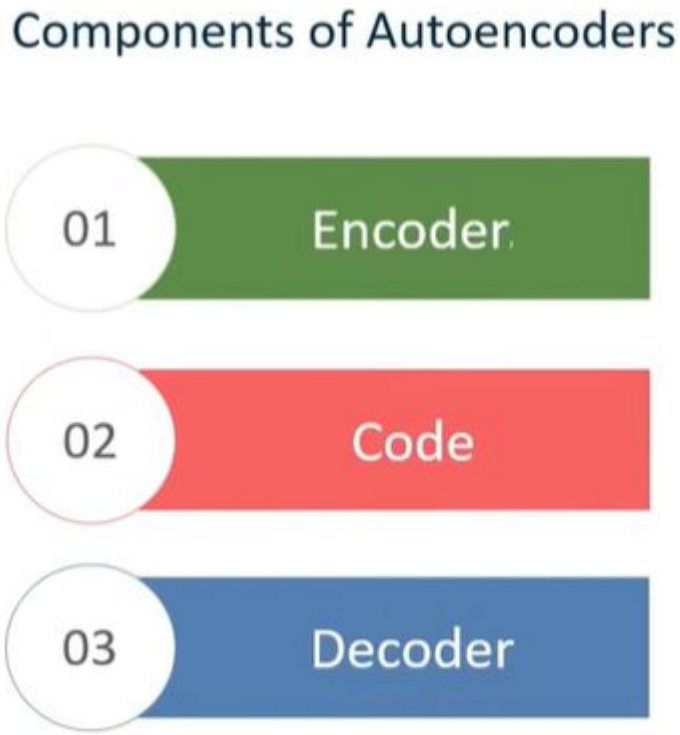
Autoencoder: Key Features

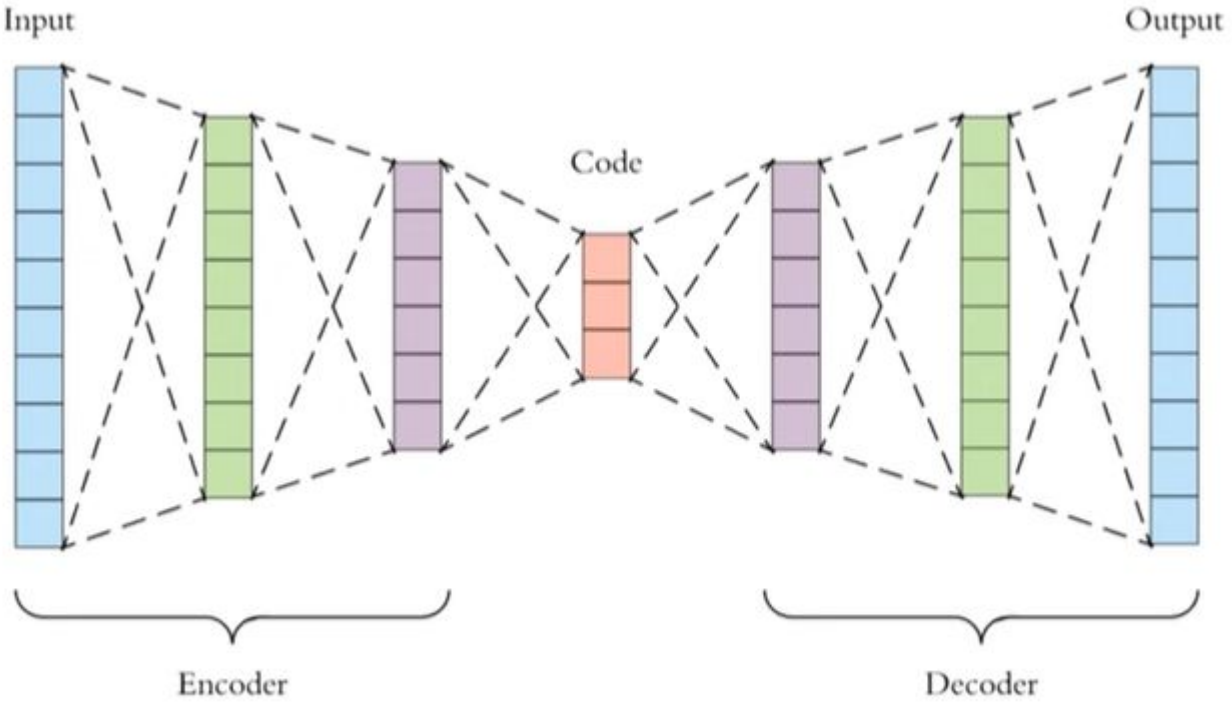


Key Facts about Autoencoders

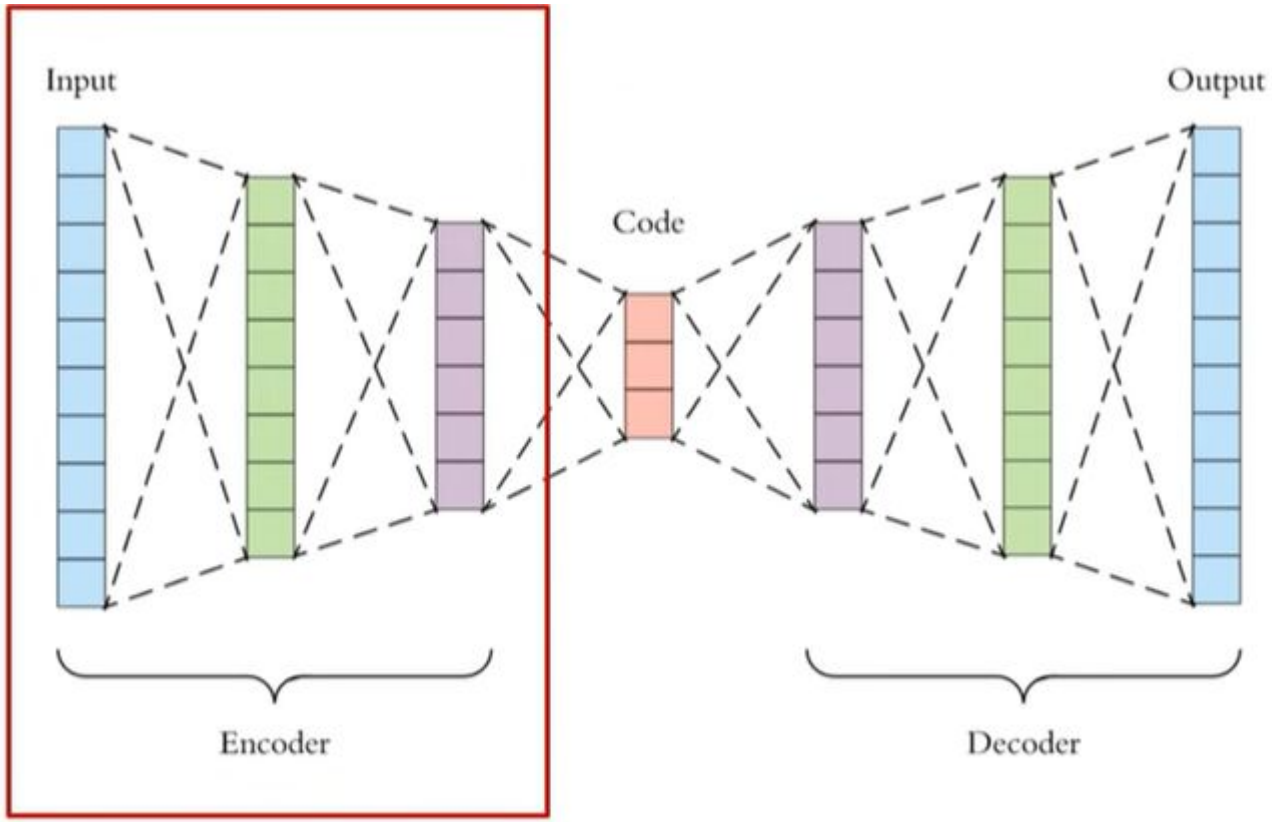
- It is an unsupervised ML algorithm similar to PCA
- It minimizes the same objective function as PCA
- It is a neural network
- The neural network's target output is its input

Components of Autoencoder





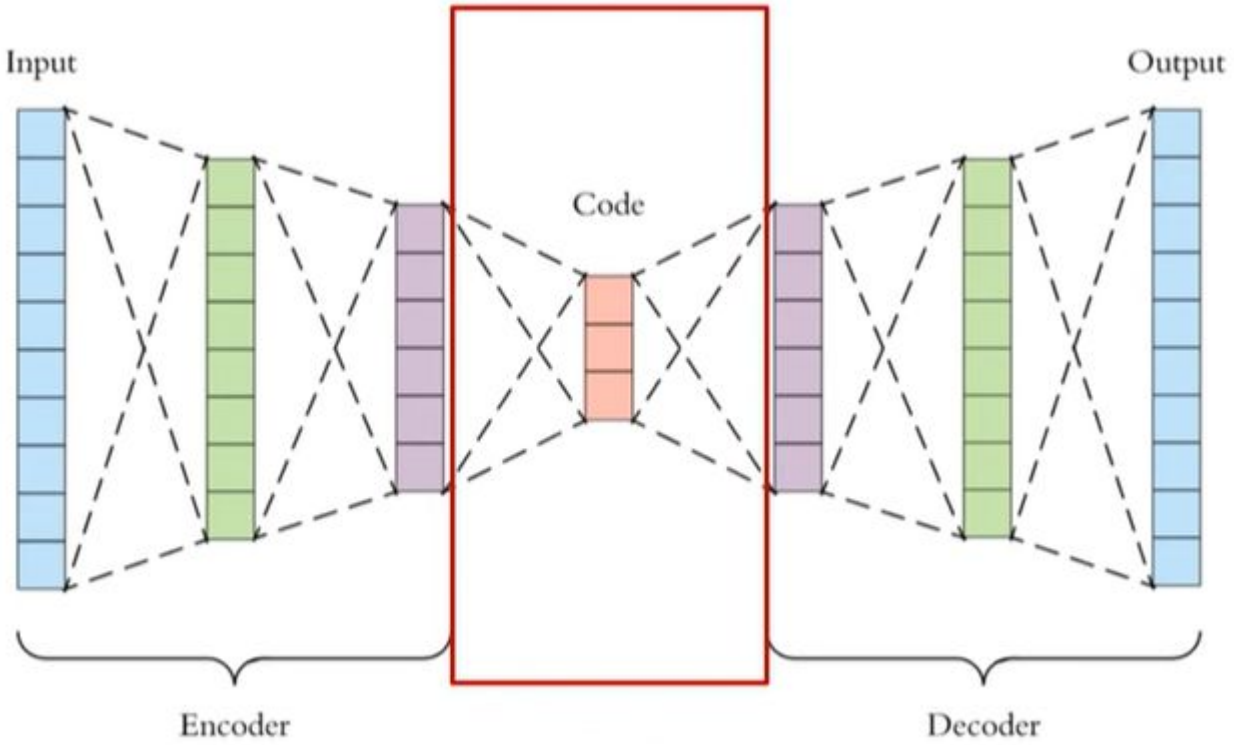
Components of Autoencoder



01 **Encoder.**

This is the part of the network that compresses the input into a latent space representation.

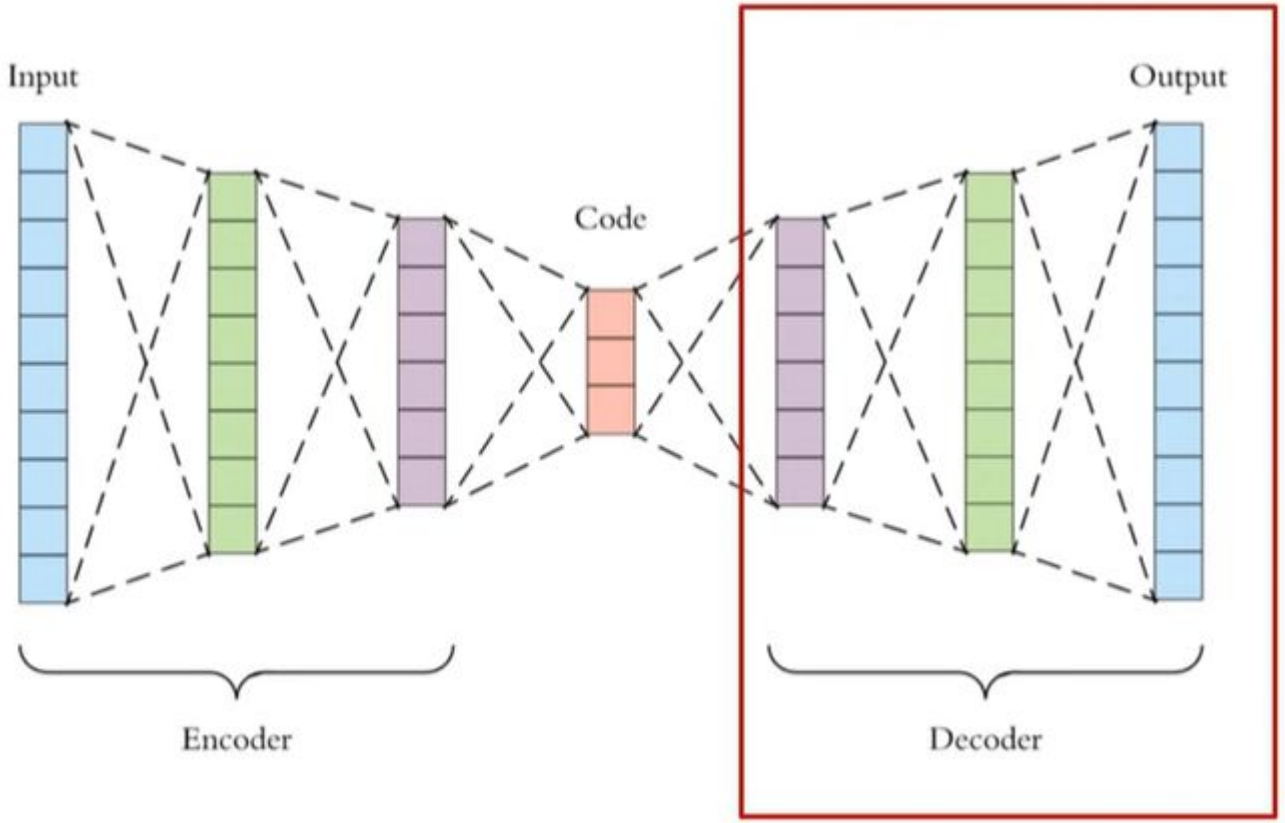
Components of Autoencoder



02 Code

This is the part of the network represents the compressed input that is fed to the decoder

Components of Autoencoder

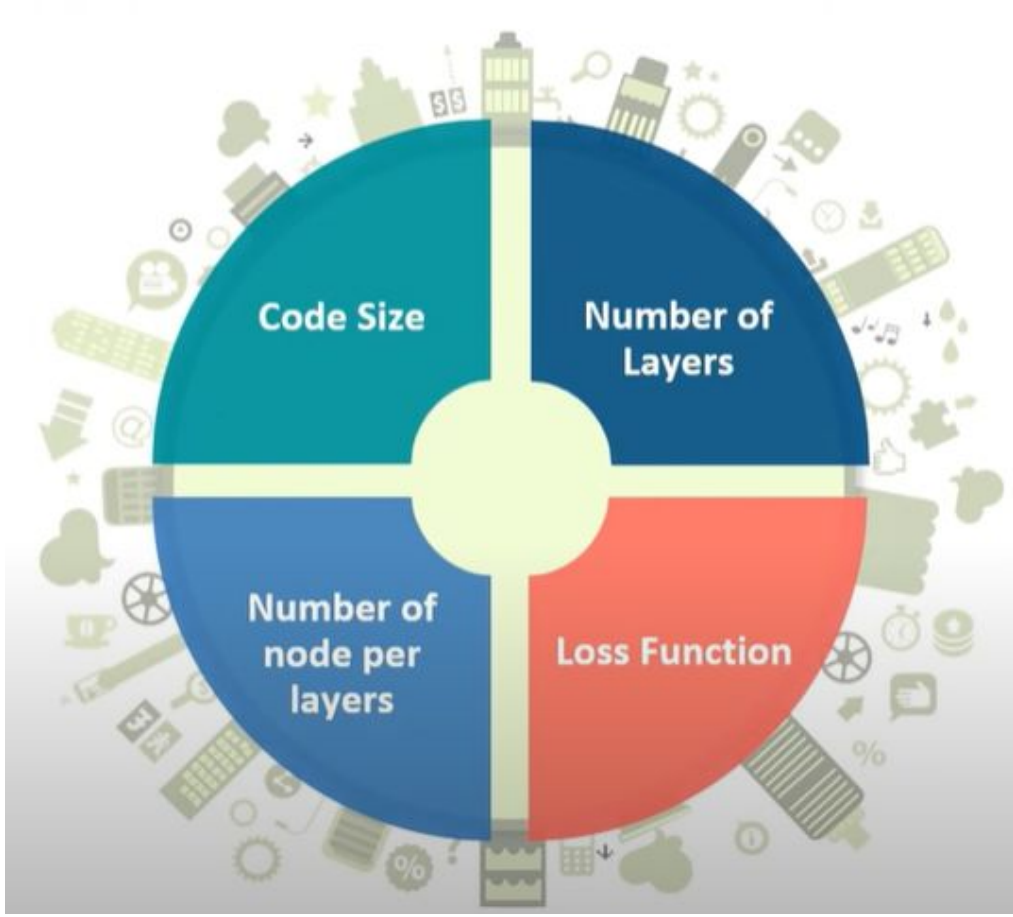


03 Decoder

This part aims to reconstruct the input from the latent space representation

Properties of Autoencoders

Hyperparameter – For Training AECD



Code Size

Smaller size results in more compression

Number of Layers

The autoencoder can have many layers

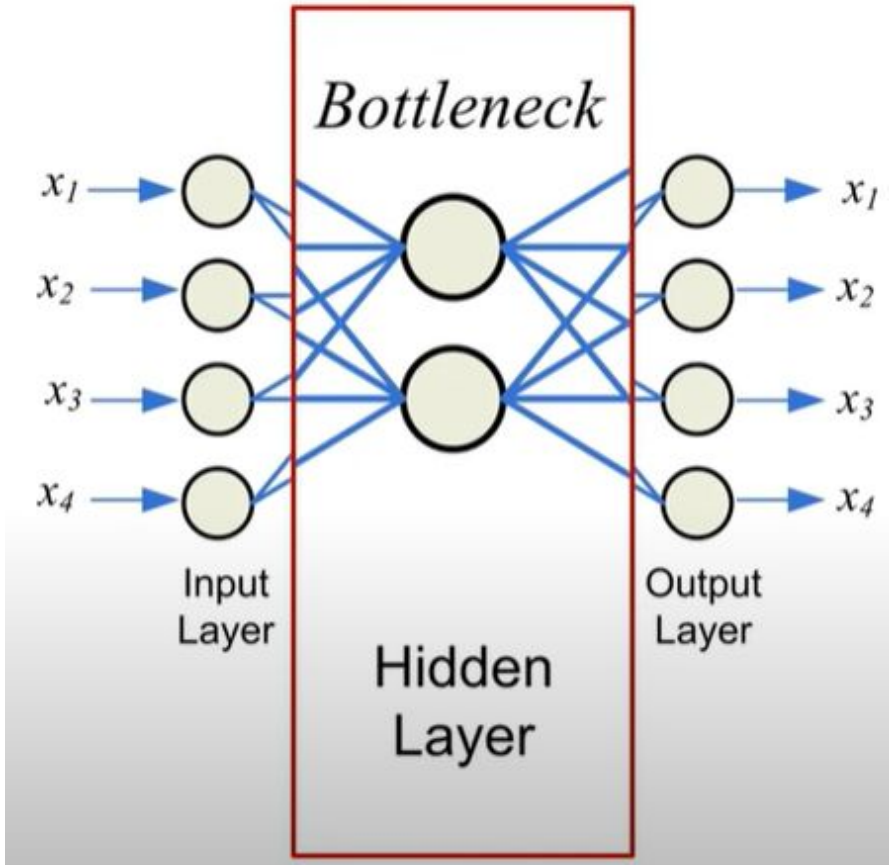
Loss Function

Mean squared error or binary cross entropy

Number of node per layers

Stacked autoencoders look like a sandwich

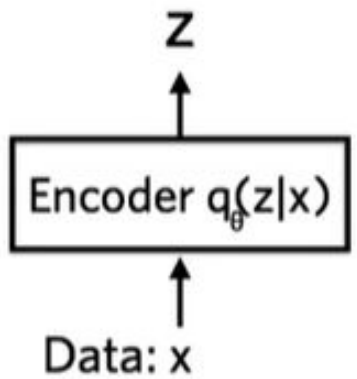
Architecture of Autoencoders

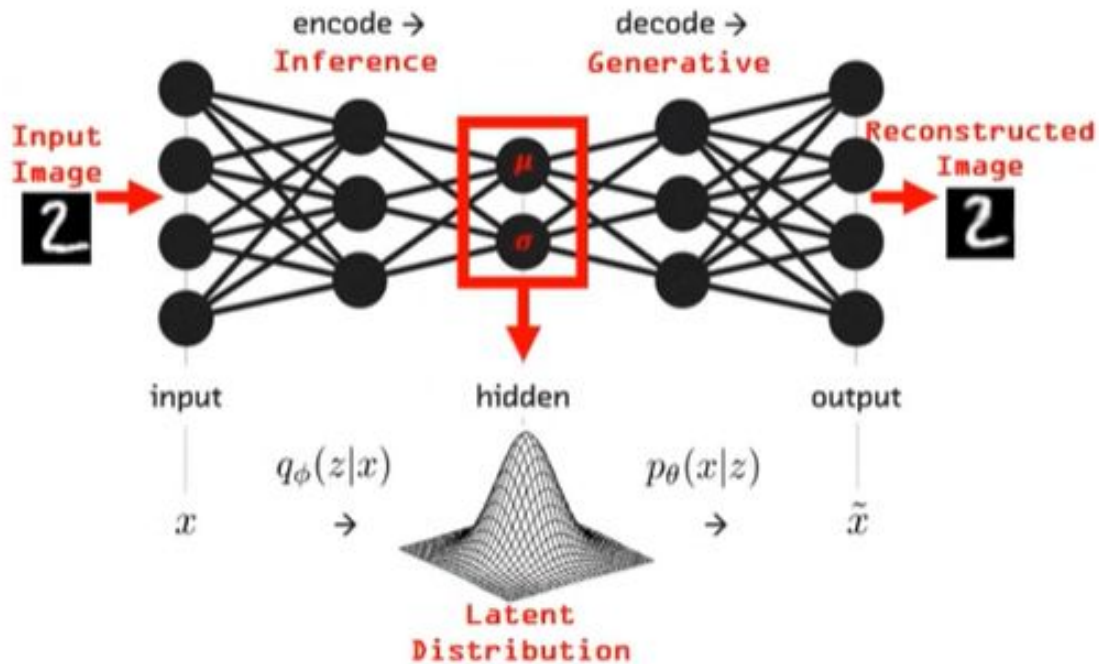


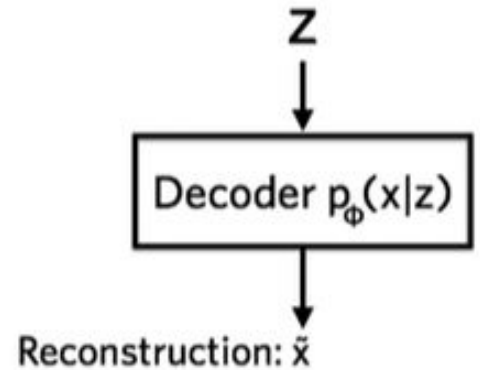
Bottleneck approach is an approach to for deciding which aspects of observed data are relevant information and what aspects can be thrown away

- Compactness of representation, measured as the compressibility
- Representation retains about some behaviourally relevant variables

Architecture of Autoencoders





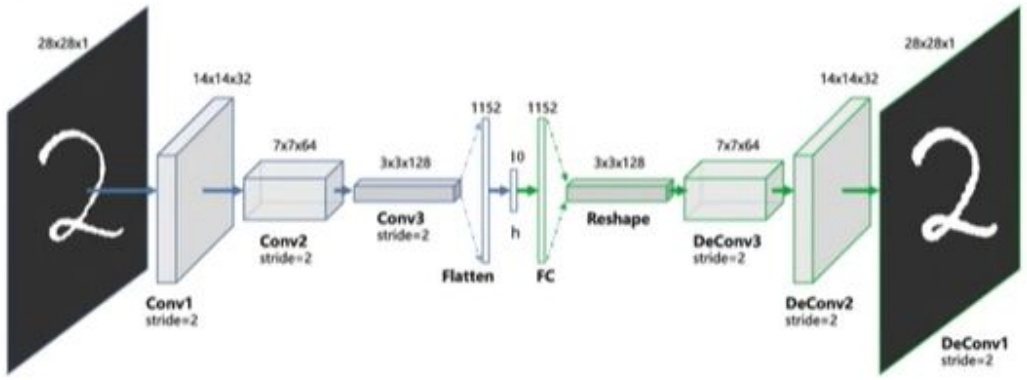


Autoencoder: Types

- Convolutional AECD
- Latent AECD
- Sparse AECD
- Deep AECD
- Contractive AECD

Convolutional AECD

Convolution Autoencoders



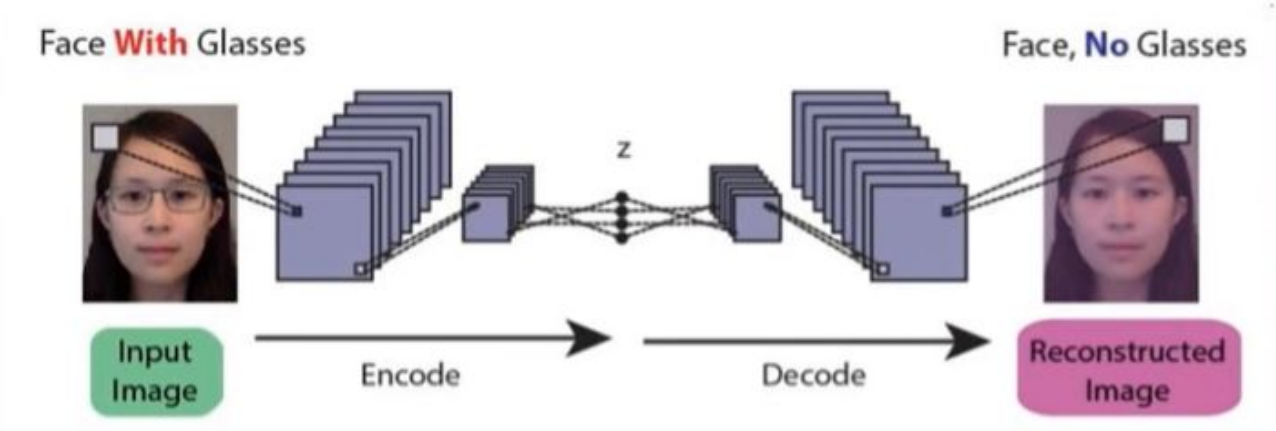
Convolution Autoencoders use the convolution operator to learn to encode the input in a set of simple signals and then try to reconstruct the input from them.

$$f(t) * g(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

Case studies of Convolutional AECD

Use Case of CAE:

1. Image Reconstruction
2. Image Colorization
3. Advanced Applications



learns to **remove noise** or **reconstruct** missing parts

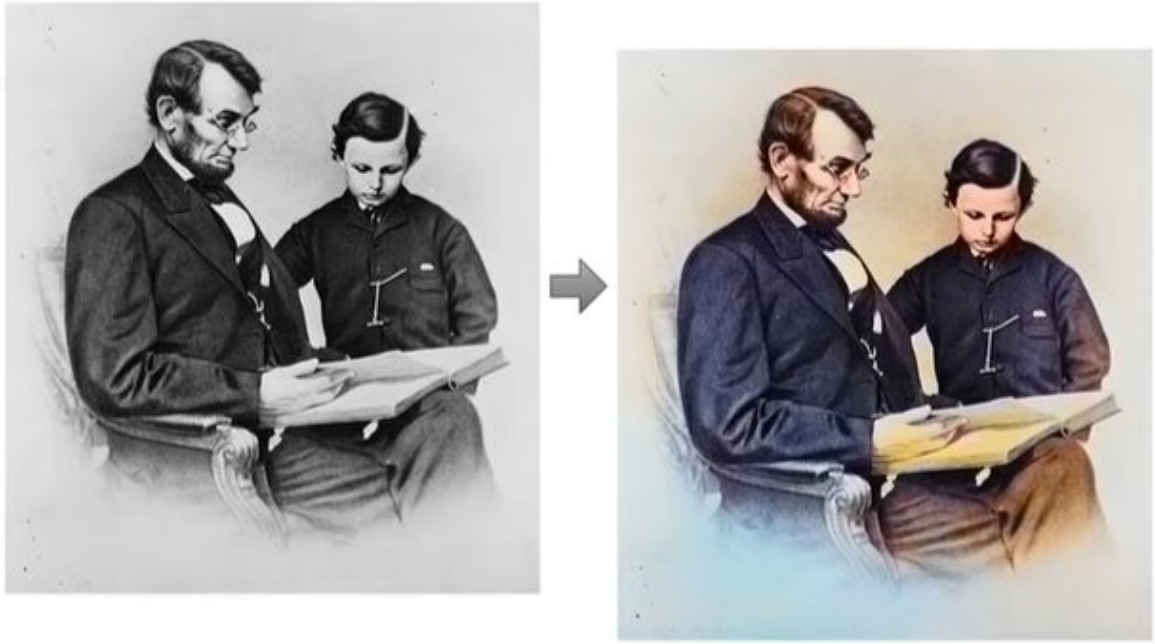
Noisy Version is converted to clean version

the network fills the gaps in the image

Case studies of Convolutional AECD

Use Case of CAE:

1. Image Reconstruction
2. **Image Colorization**
3. Advanced Applications



- maps **circles** and **squares** from an image to the same image but with Colors
- Purple is formed sometimes because of **blend** of colors, where network hesitates between circle or square.

Case studies of Convolutional AECD

Use Case of CAE:

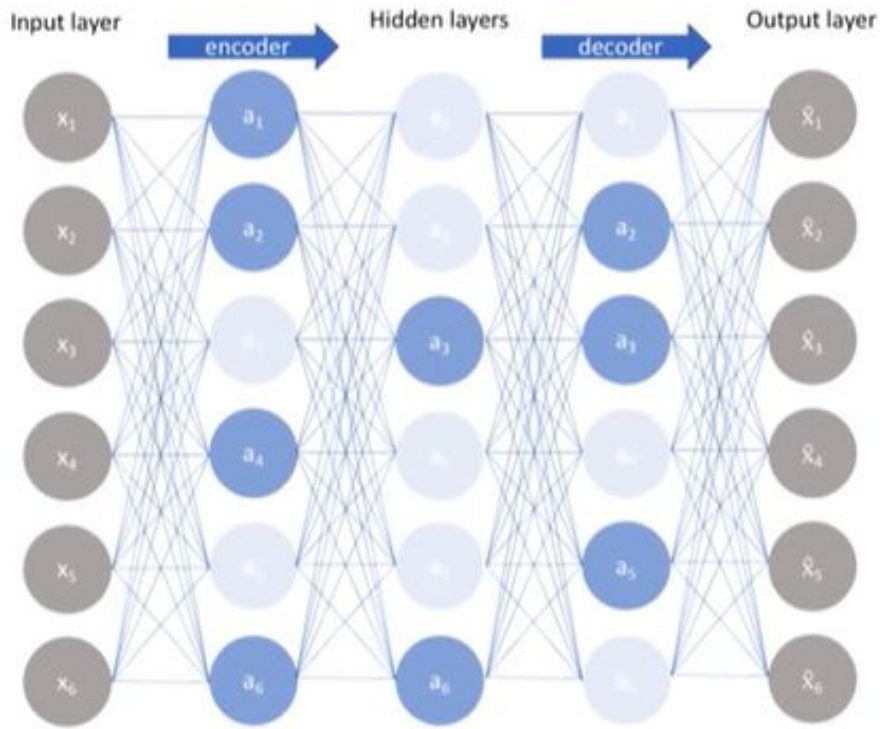
1. Image Reconstruction
2. Image Colorization
3. Advanced Applications

Fully image colorization

Latent space clustering

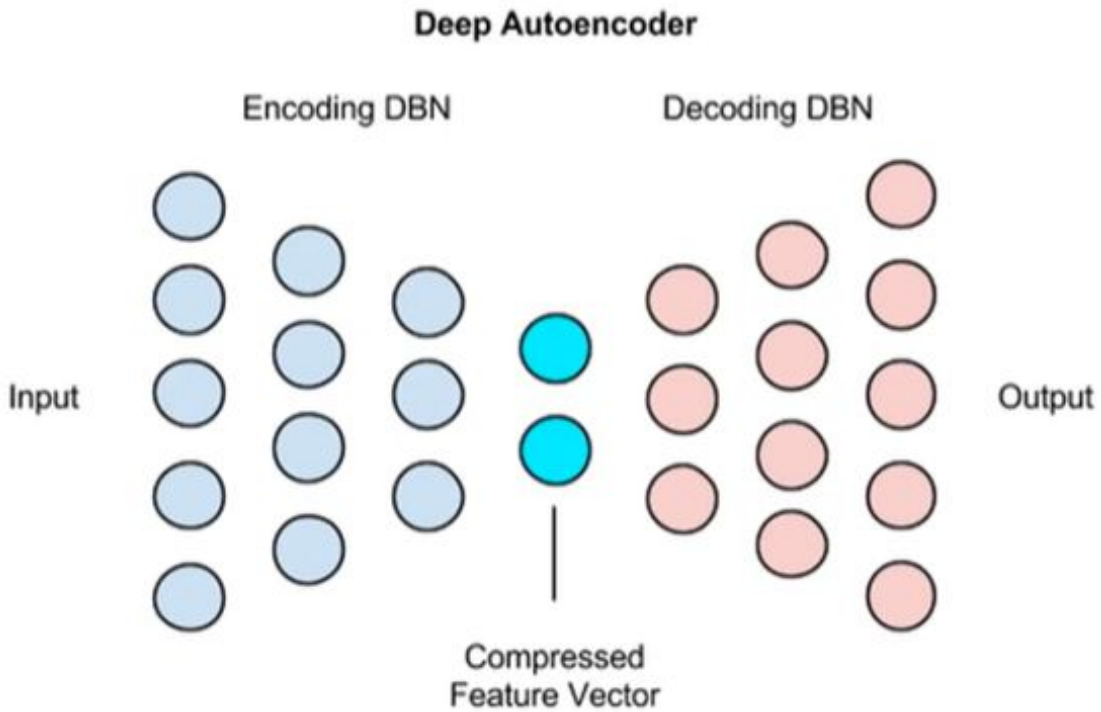
Generating higher resolution images

Sparse Autoencoders



Sparse autoencoders offer us an alternative method for introducing an information bottleneck without requiring a reduction in the number of nodes at our hidden layers

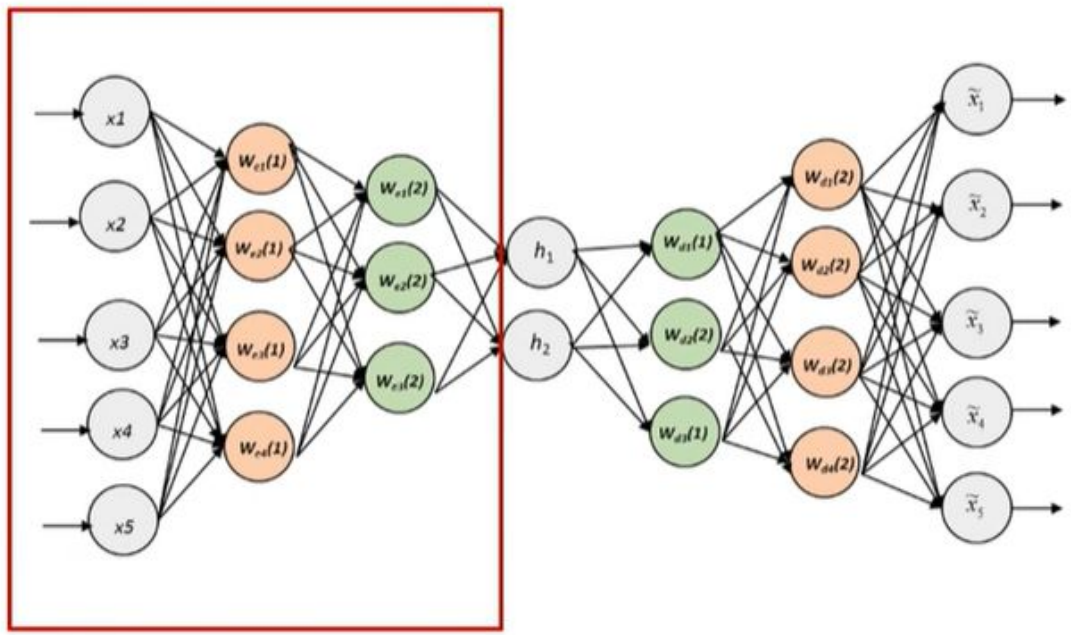
Deep Autoencoders



A ***deep autoencoder*** is composed of two, symmetrical deep-belief networks-

- First four or five shallow layers representing the encoding half of the net
- second set of four or five layers that make up the decoding half.

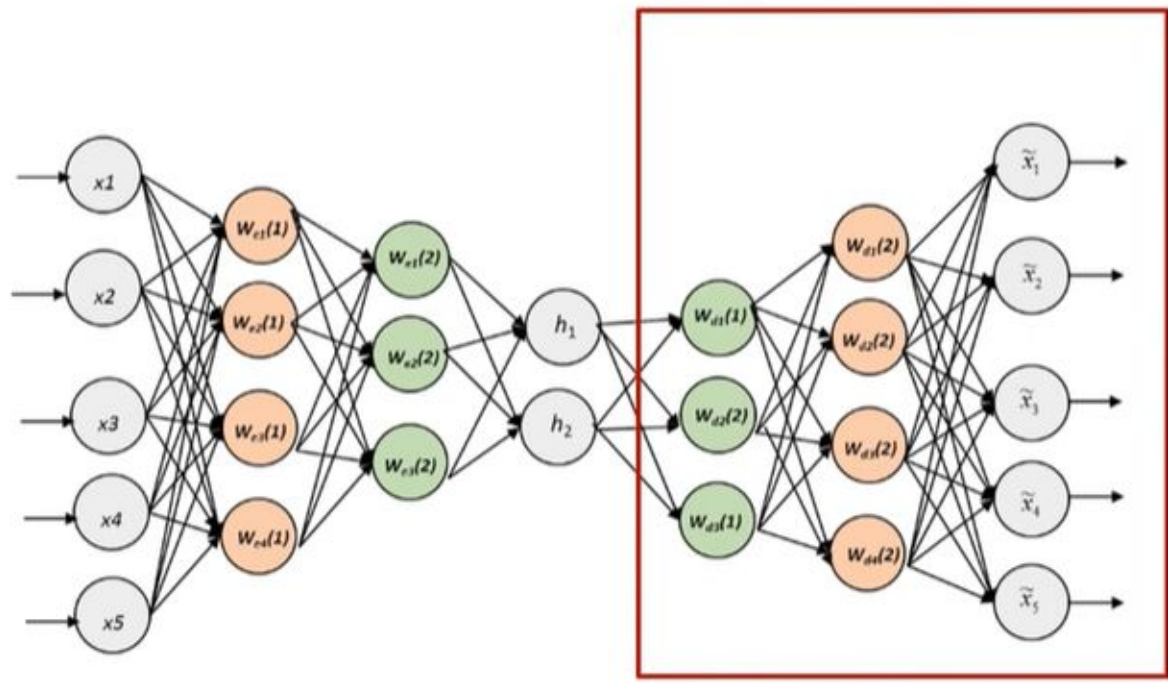
Deep Autoencoders



The first layer of the Deep Autoencoder learns first-order features in the raw input such as edges in an image

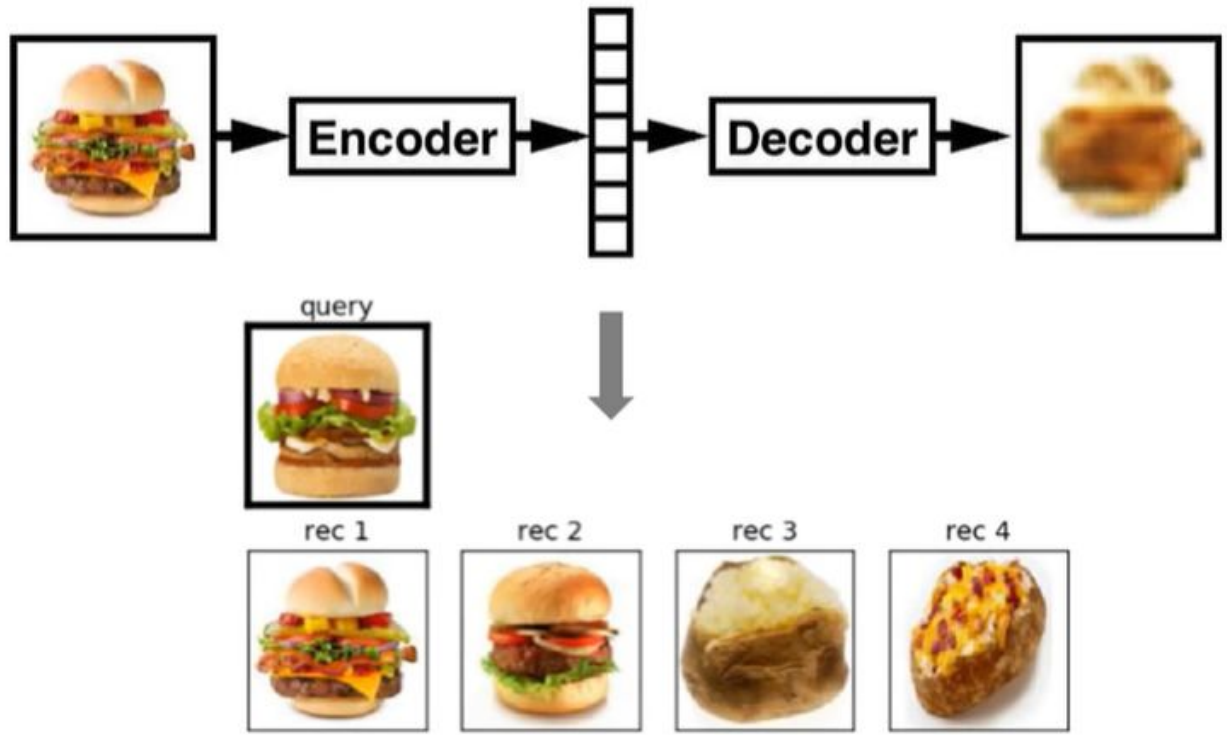
Deep Autoencoders

The second layer learns second-order features corresponding to patterns in the appearance of first-order features



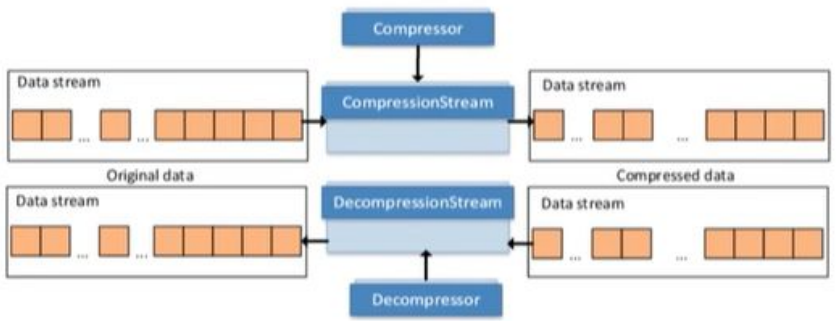
Use Case: Deep Autoencoders

1. Image Search
2. Data Compression
3. Topic Modeling & Information Retrieval



Use Case: Deep Autoencoders

1. Image Search
2. Data Compression
3. Topic Modeling & Information Retrieval

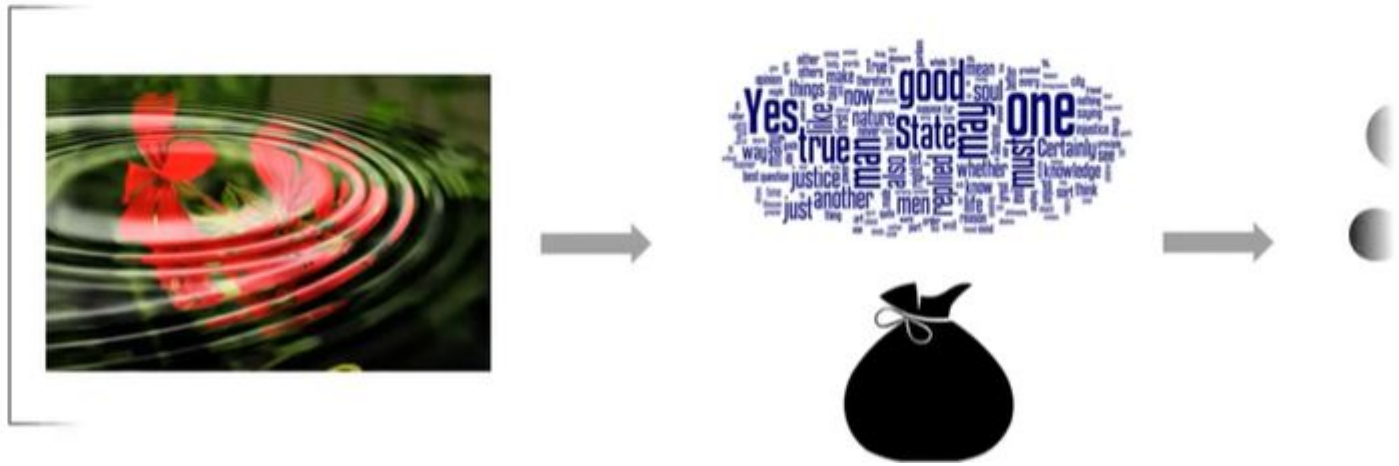


8.9M

68.34K

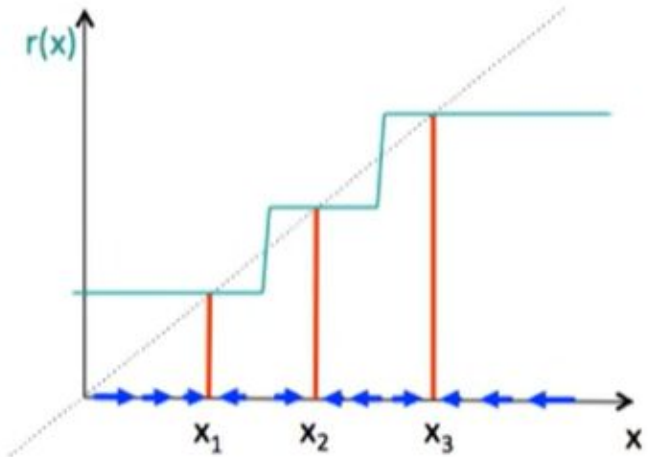
Use Case: Deep Autoencoders

1. Image Search
2. Data Compression
3. Topic Modeling & Information Retrieval



Contractive Autoencoders

Similar inputs are contracted to a constant output within a neighborhood, based on what the model observed during training



Training observations

Learned reconstruction function

Linear identity function (perfect reconstruction)

A ***contractive autoencoder*** is an unsupervised deep learning technique that helps a neural network encode unlabeled training data.

References

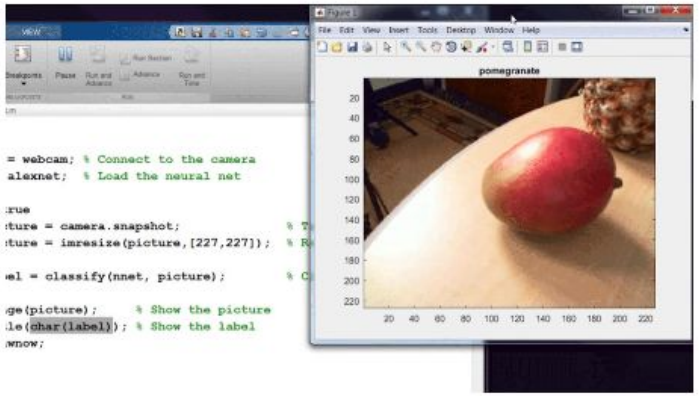
- https://www.youtube.com/watch?v=nTt_ajul8NY&t=616s

Applications of Deep Learning.

Use AlexNet to Recognize Objects with Your Webcam

Use MATLAB, a simple webcam, and a deep neural network to identify objects in your surroundings.

» Learn more



Example: Object Detection Using Deep Learning

In addition to object recognition, which identifies a specific object in an image or video, deep learning can also be used for **object detection**. Object detection algorithms like YOLO can recognize and locate the object in a scene, and can locate multiple objects within the image.

Top Applications of Deep Learning Across Industries 2022

1. Self Driving Cars
2. News Aggregation and Fraud News Detection
3. Natural Language Processing
4. Virtual Assistants
5. Entertainment
6. Visual Recognition
7. Fraud Detection
8. Healthcare
9. Personalisation's
10. Detecting Developmental Delay in Children
11. Colourisation of Black and White images
12. Adding sounds to silent movies
13. Automatic Machine Translation
14. Automatic Handwriting Generation
15. Automatic Game Playing
16. Language Translations
17. Pixel Restoration
18. Photo Descriptions
19. Demographic and Election Predictions
20. Deep Dreaming


1 Self Driving Cars
2 Entertainment
3 Visual Recognition
4 Virtual Assistants
5 Fraud Detection
6 Natural Language Processing
7 News Aggregation and Fraud News Detection
8 Detecting Developmental Delay in Children
9 Colourisation of Black and White images
10 Adding sounds to silent movies
Healthcare 11
Personalisations 12
Automatic Machine Translation 13
Automatic Handwriting Generation 14
Demographic & Election Predictions 15
16 Automatic Game Playing
17 Language Translations
18 Pixel Restoration
19 Photo Descriptions
20 Deep Dreaming

Self-learning Topics: Restricted Boltzmann Machine (RBM).

References

- https://www.mygreatlearning.com/blog/deep-learning-applications/