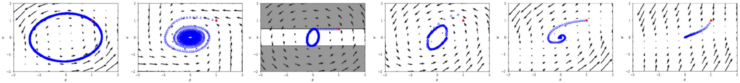
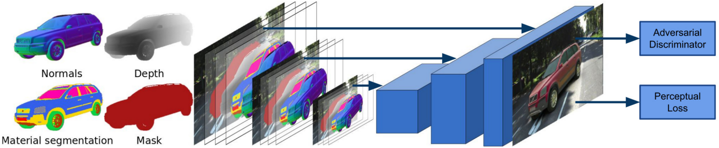
Introduction to Deep Learning

Distributed Lab

January 18, 2024

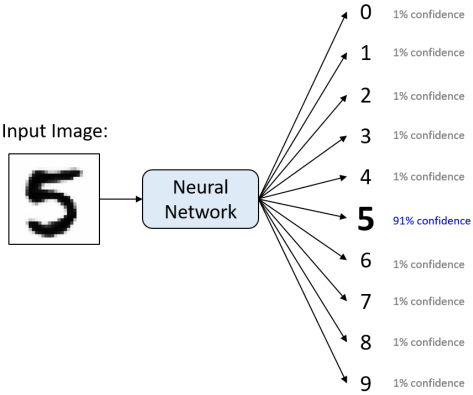


Standard GAN | Non-saturating GAN | WGAN ($n_d = 5$) | WGAN-GP ($n_d = 5$) | Gradient penalty | Gradient penalty (CR)



Spoiler

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

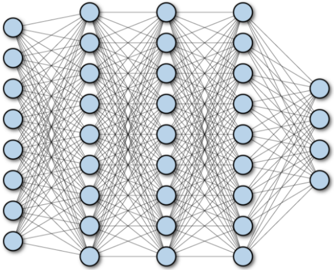


Plan

Introduction

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!



ML in Information Security: Liveness Detection

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

ML in Information Security: Face Recognition

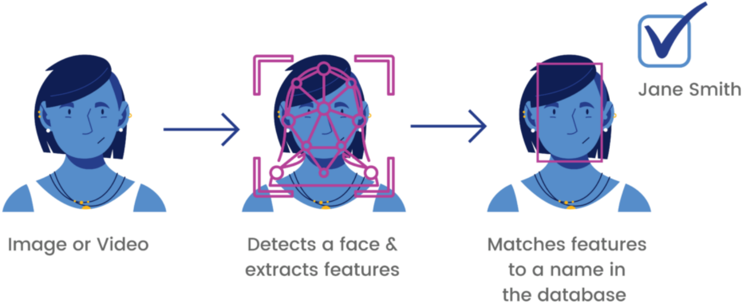


Figure: Face Recognition problem

ML in Information Security: Cancelable Biometrics

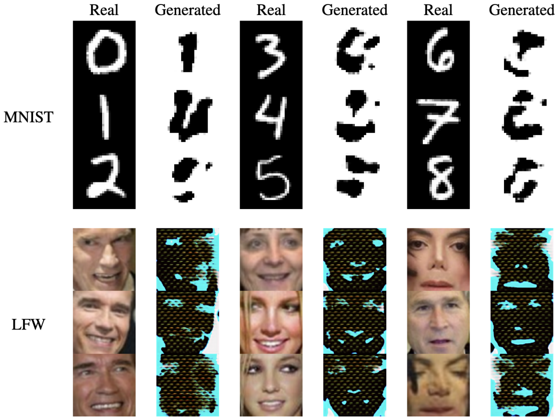


Figure: Cancelable Biometrics. Real and generated images are identifiable by the neural network

ML is fun! Pix2Pix

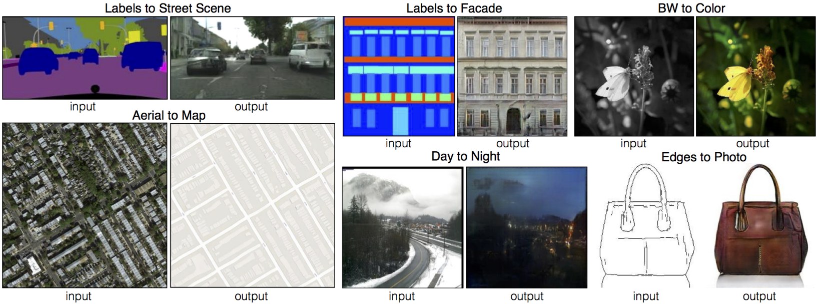


Figure: Image-to-image translation. *Pix2Pix*. See this url for reference: phillipi.github.io/pix2pix/

ML is fun! Neural Transfer

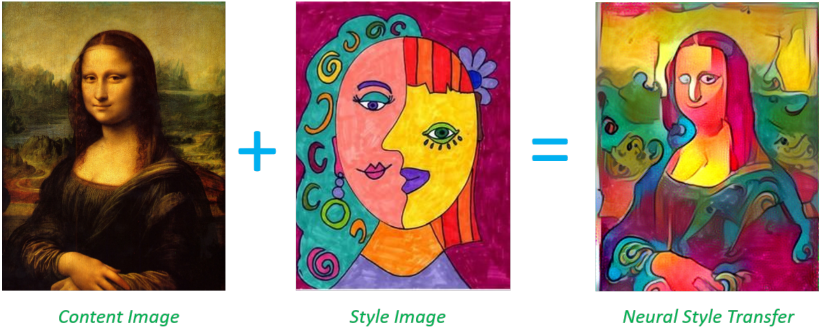


Figure: Neural transfer. See this link if you want to learn more: https://www.v7labs.com/blog/neural-style-transfer.

ML is fun! ChatGPT

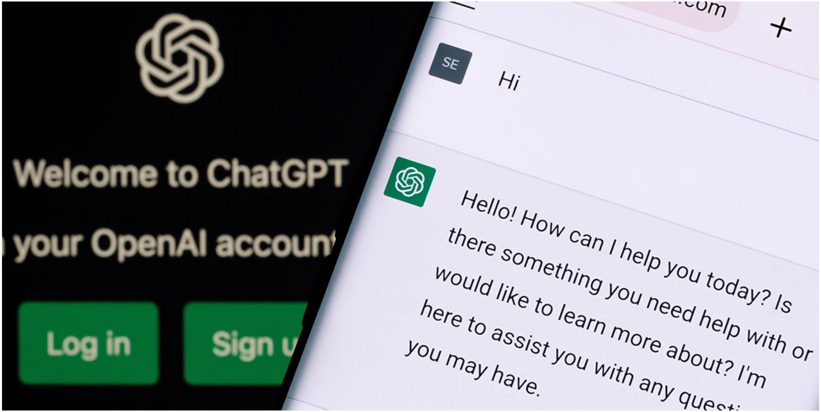
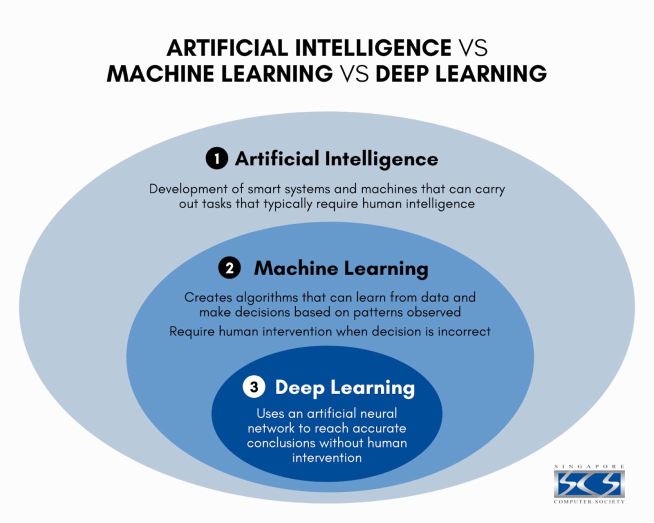


Figure: Do I actually need to explain what it is?

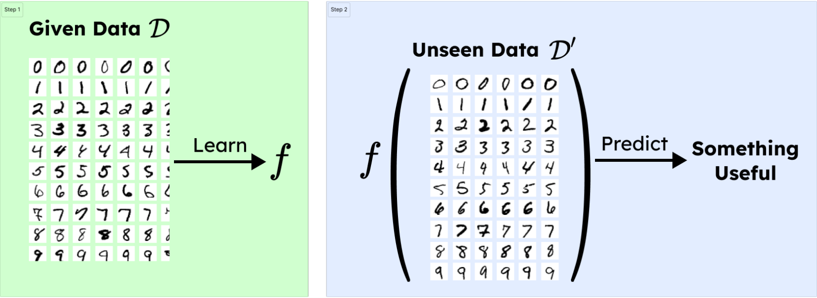
AI vs ML vs DL


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
SINGAPORE SCS COMPUTER SOCIETY

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 $\mathcal{D}_{\text{new}}$ following the same distribution as $\mathcal{D}$.

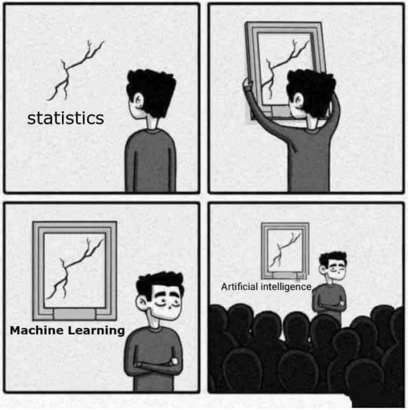


A few words about math

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.

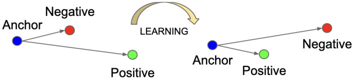


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

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,

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

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

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)

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

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 $x_1, x_2, \ldots, x_n$, 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.

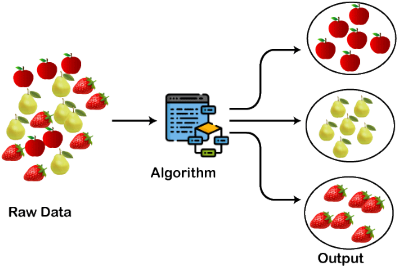


Figure: Clustering task

Reinforcement Learning

Supervised Learning

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

Example

Writing a bot that can complete levels in a videogame.

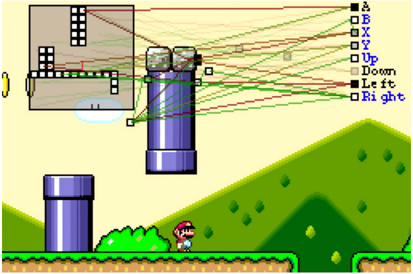


Figure: Reinforcement Learning example

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?

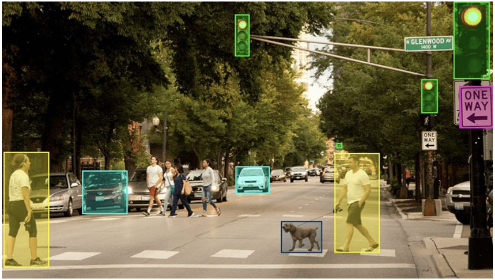
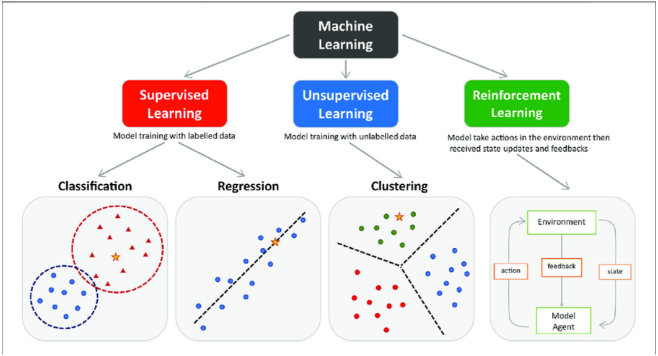


Figure: Illustration for question 1

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 $\{y_i\}_{i=1}^n$, $y_i \in \{0, \dots, 9\}$.

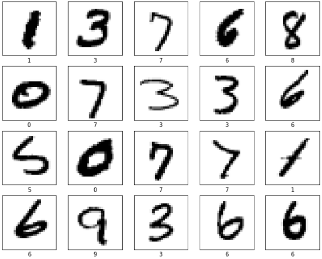


Figure: MNIST dataset

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

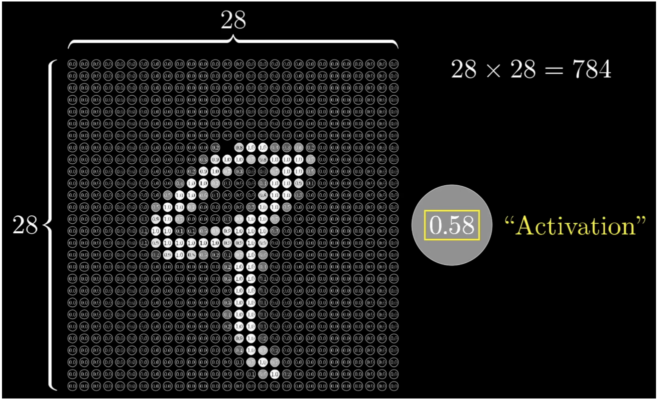


Figure: Each pixel of an image is an individual neuron, having an activation between 0 and 1. Image taken from 3Blue1Brown video.

Flattening...

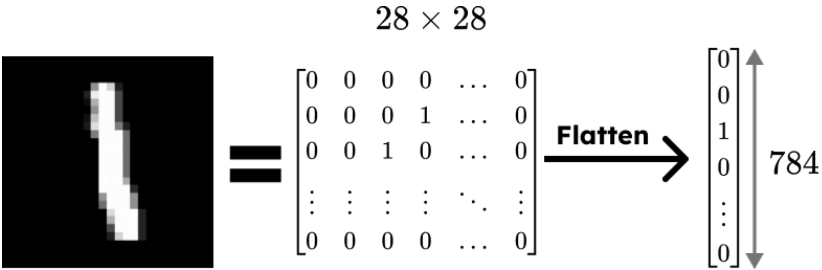


Figure: Forming a single vector from the image

Neural Network Layer

Denote i^{th} activation (the neuron's value) in the ℓ^{th} layer as $a_i^{\langle \ell \rangle}$.
Let us define activation of $a_1^{\langle 2 \rangle}$ in terms of first-layer activations $a_i^{\langle 1 \rangle}$, $i = 1, 2, \ldots, 784$.

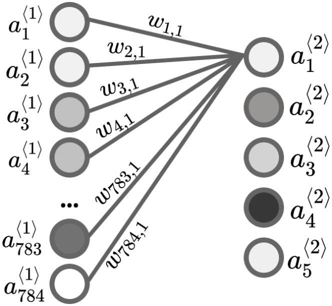


Figure: First hidden layer in our network

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, $a_1^{\langle 1 \rangle}$ contribution to the value of $a_1^{\langle 2 \rangle}$ is $w_{1,1}a_1^{\langle 1 \rangle}$.

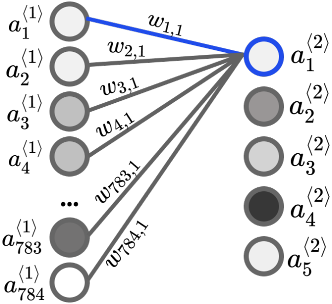
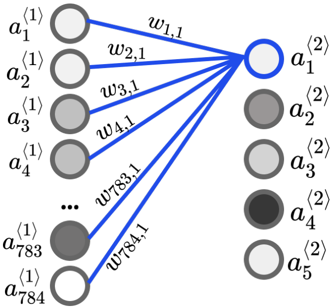


Figure: First hidden layer in our network

Total contribution

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

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



A bit of interactivity

Question #1

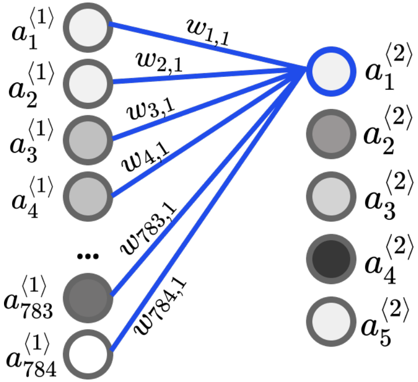
What value of $a_1^{\langle 2 \rangle}$ we would get if all $w_{i,1}$ were 0?

Question #2

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

Question #3

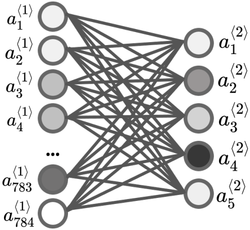
What if all $w_{i,1} = 1$?



Summing for every activation

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

$$a_1^{\langle 2 \rangle} = w_{1,1} a_1^{\langle 1 \rangle} + w_{2,1} a_2^{\langle 1 \rangle} + \cdots + w_{784,1} a_{784}^{\langle 1 \rangle}$$
$$a_2^{\langle 2 \rangle} = w_{2,1} a_1^{\langle 1 \rangle} + w_{2,2} a_2^{\langle 1 \rangle} + \cdots + w_{784,2} a_{784}^{\langle 1 \rangle}$$
$$\vdots$$
$$a_5^{\langle 2 \rangle} = w_{1,5} a_1^{\langle 1 \rangle} + w_{2,5} a_2^{\langle 1 \rangle} + \cdots + w_{784,5} a_{784}^{\langle 1 \rangle}$$



Question

How can we write this expression concisely?
Hint: Maybe vector and matrix notation could help?

A bit of Linear Algebra

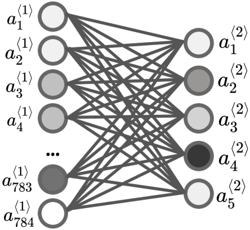
Answer:

$$\begin{bmatrix} a_1^{\langle 2 \rangle} \\ \vdots \\ a_5^{\langle 2 \rangle} \end{bmatrix} = \begin{bmatrix} w_{1,1} & \cdots & w_{784,1} \\ \vdots & \ddots & \vdots \\ w_{1,5} & \cdots & w_{784,5} \end{bmatrix} \begin{bmatrix} a_1^{\langle 1 \rangle} \\ \vdots \\ a_{784}^{\langle 1 \rangle} \end{bmatrix}$$

Or even more concisely!

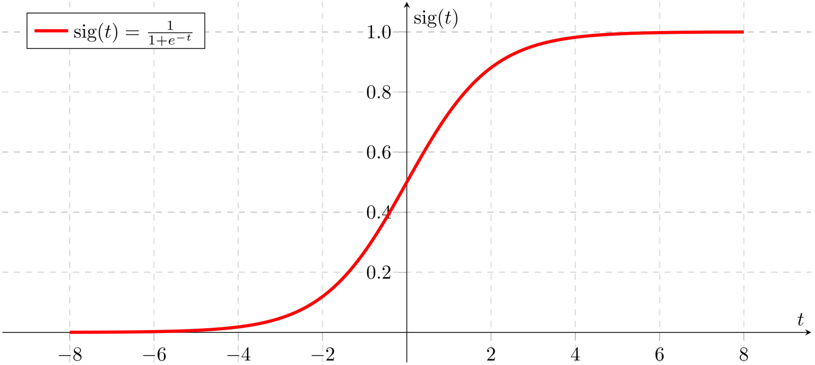
$$\mathbf{a}^{\langle \ell+1 \rangle} = \mathbf{W}^{\langle \ell \rangle} \mathbf{a}^{\langle \ell \rangle}$$

However, in this case, elements of $\mathbf{a}^{\langle \ell+1 \rangle}$ 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 $(0, 1)$.



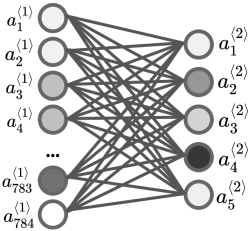
Neural Network Layer

Thus, let us do

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

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

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



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**.

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:

$$\begin{aligned} \mathbf{a}^{\langle 2 \rangle} &= \mathbf{W}^{\langle 1 \rangle}\mathbf{a}^{\langle 1 \rangle}, \\ \mathbf{a}^{\langle 3 \rangle} &= \mathbf{W}^{\langle 2 \rangle}\mathbf{a}^{\langle 2 \rangle}, \\ &\vdots \\ \mathbf{a}^{\langle n_L+1 \rangle} &= \mathbf{W}^{\langle n_L \rangle}\mathbf{a}^{\langle n_L \rangle}. \end{aligned}$$

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

Our architecture

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

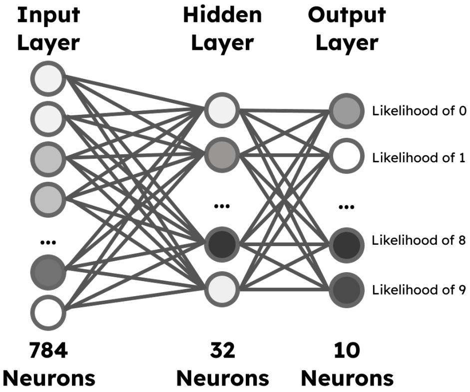


Figure: Our proposed architecture for digit recognition

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

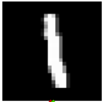
$$[0.001, 0.99, 0.01, 0.01, 0.02, 0.003, 0.1, 0.002, 0.001, 0.05]?$$

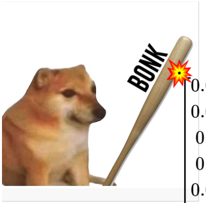
Hint: i^{th} position's value represents the likelihood of an image i.

Finding Ideas...

Question

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





$$\begin{bmatrix} 0.01 \\ 0.05 \\ 0.7 \\ 0.8 \\ 0.05 \\ 0.3 \\ 0.5 \\ 0.02 \\ 0.07 \\ 0.03 \end{bmatrix} \qquad \begin{bmatrix} 0.001 \\ 0.99 \\ 0.01 \\ 0.01 \\ 0.02 \\ 0.003 \\ 0.1 \\ 0.002 \\ 0.001 \\ 0.05 \end{bmatrix}$$

Figure: Bad and good bois

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:

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

Interactivity!

Question

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

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

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!

Tools for Deep Learning / Machine Learning

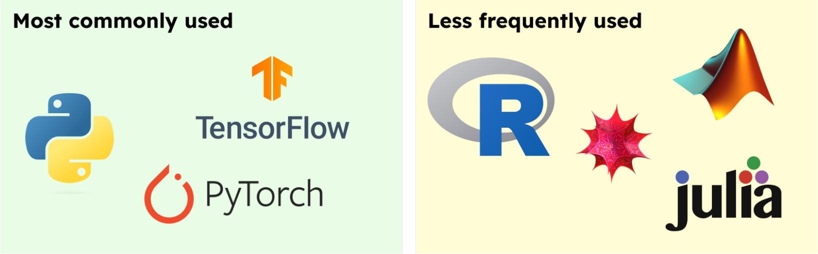


Figure: Tools used for Deep and Machine Learning

A Neural Network Playground

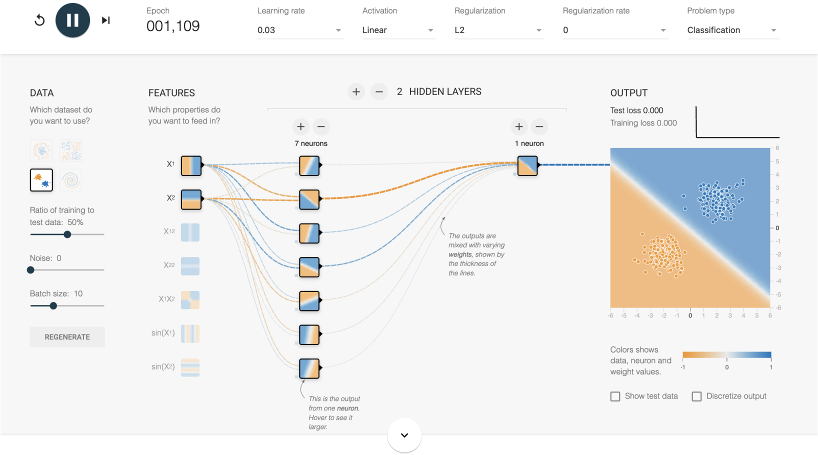


Figure: Tensorflow Playground. See https://playground.tensorflow.org

Importing packages

```
import tensorflow as tf # For defining and training the neural network
import numpy as np # For convenient use of high-dimensional arrays
from keras import datasets # For loading the MNIST dataset

print(f'Using TensorFlow {tf.__version__}')
np.set_printoptions(precision=3)

```

Listing 1: Importing packages we would use

Importing MNIST dataset

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

```

Listing 2: Importing the MNIST dataset

Defining the neural network structure

```
neural_network = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(
        32,
        name='hidden_layer',
        activation='sigmoid'
    ),
    tf.keras.layers.Dense(
        10,
        name='output_layer',
        activation='sigmoid'
    )
])
```

Listing 3: Defining the neural network structure

Fitting into the neural network

```
optimizer = tf.keras.optimizers.legacy.Adam(
    learning_rate=1e-5)
neural_network.compile(loss='mse', optimizer=optimizer
    )
neural_network.fit(
    X,
    y,
    epochs=30,
    verbose=1,
    validation_split=0.2
)

```

Listing 4: Launch the training session

Testing!

```
# Making prediction
index = 1 # Put your index to test
prediction = neural_network.predict(np.expand_dims(X[
    index], axis=0), verbose=0)

# Displaying results
print(f'Prediction in raw format: {prediction}')
show_image(X[index], np.argmax(prediction[0]))

```

Listing 5: Launch the testing

Thank you for your attention!