Design of a Facial Recognition System Based on Deep Learning Model : Siamese Neural Network

End of Semester Project

Presented by

MOUHOUB Anouar
DERAOUI Oussama

Jury members

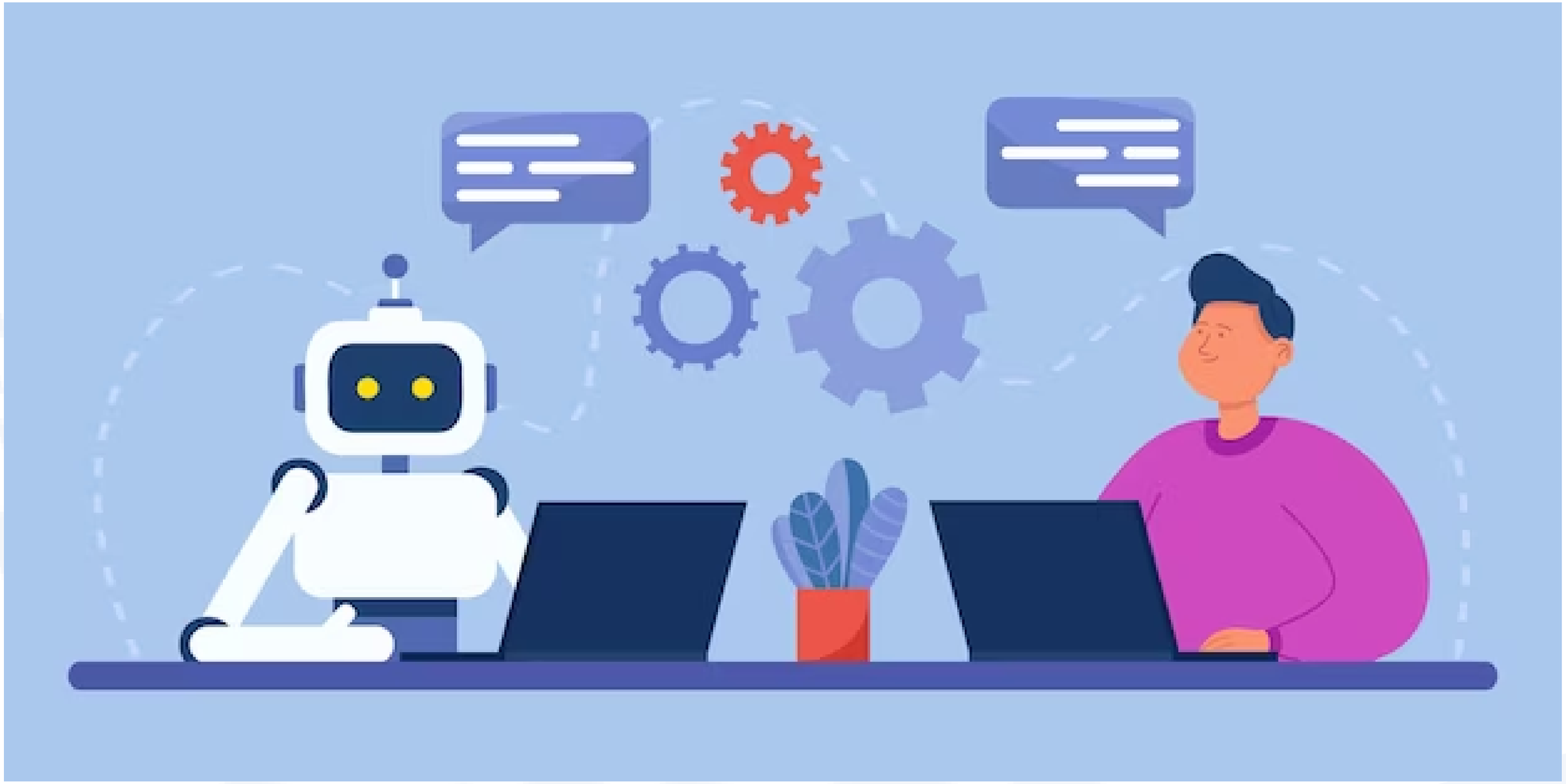
Pr. Younes JABRANE
Pr. Taoufik RACHAD

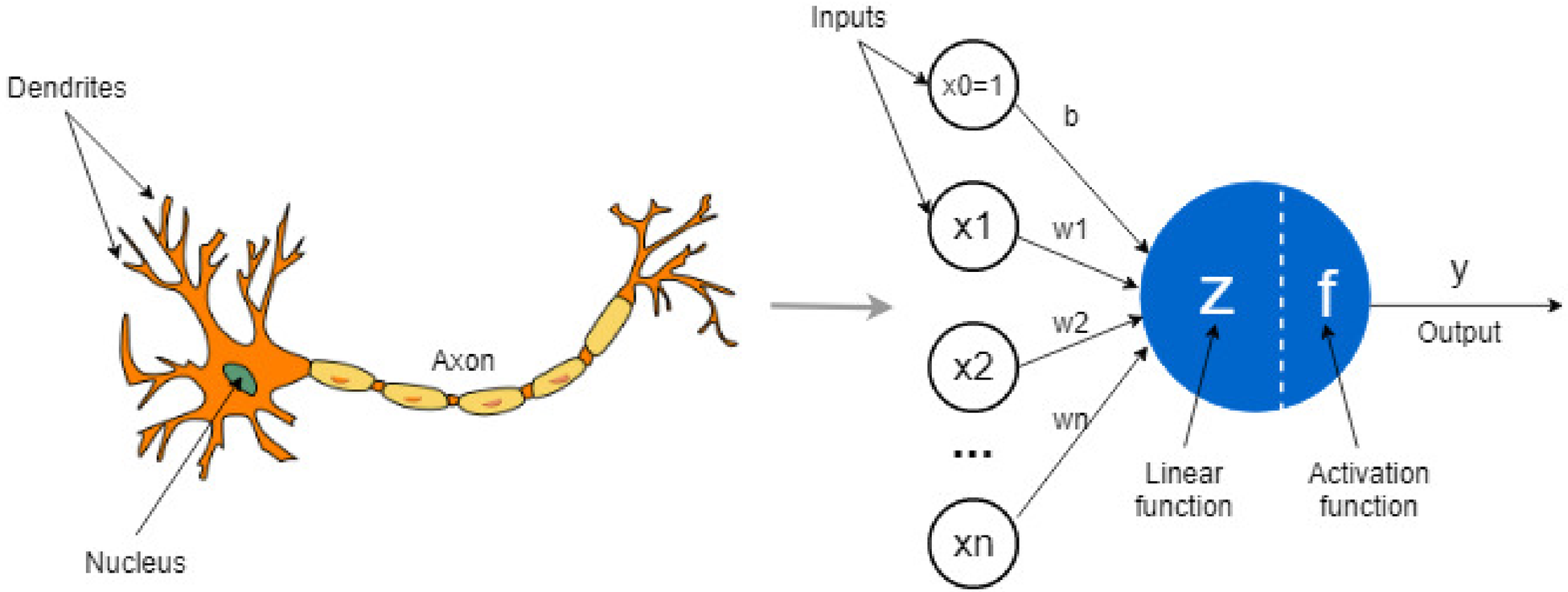
July 2023

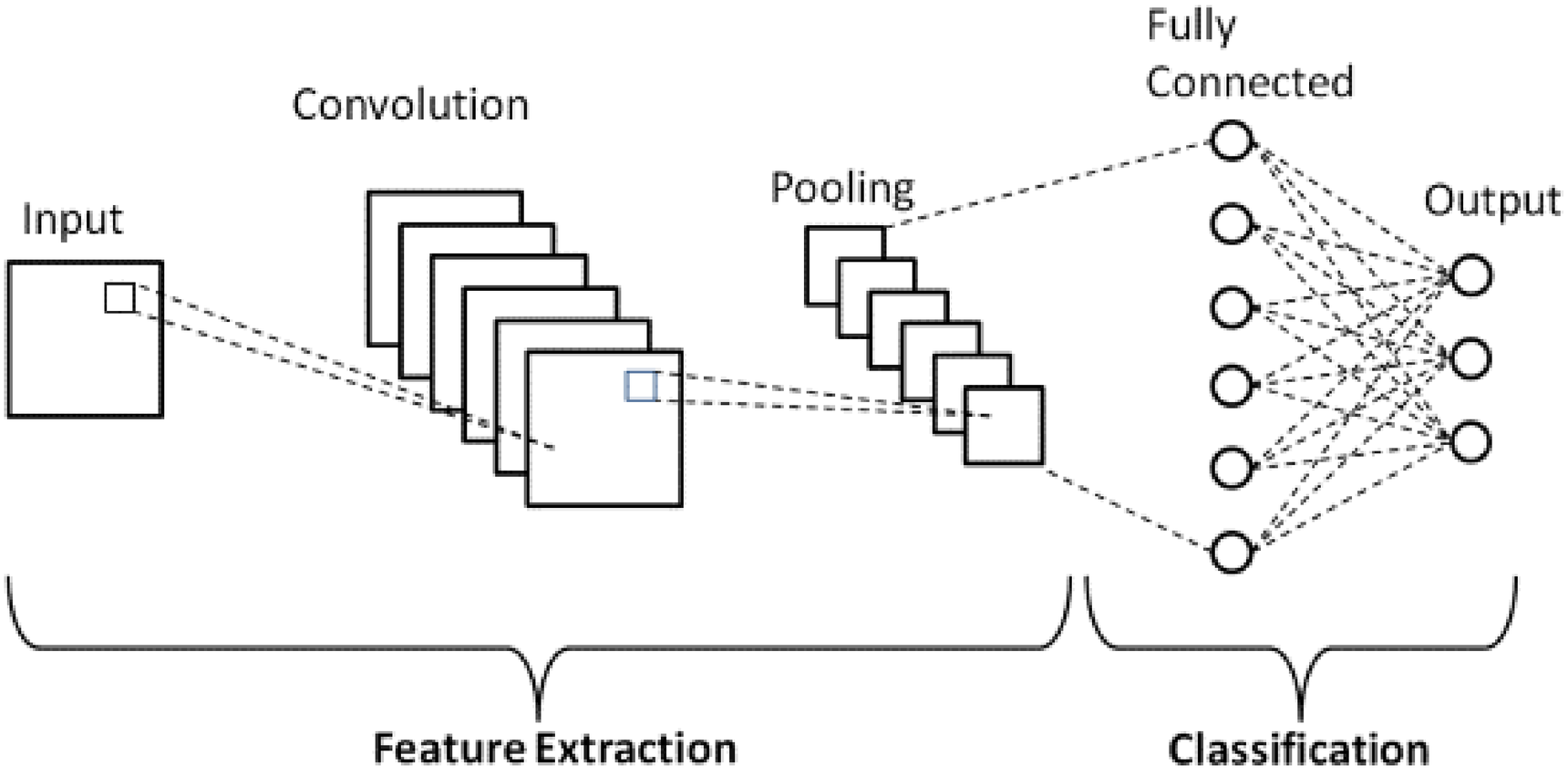
Plan

- Introduction
- Dataset Description
- Deep Neural Network
- Siamese Neural Network
- Evaluation of the Model
- Conclusion

Introduction




Dendrites
Axon
Nucleus
Inputs
x0=1
x1
x2
xn
b
w1
w2
wn
z
f
y
Output
Linear function
Activation function

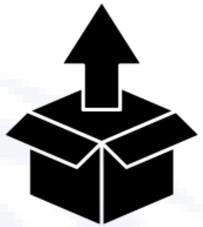

Fully
Connected
Convolution
Pooling
Input
Output
Feature Extraction
Classification

Dataset Description

Dataset : "Footballers.tgz"

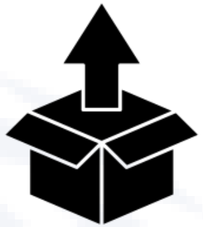
Preprocessing

Untar footballers dataset



Preprocessing

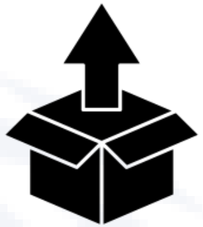
Untar footballers dataset



File Format Conversion

Preprocessing

Untar footballers dataset

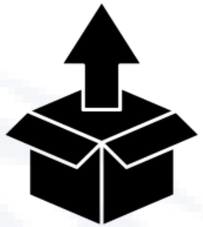


File Format Conversion

Duplicate Removal

Preprocessing

Untar footballers dataset



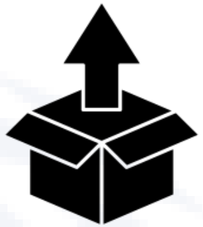
File Format Conversion

Duplicate Removal

Image Decoding

Preprocessing

Untar footballers dataset



File Format Conversion

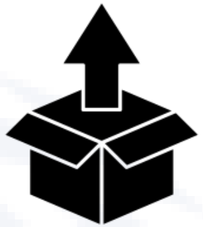
Duplicate Removal

Image Decoding

Normalization

Preprocessing

Untar footballers dataset



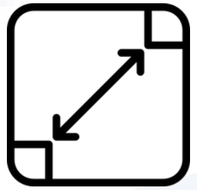
File Format Conversion

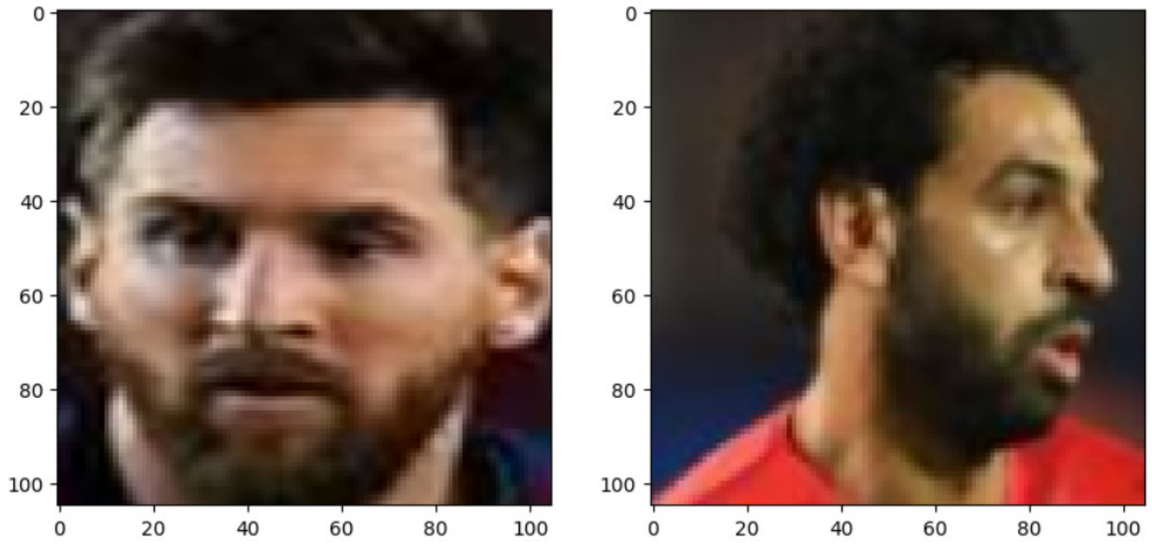
Duplicate Removal

Image Decoding

Normalization

Resizing Images





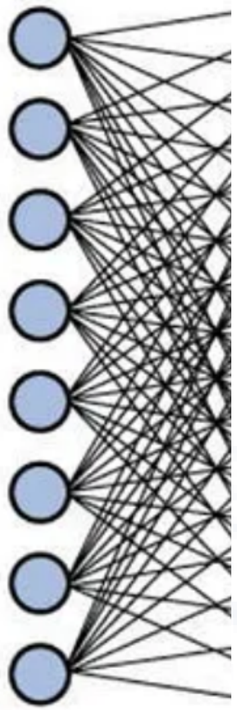
Label 0

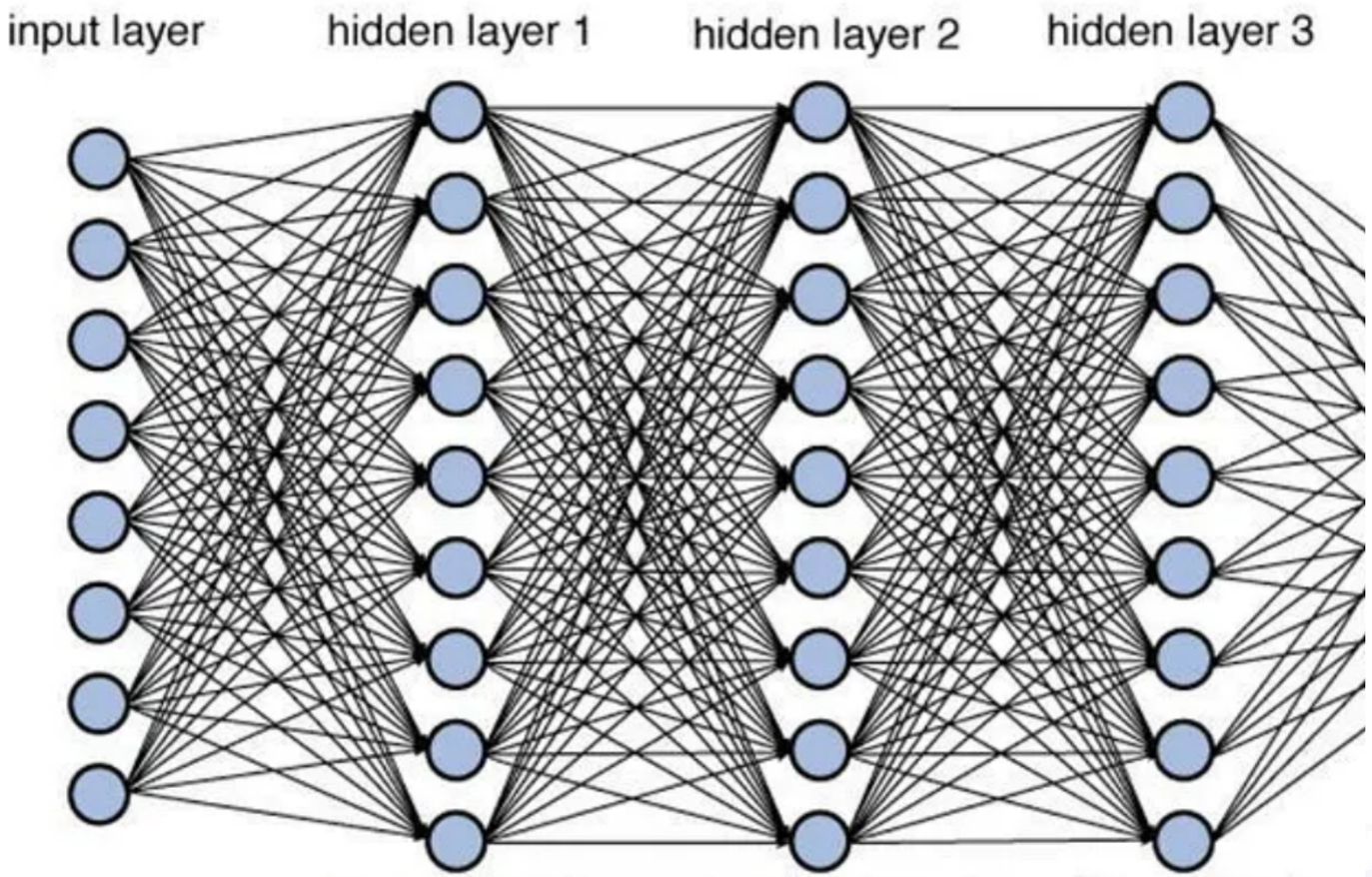
Label 1

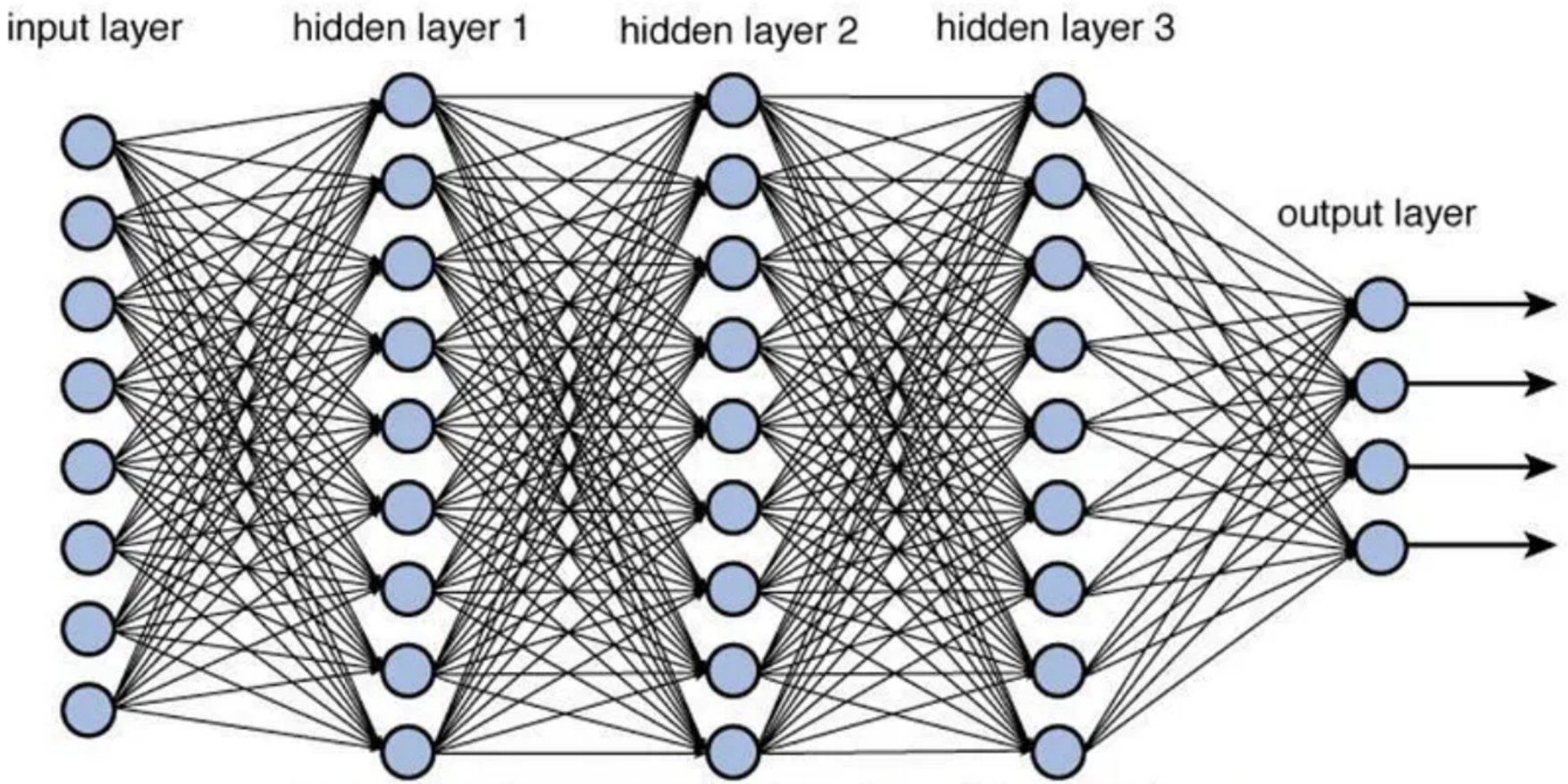
Deep Neural Network

(From scratch)

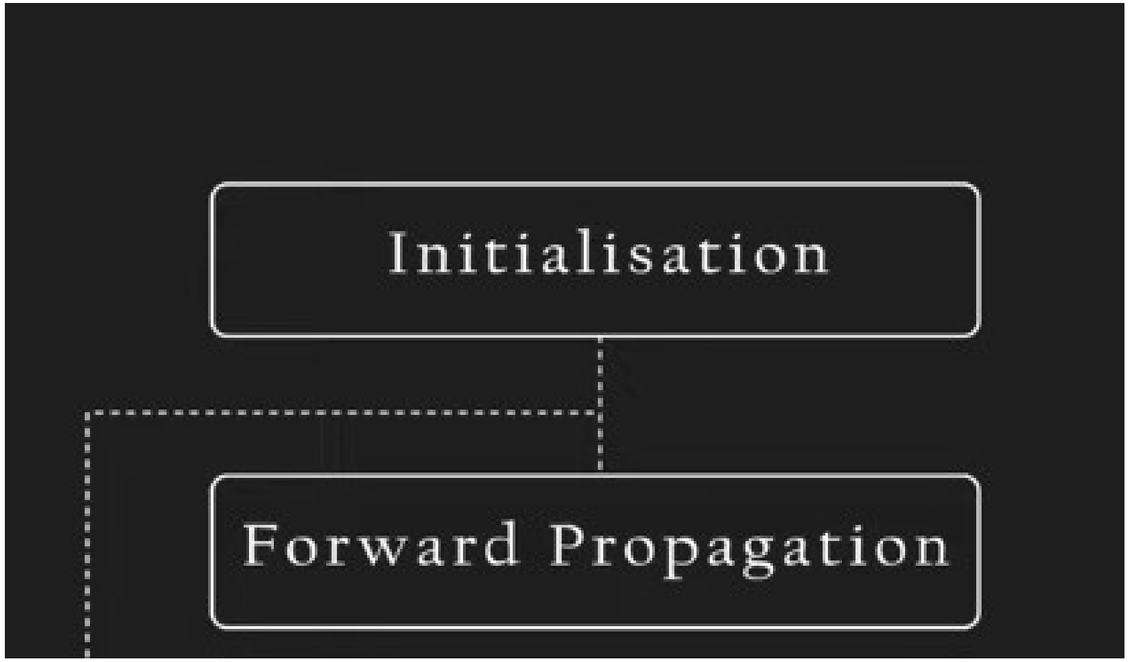
input layer

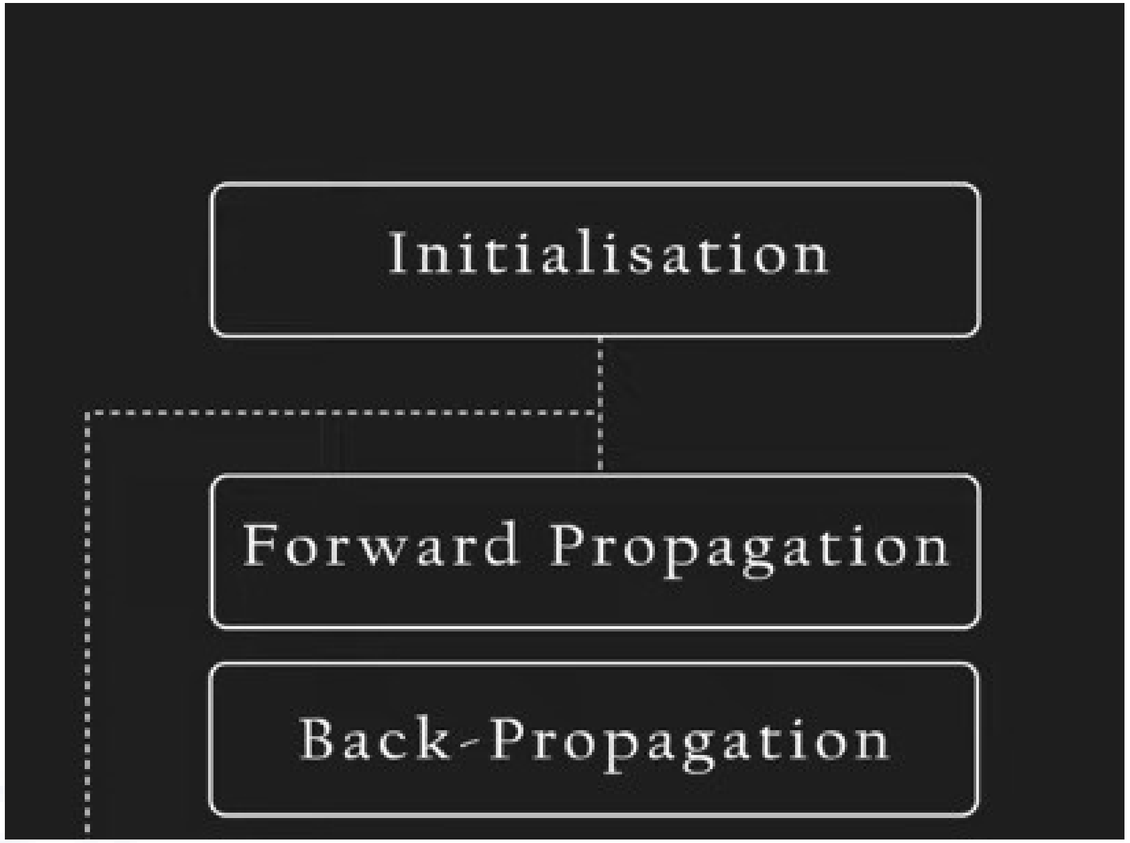


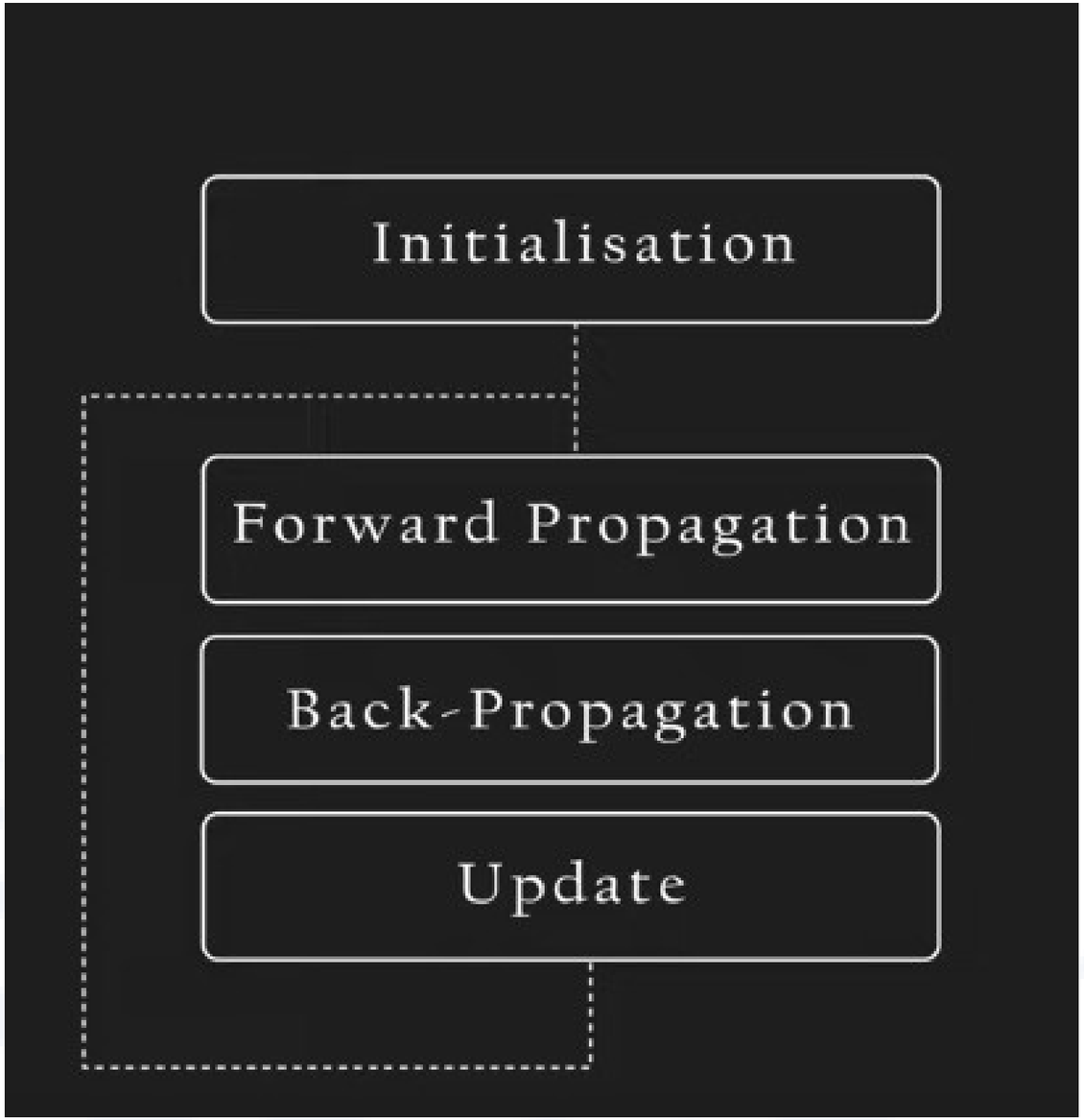

input layer
hidden layer 1
hidden layer 2
hidden layer 3


input layer
hidden layer 1
hidden layer 2
hidden layer 3
output layer


Initialisation


Initialisation
Forward Propagation


Initialisation
Forward Propagation
Back-Propagation


Initialisation
Forward Propagation
Back-Propagation
Update

Gradient Computing

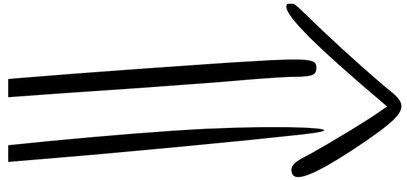
$$dZ^{[1]} = W^{[2]^T} \cdot dZ^{[2]} \times A^{[1]}(1 - A^{[1]})$$

$$dW^{[1]} = \frac{1}{m} \times dZ^{[1]} \cdot X^T$$

$$db^{[1]} = \frac{1}{m} \sum_{axe1} dZ^{[1]}$$

Gradient Computing

$$dZ^{[1]} = W^{[2]^T} \cdot dZ^{[2]} \times A^{[1]}(1 - A^{[1]})$$
$$dW^{[1]} = \frac{1}{m} \times dZ^{[1]} \cdot X^T$$
$$db^{[1]} = \frac{1}{m} \sum_{axe1} dZ^{[1]}$$

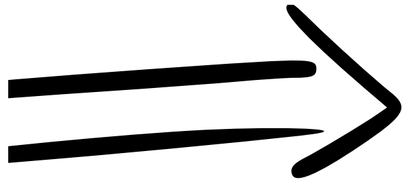


Adding one hidden layer

$$dZ^{[2]} = A^{[2]} - y$$
$$dW^{[2]} = \frac{1}{m} \times dZ^{[2]} \cdot A^{[1]^T}$$
$$db^{[2]} = \frac{1}{m} \sum_{axe1} dZ^{[2]}$$

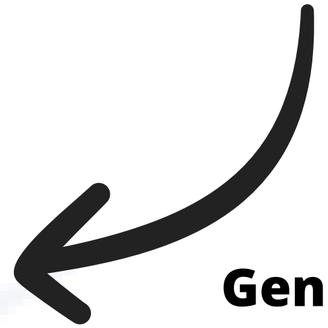
Gradient Computing

$$dZ^{[1]} = W^{[2]^T} \cdot dZ^{[2]} \times A^{[1]}(1 - A^{[1]})$$
$$dW^{[1]} = \frac{1}{m} \times dZ^{[1]} \cdot X^T$$
$$db^{[1]} = \frac{1}{m} \sum_{axe1} dZ^{[1]}$$



Adding one hidden layer

$$dZ^{[2]} = A^{[2]} - y$$
$$dW^{[2]} = \frac{1}{m} \times dZ^{[2]} \cdot A^{[1]^T}$$
$$db^{[2]} = \frac{1}{m} \sum_{axe1} dZ^{[2]}$$



Generalizing over n hidden layers

$$dZ^{[C_f]} = A^{[C_f]} - y$$
$$dW^{[c]} = \frac{1}{m} \times dZ^{[c]} \cdot A^{[c-1]^T}$$
$$db^{[c]} = \frac{1}{m} \sum_{axe1} dZ^{[c]}$$
$$dZ^{[c-1]} = W^{[c]^T} \cdot dZ^{[c]} \times A^{[c-1]}(1 - A^{[c-1]}$$

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Initialization function

Takes : Dimensions

Returns : Parameters

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Initialization function

Takes : Dimensions

Returns : Parameters

Forward Propagation

Takes : X, Parameters

Returns : Activations

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Initialization function

Takes : Dimensions

Returns : Parameters

Forward Propagation

Takes : X, Parameters

Returns : Activations

Backpropagation

Takes : Y, Activations, Parameters

Returns : Gradients

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Initialization function

Takes : Dimensions

Returns : Parameters

Forward Propagation

Takes : X, Parameters

Returns : Activations

Backpropagation

Takes : Y, Activations, Parameters

Returns : Gradients

Log loss Function

Takes : A, Y

Returns : The computed logistic loss

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Initialization function

Takes : Dimensions

Returns : Parameters

Forward Propagation

Takes : X, Parameters

Returns : Activations

Backpropagation

Takes : Y, Activations, Parameters

Returns : Gradients

Log loss Function

Takes : A, Y

Returns : The computed logistic loss

Update function

Takes : gradients, parameters, learning rate

Returns : Updated parameters

Implementation

Architecture Definition

Takes : X, Y

Returns : Dimensions

Initialization function

Takes : Dimensions

Returns : Parameters

Forward Propagation

Takes : X, Parameters

Returns : Activations

Backpropagation

Takes : Y, Activations, Parameters

Returns : Gradients

Log loss Function

Takes : A, Y

Returns : The computed logistic loss

Update function

Takes : gradients, parameters, learning rate

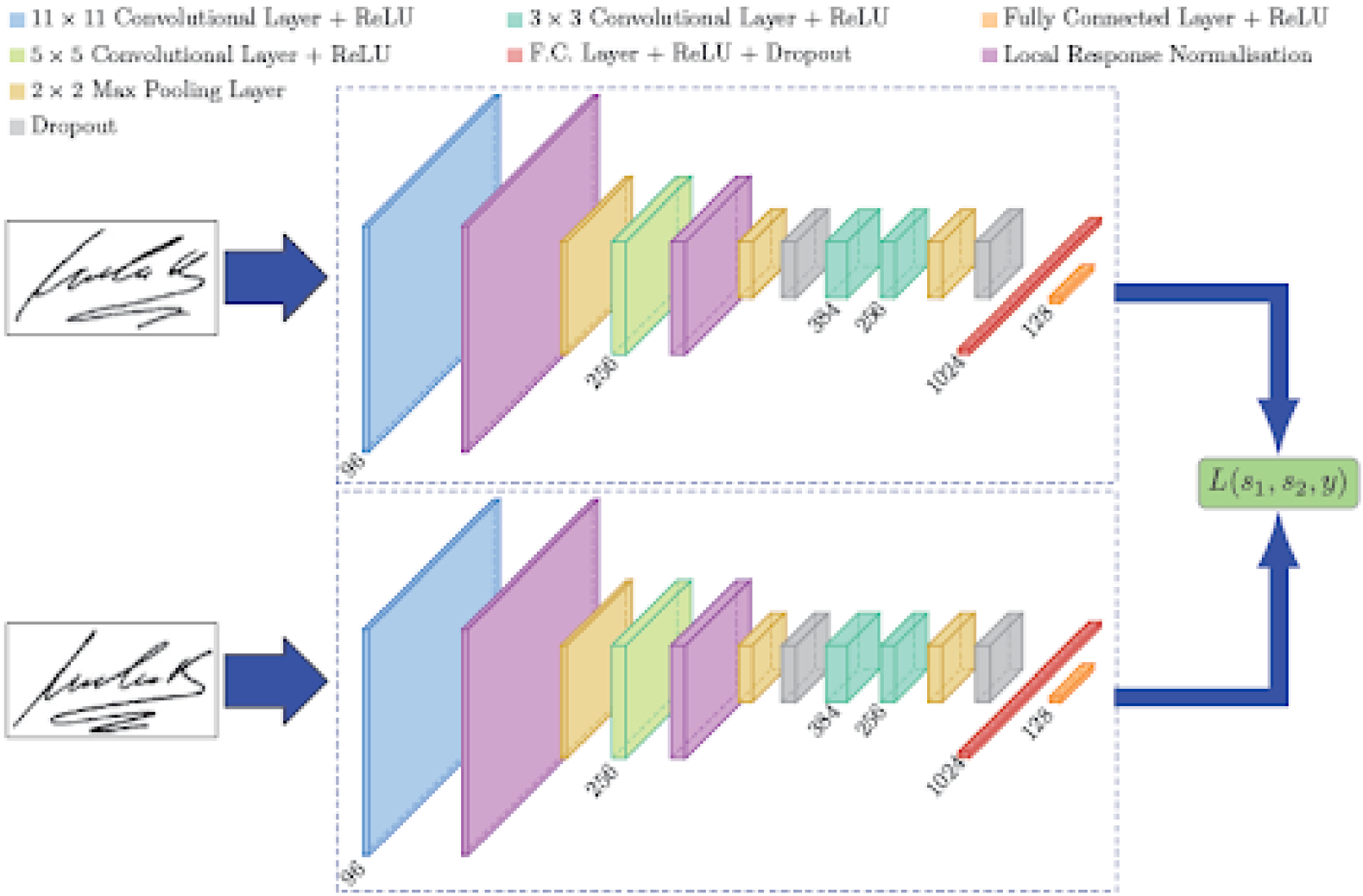
Returns : Updated parameters

Wrapping Function

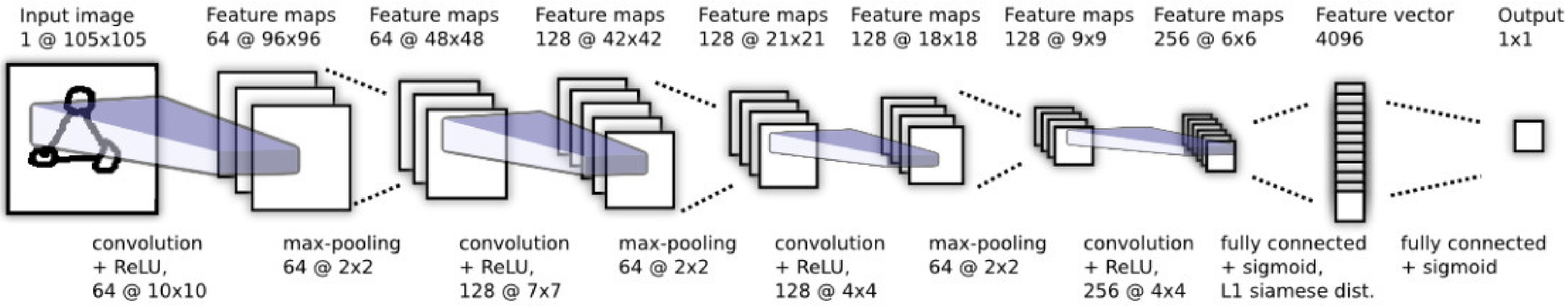
Takes : X, Y, learning Rate, N_iter

Returns : Trained parameters

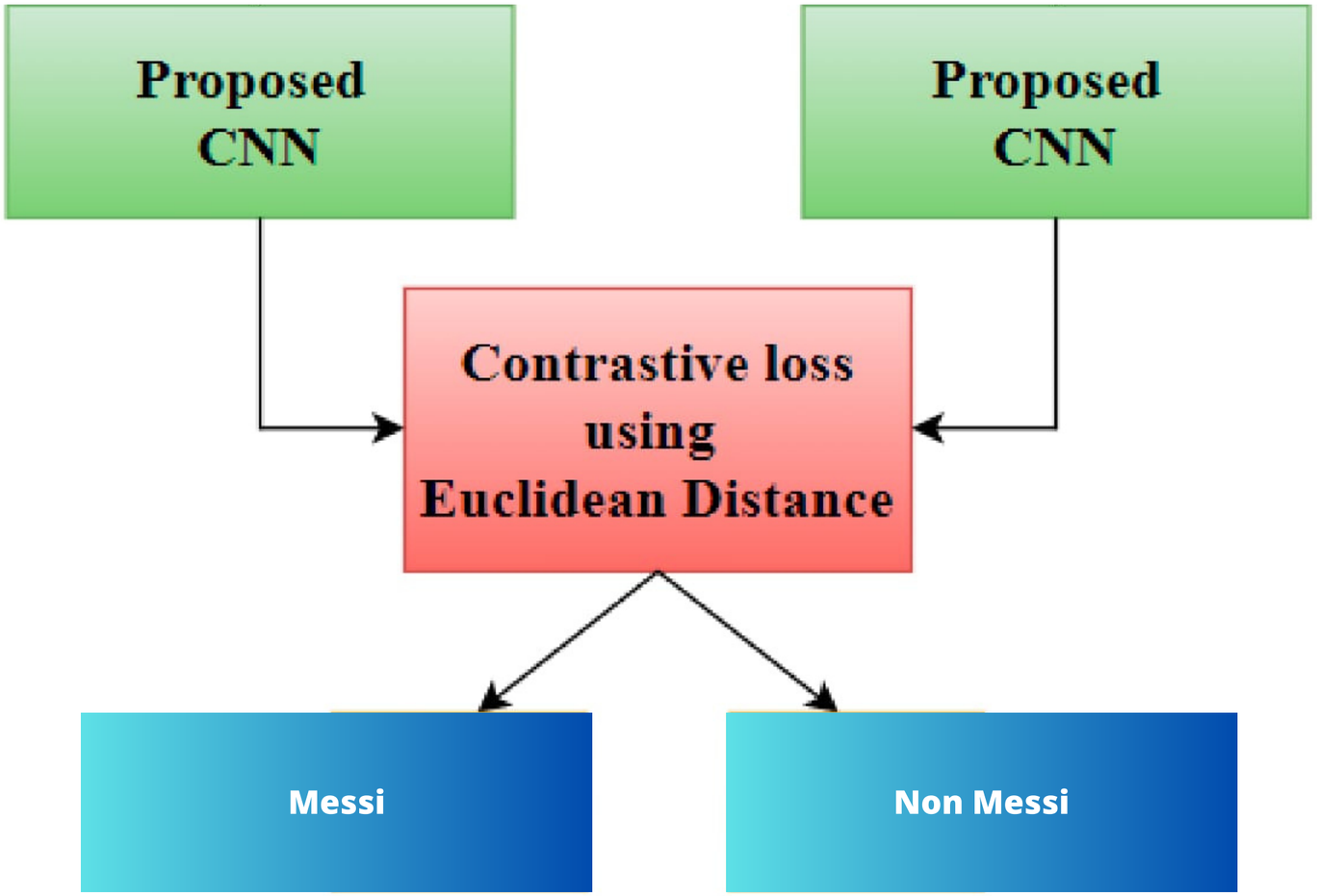
Siamese Neural Network


11 × 11 Convolutional Layer + ReLU
3 × 3 Convolutional Layer + ReLU
Fully Connected Layer + ReLU
5 × 5 Convolutional Layer + ReLU
F.C. Layer + ReLU + Dropout
Local Response Normalisation
2 × 2 Max Pooling Layer
Dropout
256
384
256
1024
128
$L(s_1, s_2, y)$

Embedding Layer



Distance Layer



Evaluation of the Model

Metrics

Recall
1.0

Precision
1.0

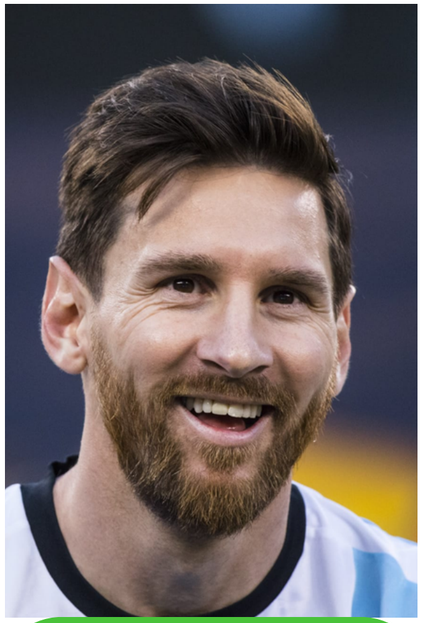
Optimization

Log Loss Function	Optimizer
Binary cross Entropy	Adam 10^(-4)

Verification & Results

Ziyech - False

Doppelganger - False



Messi - True

Thank you

For Your Attention

Any Questions?