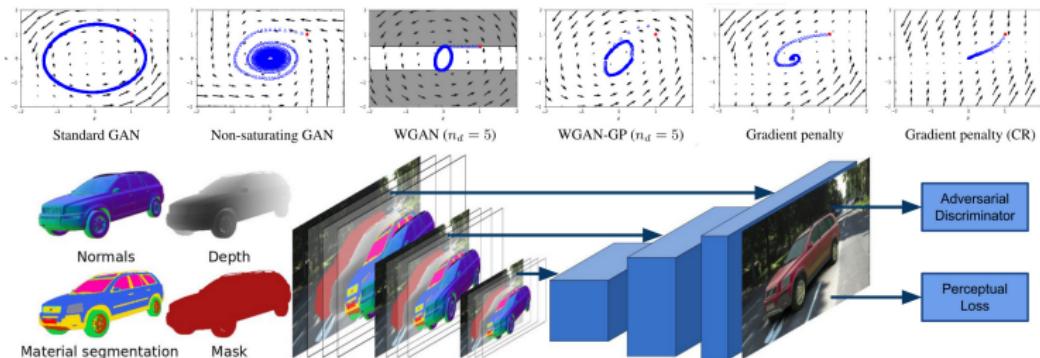


Introduction to Deep Learning

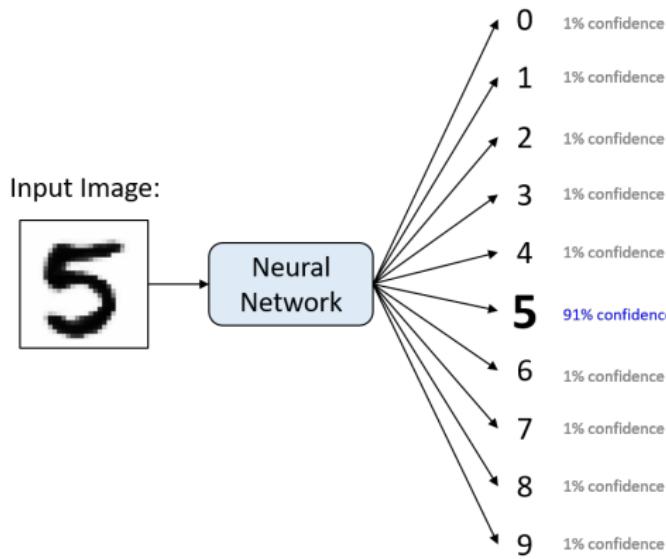
Distributed Lab

January 18, 2024



Spoiler

At the end of this lecture, we will build a digit recognition neural network!



Plan

1 Introduction

- Why do we care?
- Definitions
- Paradigms

2 Building the first Neural Network!

- Problem statement
- Forward propagation
- Making neural network adjust parameters

3 Coding Time!

- Tools
- Solution using Tensorflow

└ Introduction

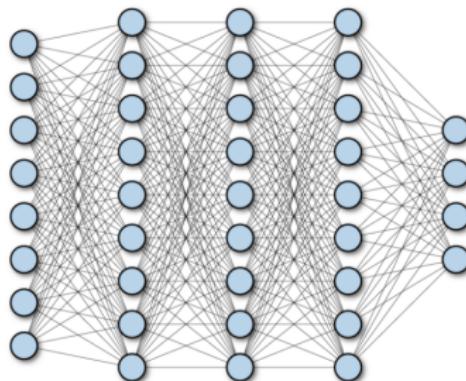
Introduction

└ Introduction

└ Why do we care?

Why do we care about Machine Learning?

- 1 Machine Learning becomes more and more used in many areas. Same might happen with blockchain and cryptography at some point.
- 2 Deep Learning is extensively used in security systems.
- 3 We have already encountered Deep Learning problems on “certain” projects.
- 4 Machine Learning is fun!



- └ Introduction

- └ Why do we care?

ML in Information Security: Liveness Detection

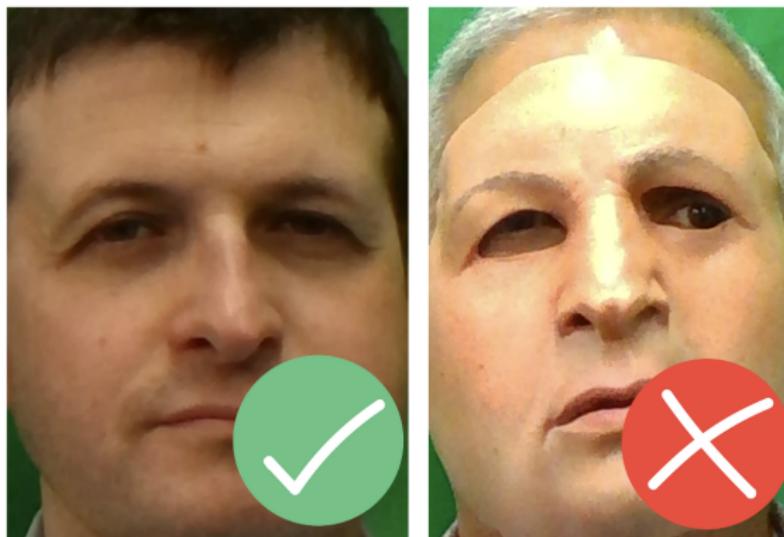


Figure: Liveness Detection. See our recently published paper:
<https://ceur-ws.org/Vol-3608/paper19.pdf>

└ Introduction

└ Why do we care?

ML in Information Security: Face Recognition

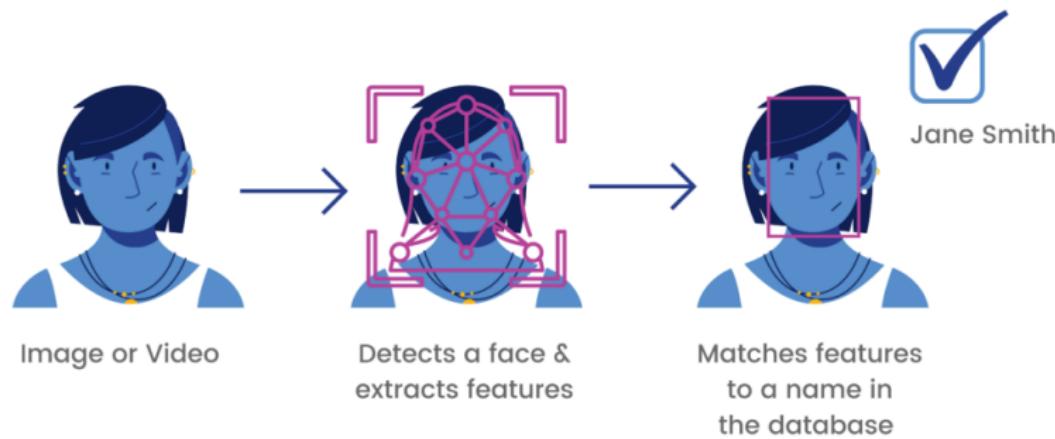


Figure: Face Recognition problem

- └ Introduction

- └ Why do we care?

ML in Information Security: Cancelable Biometrics

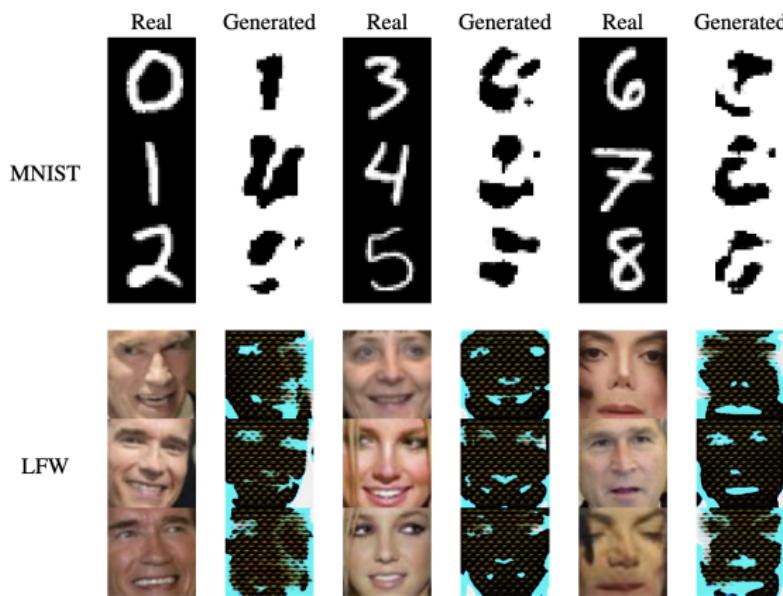


Figure: Cancelable Biometrics. Real and generated images are identifiable by the neural network

ML is fun! Pix2Pix

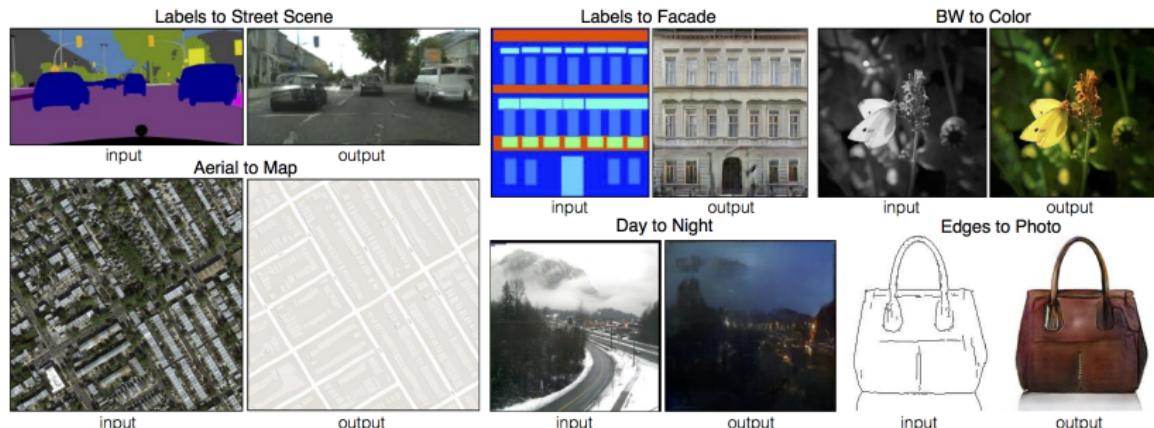


Figure: Image-to-image translation. *Pix2Pix*. See this url for reference:
phillipi.github.io/pix2pix/

- └ Introduction

- └ Why do we care?

ML is fun! Neural Transfer



Content Image



Style Image



Neural Style Transfer

Figure: Neural transfer. See this link if you want to learn more:
<https://www.v7labs.com/blog/neural-style-transfer>.

ML is fun! ChatGPT

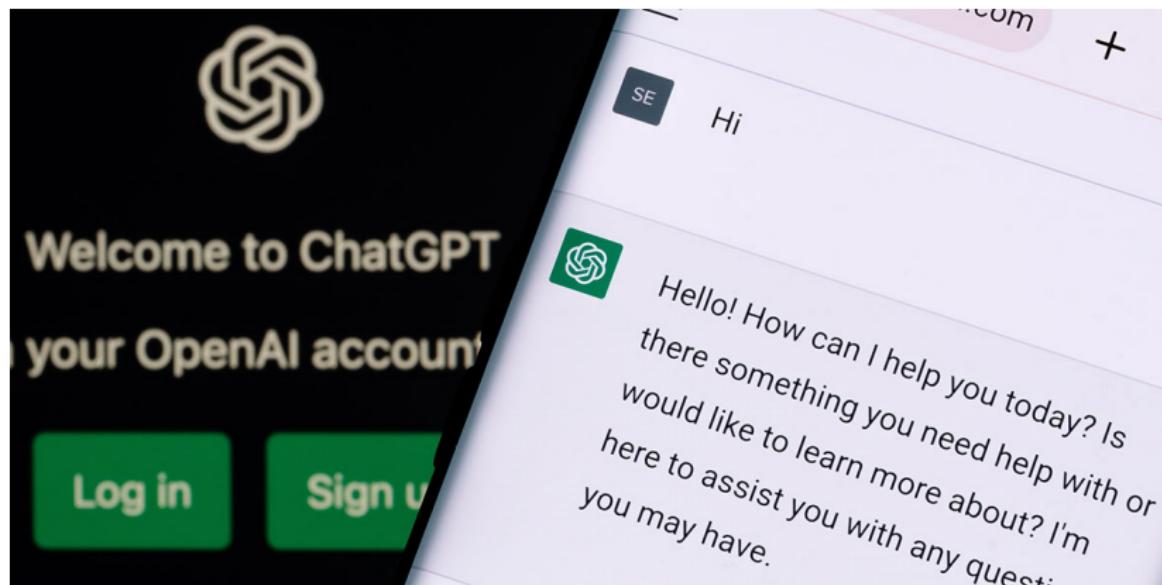


Figure: Do I actually need to explain what it is?

AI vs ML vs DL

ARTIFICIAL INTELLIGENCE VS MACHINE LEARNING VS DEEP LEARNING

① Artificial Intelligence

Development of smart systems and machines that can carry out tasks that typically require human intelligence

② Machine Learning

Creates algorithms that can learn from data and make decisions based on patterns observed

Require human intervention when decision is incorrect

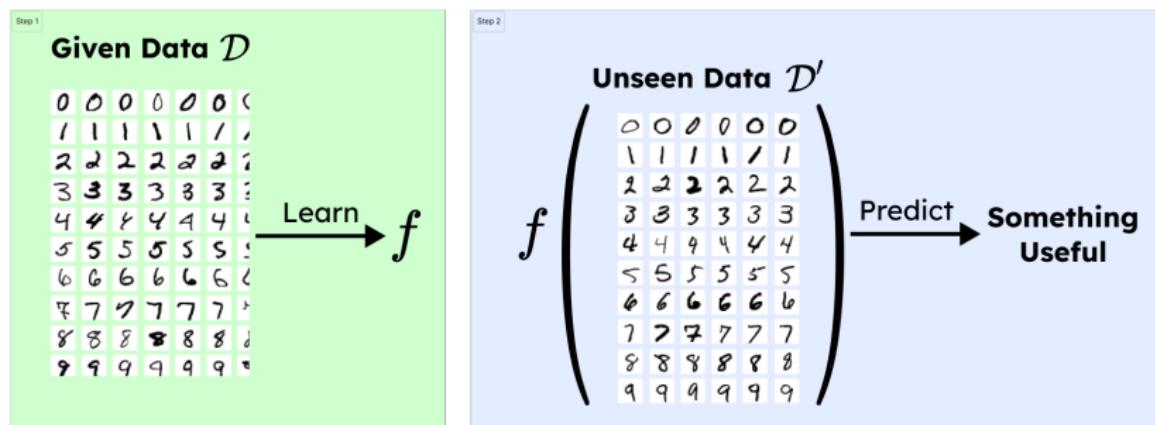
③ Deep Learning

Uses an artificial neural network to reach accurate conclusions without human intervention

What is Machine Learning?

Informal definition

Machine Learning is a field of study that learns to build a certain **function** f , based on the given data \mathcal{D} , that can give useful information about new upcoming data \mathcal{D}_{new} following the same distribution as \mathcal{D} .

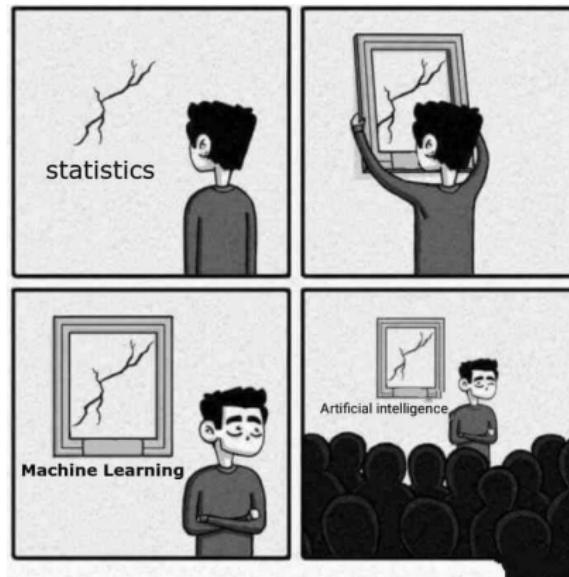


A few words about math

Does ML involve a lot of math? **Yes.**

Do you necessarily need to know *advanced* math to do ML? **No.**

Do you need *basic* understand of math to do ML? **Yes.**



Math in Papers: Good Example

Papers **love** formality, but the core is intuitive and quite often straightforward.



Figure 2. Model structure. Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.



Figure 3. The Triplet Loss minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

motivated in [19] in the context of nearest-neighbor classification. Here we want to ensure that an image x_i^a (*anchor*) of a specific person is closer to all other images x_i^p (*positive*) of the same person than it is to any image x_i^n (*negative*) of any other person. This is visualized in Figure 3.

Thus we want,

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 , \quad (1)$$

$$\forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathcal{T} . \quad (2)$$

where α is a margin that is enforced between positive and negative pairs. \mathcal{T} is the set of all possible triplets in the training set and has cardinality N .

The loss that is being minimized is then $L =$

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ .$$

Figure: FaceNet paper. See here for reference.

Supervised Learning

Supervised Learning

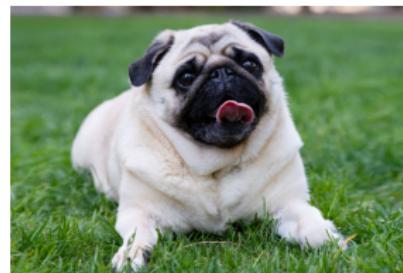
Given a pair of inputs and desired outputs (labels)

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n),$$

build a function f that can effectively map inputs to outputs.

Example

Given a set of labeled images of dogs and cats, build an NN that predicts whether an image belongs to a cat or a dog.



Dog



Cat

Figure: Supervised learning example

Unsupervised Learning

Supervised Learning

Given a dataset with unlabeled data x_1, x_2, \dots, x_n , explore patterns in data.

Example

Document analysis – given a vast library of different research papers, put them into groups according to the selected group of criteria.

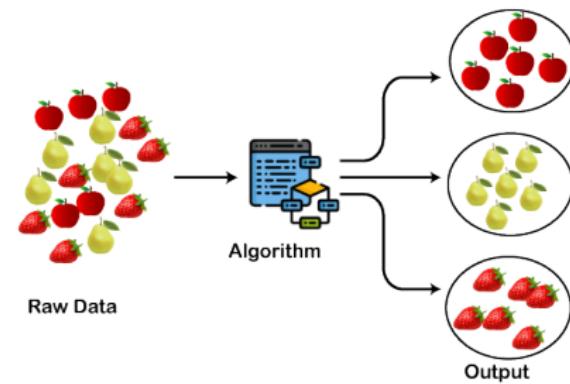


Figure: Clustering task

Reinforcement Learning

Supervised Learning

Producing actions a_1, a_2, \dots, a_n which affect the environment, and receiving rewards r_1, \dots, r_m learn to act such that it maximizes the reward.

Example

Writing a bot that can complete levels in a videogame.

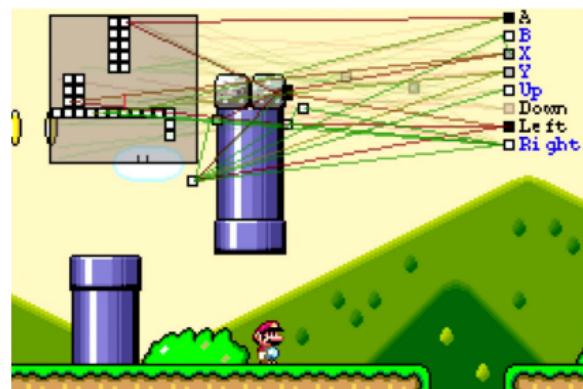


Figure: Reinforcement Learning example

└ Introduction

└ Paradigms

A bit of interactivity: object detection

Question #1

Given a set of images with people, dogs, and cars marked, build a model that detects these objects on the photo. Is that a supervised, unsupervised, or reinforcement task? Why?

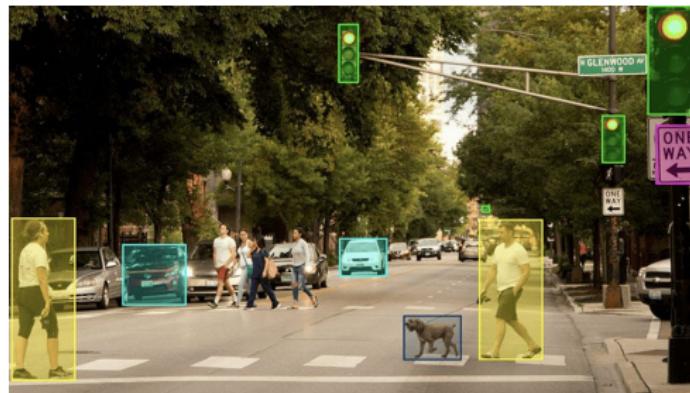
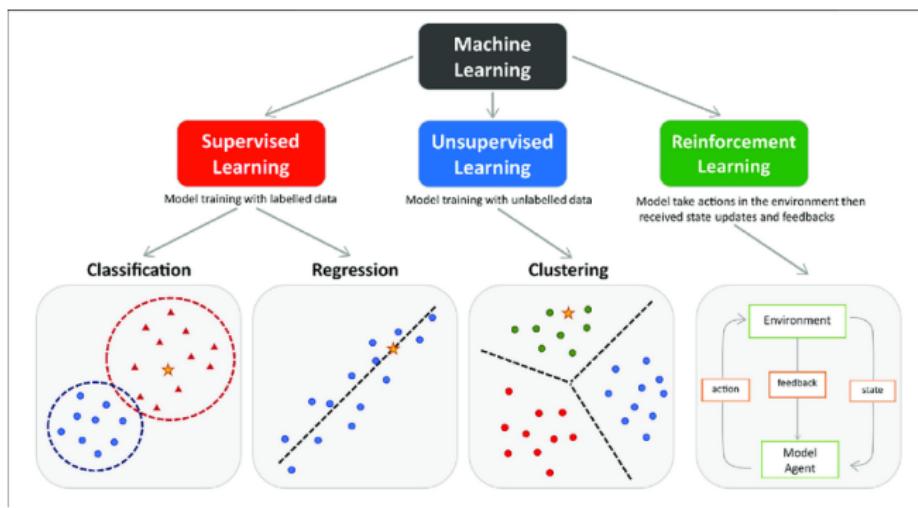


Figure: Illustration for question 1

A bit of interactivity: your own examples

Question #2

Give your own examples of (a) – supervised learning, (b) – unsupervised learning, and (c) – reinforcement learning.



Building the first Neural Network!

Problem Statement

“Hello World” in Machine Learning

Write a program that, based on the 28×28 grayscale image of a digit, predicts what digit it is. You are given a set of images $\{x_i\}_{i=1}^n$ and corresponding labels $\{y_i\}_{i=1}^n$, $y_i \in \{0, \dots, 9\}$.

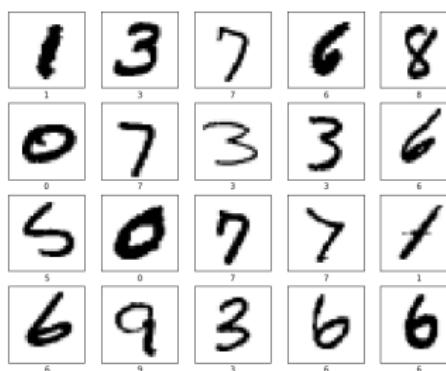


Figure: MNIST dataset

└ Building the first Neural Network!

└ Problem statement

Neural Network

To define what **Neural Network** is, we need to define:

- 1 What is a **neuron**?
- 2 How **neurons** form a **network**?

The first is simple!

Definition

Neuron is a number. Really. That's just it...

A bit more formally, this is a node in the network, possessing an **activation**, which is just a number indicating how active is the neuron.

Neurons in the context of an image

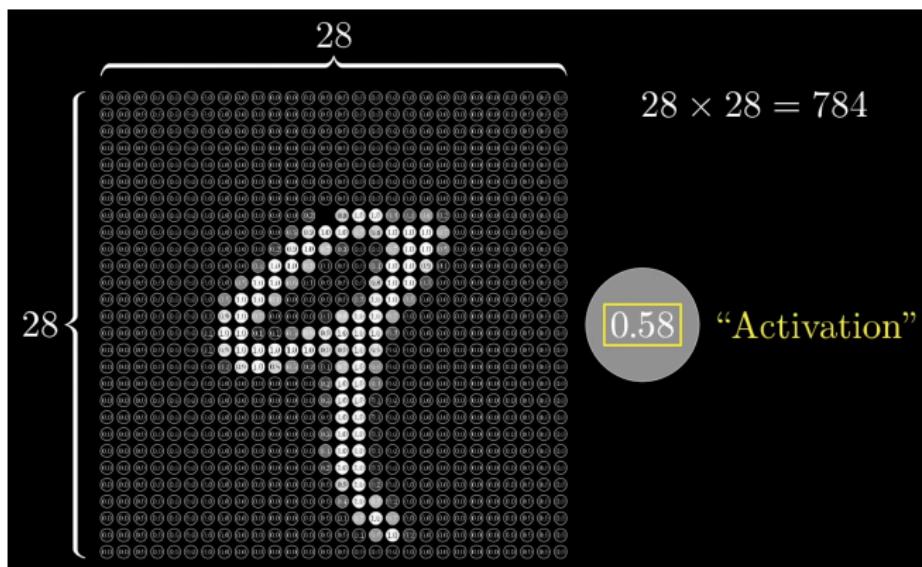


Figure: Each pixel of an image is an individual neuron, having an activation between 0 and 1. Image taken from 3Blue1Brown video.

Flattening...

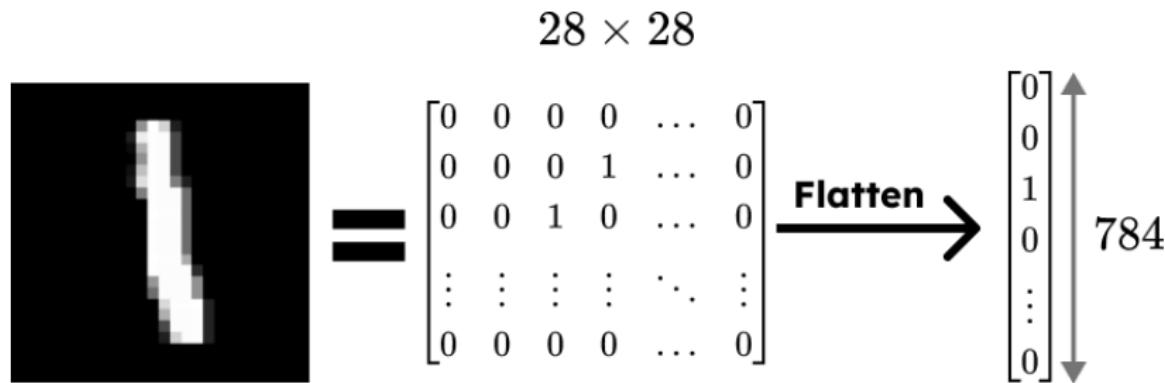


Figure: Forming a single vector from the image

Neural Network Layer

Denote i^{th} activation (the neuron's value) in the ℓ^{th} layer as $a_i^{(\ell)}$.

Let us define activation of $a_1^{(2)}$ in terms of first-layer activations $a_i^{(1)}$, $i = 1, 2, \dots, 784$.

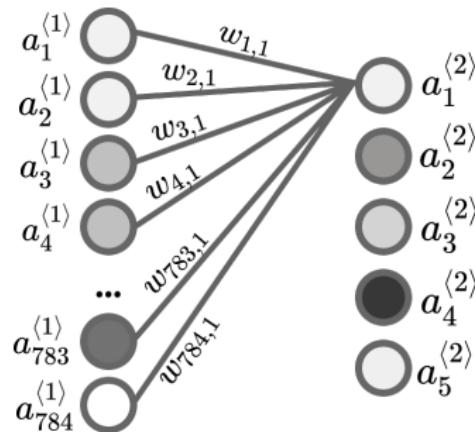


Figure: First hidden layer in our network

Contribution from the first layer

Suppose that “importance” (formally, a weight) of the first neuron in the first layer contributing to the first neuron in the second layer is $w_{1,1}$. Then, $a_1^{(1)}$ contribution to the value of $a_1^{(2)}$ is $w_{1,1}a_1^{(1)}$.

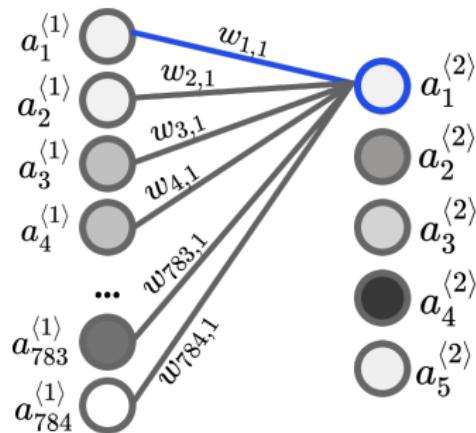


Figure: First hidden layer in our network

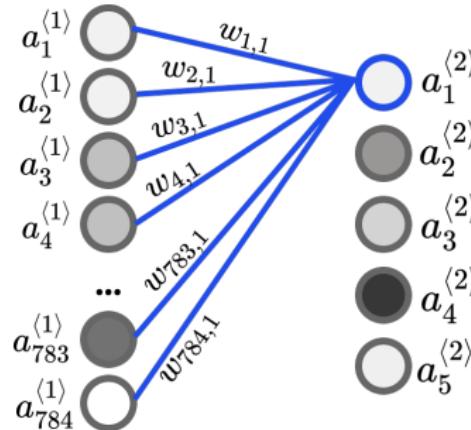
└ Building the first Neural Network!

└ Forward propagation

Total contribution

Similarly, second neuron of the first layer has a contribution of $w_{2,1}a_2^{(1)}$ to the first neuron in the second layer. Similarly, the sum of all contributions is:

$$a_1^{(2)} = w_{1,1}a_1^{(1)} + w_{2,1}a_2^{(1)} + \cdots + w_{784,1}a_{784}^{(1)} = \sum_{i=1}^{784} w_{i,1}a_i^{(1)}$$



A bit of interactivity

Question #1

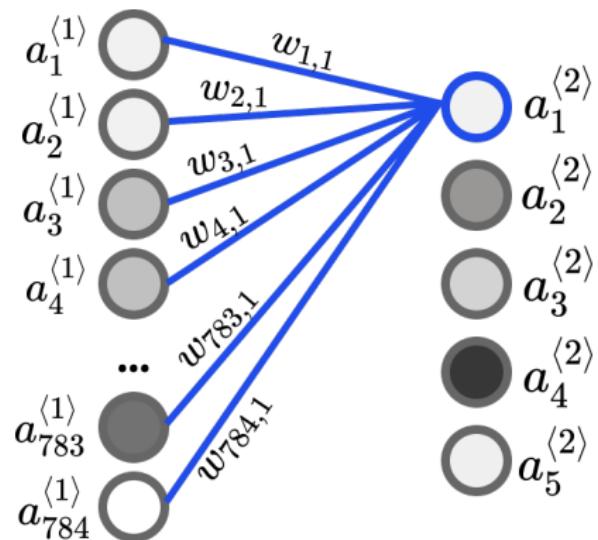
What value of $a_1^{(2)}$ we would get if all $w_{i,1}$ were 0?

Question #2

What value of $a_1^{(2)}$ we would get if all $w_{i,1}$ were 0 except for $w_{j,1} = 1$?

Question #3

What if all $w_{i,1} = 1$?



↳ Building the first Neural Network!

└ Forward propagation

Summing for every activation

And now we have **TONS** of connections:

$$a_1^{(2)} = w_{1,1}a_1^{(1)} + w_{2,1}a_2^{(1)} + \cdots + w_{784,1}a_{784}^{(1)}$$

$$a_2^{(2)} = w_{2,1}a_1^{(1)} + w_{2,2}a_2^{(1)} + \cdots + w_{784,2}a_{784}^{(1)}$$

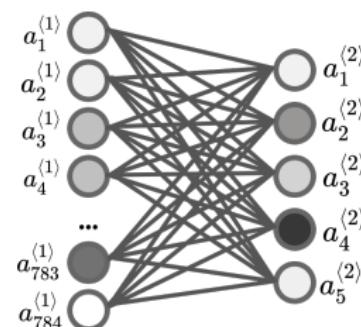
•
•
•

$$a_5^{(2)} = w_{1,5}a_1^{(1)} + w_{2,5}a_2^{(1)} + \cdots + w_{784,5}a_{784}^{(1)}$$

Question

How can we write this expression concisely?

Hint: Maybe vector and matrix notation could help?



- Building the first Neural Network!

- Forward propagation

A bit of Linear Algebra

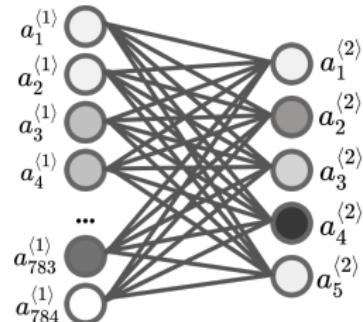
Answer:

$$\begin{bmatrix} a_1^{(2)} \\ \vdots \\ a_5^{(2)} \end{bmatrix} = \begin{bmatrix} w_{1,1} & \dots & w_{784,1} \\ \vdots & \ddots & \vdots \\ w_{1,5} & \dots & w_{784,5} \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_{784}^{(1)} \end{bmatrix}$$

Or even more concisely!

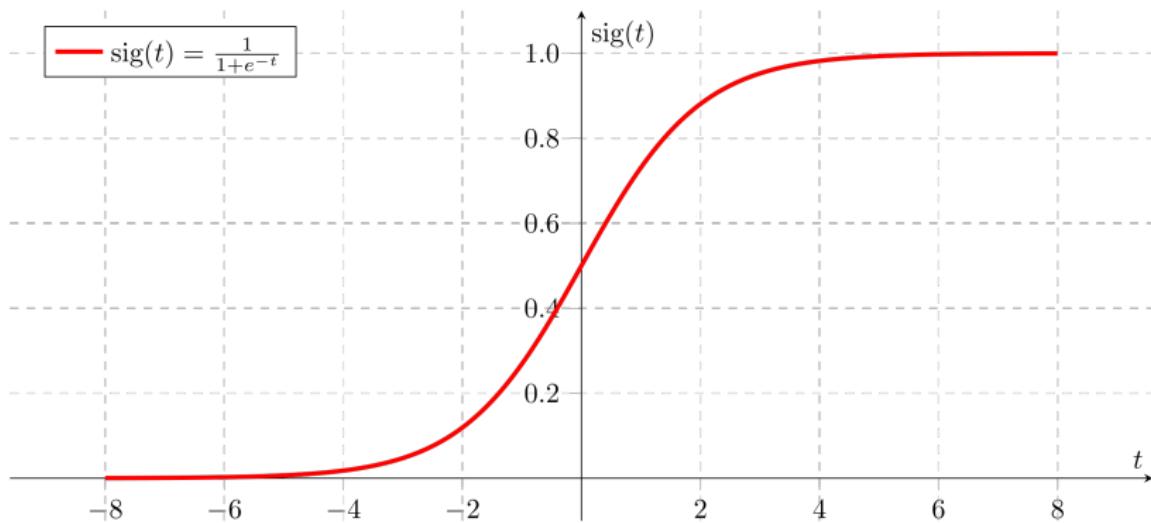
$$\mathbf{a}^{(\ell+1)} = \mathbf{W}^{(\ell)} \mathbf{a}^{(\ell)}$$

However, in this case, elements of $\mathbf{a}^{(\ell+1)}$ can take any values on the real line, but we want values in range $(0, 1)$. What to we do?



Activation function

Let us apply sigmoid function $\sigma(x)$ to all retrieved values! It will map any value to the interval $(0, 1)$.



- Building the first Neural Network!

- Forward propagation

Neural Network Layer

Thus, let us do

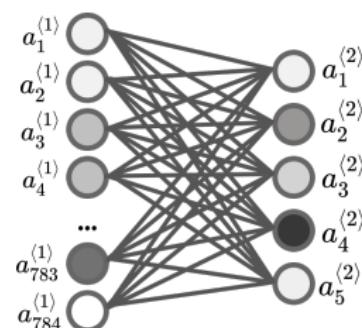
$$\mathbf{a}^{\langle \ell+1 \rangle} = \sigma(\mathbf{W}^{\langle \ell \rangle} \mathbf{a}^{\langle \ell \rangle})$$

To make neural network a bit more complicated, let's add an “offset”:

$$\mathbf{a}^{\langle \ell+1 \rangle} = \sigma(\mathbf{W}^{\langle \ell \rangle} \mathbf{a}^{\langle \ell \rangle} + \mathbf{b}^{\langle \ell \rangle})$$

Definitions

Bias is an “offset” vector in each layer ($\mathbf{b}^{\langle \ell \rangle}$ for our notation). Function applied to the obtained vector is called an **activation function**.



└ Building the first Neural Network!

└ Forward propagation

Another motivation for activation function

Suppose that we:

- Don't have any activation function.
- Neural network consists of $n_L + 1$ layers with zero biases.

Then, based on our definition of a neural network:

$$\mathbf{a}^{(2)} = \mathbf{W}^{(1)} \mathbf{a}^{(1)},$$

$$\mathbf{a}^{(3)} = \mathbf{W}^{(2)} \mathbf{a}^{(2)},$$

⋮

$$\mathbf{a}^{(n_L+1)} = \mathbf{W}^{(n_L)} \mathbf{a}^{(n_L)}.$$

Therefore, $\mathbf{a}^{(n_L+1)} = \mathbf{W}^{(n_L)} \mathbf{W}^{(n_L-1)} \dots \mathbf{W}^{(1)} \mathbf{a}^{(1)} = \widetilde{\mathbf{W}} \mathbf{a}^{(1)}$ – that's a **linear** dependence – no good.

└ Building the first Neural Network!

└ Forward propagation

Our architecture

By putting everything together, we have a complex function $f(x; \theta)$ where θ is a set of weights and biases.

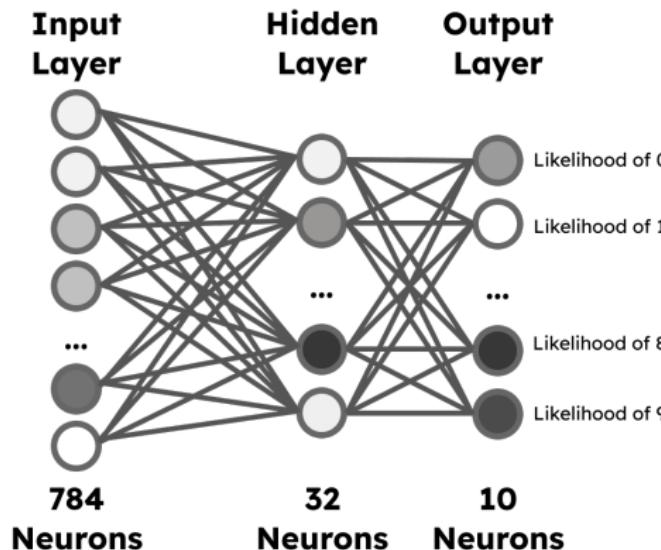


Figure: Our proposed architecture for digit recognition

└ Building the first Neural Network!

 └ Making neural network adjust parameters

Loss function motivation

Until this point, we haven't used dataset, so let us discuss how to apply it.

Core idea: neural network is nothing but a **HUGE** parametric function $f(\mathbf{x}; \theta)$ with **A LOT** of parameters θ , which somehow outputs us a label based on input \mathbf{x} .

Question

After inputting image of a digit 1, you get the following output:

[0.01, 0.05, 0.7, 0.8, 0.05, 0.3, 0.5, 0.02, 0.07, 0.03]

Is it a good result? What about

[0.001, 0.99, 0.01, 0.01, 0.02, 0.003, 0.1, 0.002, 0.001, 0.05]?

Hint: i^{th} position's value represents the likelihood of an image i .

Finding Ideas...

Question

What measure can you propose to indicate how bad the neural network performs?

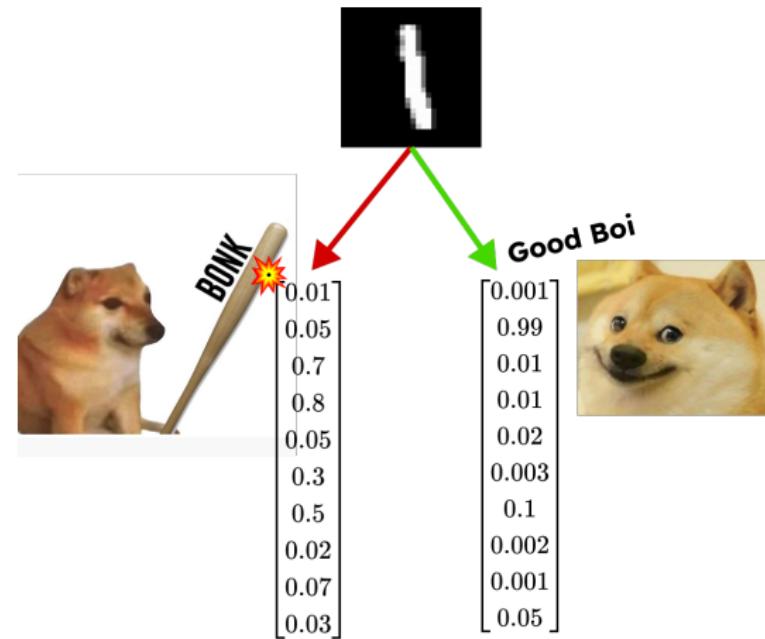


Figure: Bad and good bois

- Building the first Neural Network!

- Making neural network adjust parameters

Mean-squared error

Suppose that the true label is a vector \mathbf{y} , where the i^{th} position is 1, if the corresponding image \mathbf{x} is of digit i , and 0 otherwise.

Example

If \mathbf{x} is an image of 5, then the corresponding label \mathbf{y} is

$$\mathbf{y} = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$$

Ideally, we want our neural network to output \mathbf{y} , but it outputs $f(\mathbf{x}; \theta) = \hat{\mathbf{y}}$. Define an error as
 $\ell(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \sum_{i=1}^{10} (y_i - \hat{y}_i)^2$. Our goal is to minimize this error for all pairs of (\mathbf{x}, \mathbf{y}) from our dataset:

$$L(\theta) = \sum_{i=1}^n \ell(\mathbf{y}_i, f(\mathbf{x}_i; \theta))$$

- Building the first Neural Network!

- Making neural network adjust parameters

Interactivity!

Question

Suppose we want to have a prediction $\hat{\mathbf{y}}$ as close as possible to the true label \mathbf{y} . Is the loss function below a good candidate? Why?

$$\ell(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{\|\mathbf{y} - \hat{\mathbf{y}}\|}$$

└ Building the first Neural Network!

 └ Making neural network adjust parameters

What is takes to build a supervised neural network?

- 1 Find the dataset: a set of inputs and desired outputs.
- 2 Choose the structure: number of layers, activation functions, what is “input” and what is “output”.
- 3 Choose the loss function: what is considered to be a “good” and “bad” outputs.
- 4 Defining training parameters: speed of learning, optimizer (we do not discuss it today).

Key takeaway

Building a neural network is nothing but specifying a function $f(X; \theta)$ with tons of parameters, feeding a list of inputs and outputs, and finding such parameters $\hat{\theta}$, which minimizes the error on the given dataset.

└ Coding Time!

Coding Time!

└ Coding Time!

└ Tools

Tools for Deep Learning / Machine Learning

Most commonly used

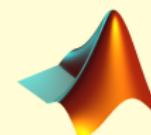


TensorFlow



PyTorch

Less frequently used



julia

Figure: Tools used for Deep and Machine Learning

Coding Time!

Tools

A Neural Network Playground

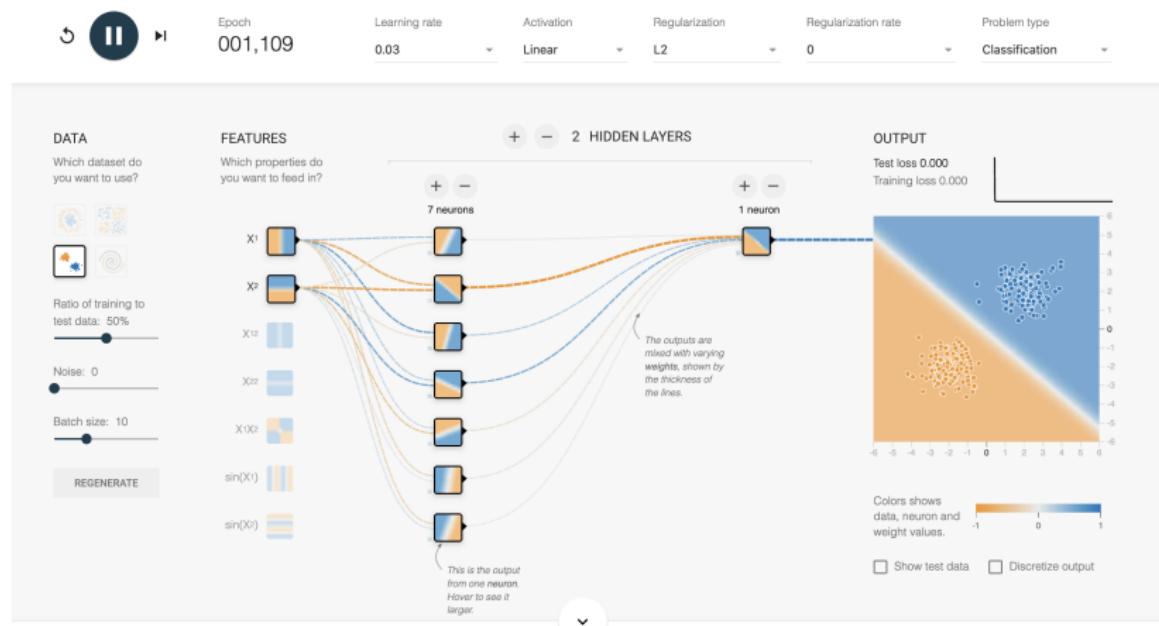


Figure: Tensorflow Playground. See <https://playground.tensorflow.org>

- Coding Time!

- Solution using Tensorflow

Importing packages

```
1 import tensorflow as tf # For defining and training  
    the neural network  
2 import numpy as np # For convenient use of high-  
    dimensional arrays  
3 from keras import datasets # For loading the MNIST  
    dataset  
4  
5 print(f'Using TensorFlow {tf.__version__}')  
6 np.set_printoptions(precision=3)  
7
```

Listing 1: Importing packages we would use

- Coding Time!

- Solution using Tensorflow

Importing MNIST dataset

```
1 (X, y), _ = datasets.mnist.load_data()  
2 X = X / 255.0  
3 y = np.array([[1.0 if i == label else 0.0 for i in  
               range(10)] for label in y])  
4
```

Listing 2: Importing the MNIST dataset

- Coding Time!

- Solution using Tensorflow

Defining the neural network structure

```
1 neural_network = tf.keras.models.Sequential([
2     tf.keras.layers.Flatten(input_shape=(28,28)),
3     tf.keras.layers.Dense(
4         32,
5         name='hidden_layer',
6         activation='sigmoid'
7     ),
8     tf.keras.layers.Dense(
9         10,
10        name='output_layer',
11        activation='sigmoid'
12    )
13 ])
14 ])
```

Listing 3: Defining the neural network structure

- Coding Time!

- Solution using Tensorflow

Fitting into the neural network

```
1 optimizer = tf.keras.optimizers.legacy.Adam(  
2     learning_rate=1e-5)  
3 neural_network.compile(loss='mse', optimizer=optimizer  
4 )  
5 neural_network.fit(  
6     X,  
7     y,  
8     epochs=30,  
9     verbose=1,  
10    validation_split=0.2  
11 )
```

Listing 4: Launch the training session

- Coding Time!

- Solution using Tensorflow

Testing!

```
1 # Making prediction
2 index = 1 # Put your index to test
3 prediction = neural_network.predict(np.expand_dims(X[
    index], axis=0), verbose=0)
4
5 # Displaying results
6 print(f'Prediction in raw format: {prediction}')
7 show_image(X[index], np.argmax(prediction[0]))
8
```

Listing 5: Launch the testing

- Coding Time!

- Solution using Tensorflow

Thank you for your attention!