Deep Learning

ML

No Brain activity

DL

Full Brain activity

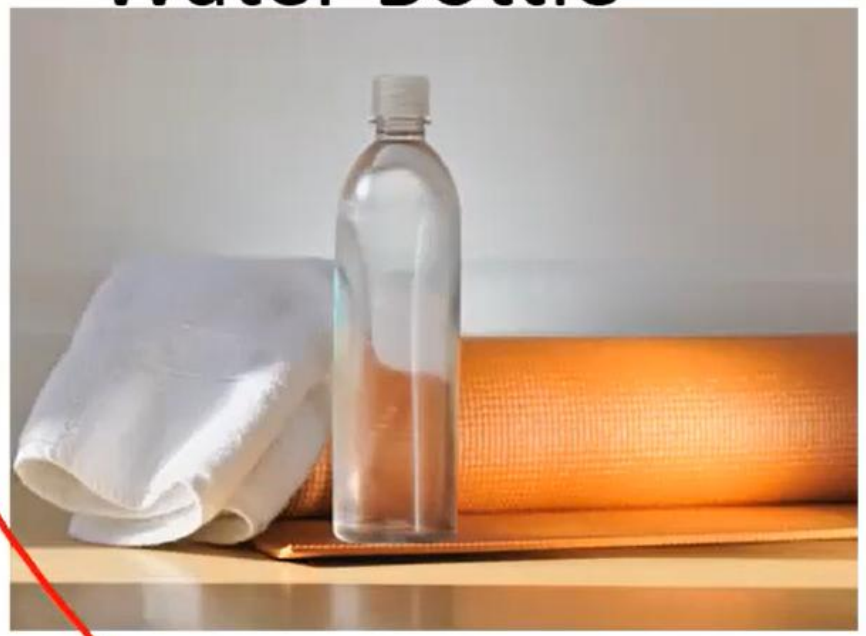
NN

Traditional Vs Artificial Intelligence

Traditional Method

Water Bottle

Water Bottle



Water Bottle

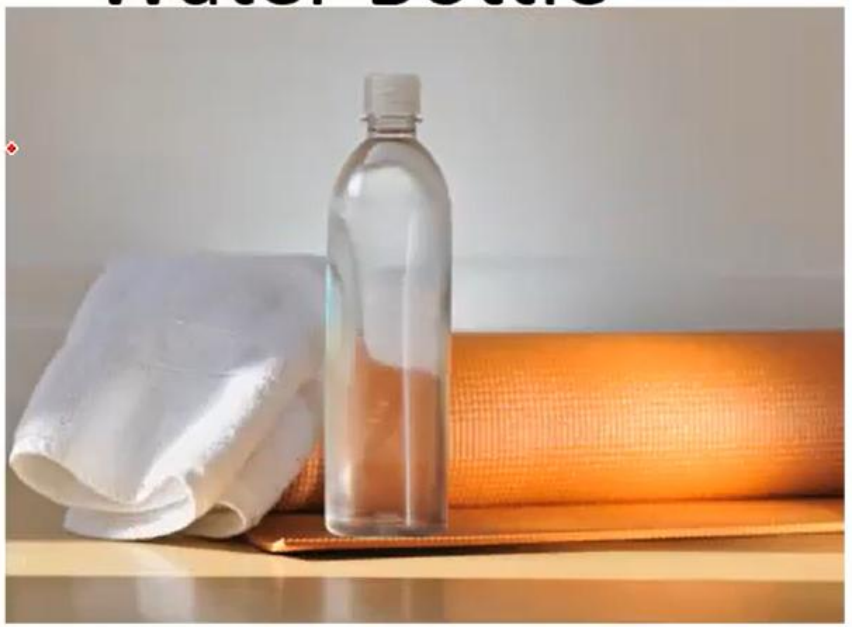
Water Bottle

Traditional Vs Artificial Intelligence

Learning Method

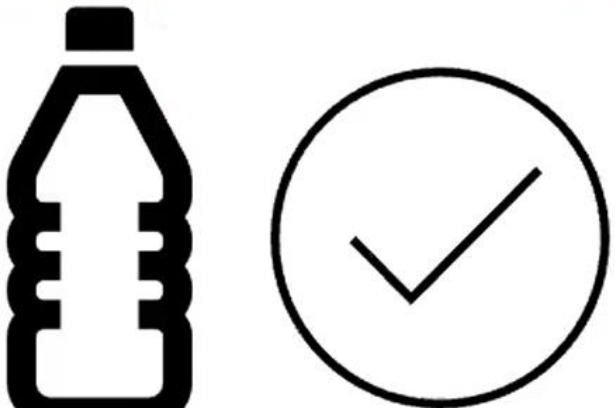
Water Bottle

Water Bottle



Water Bottle

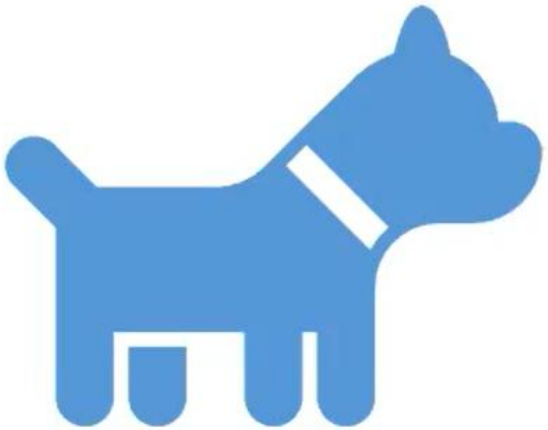
Water Bottle



Traditional Vs Artificial Intelligence

Traditional Method

Dog



Dog

Dog

Dog

Dog

Traditional Vs Artificial Intelligence

Learning Method

Dog

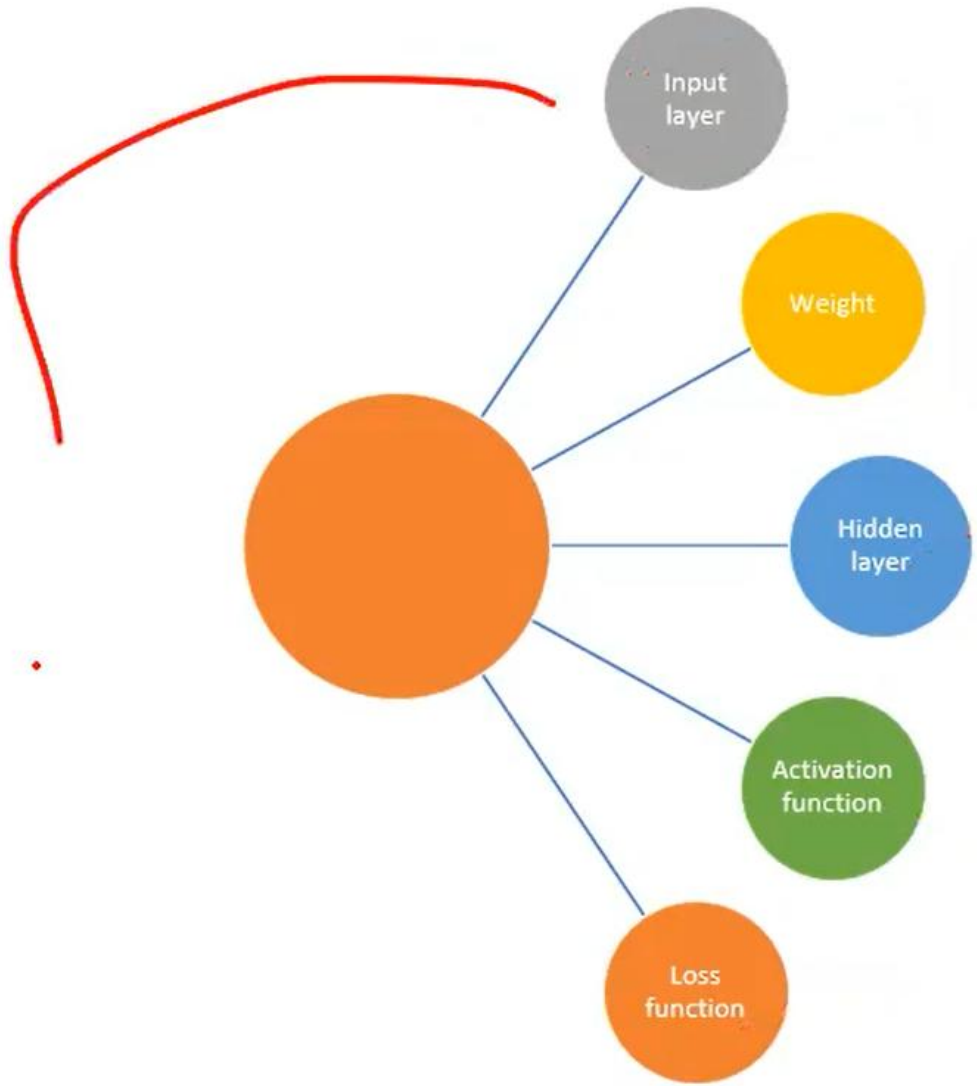
Dog

Dog

Dog

Dog

Artificial Neural Network



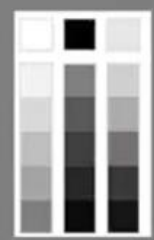
How computer understands the image

Types of Image

Black & White Image

Black : 0
White: 1

Gray Scale Image



Black : 0
White: 255

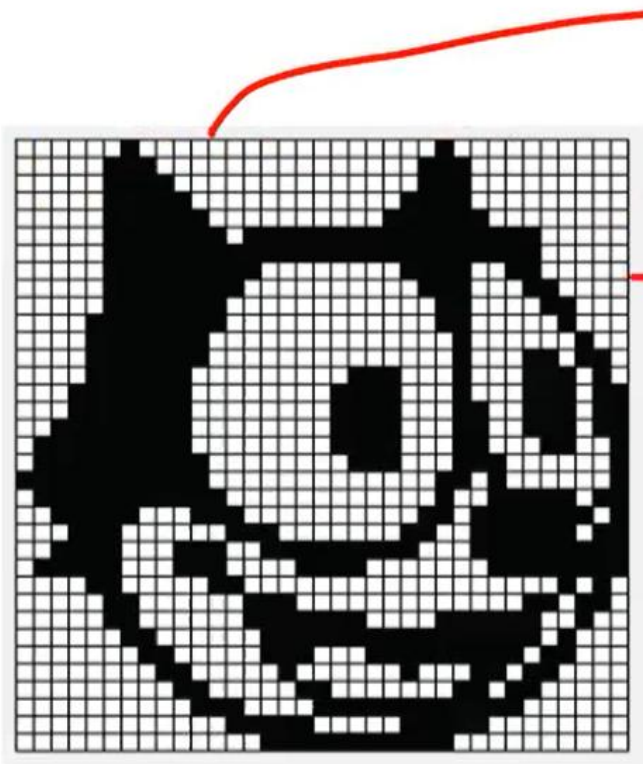
Color Image(BGR)

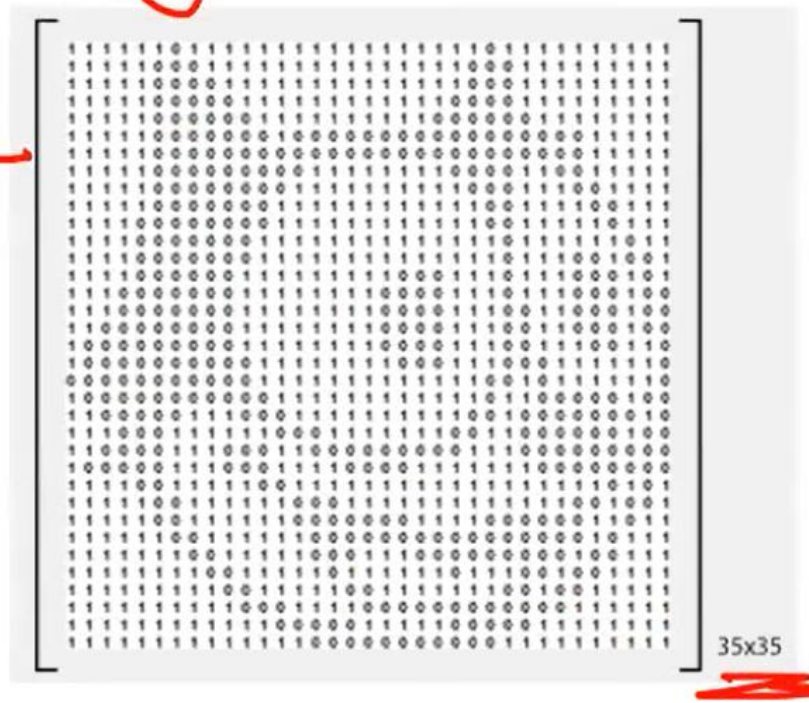
Red: 0 - 255
Blue: 0 - 255
Green: 0 - 255

How computer understands the image

Black & White Image

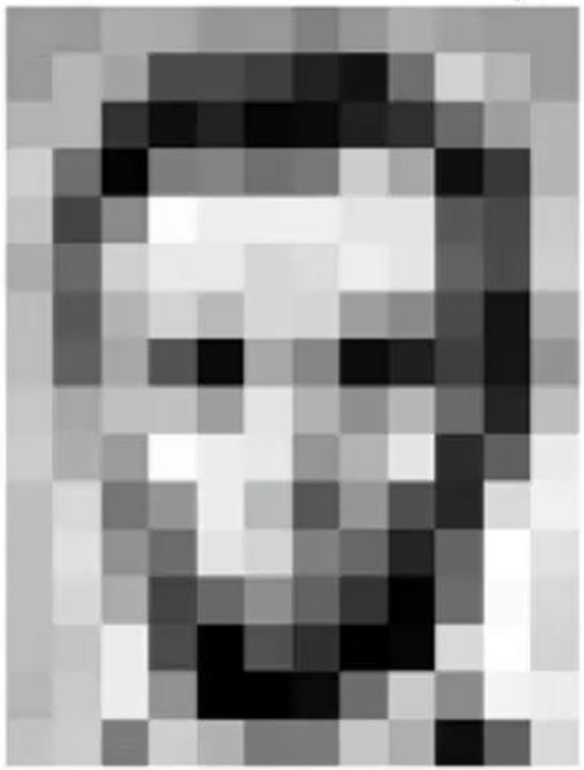
Matrix

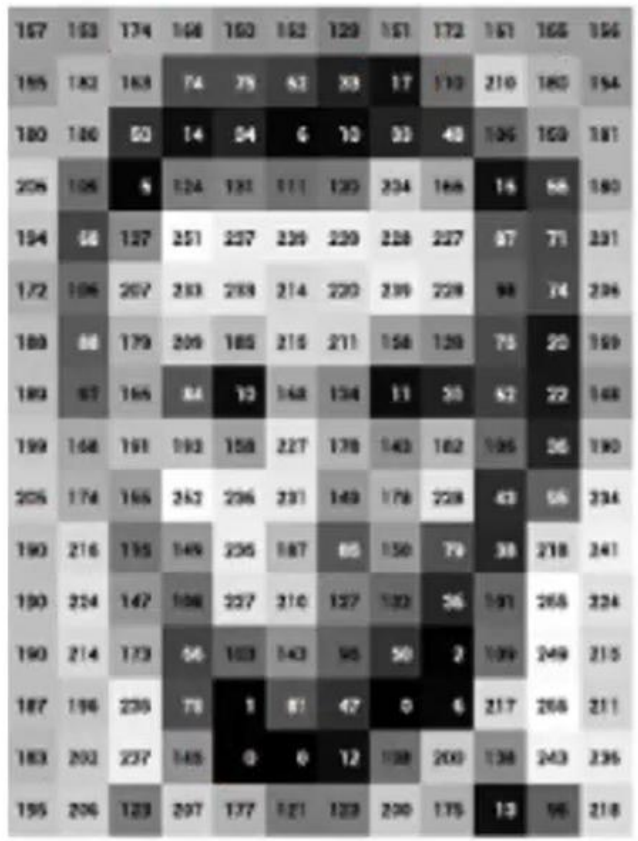


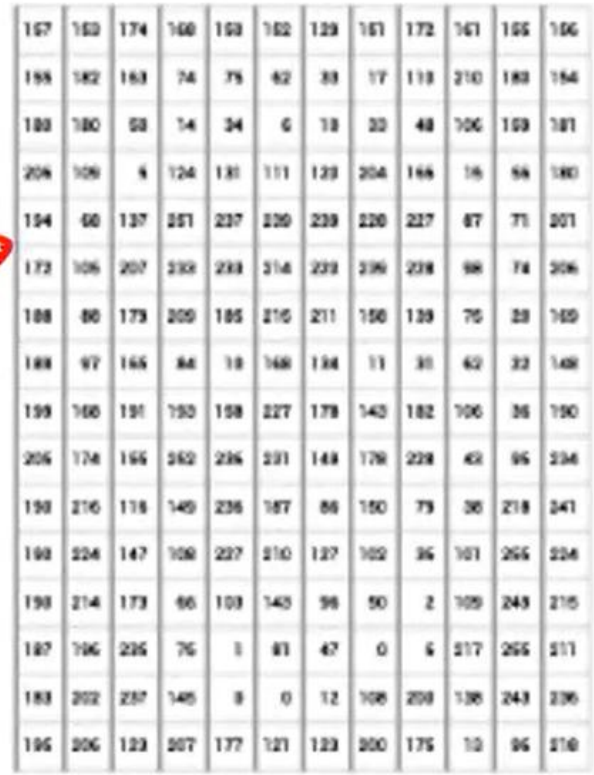


How computer understands the image

Gray Scale Image







How computer understands the image

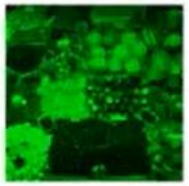
Color Image

Black-Red dark to - white

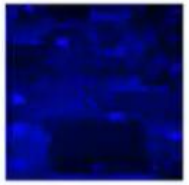
RGB 

R 

Red: 0 - 255

G 

Green: 0 - 255

B 

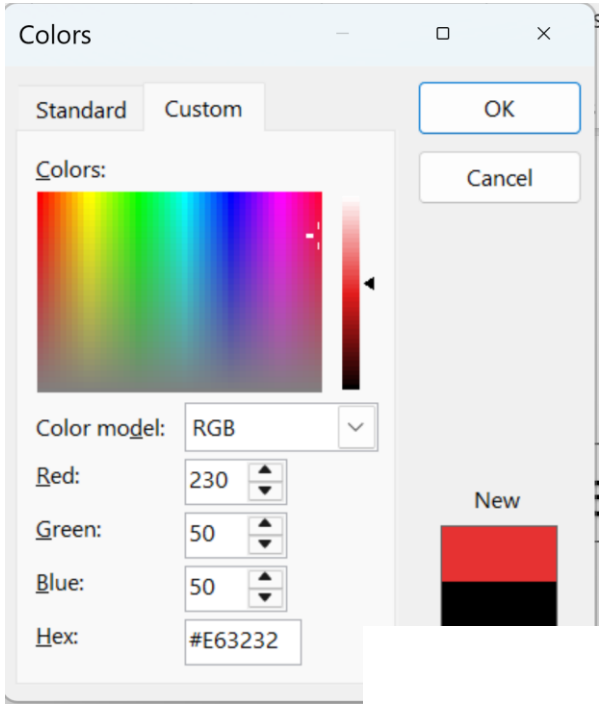
Blue: 0 - 255

Red 

Green 

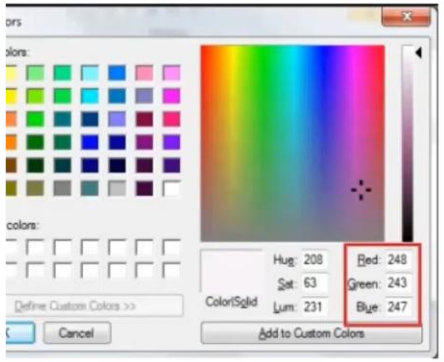
Blue 

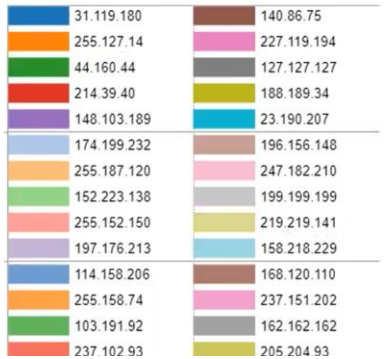
RGB 

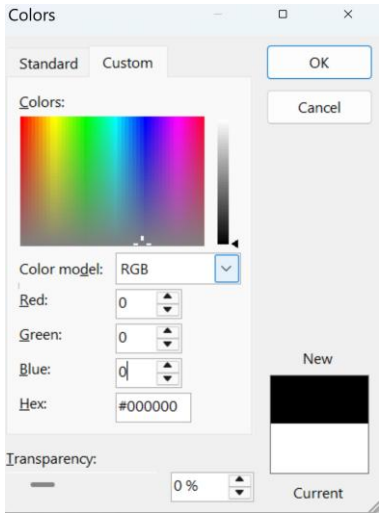


How computer understands the image

Color Image

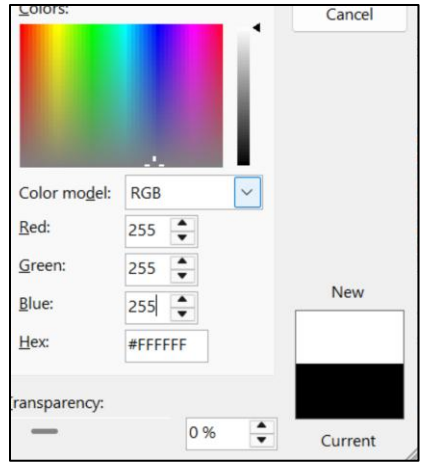


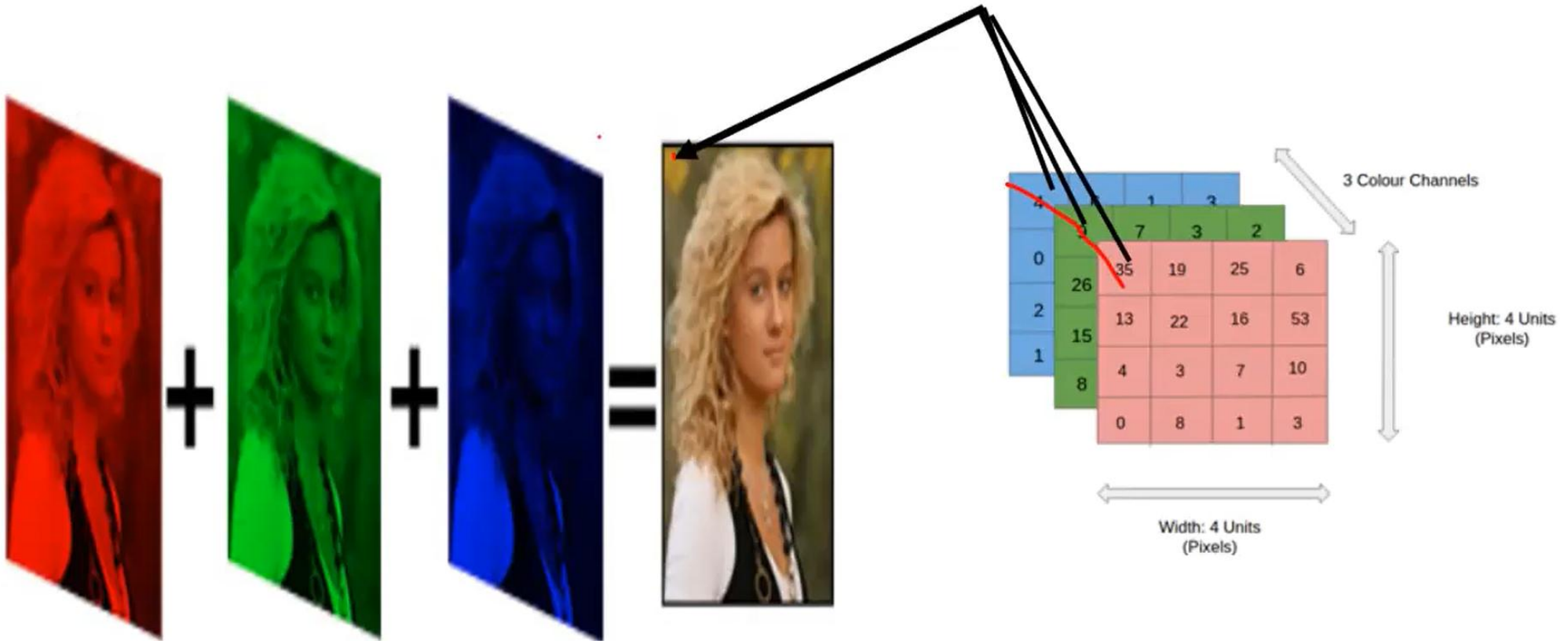




Colors on the color wheel created using combinations of **RGB** values

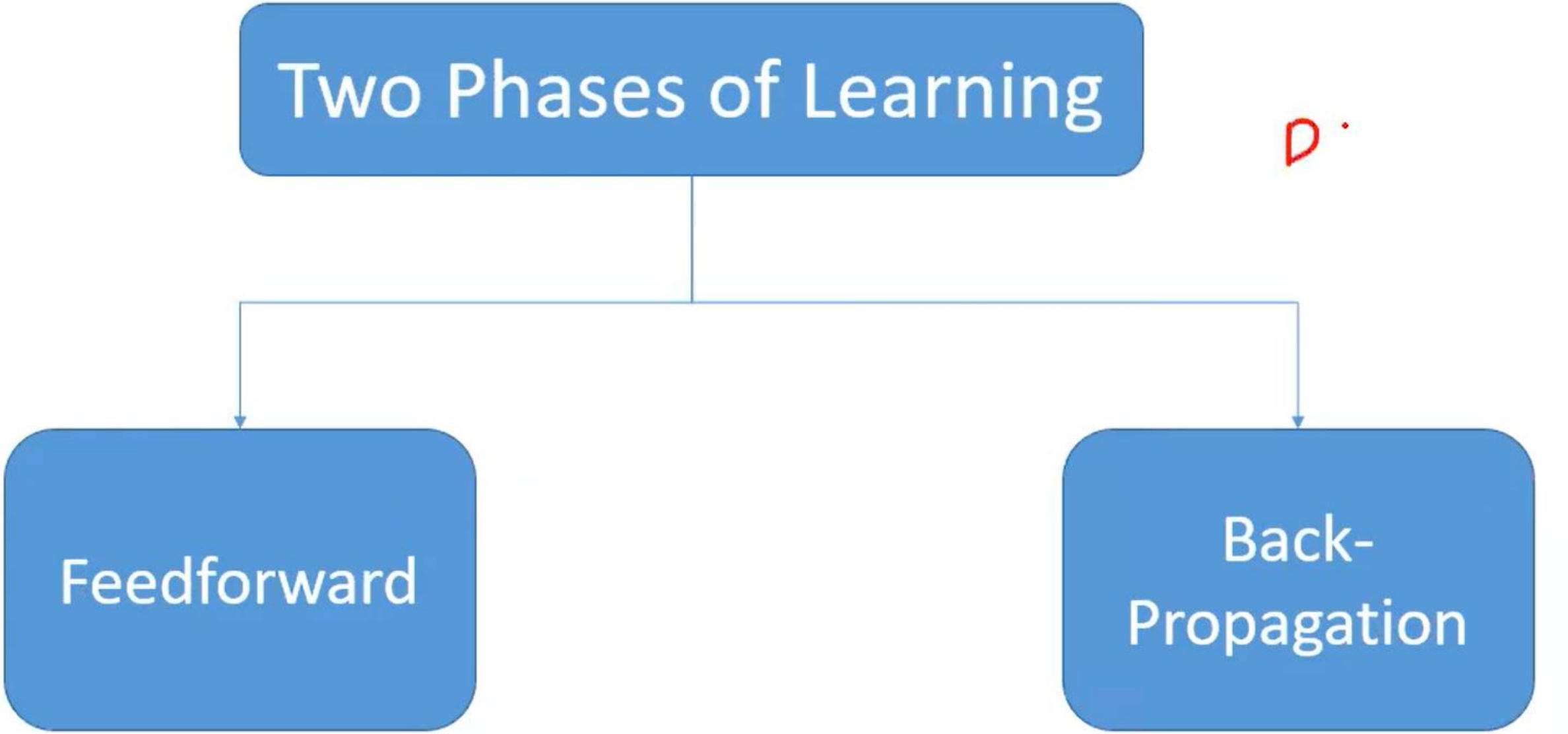
R	255	255	255	125	0	0	0	0	0	125	255	255
G	0	125	255	255	255	255	255	125	0	0	0	0
B	0	0	0	0	0	125	255	255	255	255	255	125





Row x Column x Channel

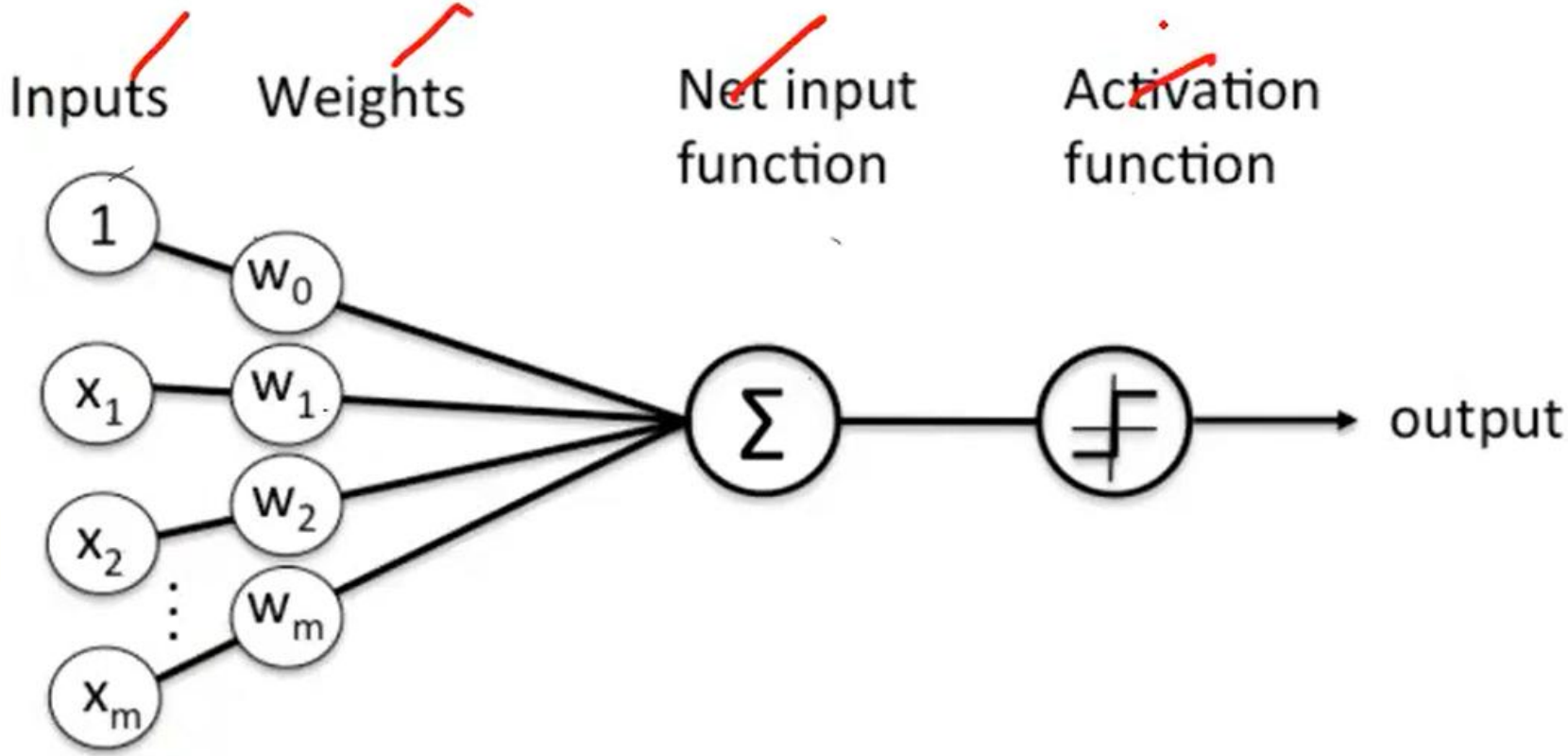
Row x Column x Channel 33 x 35 x 3



Linear Algebra, net input function, activate function(Relu)

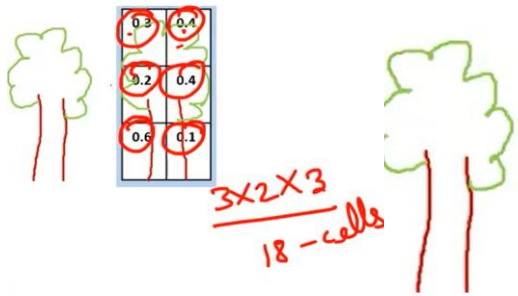
Chain rule – loss fuction (Loss and cost function)

Single Perceptron



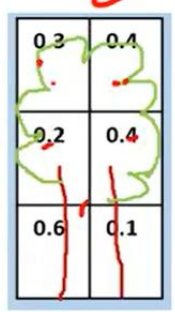
Schematic of Rosenblatt's perceptron.

ANN
Feed Forward

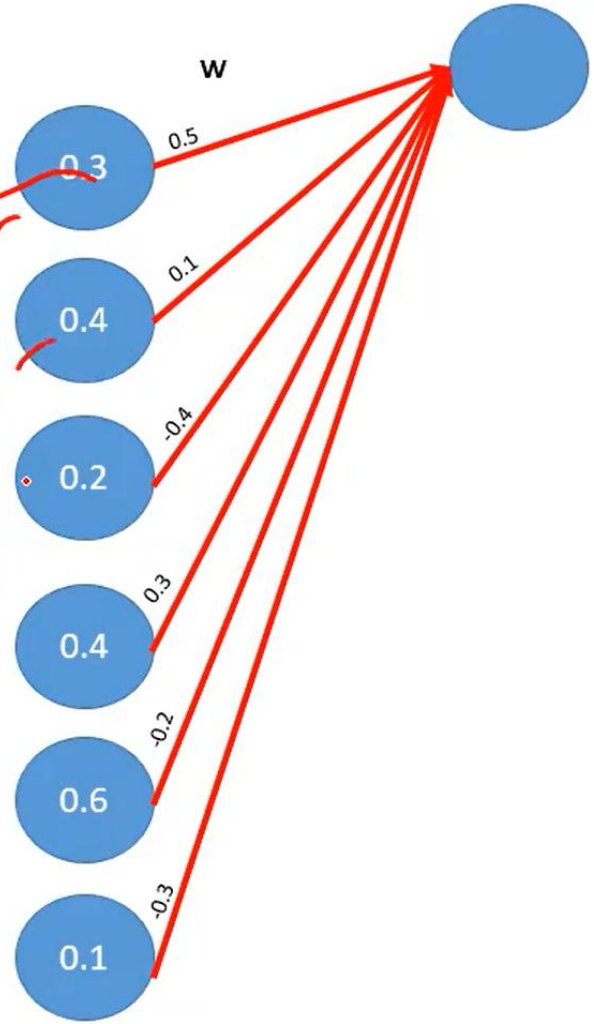


3x2x3
18 - cells

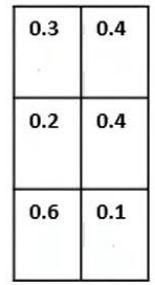
ANN
Feed Forward
i



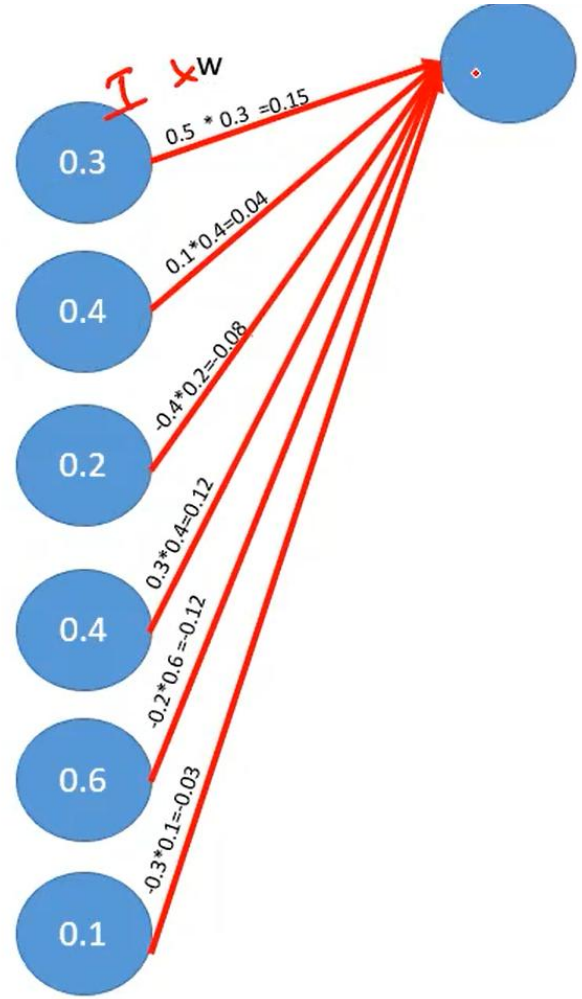
w

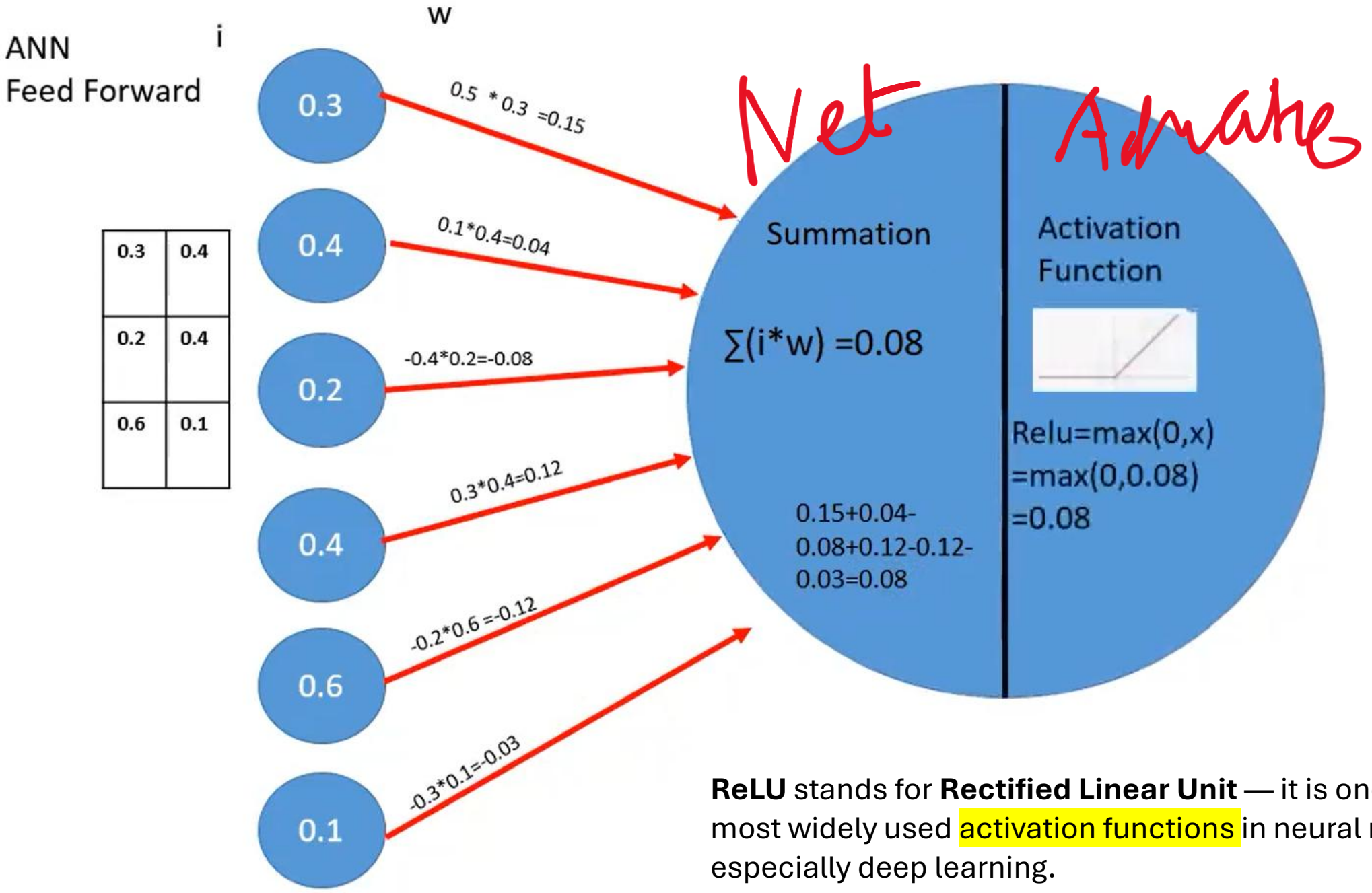


ANN
Feed Forward
i



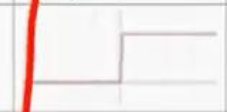
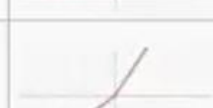
I x w

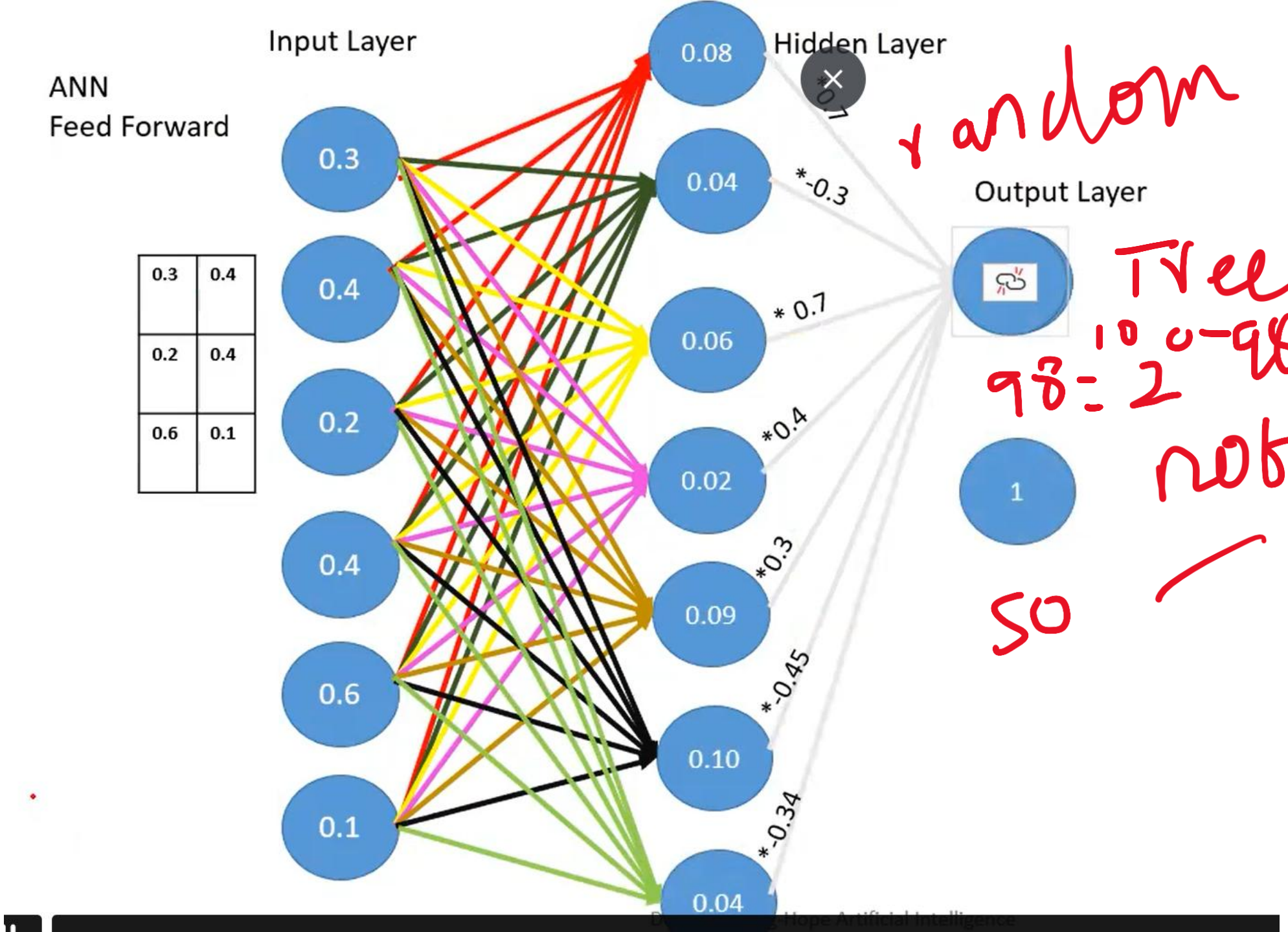




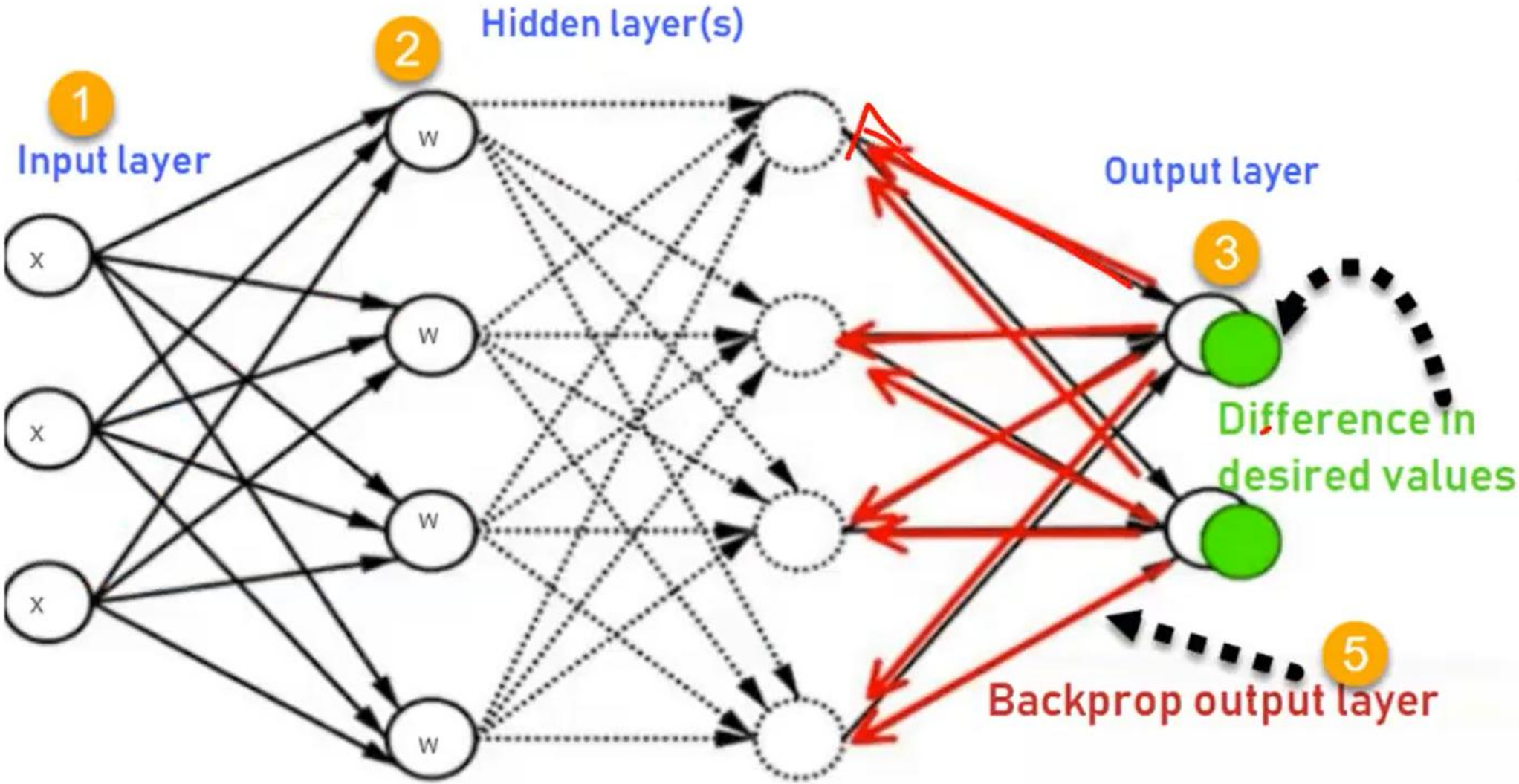
ReLU stands for **Rectified Linear Unit** — it is one of the most widely used activation functions in neural networks, especially deep learning.

Types of Activation Function

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

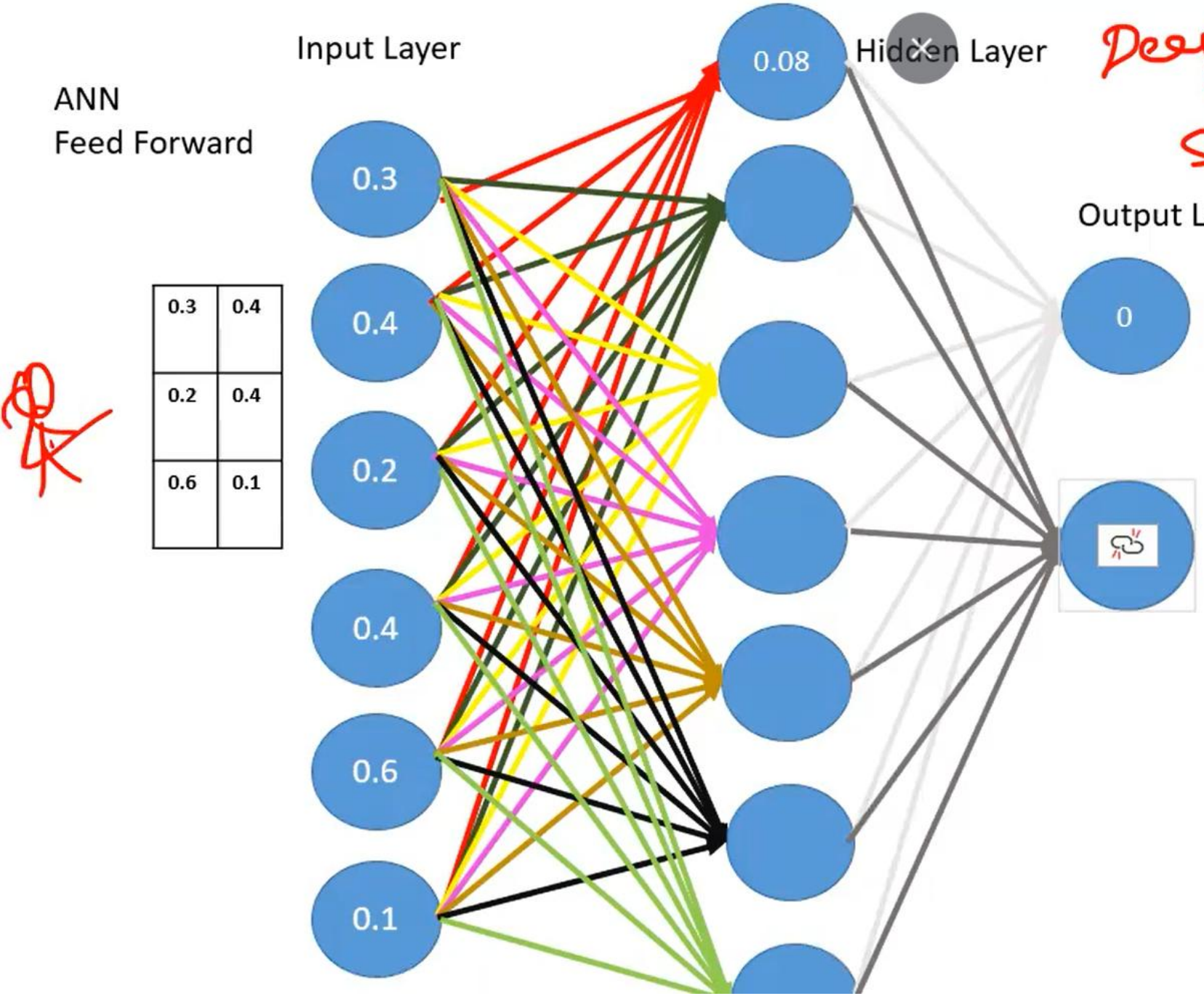

ANN
Feed Forward
Input Layer
Hidden Layer
Output Layer
0.3
0.4
0.2
0.4
0.6
0.1
0.08
0.04
0.06
0.02
0.09
0.10
0.04
1
*-0.1
*-0.3
* 0.7
*0.4
*0.3
*-0.45
*-0.34
0.3 0.4
0.2 0.4
0.6 0.1
random
Tree
98 = 2^10 - 98
not
so

Back Propagation



How Backpropagation Works

ANN
Feed Forward

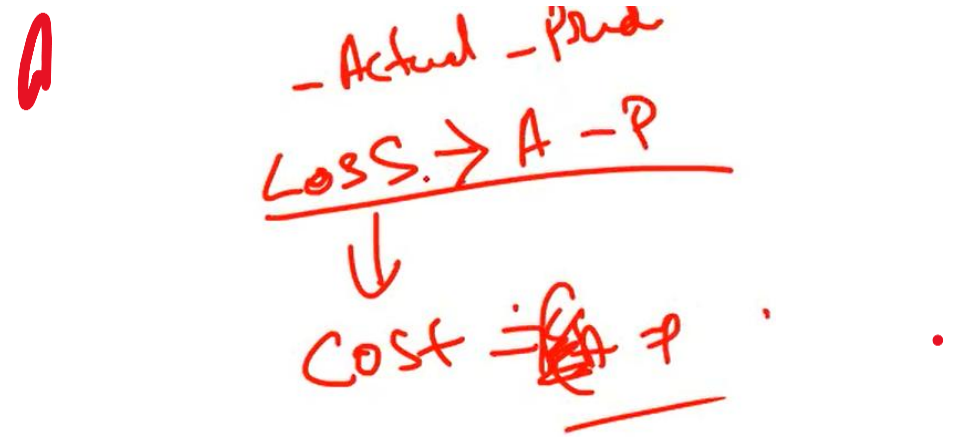


0.3	0.4
0.2	0.4
0.6	0.1

Deep learn
supervised → I/O
Loss fun-
cost func

} Loss

$$cost = \frac{Loss_1 + 2 + 4 + \dots}{3}$$

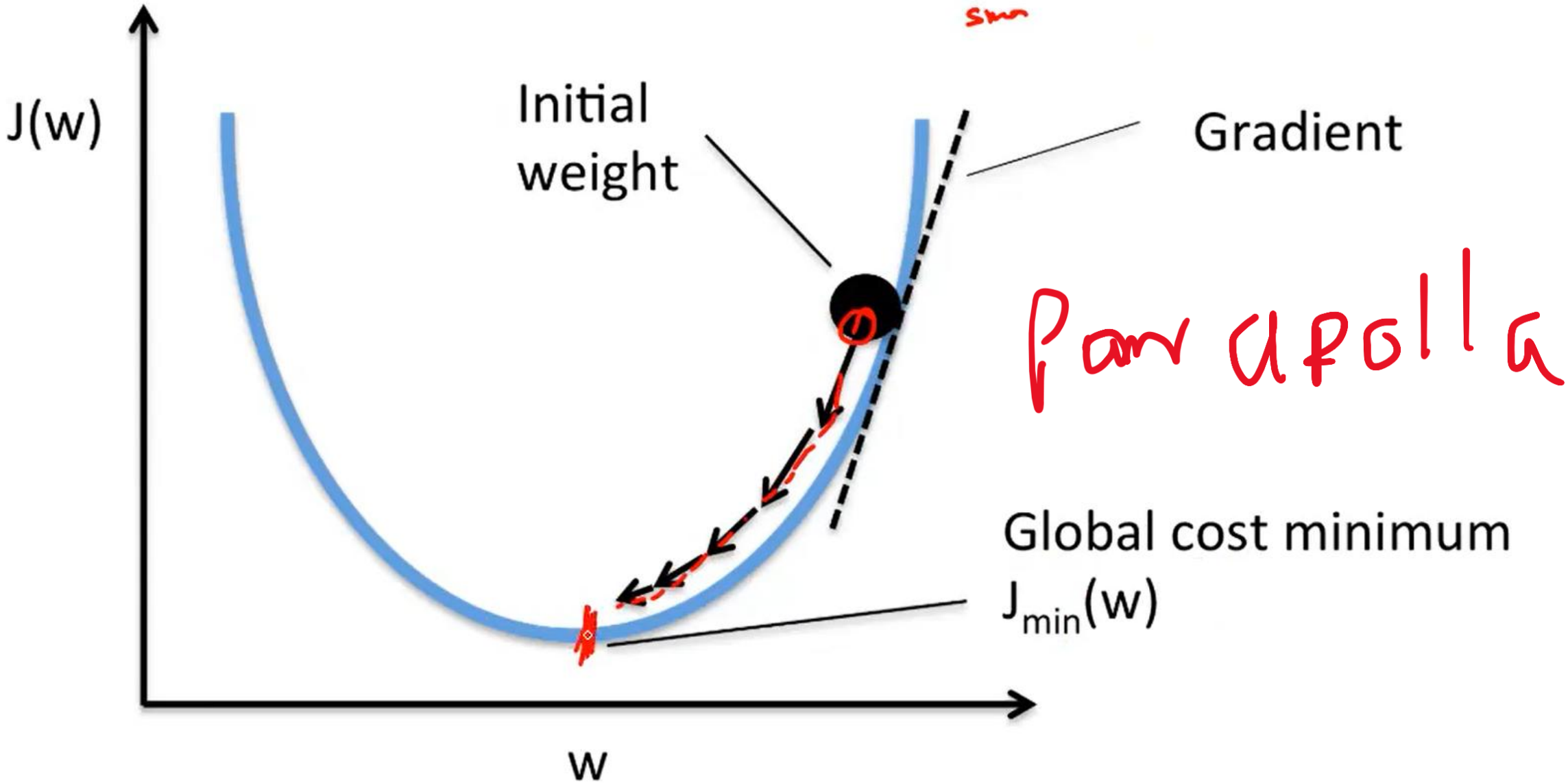


🔥 **Example:**
Actual value = **1** (cat)
Model predicted = **0.3**
Loss = **difference → how bad the prediction is**
💡 **Simple definition:**
Loss function = Error for one prediction.

⭐ **Cost Function — Easy Explanation**
The **cost function** is the **average loss** for the **entire training dataset**.
Think of it like:
•**Loss** = error for one example
•**Cost** = average error for all examples
🔥 **Example:**
If you train on 1,000 images:
•Each image gives a **loss**
•All losses together → **cost function**
💡 **Simple definition:**
Cost function = Total/average error of all predictions in training.

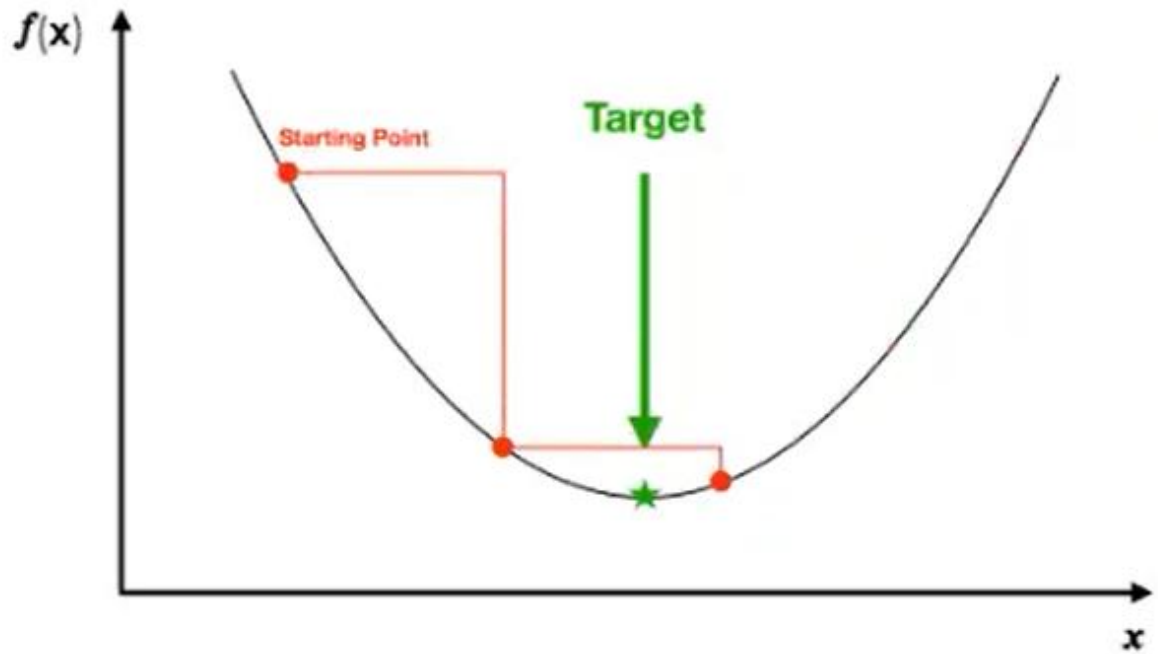
Term	Easy Meaning	Example
Deep Learning	Model with many layers learning patterns automatically	Identifying cats in images
Loss Function	Error for one prediction	1 − 0.3 = loss
Cost Function	Average of all losses	Mean error for all 1000 images

Learning Rate | Gradient Descent



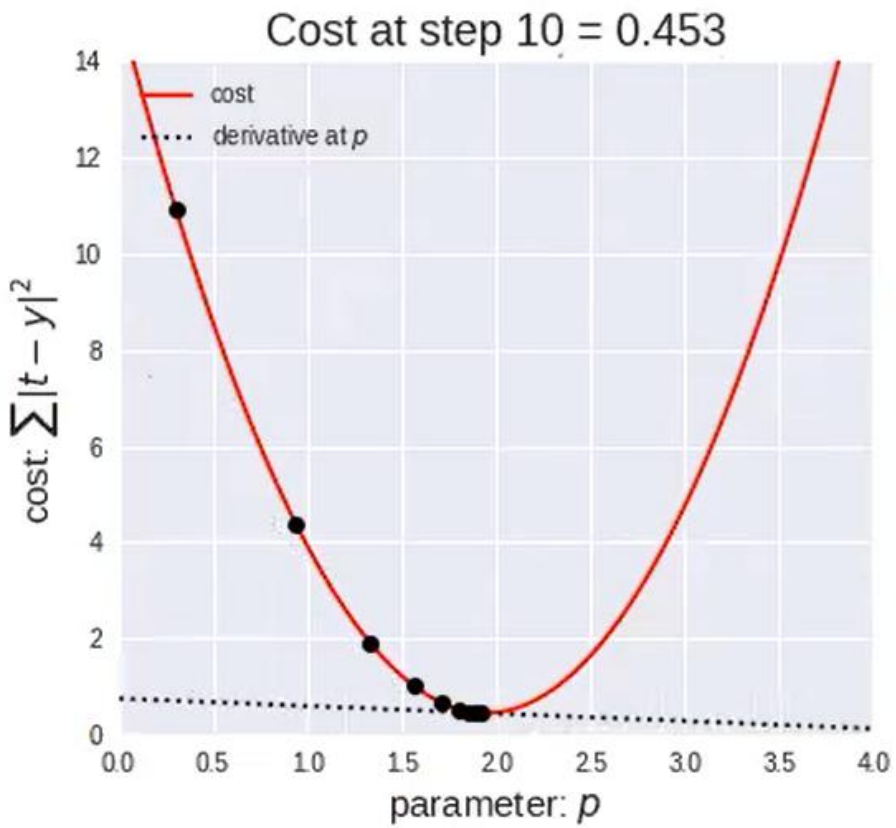
Learning Rate | Gradient Descent

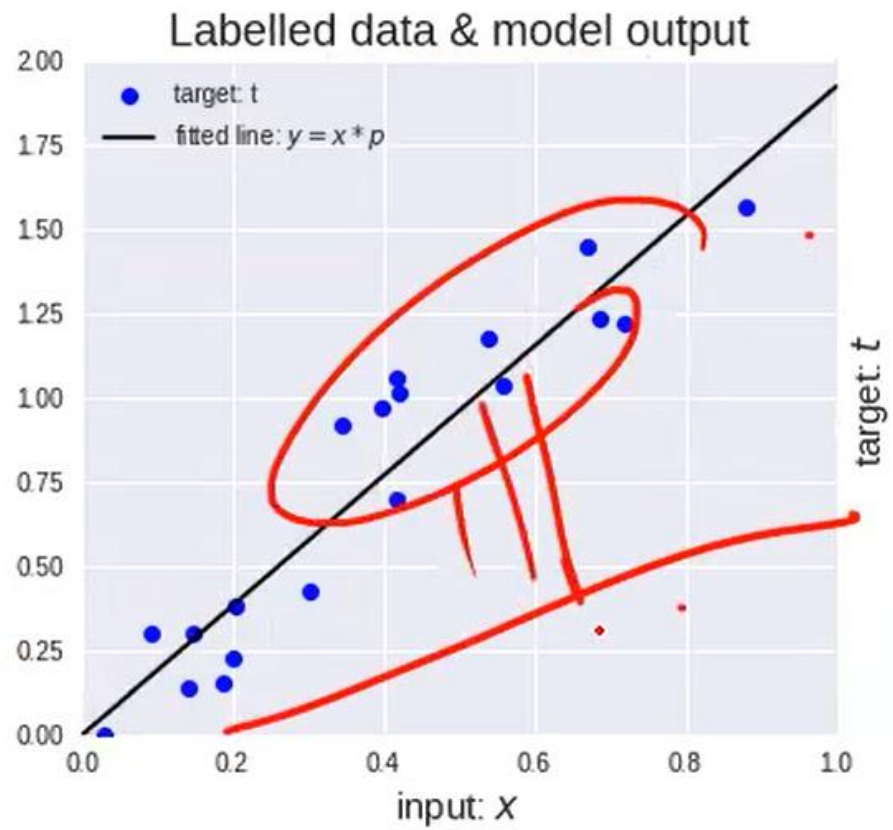
Learning rate too slow



Learning rate too fast

Learning Rate | Gradient Descent





Epoch | Batch Size | Iteration

Epoch

One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.

Epoch | Batch Size | Iteration

Batch Size

Total number of training examples present in a single batch.

(64) Neural networks - YouTube

youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

YouTube IN

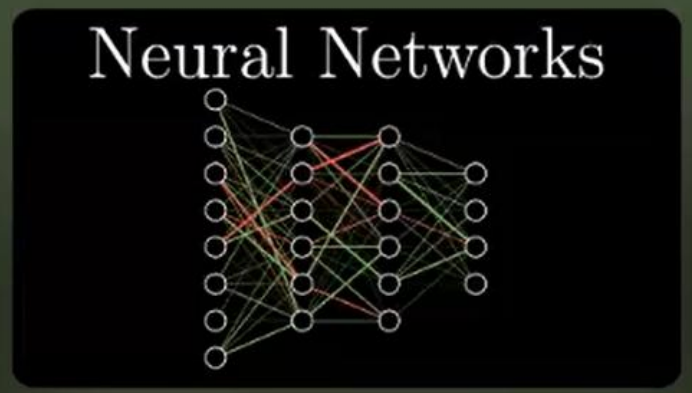
3 blue 1 brown

Home

Shorts

ubscriptions

Library



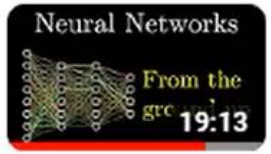
Neural networks

3Blue1Brown

4 videos Last updated on Aug 1, 2018

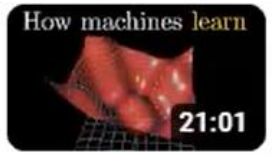
Play all

Season 3

1 

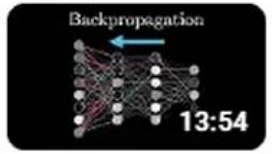
But what is a neural network? | Chapter 1, Deep learning

3Blue1Brown

2 

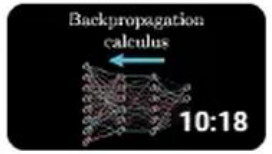
Gradient descent, how neural networks learn | Chapter 2, Deep learning

3Blue1Brown

3 

What is backpropagation really doing? | Chapter 3, Deep learning

3Blue1Brown

4 

Backpropagation calculus | Chapter 4, Deep learning

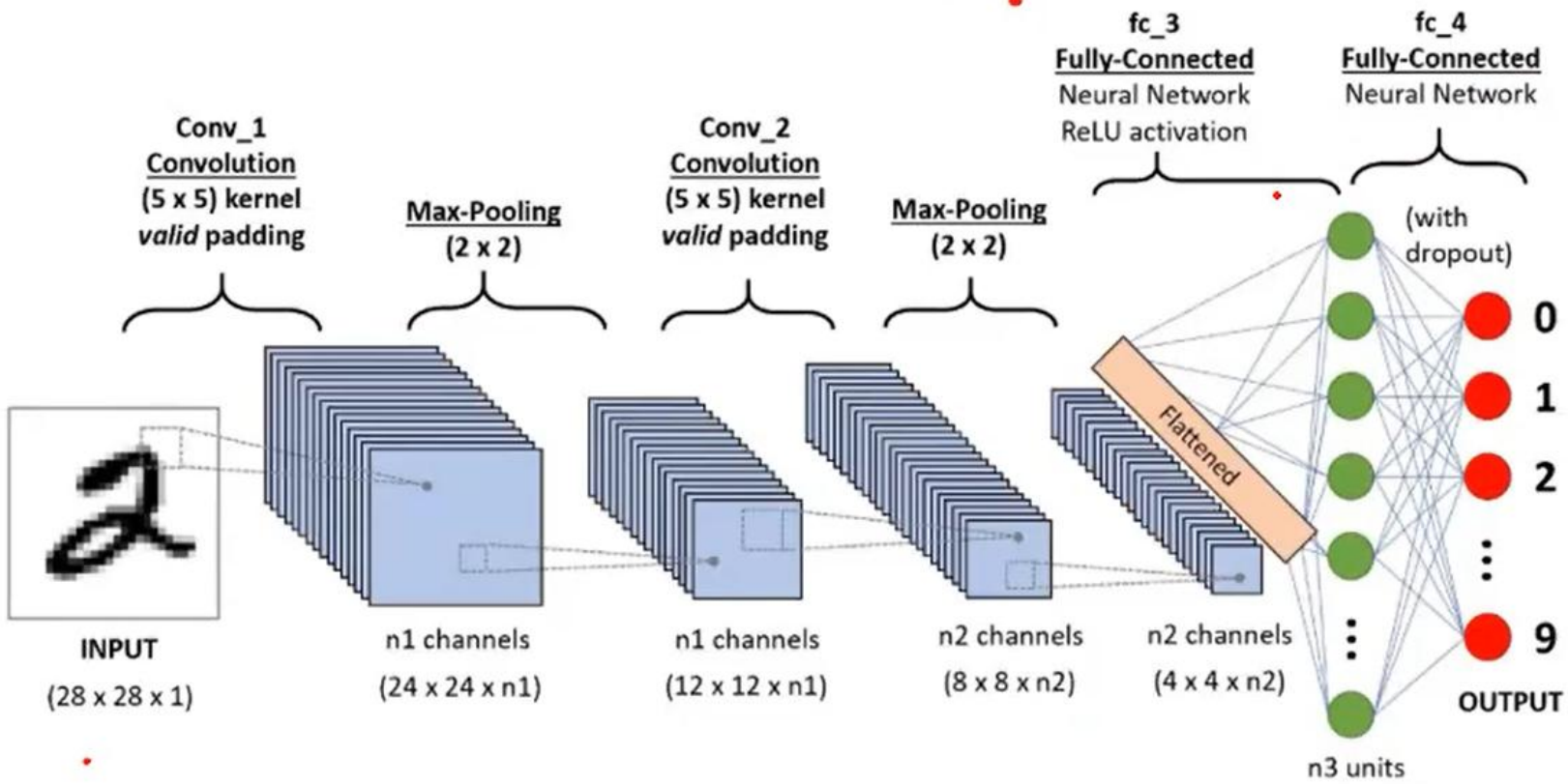
3Blue1Brown

Convolutional Neural Network

BY

Ramisha Rani Aravind

Common Architectures In Convolutional Neural Networks

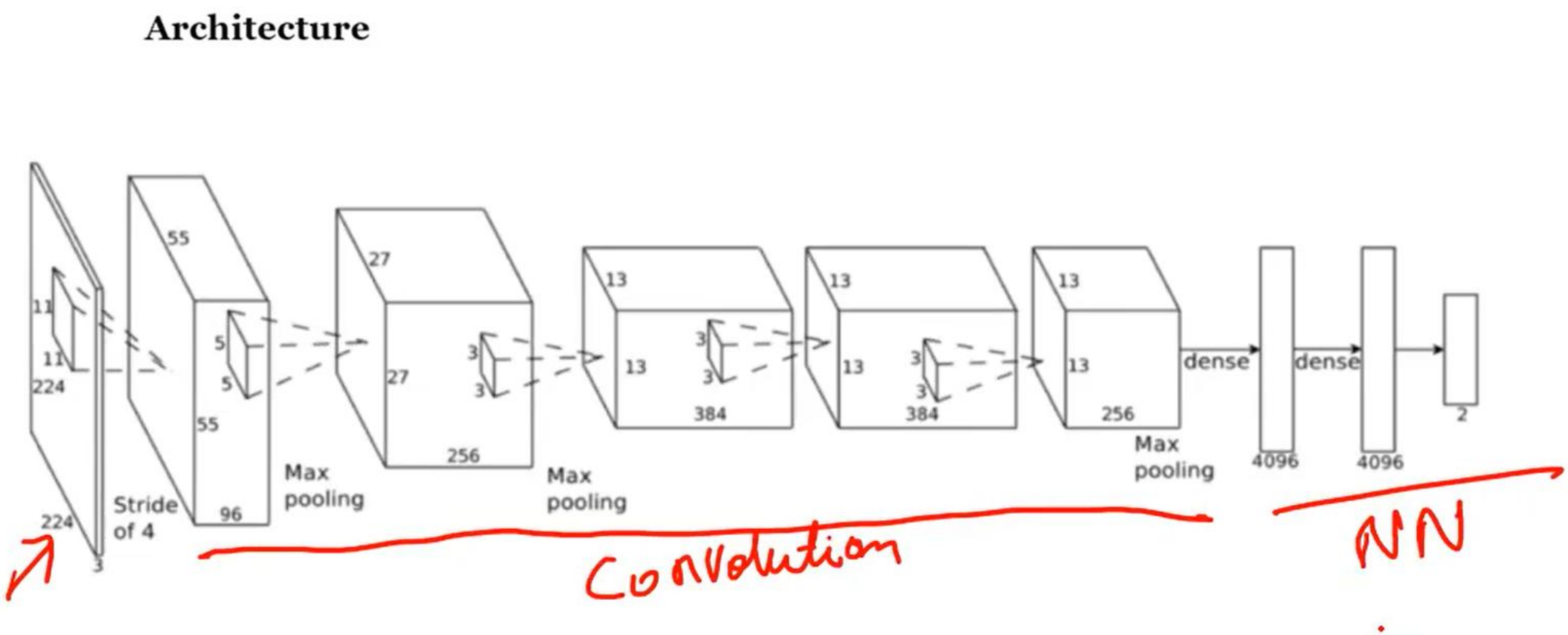


A CNN sequence to classify handwritten digits

Image source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

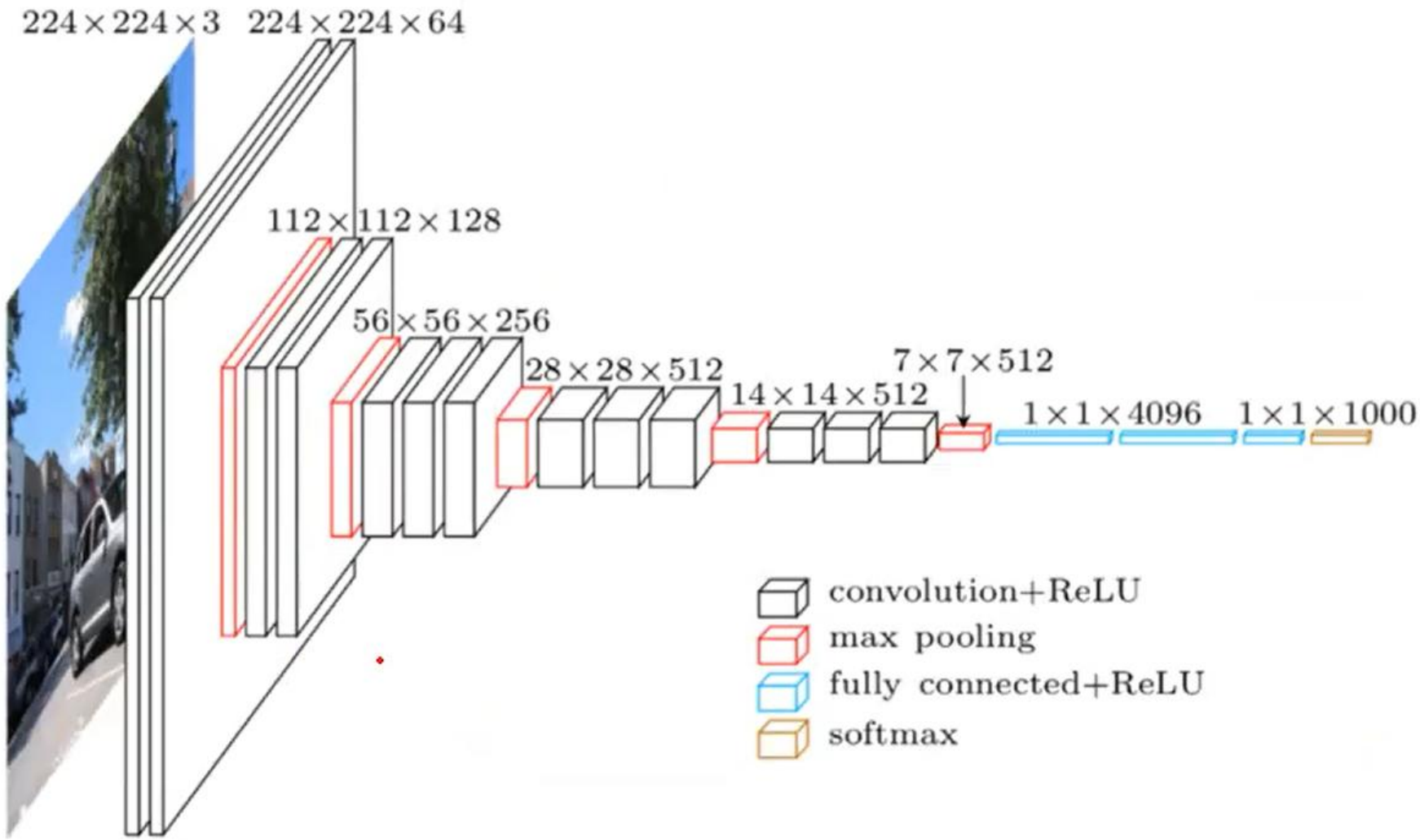
Common Architectures In Convolutional Neural Networks

AlexNet



Common Architectures In Convolutional Neural Networks

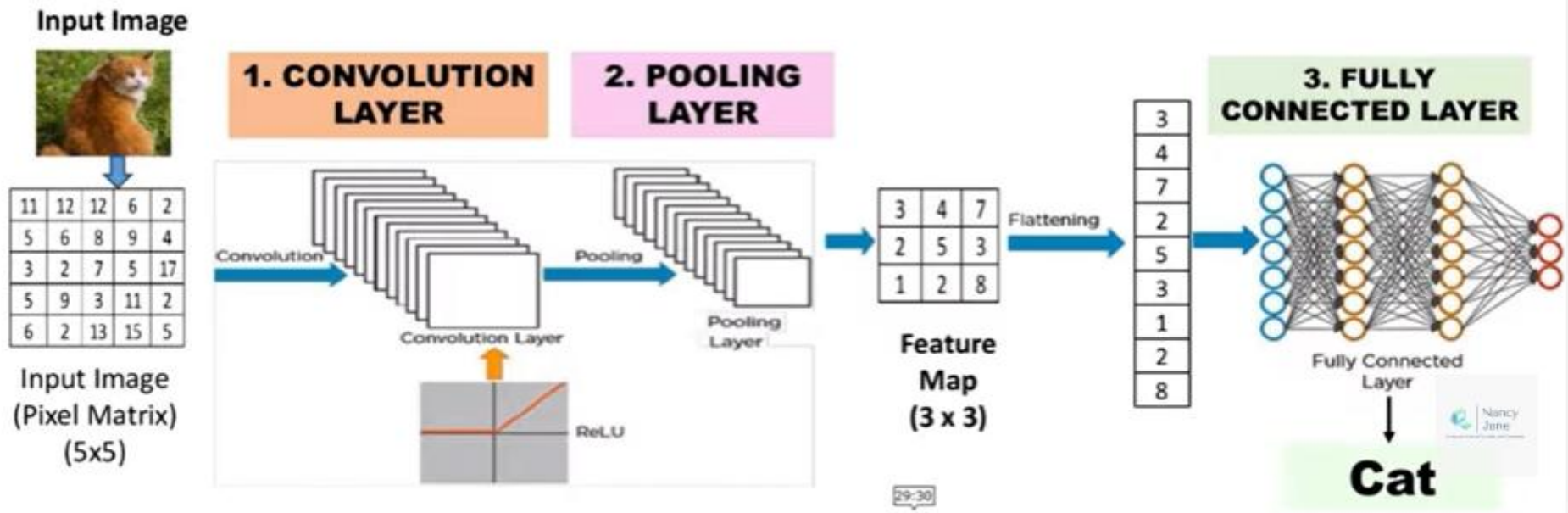
VGG-16



CNN Architecture for Image Classification

Three Layers in CNN

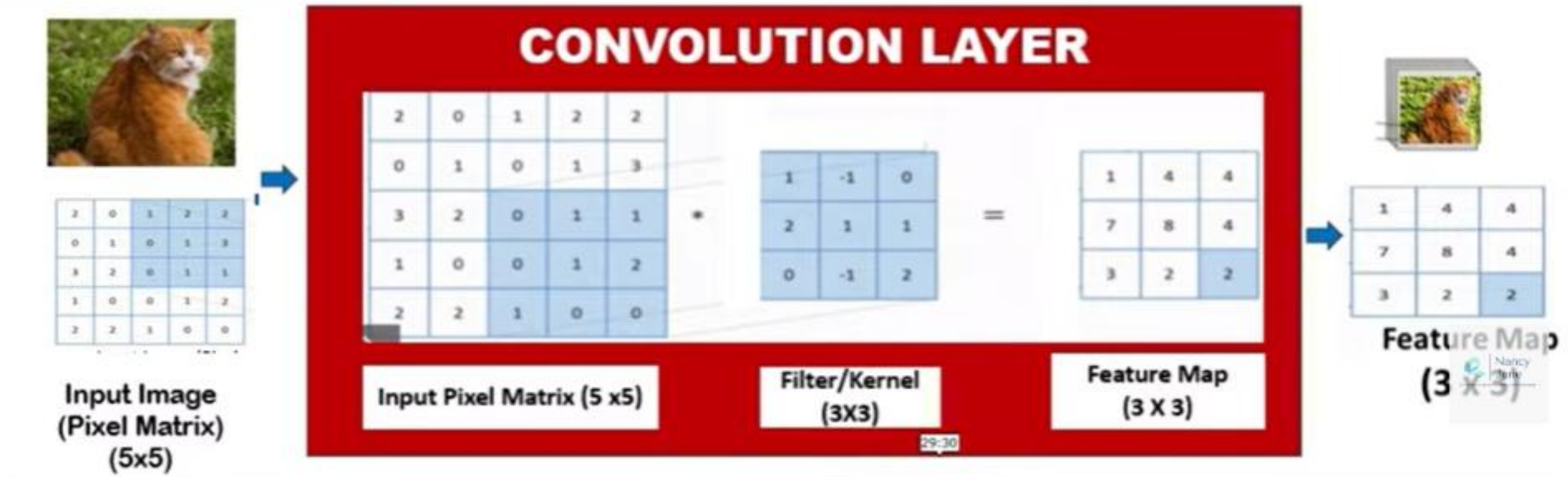
1. **CONVOLUTION LAYER**
2. **POOLING LAYER**
3. **FULLY CONNECTED LAYER**



1. CONVOLUTION LAYER

- ✓ Core computational layer in CNN .
- ✓ It performs **convolution operation** to extract significant features from input.
- ✓ Convolution operations are done with **input** data & weighted **filters or kernel**.
- ✓ **Feature Map** are generated after convolution

PARAMETER	EXAMPLE
Input size (M X M)	*Input (5 X 5)*
Filter Size (N X N)	*Filter (3 x3)*
number of Filters (F)	*F =1*
Stride & Padding	*Stride =1 & Padding =No*
Activation Function(if any)	Relu (Recommended)



PADDING LAYER

Original Image
Pixel Matrix
5 x 5

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

Image
Pixel Matrix + Padding =1
6 x 6

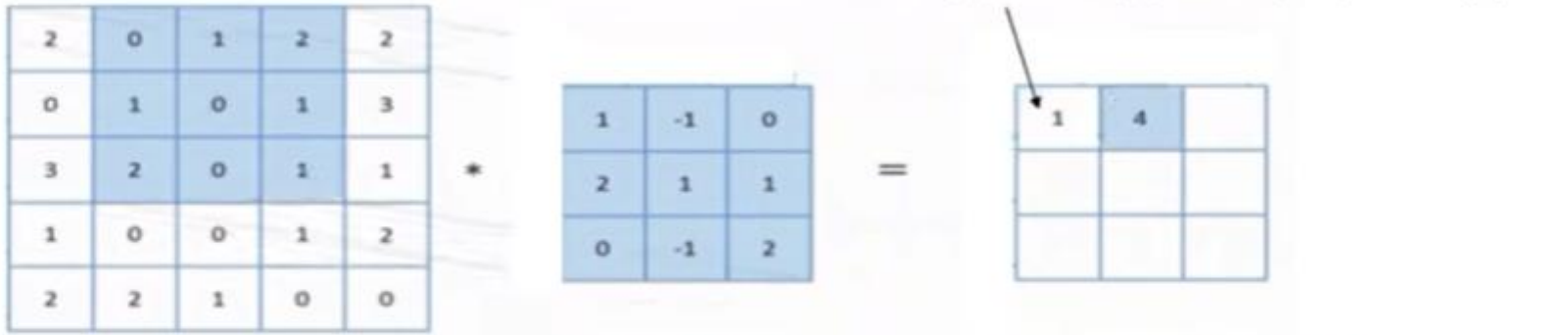
	2	0	1	2	
	0	1	0	1	
	3	2	0	1	
	1	0	0	1	

Padding Layer

29:30

CONVOLUTION -Operation Demo

$$A = 2*(1)+0*(-1)+1(0)+0*(2)+1*(1)$$
$$+0*(1)+3*(0)+2*(-1)+0*(2)$$



Input Image (5x5)
Pixel Matrix

Filter (3x 3)

Feature Map (3x3)

- Convolution operations are done with weighted **filters or kernel** & **input** data.
- **Feature Map** are generated after convolution

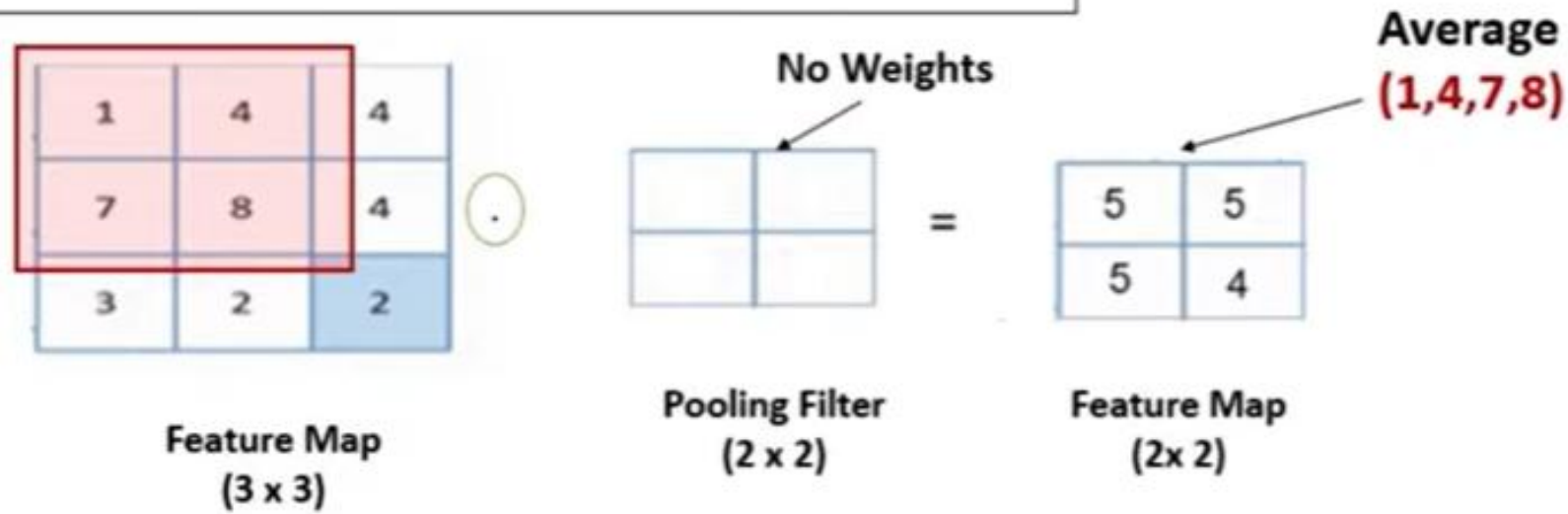
PARAMETER	EXAMPLE
Input size (M X M)	*Input (5 X 5)*
Filter Size (N X N)	*Filter (3 x3)*
number of Filters (F)	*F =1*
Stride & Padding	*Stride =1 & Padding =No*
Activation Function(if any)	Relu (Recommended)

Nancy Jane

2. POOLING LAYER

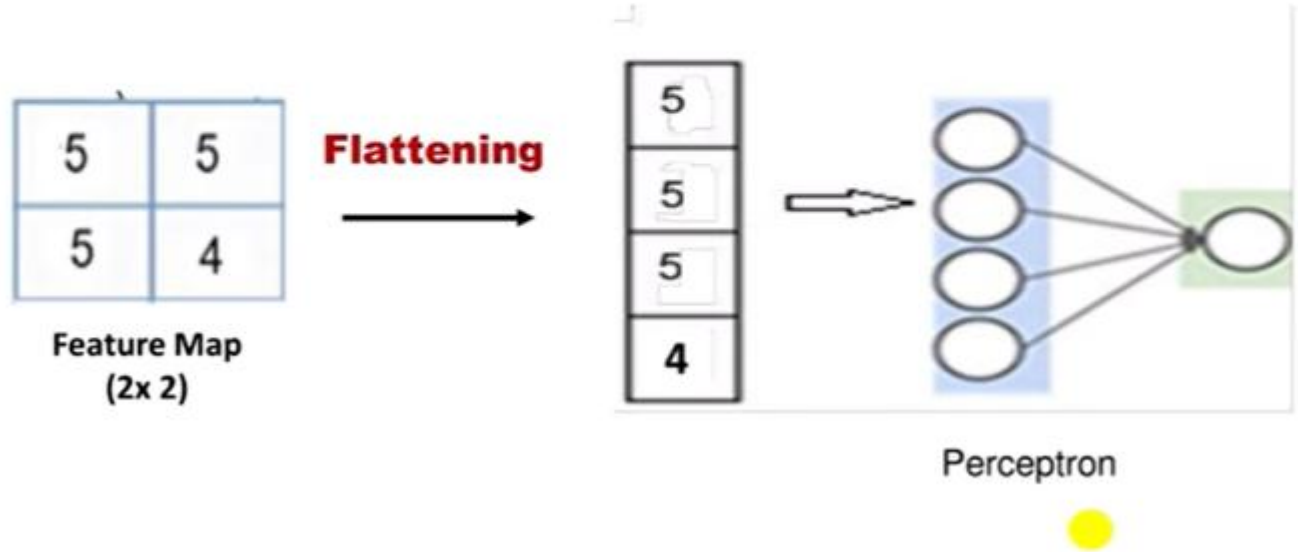
- ❑ Pooling compresses feature maps by reducing their size.
- ❑ It keeps essential features while improving shift and noise invariance.
- ❑ Types of Pooling : Average pooling, Max Pooling etc.
- ❑ No weights in pooling filter

PARAMETER	EXAMPLE
Feature Map	*Input (3 X 3)*
Pooling Filter Size	*Filter (2 x2)*
Pooling Type	*Average Pooling*
Stride & Padding	*Stride =1 & Padding =No*



3. FULLY CONNECTED LAYER

- Also referred to as Dense layer, it uses Dense Neural Network or Multilayer Perceptron to classify the flattened features



Input Image

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

Input Pixel Matrix (5 x 5)

CNN LAYER

CONVOLUTION LAYER

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

*

Convolution Filter

1	-1	0
2	1	1
0	-1	2

1	4	4
7	8	4
3	2	2

Feature Map (3 X 3)

POOLING LAYER

1	4	4
7	8	4
3	2	2

⊛

Pooling Filter

5	5
5	4

Feature Map (2 x 2)

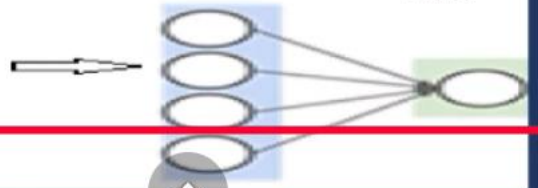
Flattening

5
5
5
4

FULLY CONNECTED LAYER

5
5
5
4

MLP

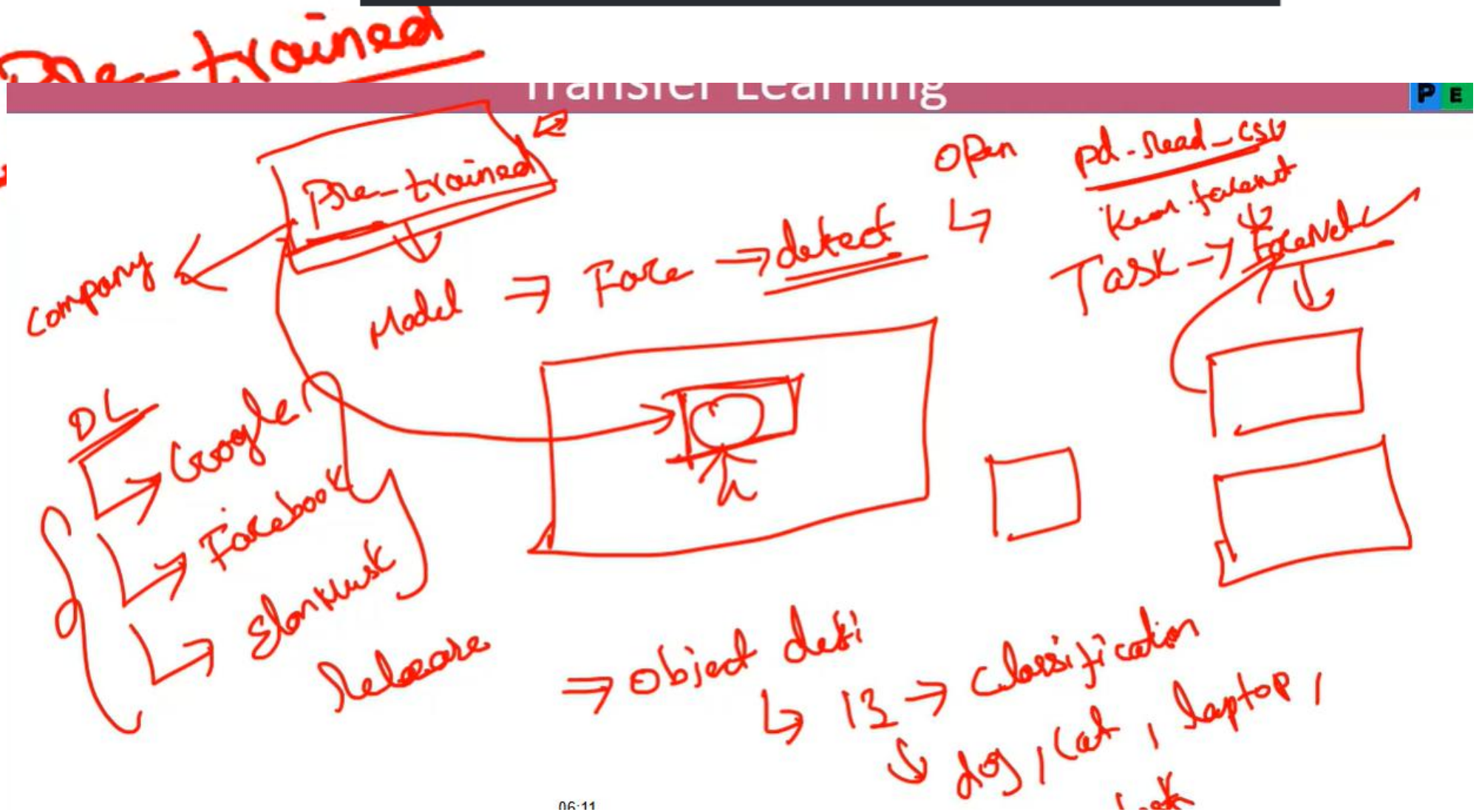


Cat

1. Feature Learning

2. Classification

Transfer Learning



🔥 1. CNN-Based Pre-Trained Models (Image Processing)

✓ For: image classification, feature extraction, transfer learning, detection

1️⃣ VGG16 / VGG19

- **Use:** Simple, classic architecture
- **Good for:** Transfer learning, small datasets
- **Strength:** Easy to understand, stable
- **Weakness:** Heavy, many parameters

2️⃣ ResNet (ResNet50, ResNet101, ResNet152)

- **Use:** General image classification
- **Good for:** Deep networks (skip connections)
- **Strength:** Best baseline for transfer learning
- **Weakness:** Larger model size

3️⃣ InceptionV3 / InceptionResNetV2

- **Use:** High-accuracy image classification
- **Good for:** Complex feature extraction
- **Strength:** Efficient + powerful
- **Weakness:** Architecture is complex

4️⃣ MobileNet (V1, V2, V3)

- **Use:** Mobile apps, real-time inference
- **Good for:** Low-power devices
- **Strength:** Lightweight & fast
- **Weakness:** Slightly lower accuracy

5️⃣ EfficientNet (B0–B7)

- **Use:** Modern high-accuracy model
- **Good for:** Classification, feature extraction
- **Strength:** Best accuracy vs speed balance
- **Weakness:** Larger versions need GPU

6️⃣ Xception

- **Use:** High-end image classification
- **Strength:** Depthwise separable conv
- **Good for:** Transfer learning on large datasets
- **Weakness:** Heavier model

7️⃣ DenseNet (DenseNet121/169/201)

- **Use:** Deep feature extraction
- **Strength:** Feature reuse (dense connections)
- **Weakness:** More memory usage

8️⃣ NASNet

- **Use:** Architectures designed by neural search
- **Strength:** Very high accuracy
- **Weakness:** Huge and slow

🎯 Use-Case Recommendation (Simple)

Task	Best Pre-trained Models
General image classification	ResNet50, EfficientNet, VGG16
Mobile apps / fast inference	MobileNetV2, EfficientNetB0
High accuracy	InceptionV3, Xception, EfficientNetB7
Small datasets	VGG16, ResNet50
Transfer learning	ResNet50, MobileNet, EfficientNet
Medical imaging	DenseNet, EfficientNet

🧠 2. NLP Pre-Trained Models (Text / Language)

✓ For: text classification, translation, QA, chatbots

1️⃣ BERT

- **Use:** NLP understanding (classification, Q&A)
- **Strength:** Bidirectional context understanding

2️⃣ GPT (2, 3, 3.5, 4, etc.)

- **Use:** Text generation, chatting, story writing
- **Strength:** Best at generative tasks

3️⃣ RoBERTa

- **Use:** Better version of BERT
- **Strength:** Strong for classification tasks

4️⃣ DistilBERT

- **Use:** Lightweight NLP
- **Strength:** Faster and smaller model

5️⃣ T5

- **Use:** Text-to-text tasks (summaries, translation)
- **Strength:** Very flexible

🔥 3. Audio / Speech Pre-Trained Models

1️⃣ Whisper (OpenAI)

- **Use:** Speech-to-text transcription
- **Strength:** Multilingual, high accuracy

2️⃣ Wav2Vec 2.0

- **Use:** Audio classification, speech recognition

🌐 4. Vision-Language Models (Image + Text)

1️⃣ CLIP

- **Use:** Image-text matching, search by text
- **Strength:** Learns joint embeddings

2️⃣ BLIP / BLIP-2

- **Use:** Image captioning, image-to-text

3️⃣ Flamingo / Vision Transformers (ViT)

- **Use:** Advanced image understanding

🎨 5. Video Pre-Trained Models

1️⃣ I3D (Inflated 3D ConvNet)

- **Use:** Video action recognition

2️⃣ TimeSformer

- **Use:** Transformer-based video classification

🧩 6. Object Detection Pre-Trained Models

1️⃣ YOLOv5 / YOLOv8 / YOLOv11

- **Use:** Real-time object detection
- **Strength:** Fastest & most accurate today

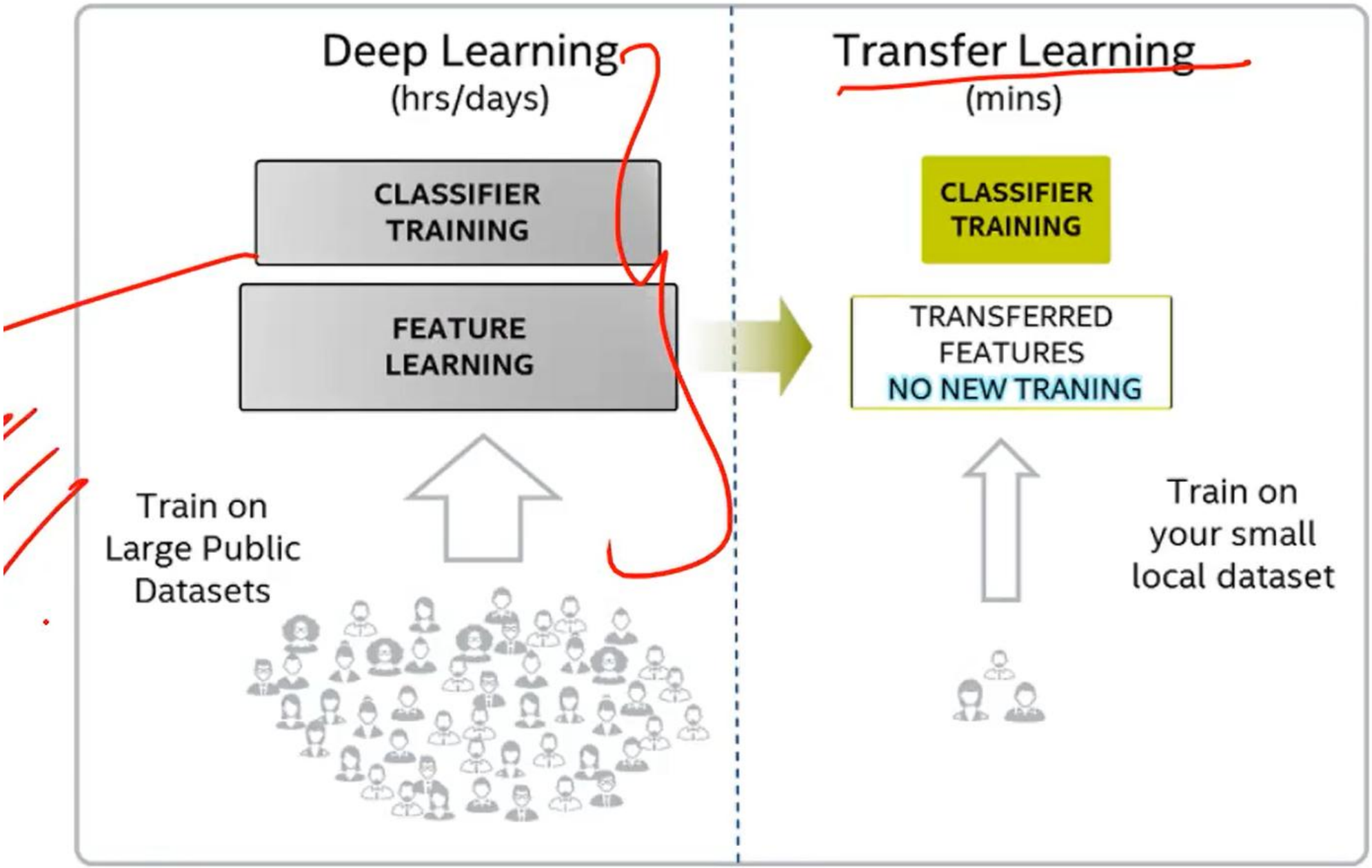
2️⃣ Faster R-CNN

- **Use:** High accuracy detection

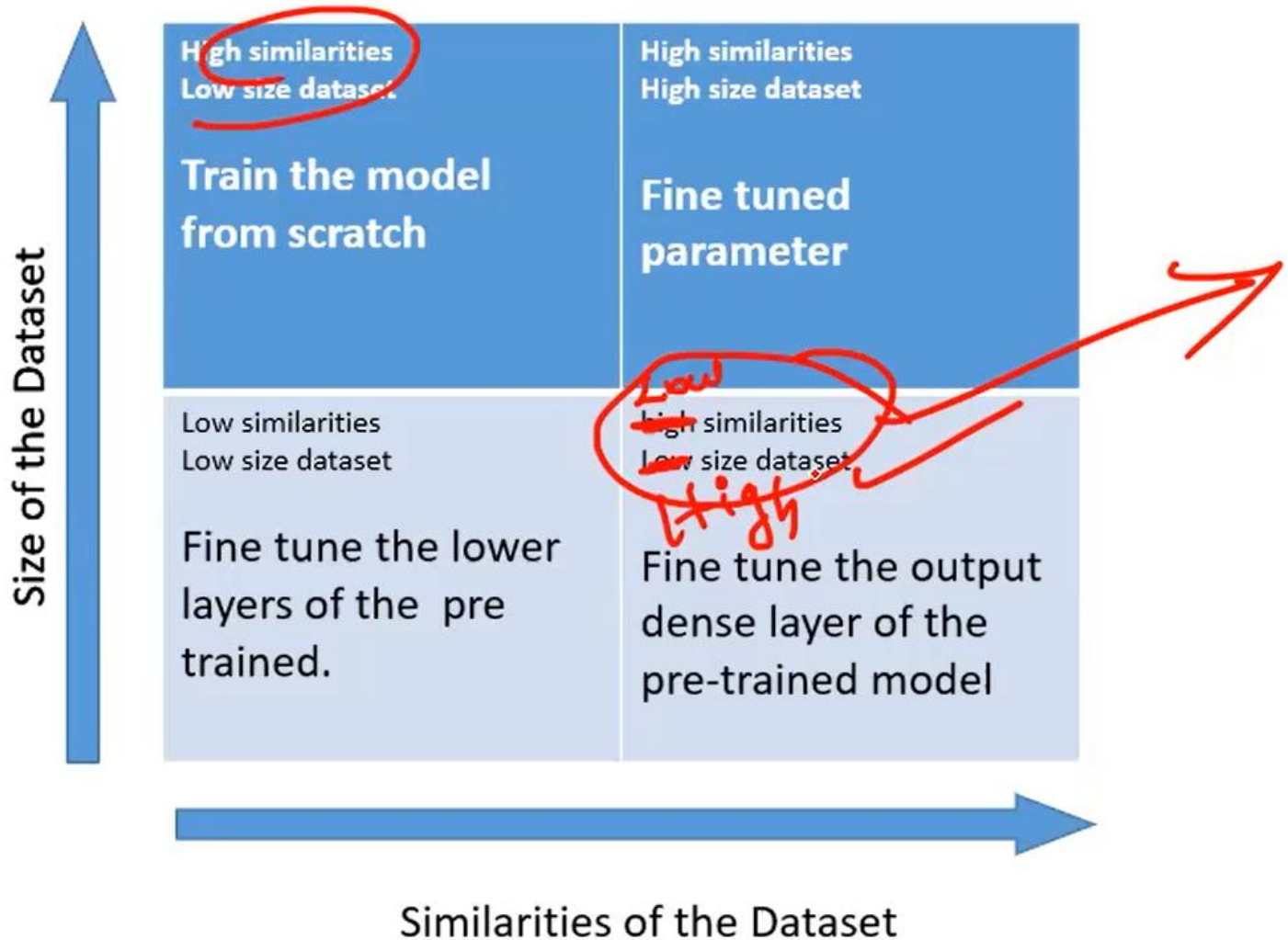
3️⃣ SSD

- **Use:** Lightweight detector

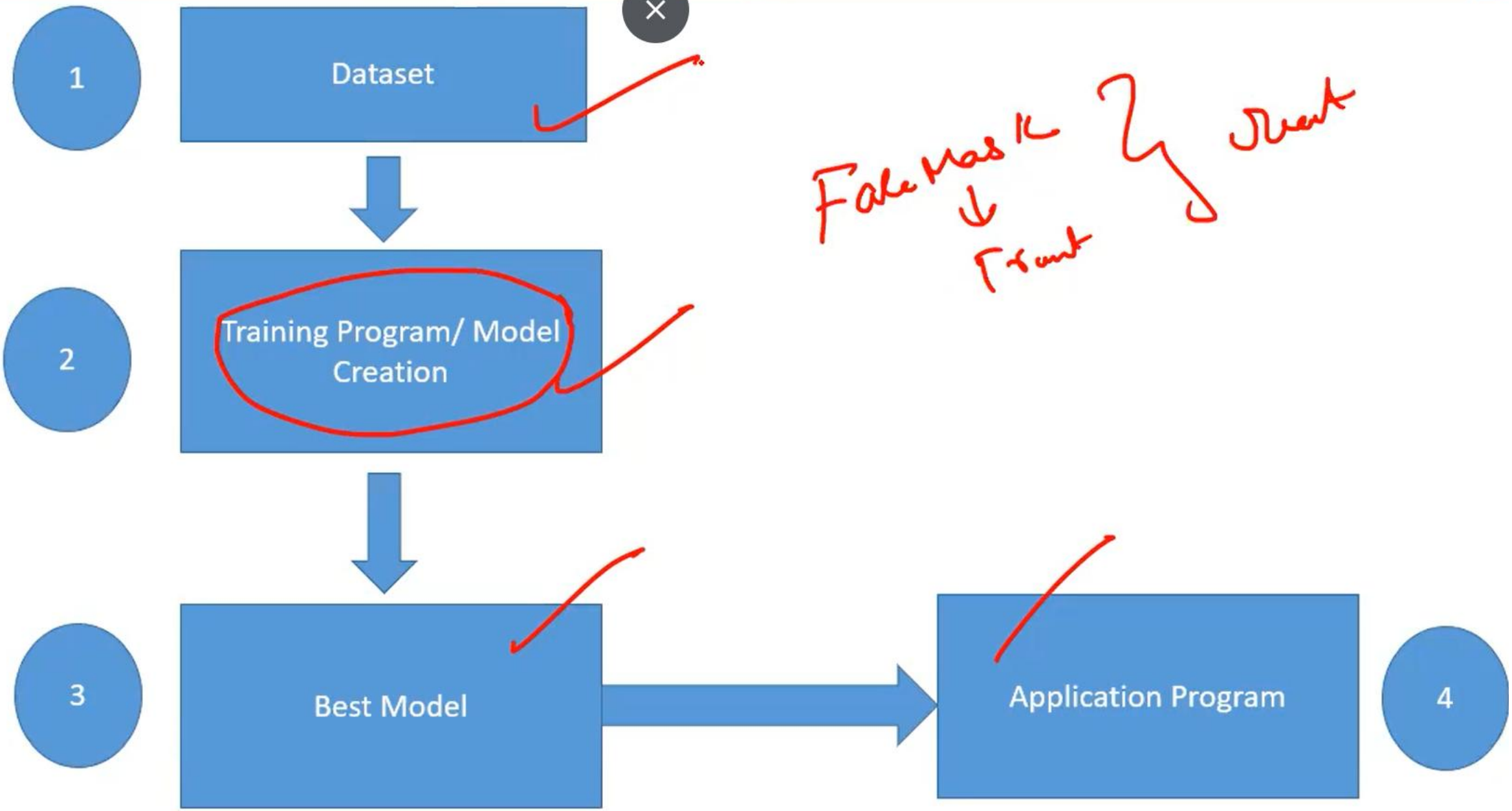
Transfer Learning

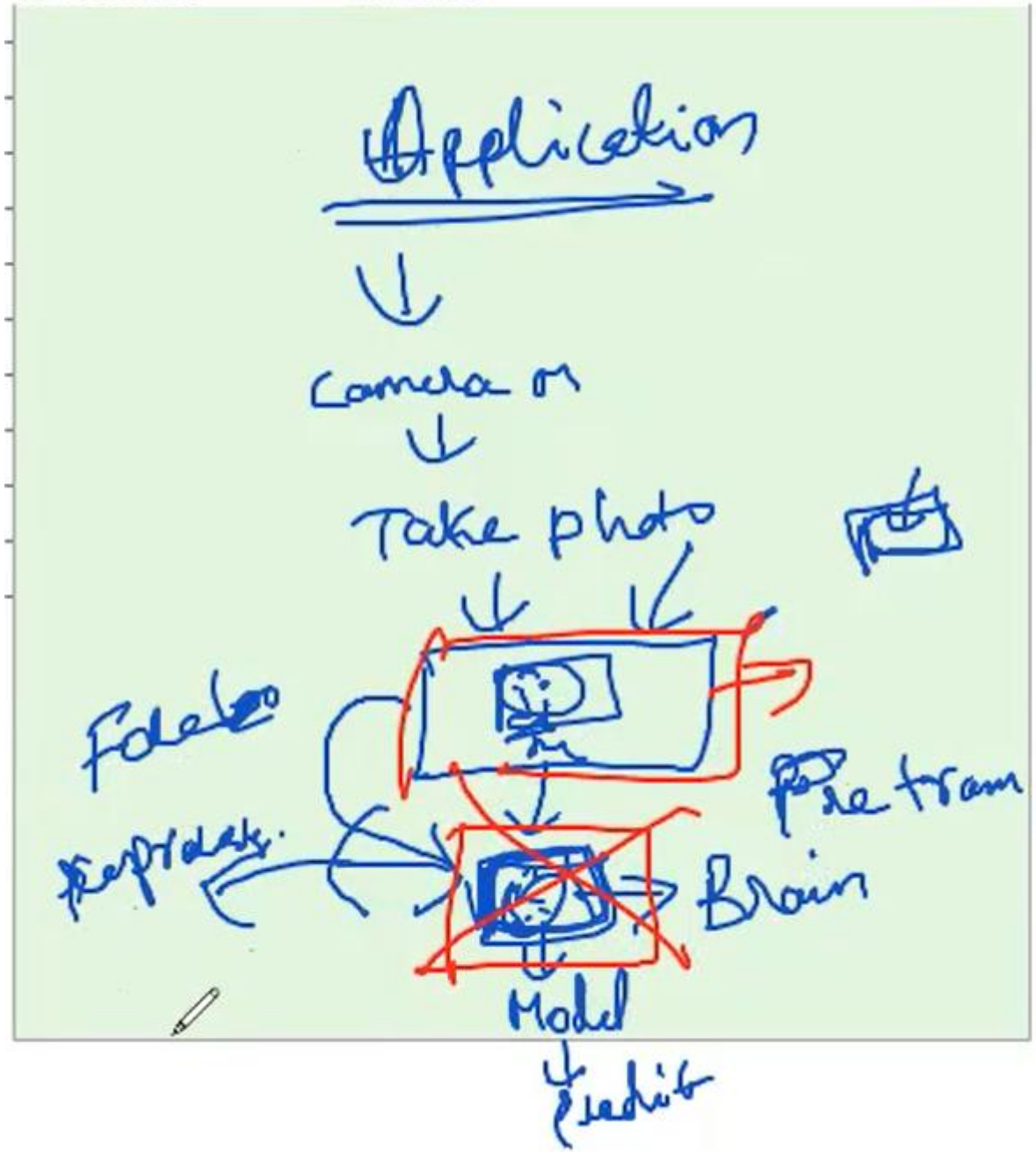


Transfer Learning



Deep Learning - Flow



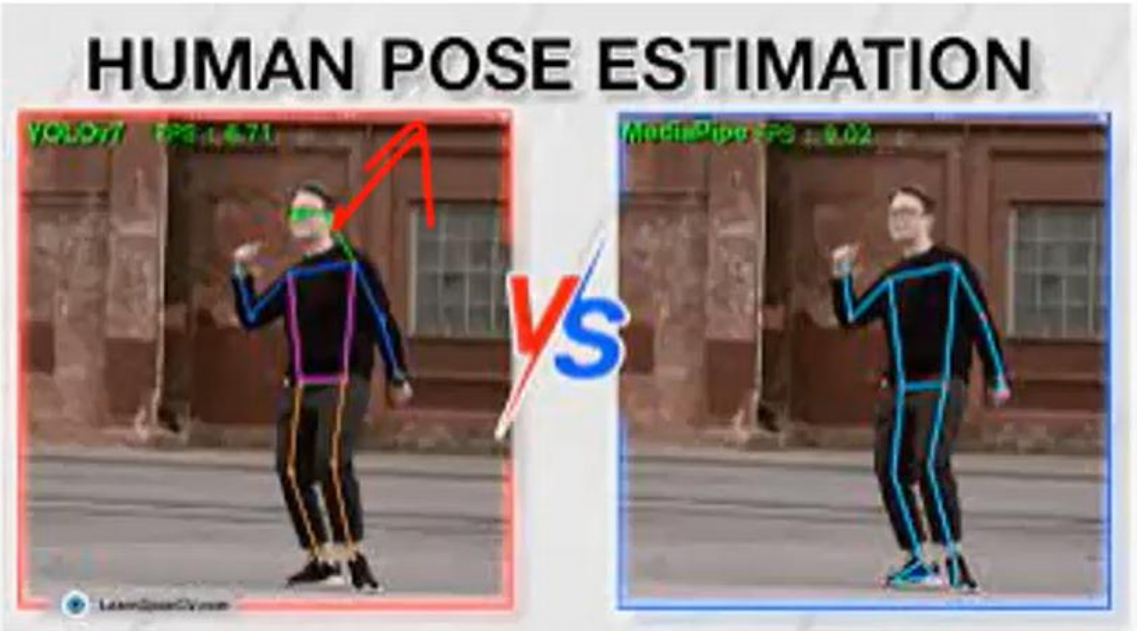


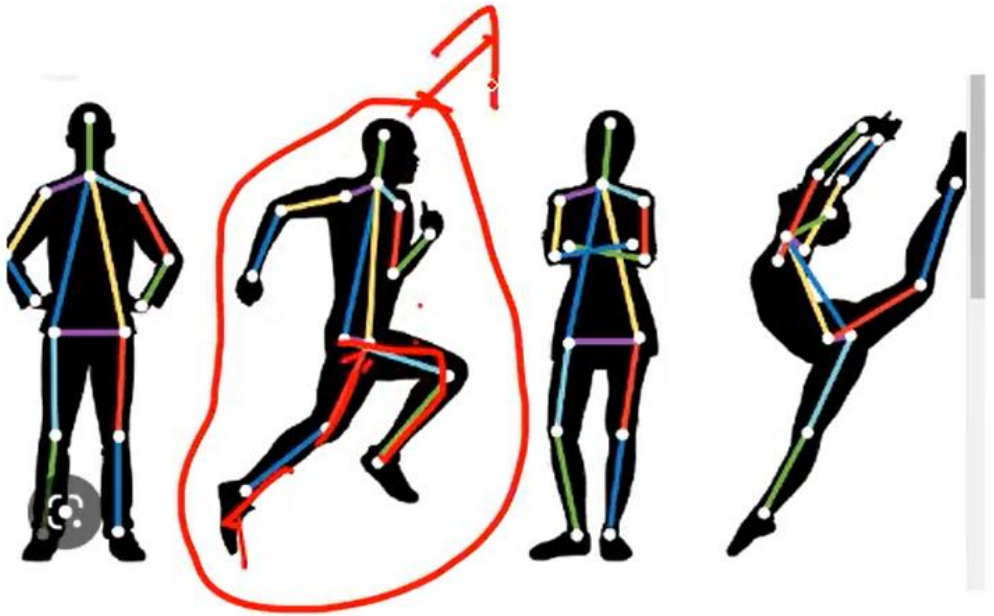
Input

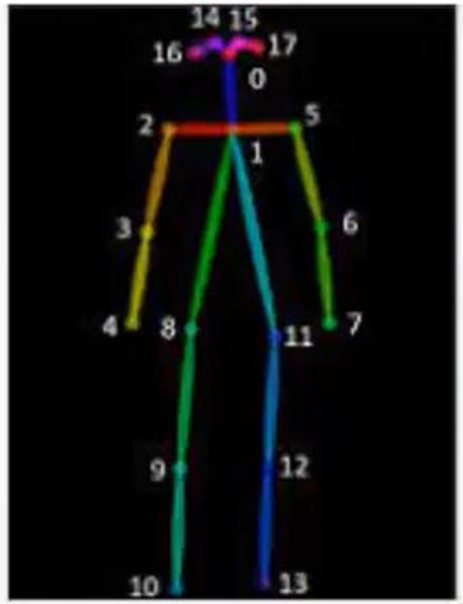
Brain

Input

Popular Deep Learning Model





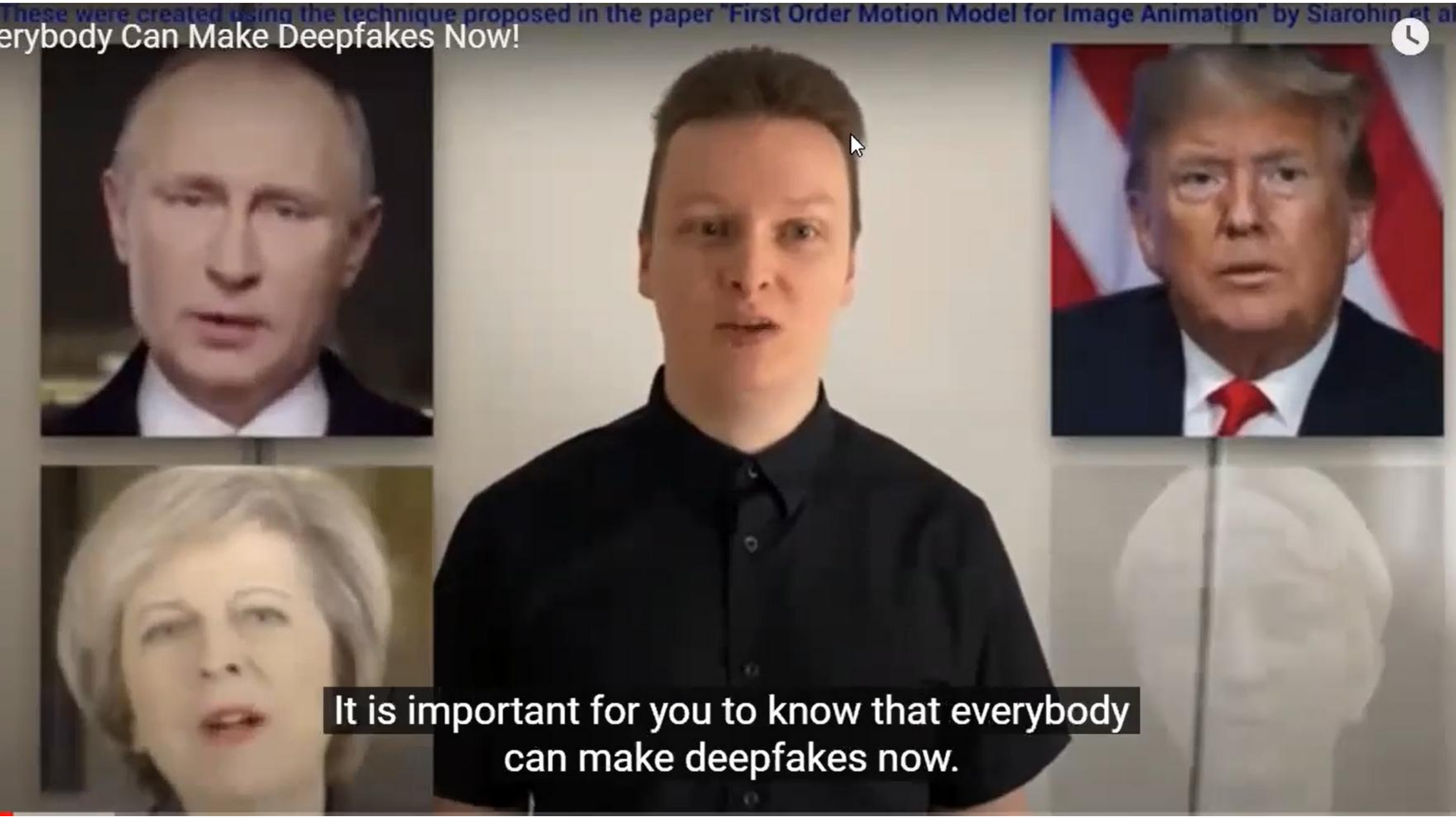


Popular Deep Learning Model

Popular Deep Learning Model

Deep Fake

https://www.youtube.com/watch?v=mUfJOQKdtAk


These were created using the technique proposed in the paper "First Order Motion Model for Image Animation" by Siarohin et al
erybody Can Make Deepfakes Now!
It is important for you to know that everybody
can make deepfakes now.

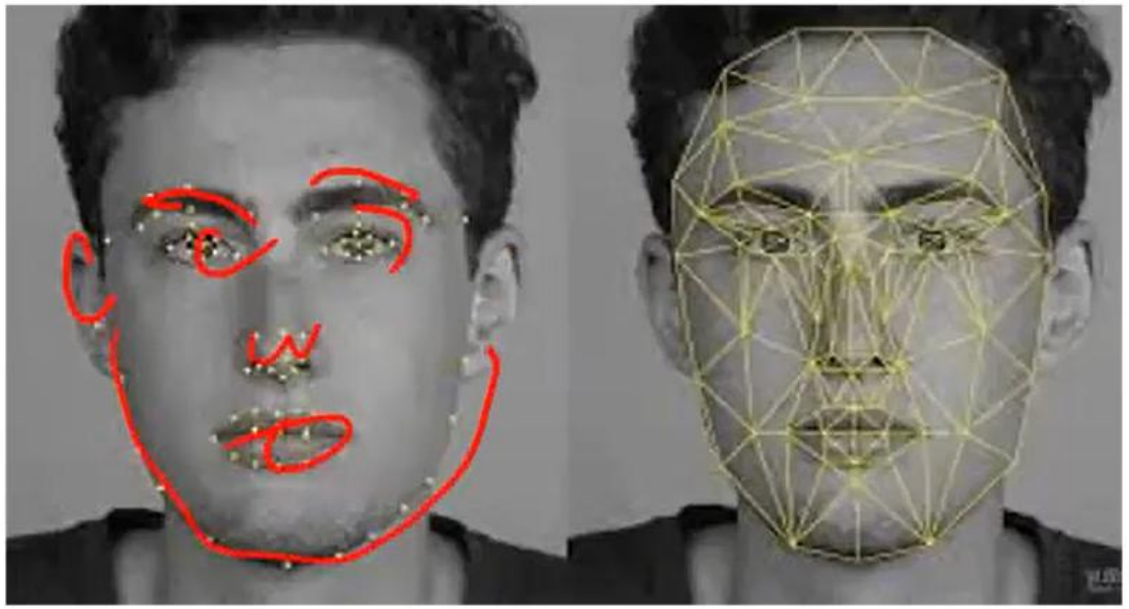
Popular Deep Learning Model

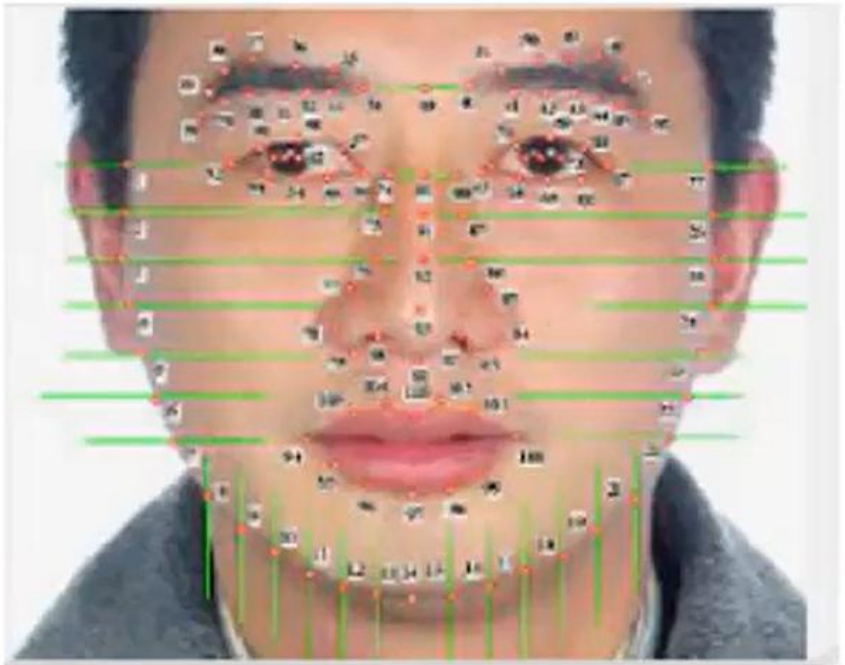
The Different Types of Generative Adversarial Networks (GANs) are:

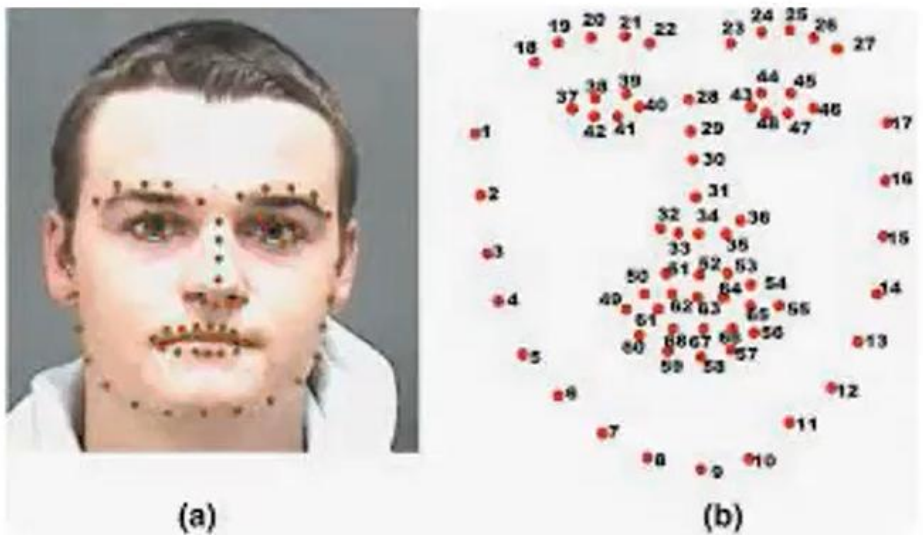
- Vanilla GAN.
- Conditional Gan (CGAN)
- Deep Convolutional GAN (DCGAN)
- CycleGAN.
- Generative Adversarial Text to Image Synthesis.
- Style GAN.
- Super Resolution GAN (SRGAN)

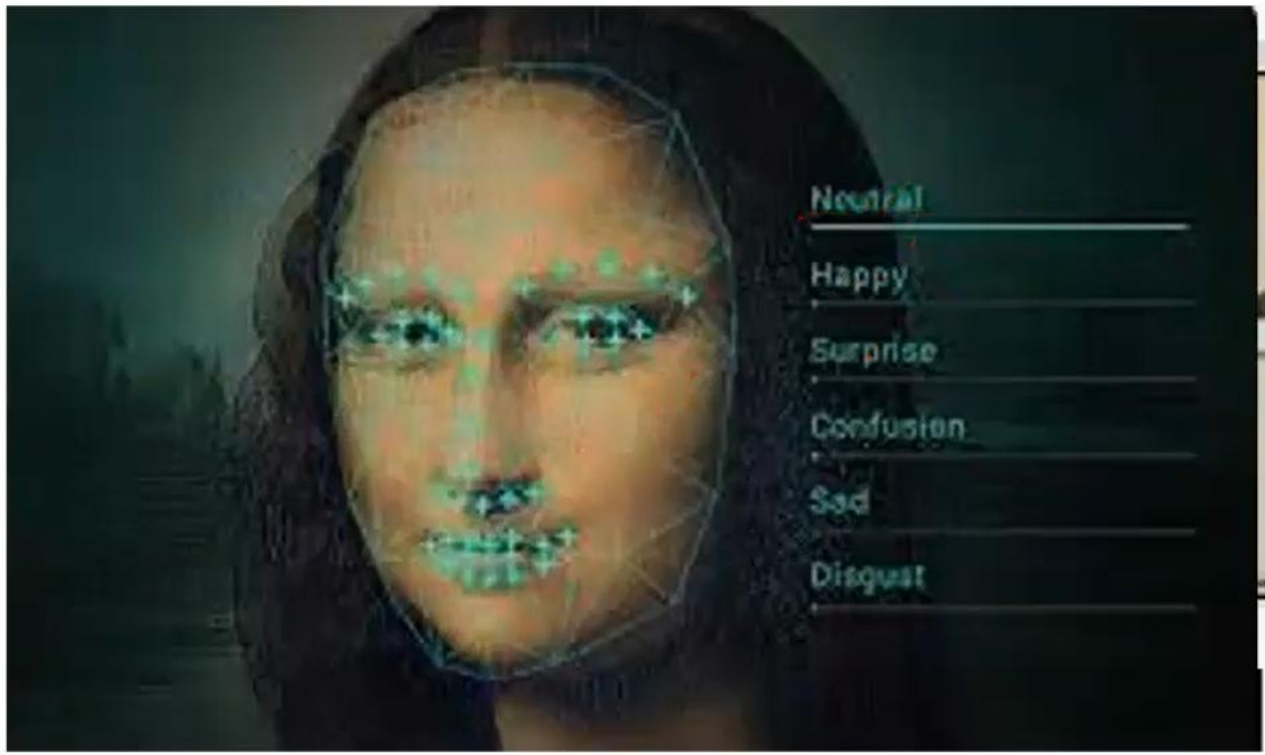
16:22

Popular Deep Learning Model








Neutral
Happy
Surprise
Confusion
Sad
Disgust

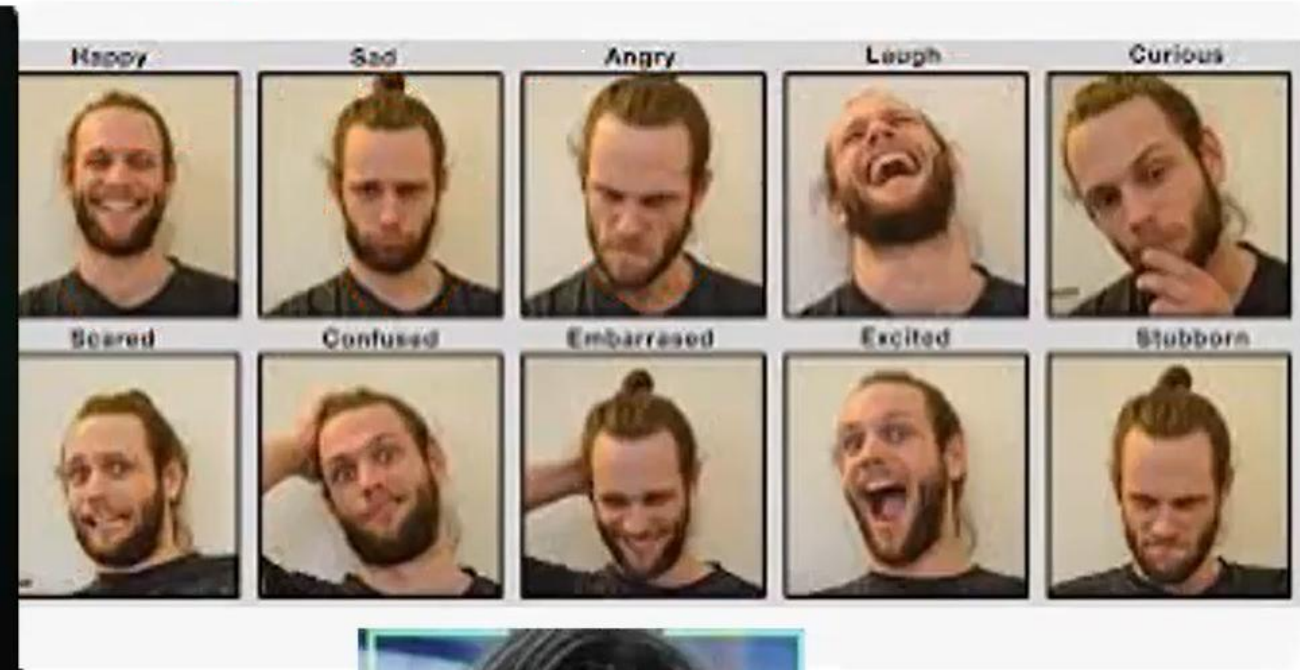

Happy
Sad
Angry
Laugh
Curious
Scared
Confused
Embarrased
Excited
Stubborn

Image Captioning – DL and NLP

A young boy is playing basketball. 	Two dogs play in the grass. 	A dog swims in the water. 	A little girl in a pink shirt is swinging. 
A group of people walking down a street. 	A group of women dressed in formal attire. 	Two children play in the water. 	A dog jumps over a hurdle.