SA-MIRI Course Schedule (22/10 Tentative)

DL basics Week

day	Wednesday	day	Friday
22/10	7. Neural Networks: Concepts and First Steps Pf part 1 Presentation Pa Presentation Pb	24/10	8. Training Neural Networks: Basics, CNNs, and Deployment 9. Getting Started with PyTorch Pf part 2
29/10	10. Introduction to Parallel Training of Neural Networks Presentation Pc Presentation Pd Presentation Pe	31/10	Midterm period (no class)
05/11	~~(Midterm period) PRESENTATIONS – I~~ Midterm period (no class)	07/11	Pg
12/11	11. Practical Guide to Efficient Training with PyTorch 12. Parallelizing Model Training with Distributed Data Parallel Presentation Pf	14/11	Ph
19/11	Ph Presentation Pg	21/11	Pi
26/11	Honoris Causa Dr. Oriol Vinyals (no class)	28/11	Pi
03/12	13. Introduction to Large Language Models 14. End-to-End Large Language Models Workflow Presentation Ph	05/12	Pj
10/12	15. Exploring Optimization and Scaling of LLMs Presentation Pi	12/12	Pk*
17/12	Pk* 16. Looking Forward: Supercomputing and AI Futures Presentation Pj	19/12	Pl * “attendance not required (for students returning home)”

* No presentation, submission only.

SA-MIRI Course Schedule (Tentative)

Practical Session	acronim	Book Chapter included
Pa	GETTING STARTED	task 2.1 – Log into MareNostrum 5 task 2.2 – Change your password task 2.3 – (Optional) Enable passwordless ssh authentication task 2.4 – Transfer files using scp task 2.5 – (Optional) Mount the MN5 filesystem on your laptop task 3.1 – Compare icx and gcc compiler optimizations task 3.2 – Reflecting on slurm job prioritization task 3.3 – Submit your first slurm job
Pb	CONTAINERS	task 3.4 – Install docker in your platform task 3.5 – Download docker image task 3.6 – Run docker image task 3.7 – Stop a docker container task 3.8 – Run docker with port mapping task 3.9 – Start the jupyter notebook server task 3.10 – Create and run a test notebook
Pc	MPI	task 4.1 – Compile and run your first mpi program task 4.2 – Observe node distribution using hostnames task 4.3 – Point-to-point communication task 4.4 – Write and run the sequential program that estimate π task 4.5 – Write and run the parallel mpi code to estimate the value of π task 4.6 – Analysis using Gustafson's law to estimate the value of π task 4.7 – Experimenting with scatter and gather
Pd	CUDA	task 5.1 – Your first hello world in cuda task 5.2 – Dimensionality of a thread block and grid task 5.3 – Investigating parallel execution with multiple threads task 5.4 – Element-wise vector addition using cuda task 5.5 – Parallel matrix multiplication with cuda task 5.6 – Running cuda jobs with slurm task 5.7 – Profiling matrix multiplication on the gpu task 5.8 – Compute-bound vs memory-bound
Pe	CUDA-aware MPI	task 6.1 – Reflecting on cuda's execution model task 6.2 – Precision trade-offs: true or false? task 6.3 – Submit and validate the first performance run task 6.4 – Understand the metrics collection task 6.5 – Explore the effect of optimized compilation flags task 6.6 – Evaluate the impact of the cub library task 6.7 – Benchmarking the impact of gpu count and problem size
Pf	DL COLABS	task 7.1 – Set up your google colab environment task 7.2 – Execute the provided notebook step-by-step task 7.3 – Improve the accuracy of your model task 8.1 – Improving a basic cnn model task 8.2 – Exporting the python script task 8.3 – Running your first natural network on a login node task 8.4 – Submitting your first gpu dl training with slurm task 9.1 – Comparative implementation in pytorch and tensorflow

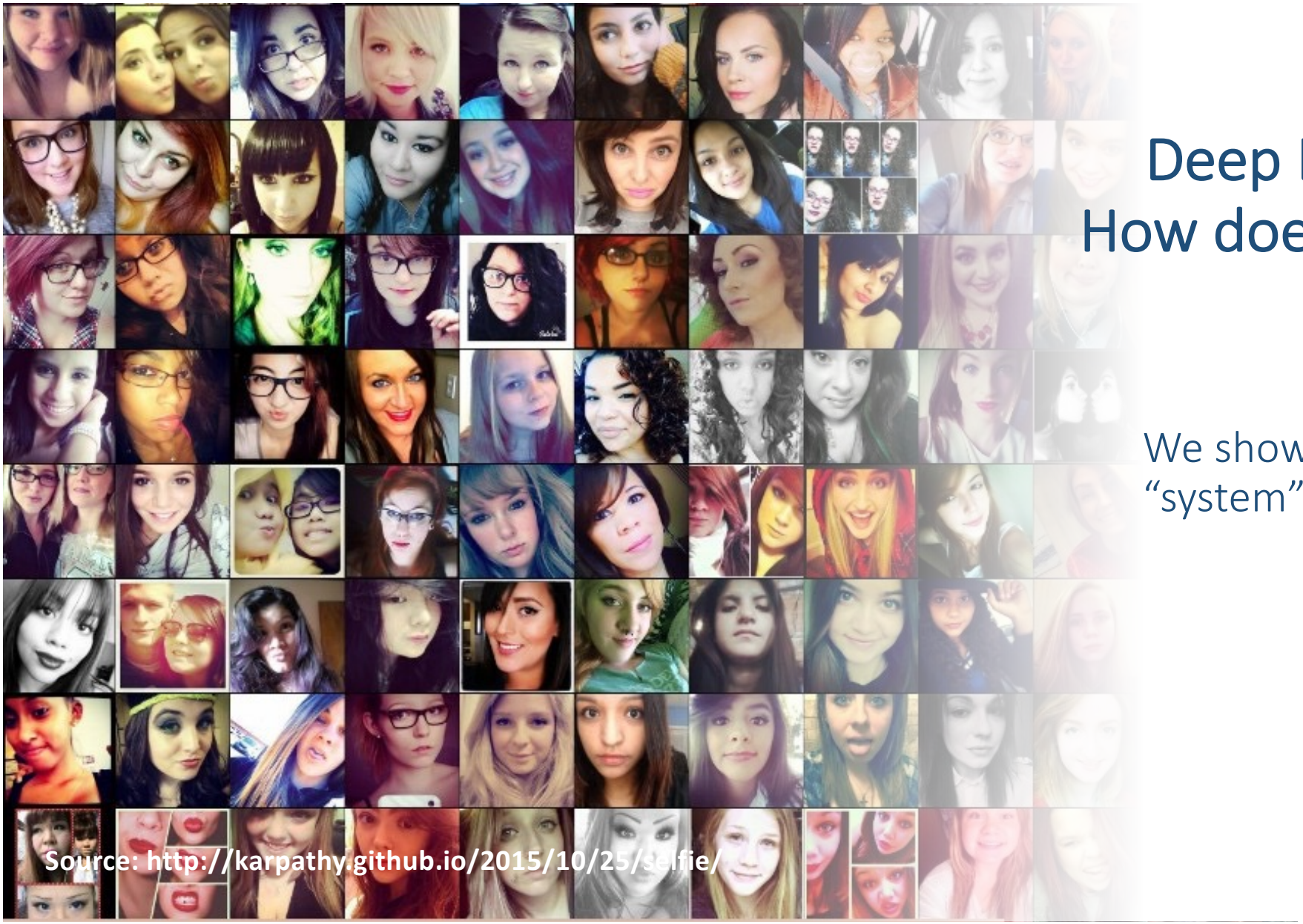
Practical Session	acronim	Book Chapter included
Pg	SCALING TF	task 10.1 – Task setup and file structure task 10.2 – Code review and understanding task 10.3 – Training on cpu (baseline performance) task 10.4 – Gpu execution time and cpu vs gpu comparison task 10.6 – Analyze the impact of gpu parallelism task 10.7 – Parallelization of resnet152 task 10.8 – Comparing parallel training performance across model sizes task 10.9 – Resnet101v2 scalability analysis task 10.10 – Interpreting results with Amdahl's and Gustafson's laws task 10.11 – Applying Amdahl's and Gustafson's laws to Resnet101v2
Ph	OPTIMIZE PT	task 11.1 – Find the maximum viable batch size task 11.2 – Investigating dataloader bottleneck with a lightweight model task 11.3 – Optimizing dataloaders with appropriate num_workers task 11.4 – Confirming dataloader efficiency with vit task 11.5 – Enable mixed precision with vit + micro-224 task 11.6 – Reproducing the effect of torch.compile() task 11.7 – Report your conclusions task 11.8 – Minor code tweaks for a final throughput boost
Pi	SCALING PT	task 12.1 – Reproducing distributed training results on mn5 task 12.2 – Analyze and compare scaling efficiency task 12.3 – Investigate diminishing returns in training time task 12.4 – Find the sweet spot for your use case
Pj	END-TO-END HF	task 14.1 – Obtain your hugging face access token task 14.2 – Download and run the hello world in google colab task 14.3 – Download the model and dataset locally using huggingface-cli task 14.4 – Transfer the model and dataset to MareNostrum 5 task 14.5 – Run the inference and fine-tuning script on mn5 task 14.6 – Compare execution in colab vs mn5
Pk	OPT & SCAL LLM	task 15.1 – Baseline experiment using facebook/opt-1.3b task 15.2 – Finding the out-of-memory limit task 15.3 – Mixed precision training task 15.4 – Model precision task 15.5 – Increasing batch size with model precision task 15.6 – Enabling flash attention task 15.7 – Increasing batch size with flash attention task 15.8 – Using the liger kernel task 15.9 – Augmenting batch size due to liger kernels task 15.10 – Scaling on multiple gpus task 15.11 – Final reflections
Pl	FUTURE	task 16.1 – Mapping the pendulum shifts of the triad task 16.2 – Exploring the new giants of compute task 16.3 – Powering the future of ai task 16.4 – Charting the role of quantum in future supercomputing task 16.5 – From data farms to ai preserves task 16.6 – Embodiment as a source of learning

GETTING STARTED WITH DEEP LEARNING (a recap combining Chapters 7, 8, and 9)

SUPERCOMPUTERS ARCHITECTURE
Master in Innovation and Research in Informatics
Jordi TORRES.AI

How does Deep Learning Works?

Jordi TORRES.AI

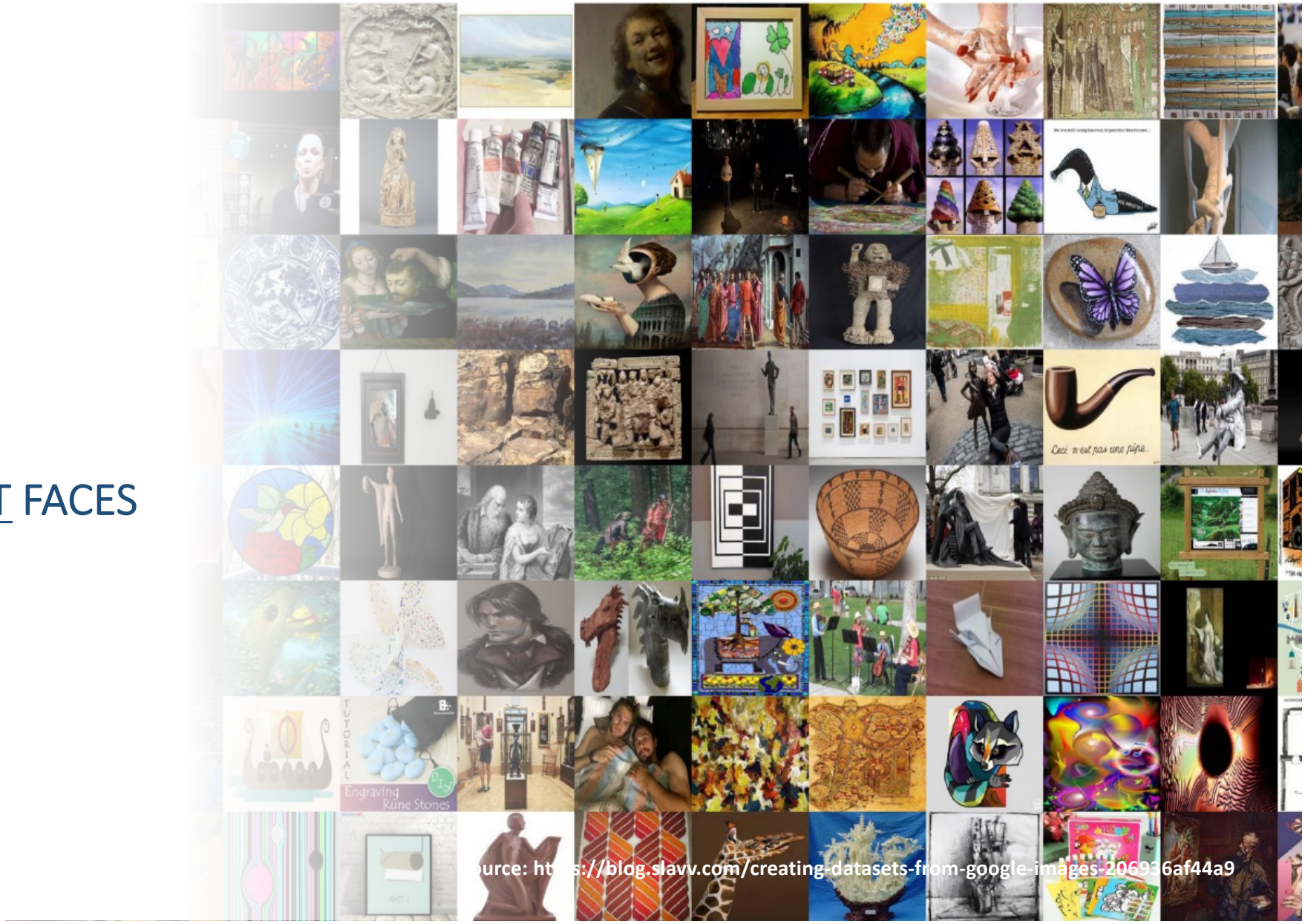


Deep Learning
How does it works?

We show to the
“system” many FACES

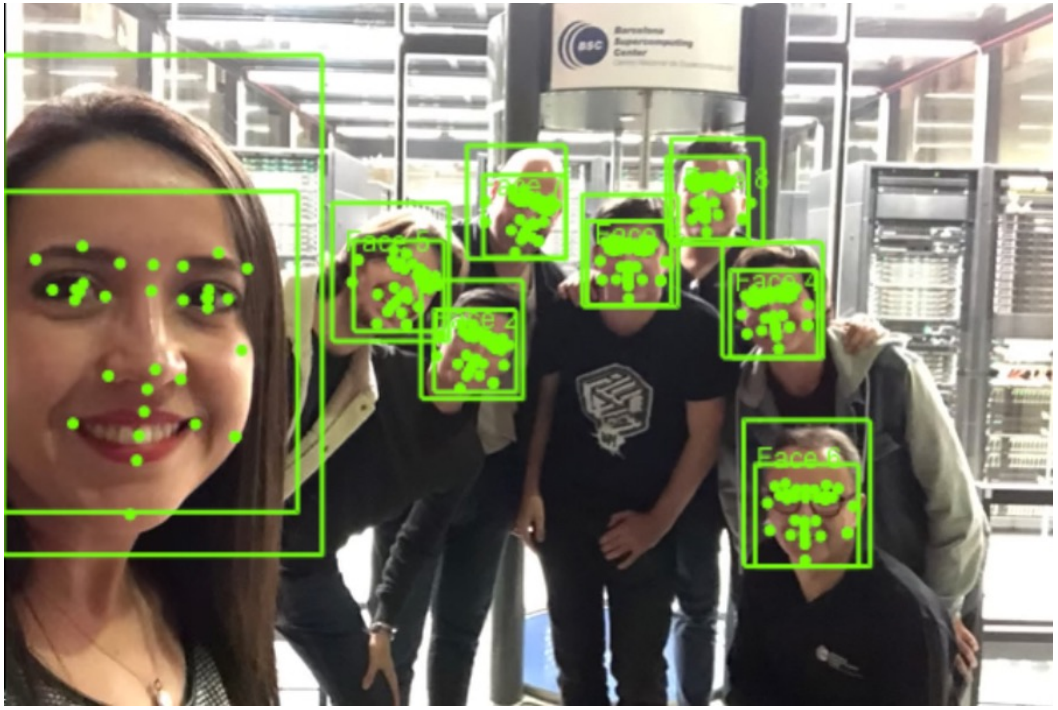
Source: http://karpathy.github.io/2015/10/25/selfie/

And also, NOT FACES



Source: https://blog.slavv.com/creating-datasets-from-google-images-206936af44a9

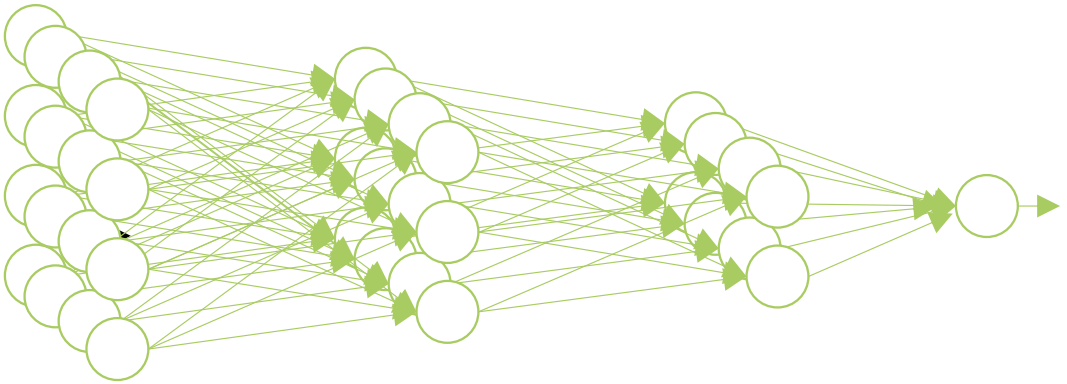
Et voilà!

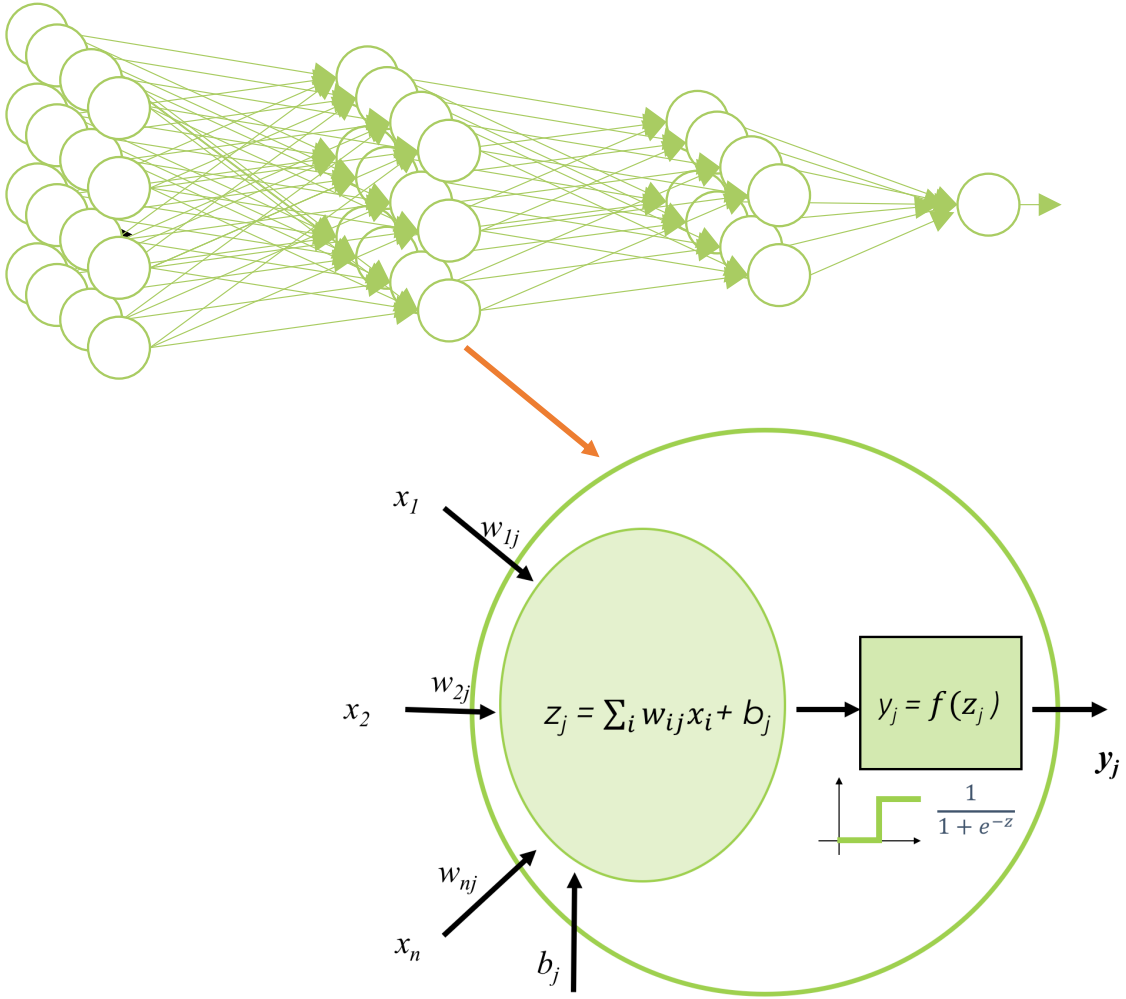


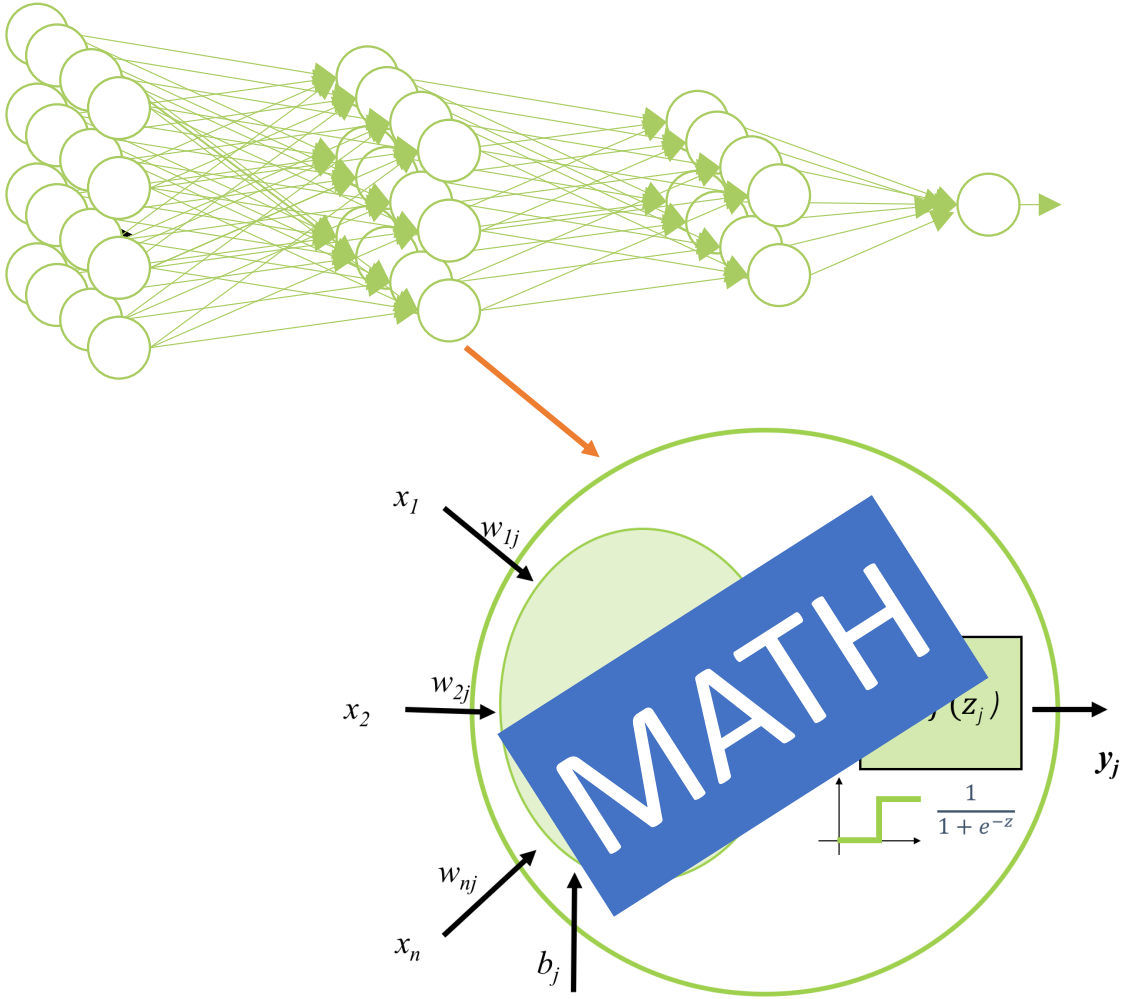
So what's all about?

Deep Learning:

Matematic models




x_1
w_{1j}
x_2
w_{2j}
$z_j = \sum_i w_{ij} x_i + b_j$
$y_j = f(z_j)$
y_j
$\frac{1}{1+e^{-z}}$
w_{nj}
x_n
b_j

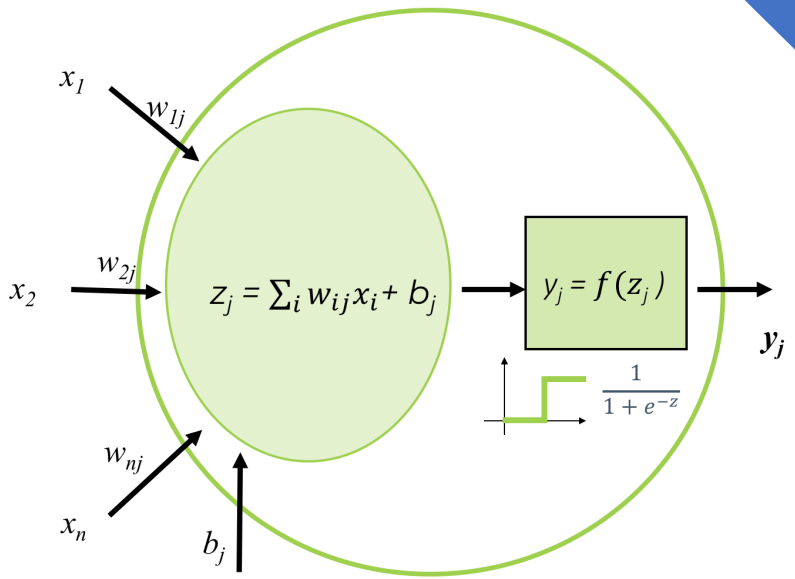

x_1
w_{1j}
x_2
w_{2j}
w_{nj}
x_n
b_j
MATH
(z_j)
y_j
$\frac{1}{1+e^{-z}}$

Basic machine learning terminology

- Model: defines the relation between features and labels

$$y = wx + b$$

  - $\boldsymbol{y}$: Labels
  - $\boldsymbol{x}$: Features
  - $\boldsymbol{w}$:weights
  - $\boldsymbol{b}$: bias

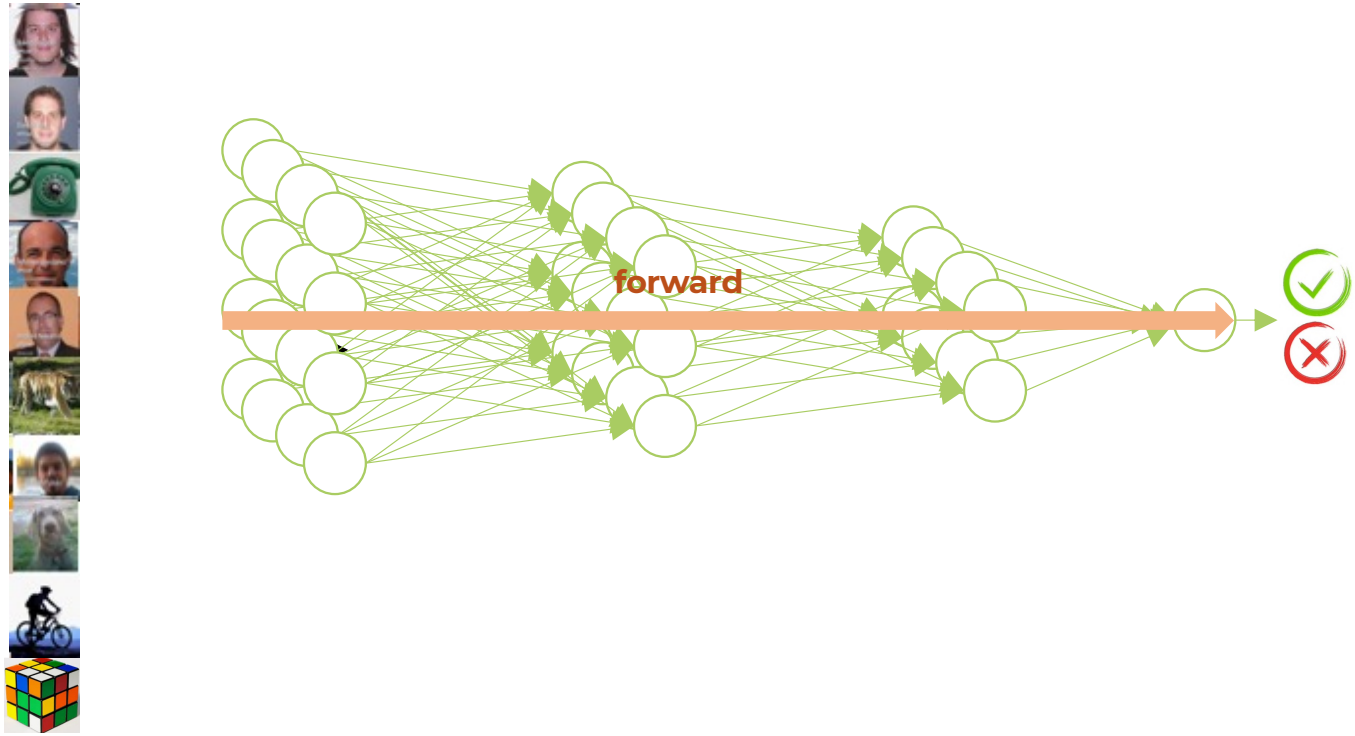


Basic machine learning terminology

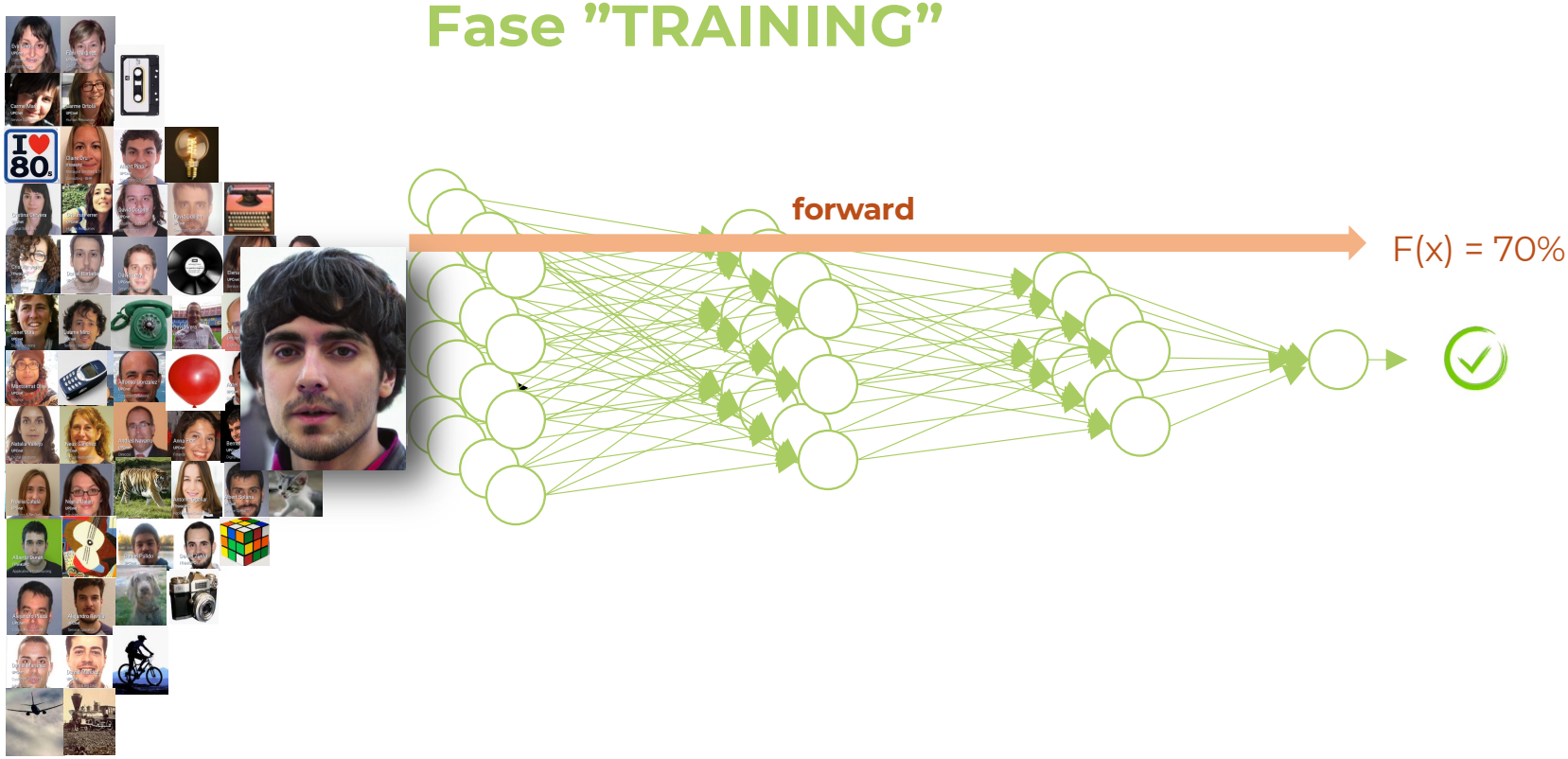
$$y = \sum_i w_i x_i + b$$

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b = \mathbf{w}^T \cdot \mathbf{x} + b$$

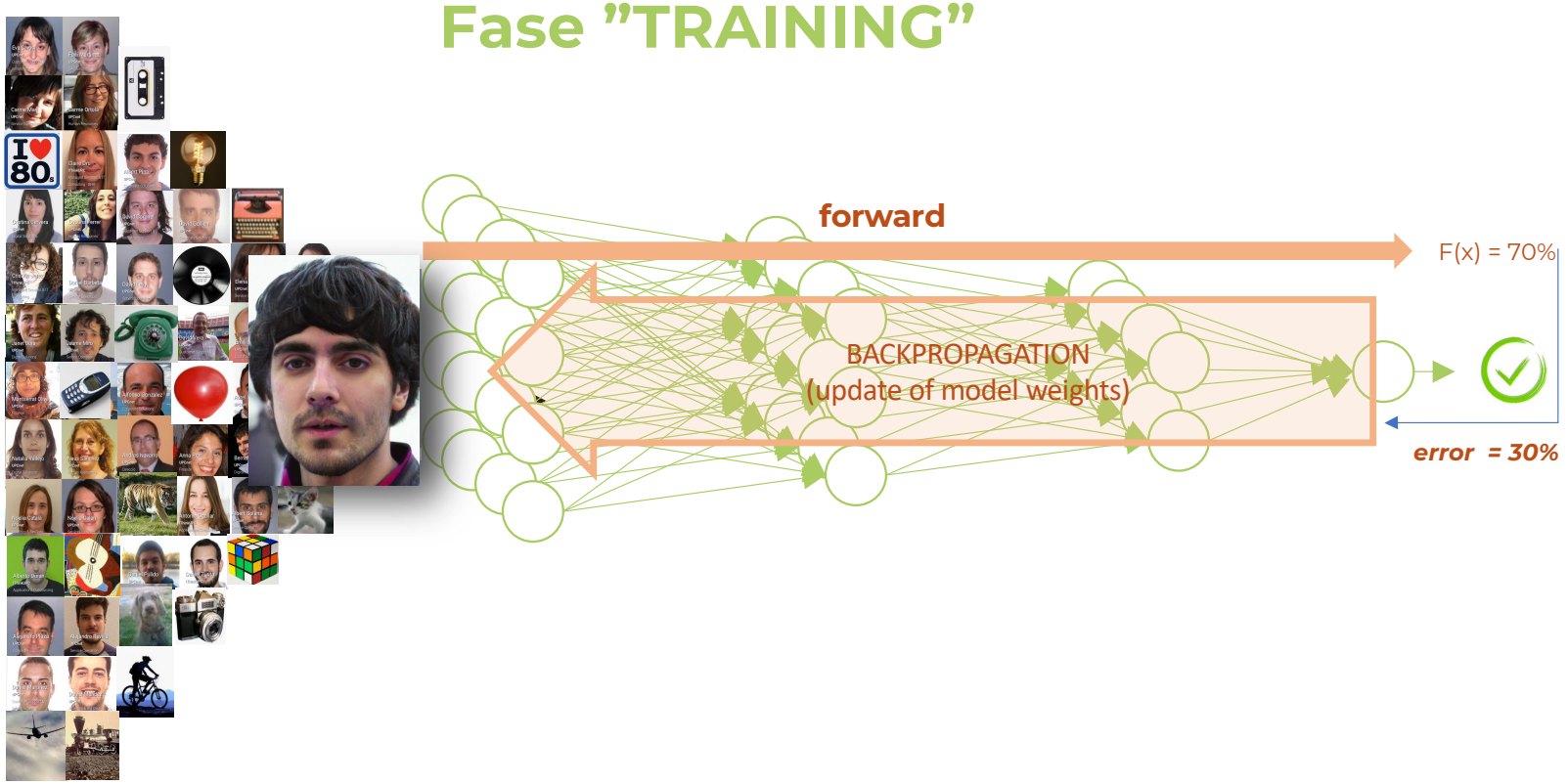
$$y = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w}^T \cdot \mathbf{x}$$


forward

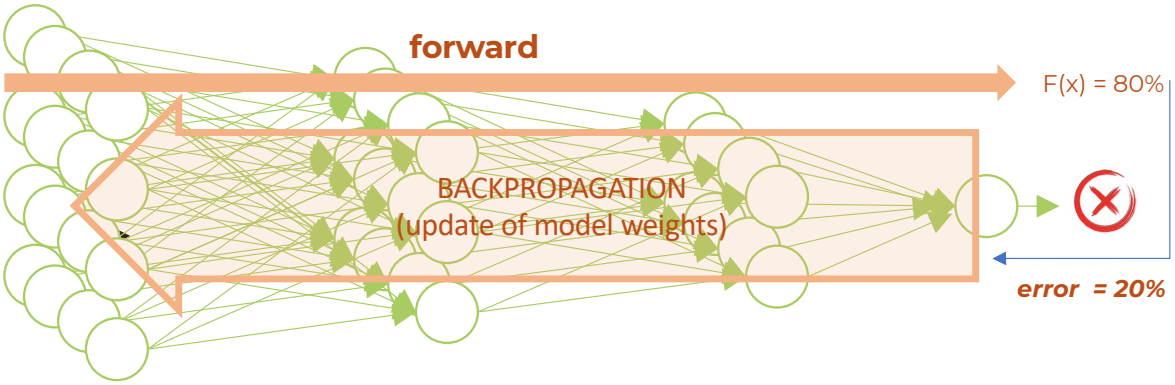
Fase "TRAINING"



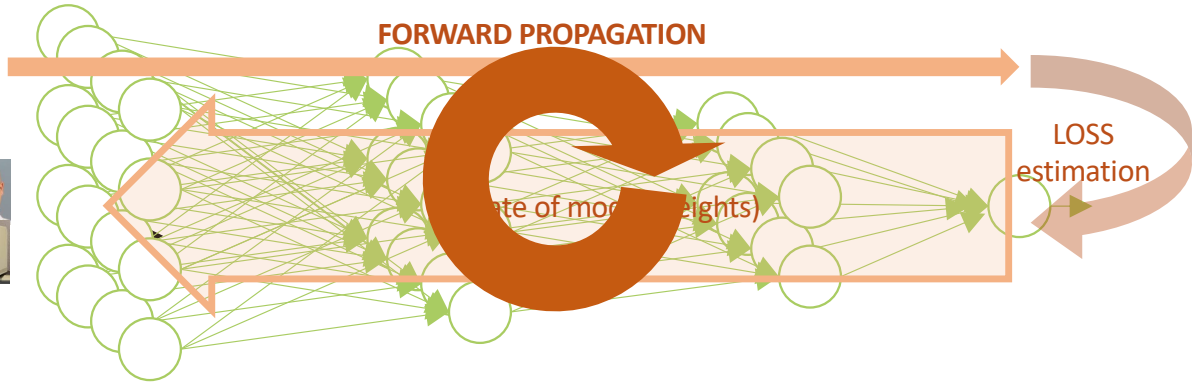
Fase "TRAINING"



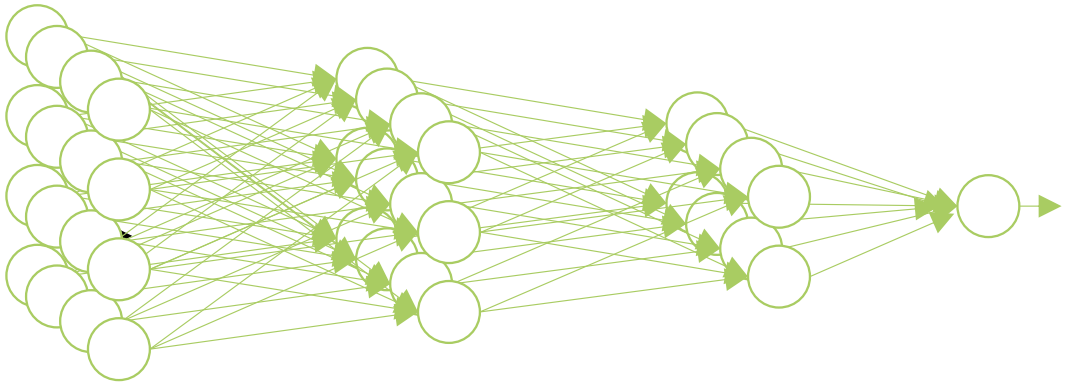
Fase ”TRAINING”



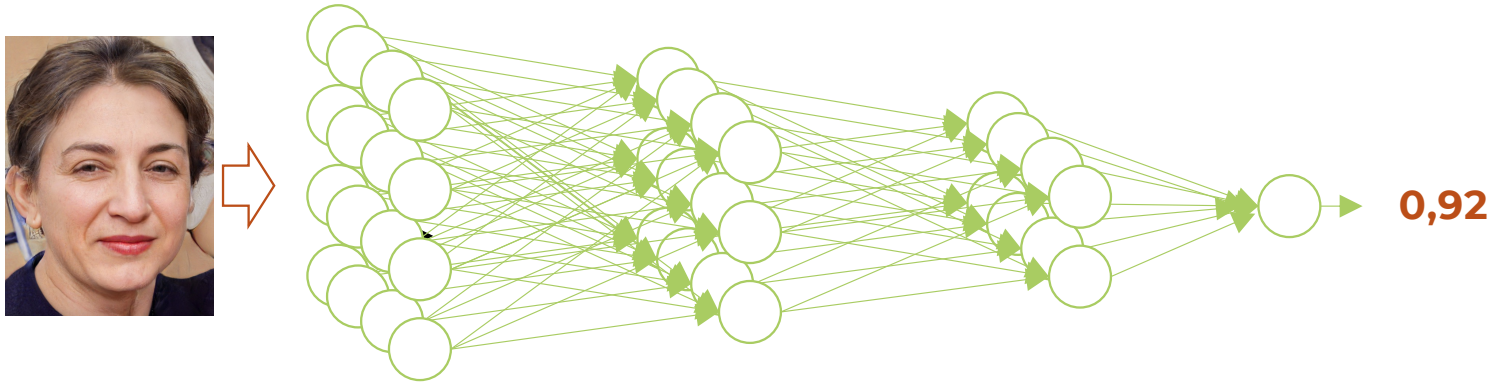
Fase "TRAINING"



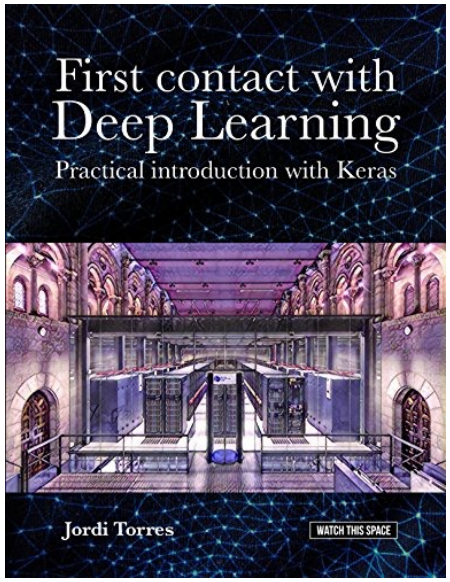
After ”TRAINING” the model is ready for “INFERENCE”

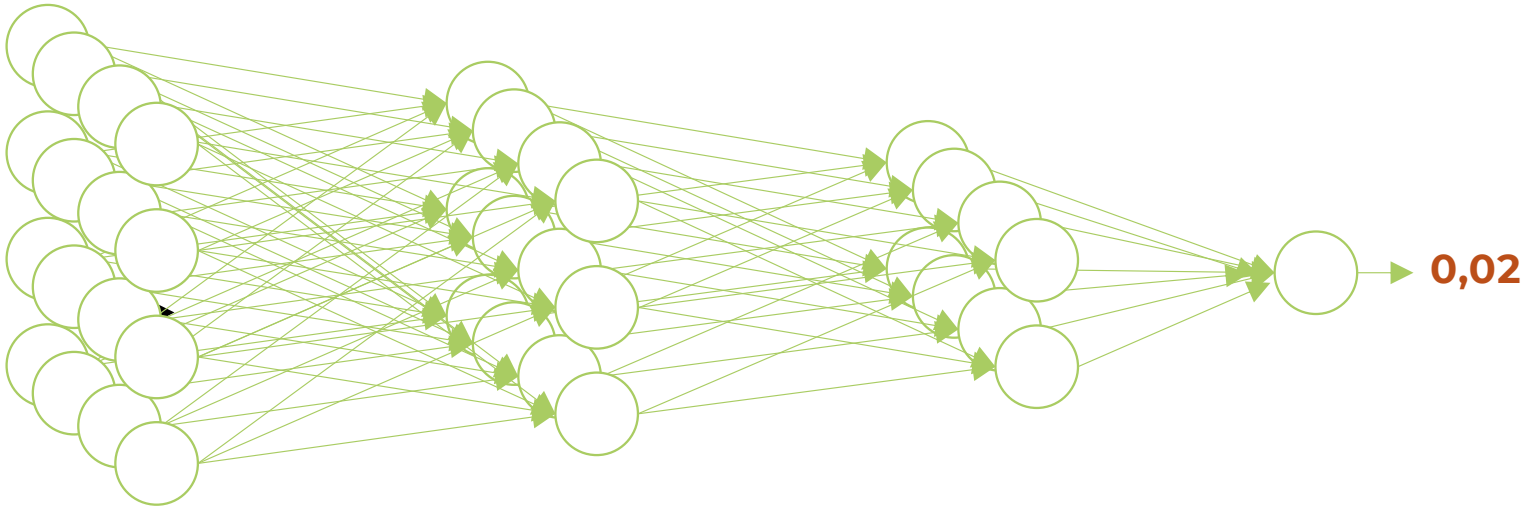


Fase "INFERENCE"

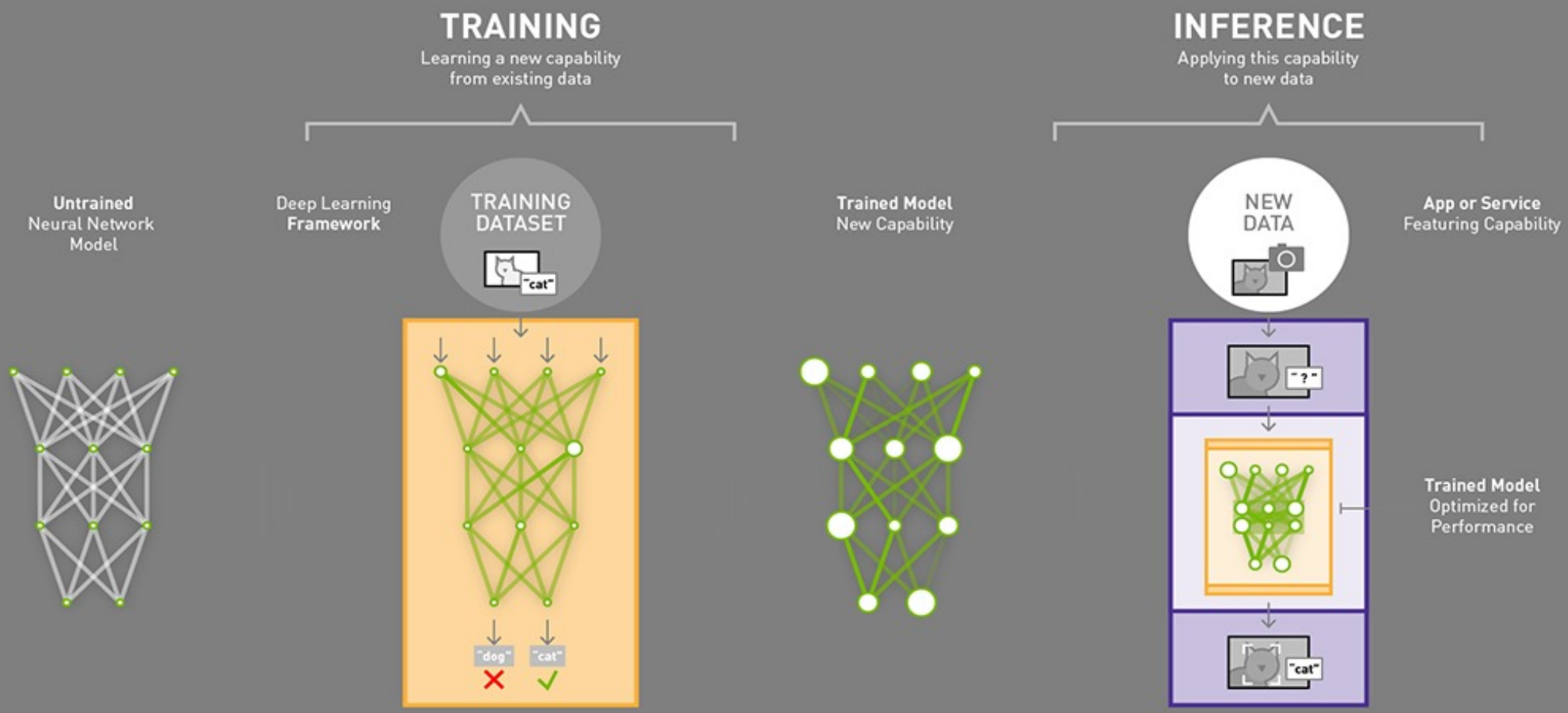


Fase "INFERENCE"





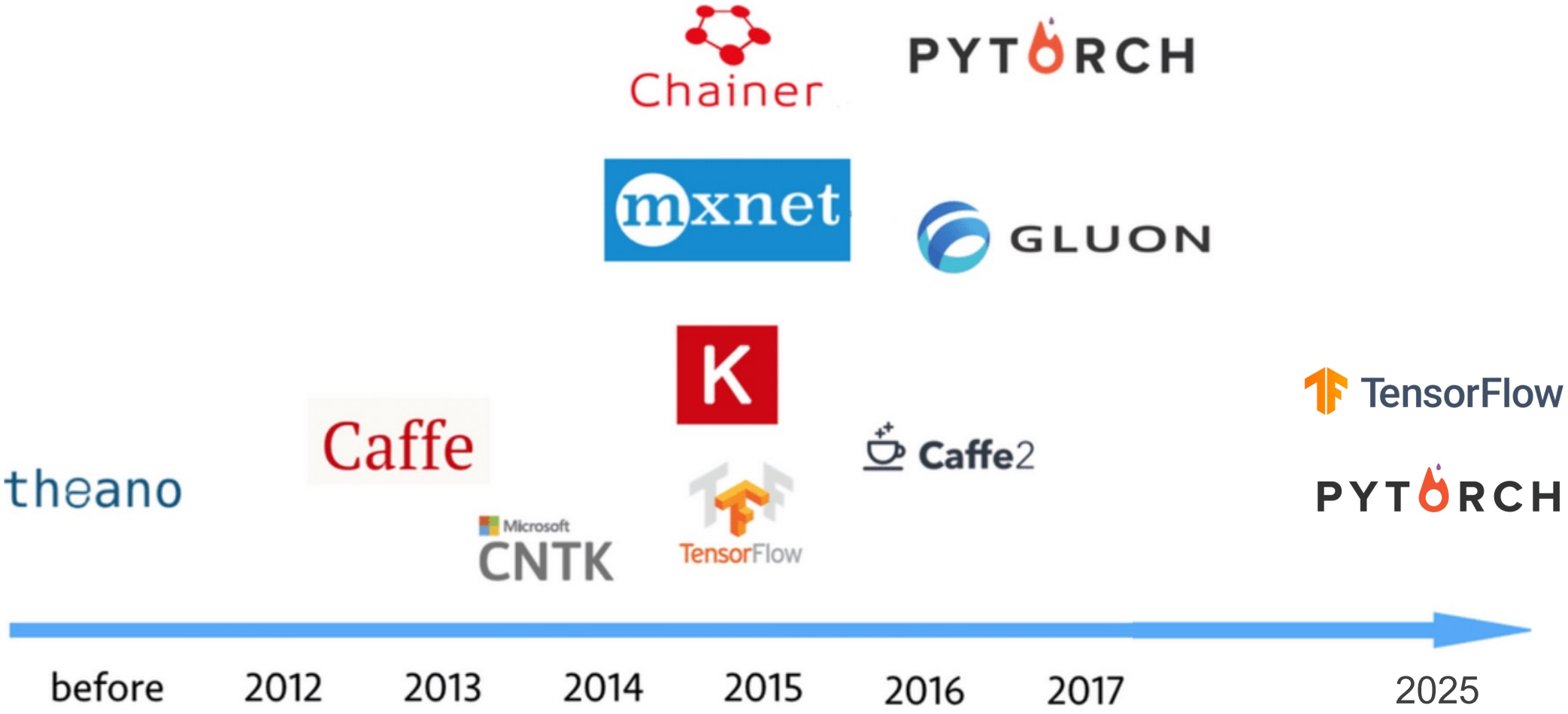
Deep Learning: Training vs Inference



Source: https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai/

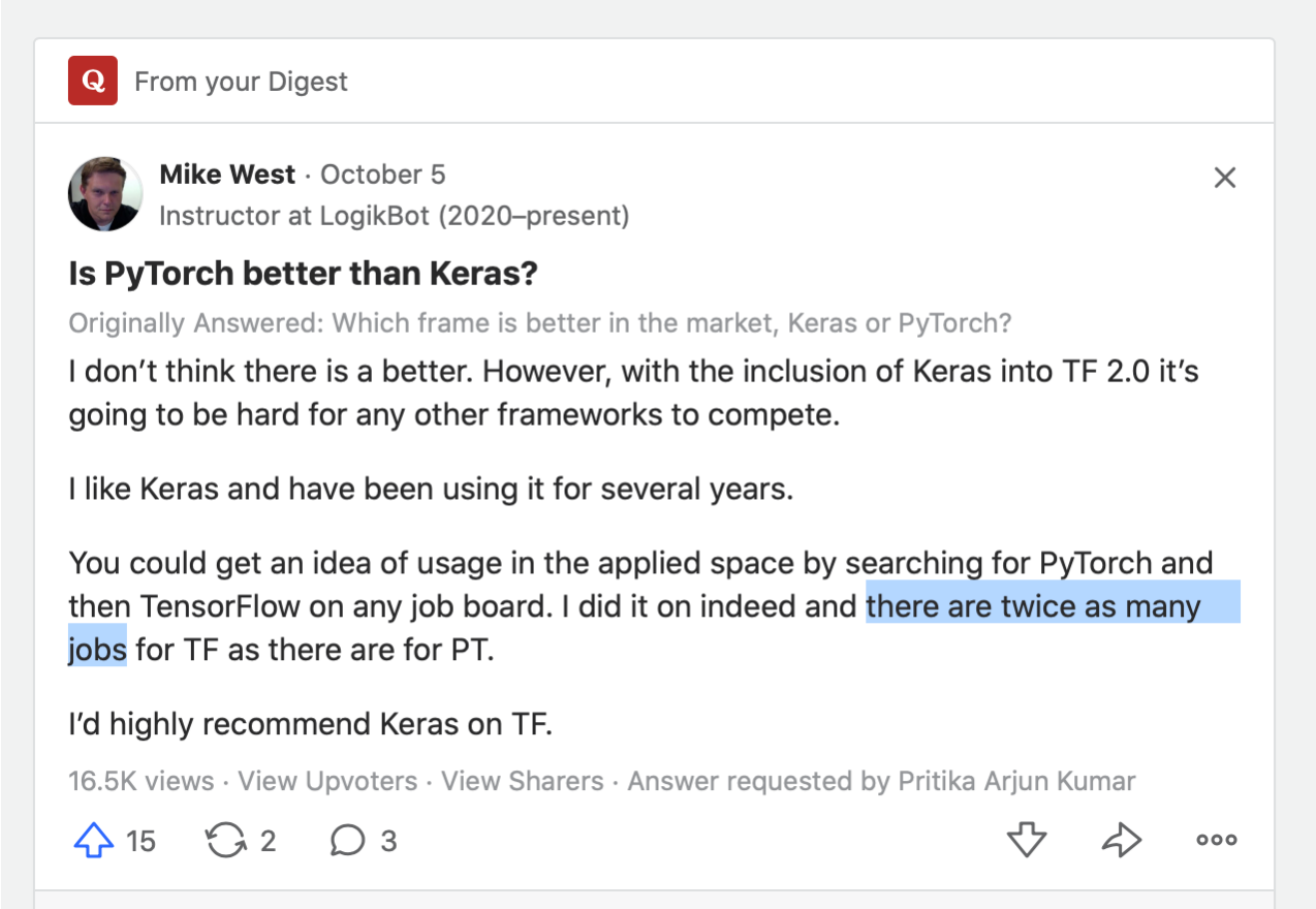
Working environment?

Jordi TORRES.AI



source: https://www.researchgate.net/publication/349108714_DLBench_a_comprehensive_experimental_evaluation_of_deep_learning_frameworks

TensorFlow vs PyTorch?



From your Digest

Mike West · October 5
Instructor at LogikBot (2020–present)

Is PyTorch better than Keras?

Originally Answered: Which frame is better in the market, Keras or PyTorch?

I don't think there is a better. However, with the inclusion of Keras into TF 2.0 it's going to be hard for any other frameworks to compete.

I like Keras and have been using it for several years.

You could get an idea of usage in the applied space by searching for PyTorch and then TensorFlow on any job board. I did it on indeed and there are twice as many jobs for TF as there are for PT.

I'd highly recommend Keras on TF.

16.5K views · View Upvoters · View Sharers · Answer requested by Pritika Arjun Kumar

15 2 3

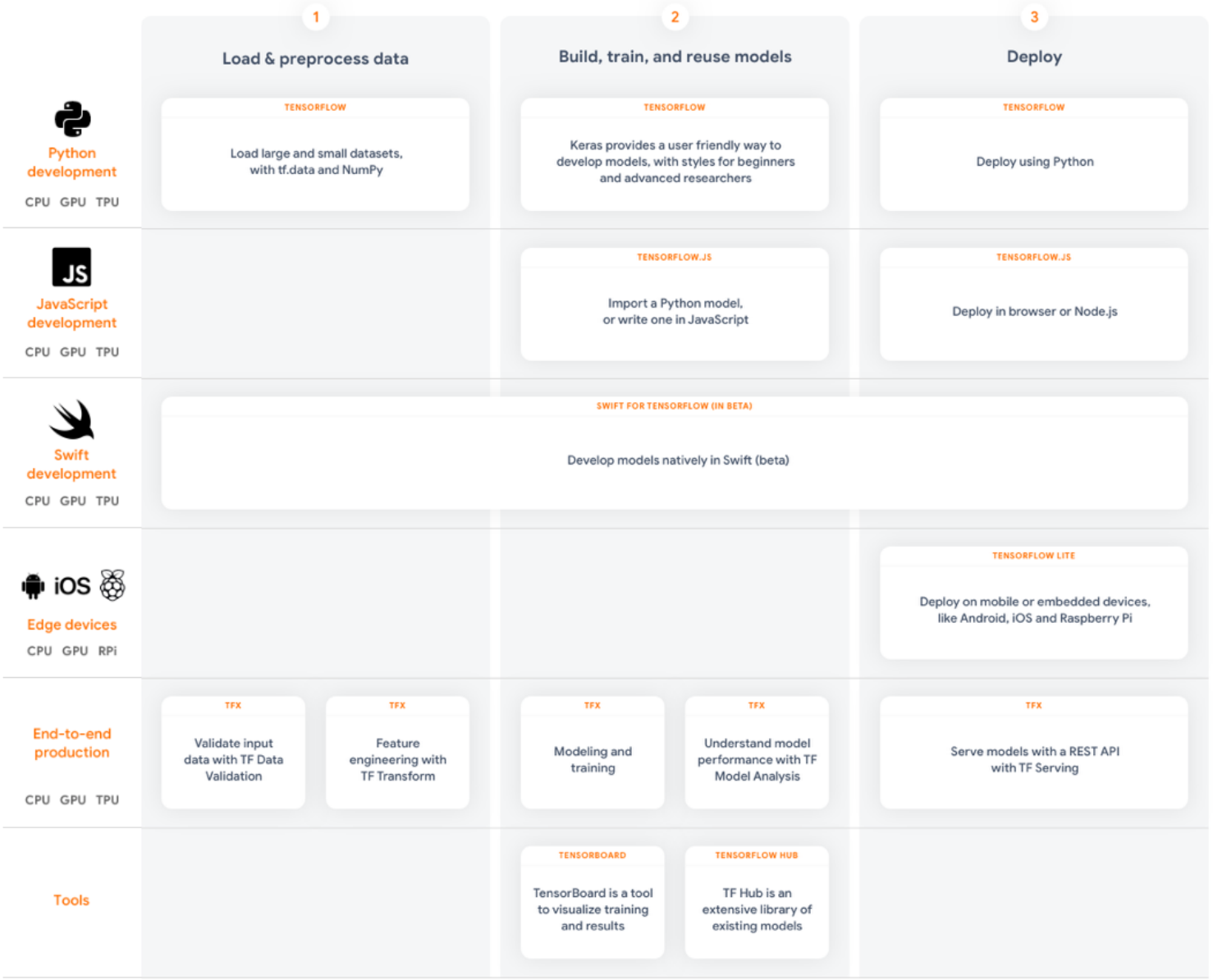
TensorFlow 2.*

- During **2019 TensorFlow Dev Summit**, Google announced a major upgrade on the framework → the TensorFlow 2.0 Alpha version.

  Latest stable release: 2.20 (13 August 2025)

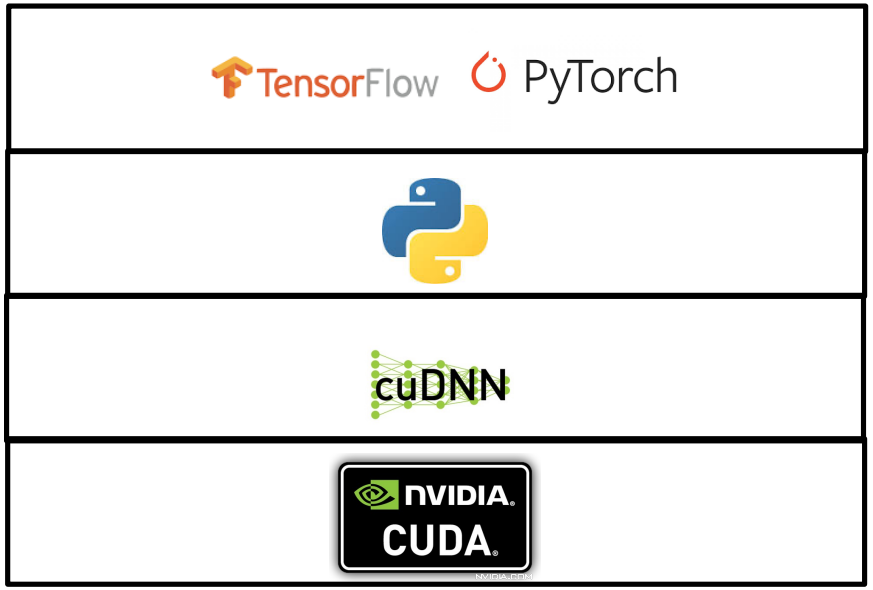
- **TensorFlow 2.0 focuses on simplicity and ease of use:**
  - with updates like eager execution
  - flexible model building on any platform
  - **intuitive higher-level APIs using**  (developed by François Chollet)

TensorFlow Ecosystem



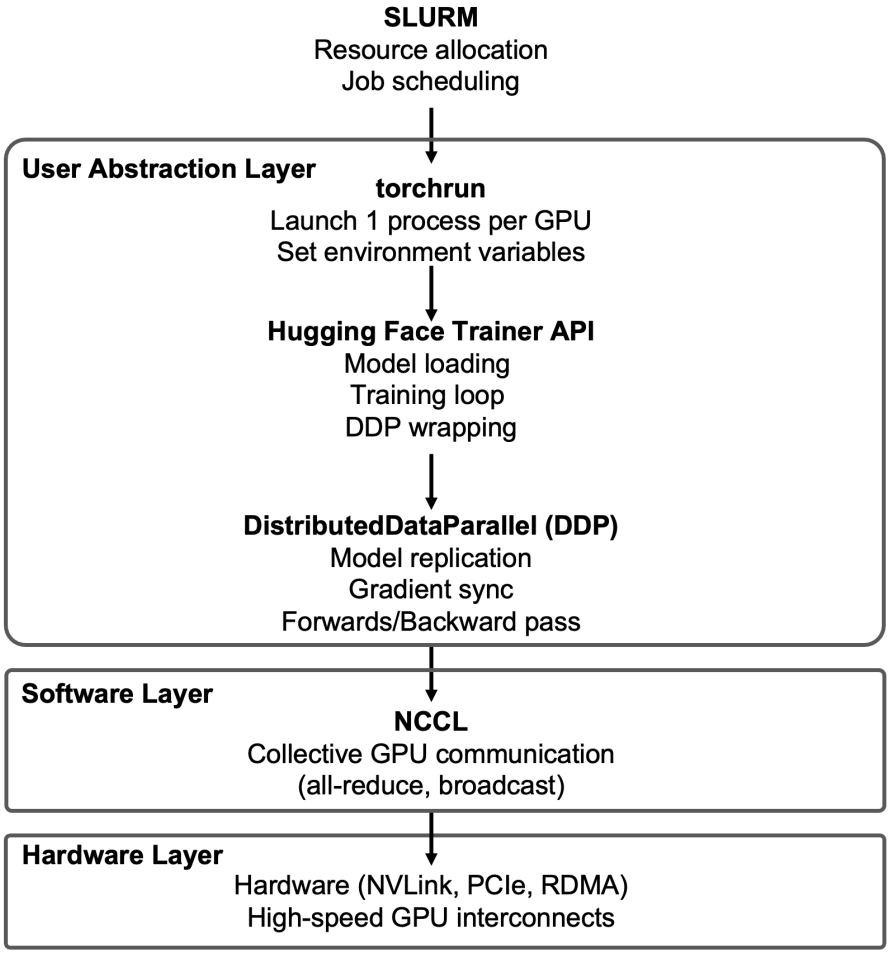
Deep Learning Software Stack

- Our case study: basic



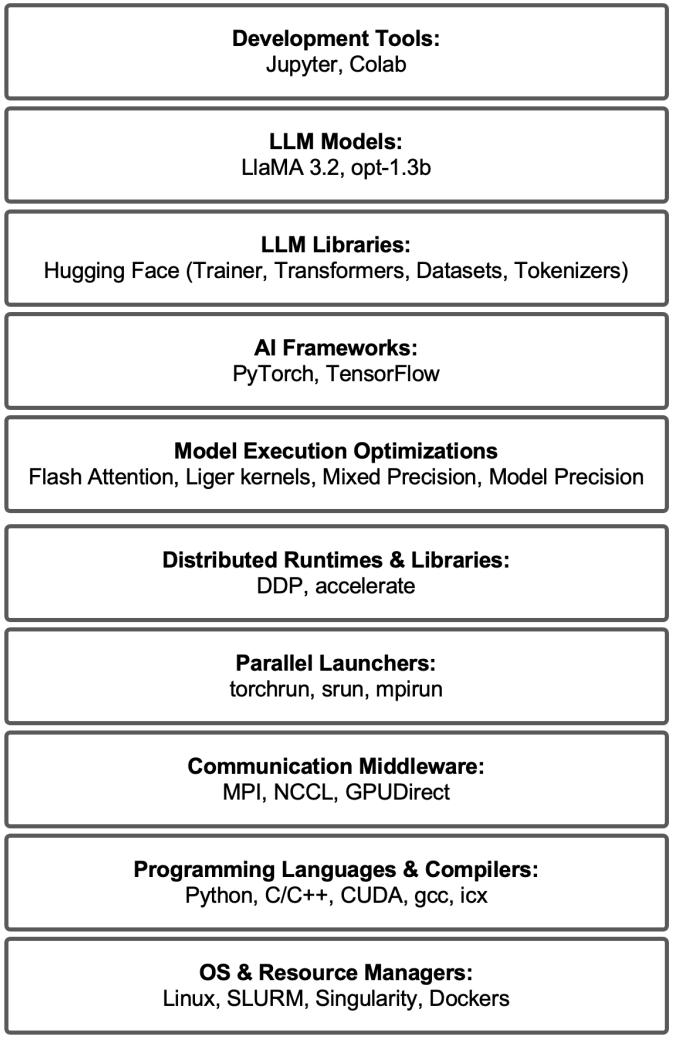
Deep Learning Software Stack

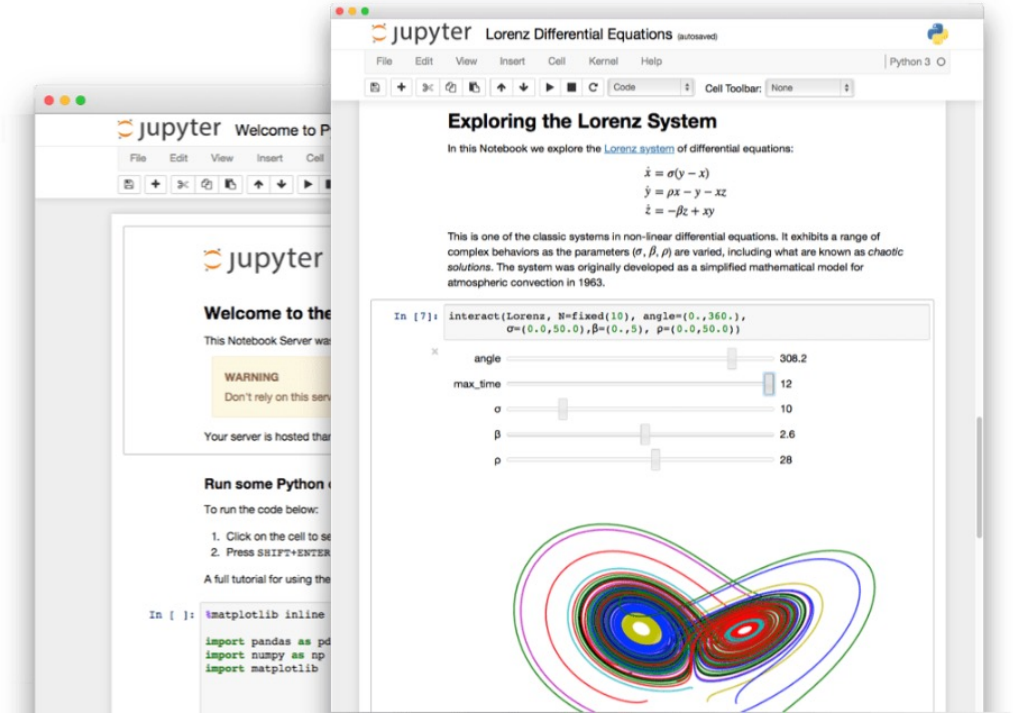
- Our case study: workflow



Deep Learning Software Stack

- Our case study



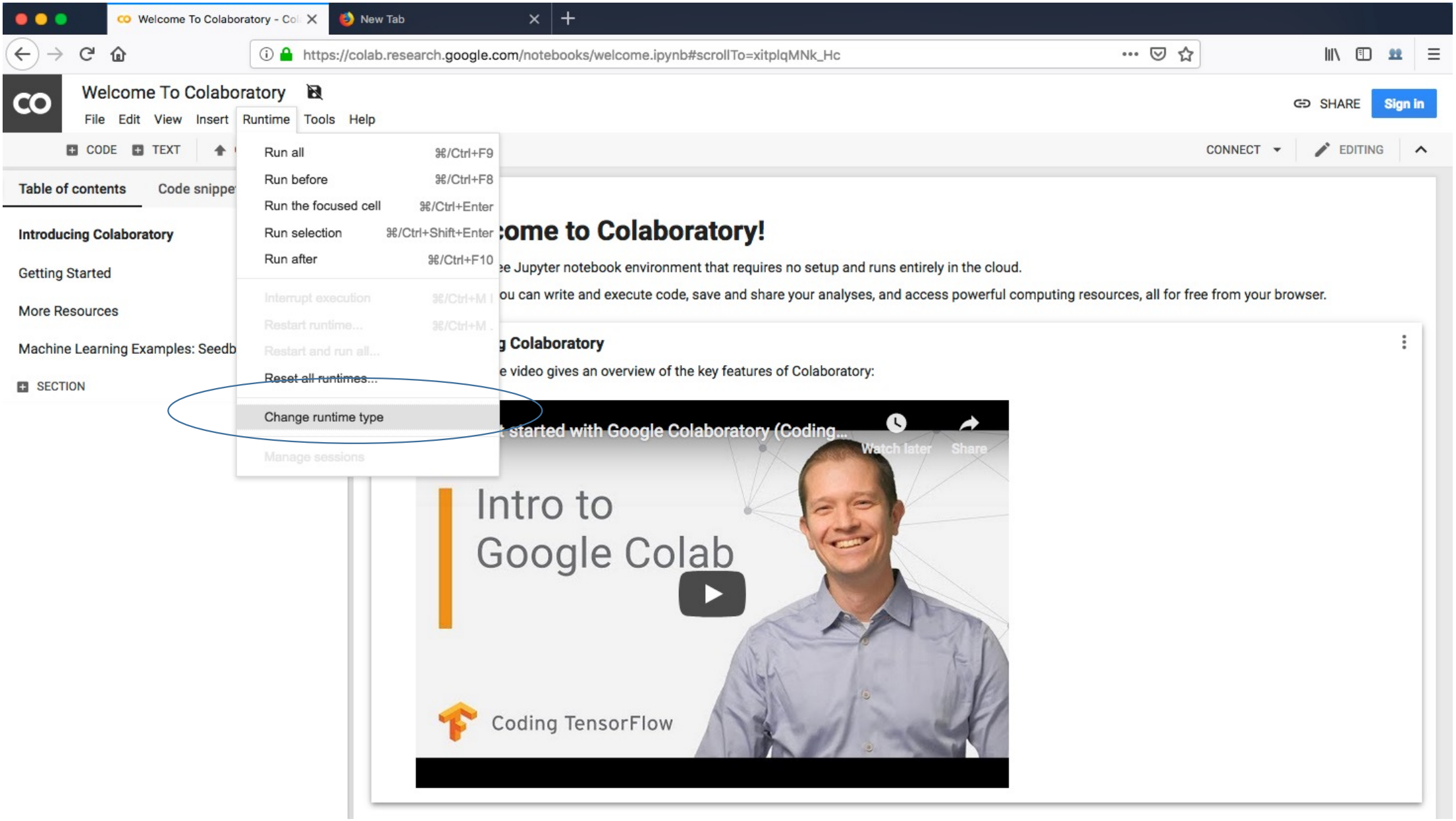


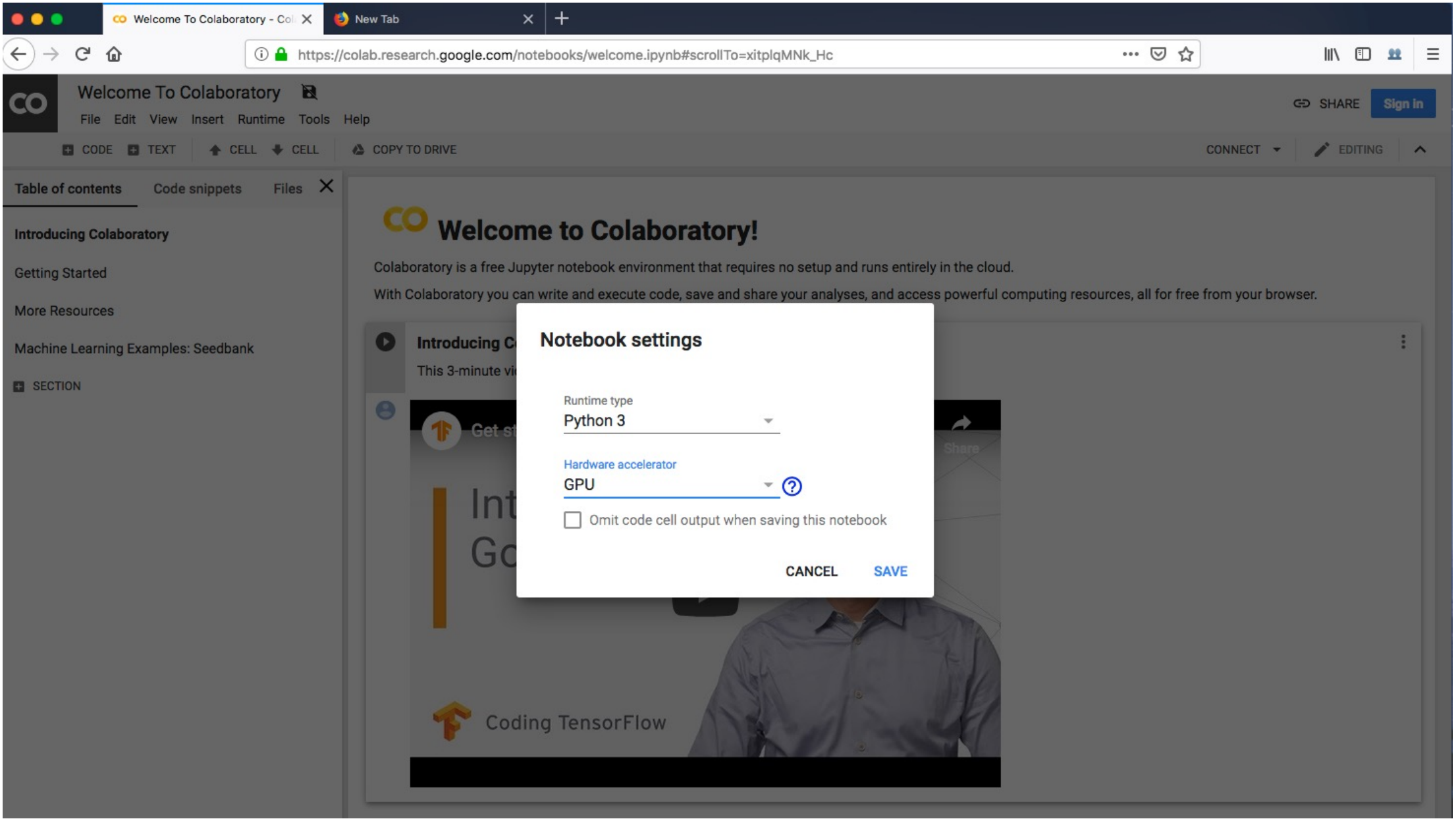
The Jupyter Notebook

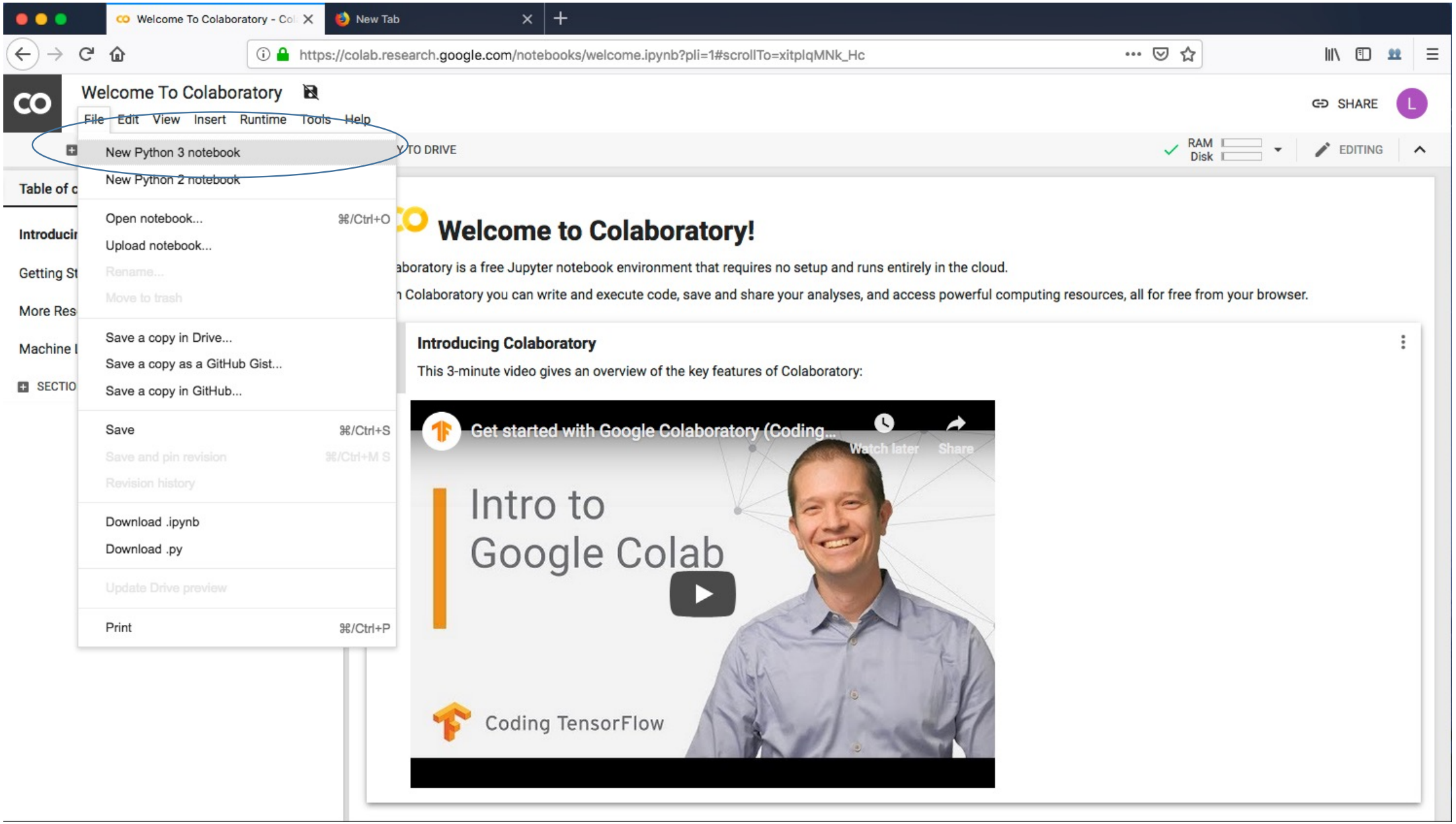
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Working Environment Colab

Google Colab
https://colab.research.google.com/ ▾ Traducir esta página
Colaboratory notebooks are stored in **Google Drive** and can be shared just as you ... **Colab** also supports connecting to a Jupyter runtime on your local machine.


Welcome To Colaboratory - Col
New Tab
https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=xitplqMNk_Hc
Welcome To Colaboratory
File Edit View Insert Runtime Tools Help
SHARE
Sign in
CODE TEXT
CONNECT
EDITING
Table of contents
Code snippe
Introducing Colaboratory
Getting Started
More Resources
Machine Learning Examples: Seedb
SECTION
Run all ⌘/Ctrl+F9
Run before ⌘/Ctrl+F8
Run the focused cell ⌘/Ctrl+Enter
Run selection ⌘/Ctrl+Shift+Enter
Run after ⌘/Ctrl+F10
Interrupt execution ⌘/Ctrl+M I
Restart runtime... ⌘/Ctrl+M .
Restart and run all...
Reset all runtimes...
Change runtime type
Manage sessions
come to Colaboratory!
e Jupyter notebook environment that requires no setup and runs entirely in the cloud.
ou can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.
g Colaboratory
e video gives an overview of the key features of Colaboratory:
started with Google Colaboratory (Coding...
Watch later
Share
Intro to
Google Colab
Coding TensorFlow


Welcome To Colaboratory - Col
New Tab
https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=xitplqMNk_Hc
Welcome To Colaboratory
File Edit View Insert Runtime Tools Help
SHARE
Sign in
CODE TEXT CELL CELL COPY TO DRIVE
CONNECT
EDITING
Table of contents
Code snippets
Files
Introducing Colaboratory
Getting Started
More Resources
Machine Learning Examples: Seedbank
SECTION
Welcome to Colaboratory!
Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud.
With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.
Notebook settings
Runtime type
Python 3
Hardware accelerator
GPU
Omit code cell output when saving this notebook
CANCEL
SAVE
Coding TensorFlow


Welcome To Colaboratory - Col
New Tab
https://colab.research.google.com/notebooks/welcome.ipynb?pli=1#scrollTo=xitplqMNk_Hc
Welcome To Colaboratory
File Edit View Insert Runtime Tools Help
SHARE
New Python 3 notebook
New Python 2 notebook
Open notebook... ⌘/Ctrl+O
Upload notebook...
Rename...
Move to trash
Save a copy in Drive...
Save a copy as a GitHub Gist...
Save a copy in GitHub...
Save ⌘/Ctrl+S
Save and pin revision ⌘/Ctrl+M S
Revision history
Download .ipynb
Download .py
Update Drive preview
Print ⌘/Ctrl+P
TO DRIVE
RAM
Disk
EDITING
Table of c
Introducir
Getting St
More Res
Machine l
SECTIO
Welcome to Colaboratory!
aboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud.
n Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.
Introducing Colaboratory
This 3-minute video gives an overview of the key features of Colaboratory:
Get started with Google Colaboratory (Coding...
Watch later
Share
Intro to
Google Colab
Coding TensorFlow

Steps to create your first neural network

Jordi TORRES.AI

Steps to create a model with TensorFlow

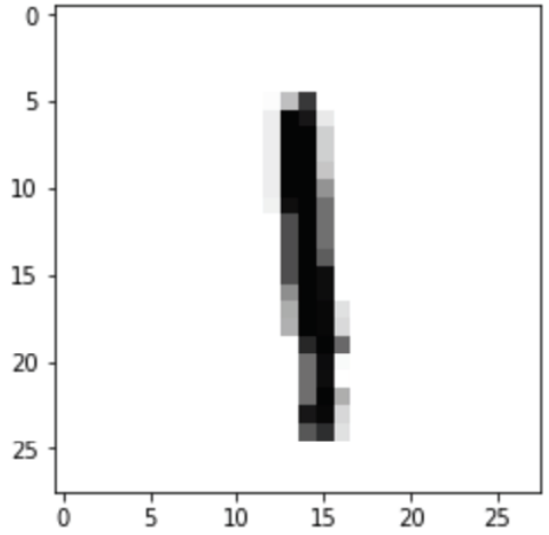
1	2	3	4	5	6
Load Data & Prepare Data	Define Model	Compile Model	Fit Model	Evaluate Model	Use the Model

1. Load and prepare data

- Case study: “The MNIST database”
  - Dataset of handwritten digits classification
  - 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images.

1. Load and prepare data

- Features (Images): matrix of 28x28 pixels with values [0, 255]
- Labels: values [0, 9]



1. Load and prepare data: Example

- Feature (Image): matrix of 28x28 pixels with values [0, 255]
- Label: 1

```
[[   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   5  63 197   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  20 254 230  24   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  20 254 254  48   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  20 254 255  48   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  20 254 254  57   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  20 254 254 108   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0  16 239 254 143   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0 178 254 143   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0 178 254 143   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0 178 254 162   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0 178 254 240   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0 113 254 240   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0  83 254 245  31   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0  79 254 246  38   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0 214 254 150   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0 144 241   8   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0 144 240   2   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0 144 254  82   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0 230 247  40   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0 168 209  31   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

1. Load and prepare data

```
In [2]: mnist = tf.keras.datasets.mnist

        (x_train, y_train), (x_test, y_test) = mnist.load_data(path='/gpfs/projects/nct00/nct00(
```

2. Define model

In [24]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

3. Compile model

In [26]:
```
model.compile(loss="categorical_crossentropy",
              optimizer="sgd",
              metrics = ['accuracy'])
```

4. Fit model

```
[24] model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 6s 98us/sample - loss: 0.4039 - accuracy: 0.8930
Epoch 2/5
60000/60000 [==============================] - 6s 94us/sample - loss: 0.1958 - accuracy: 0.9441
Epoch 3/5
60000/60000 [==============================] - 5s 88us/sample - loss: 0.1421 - accuracy: 0.9590
Epoch 4/5
60000/60000 [==============================] - 5s 86us/sample - loss: 0.1101 - accuracy: 0.9688
Epoch 5/5
60000/60000 [==============================] - 5s 86us/sample - loss: 0.0878 - accuracy: 0.9750
<tensorflow.python.keras.callbacks.History at 0x7fbd8af52ba8>
```

5. Evaluate model

```
[25] test_loss, test_acc = model.evaluate(x_test, y_test)
     print ('Test accuracy:', test_acc)
```

```
10000/10000 [==============================] - 0s 48us/sample - loss: 0.0955 - accuracy: 0.9716
Test accuracy: 0.9716
```

6. Use the model

```
[31] predictions = model.predict(x_test)
     print (predictions)
```

```
[[1.26404475e-05 2.49308573e-06 4.45205660e-04 ... 9.93939281e-01
  2.26089851e-05 3.87840439e-04]
 [1.40489414e-04 5.95517922e-03 9.92044926e-01 ... 7.36168673e-08
  8.24517483e-05 5.90428044e-08]
 [4.98437309e-08 9.97506320e-01 1.01395021e-03 ... 9.72060545e-04
  2.70858494e-04 1.64484100e-05]
 ...
 [1.58638017e-07 4.75961940e-07 7.93745471e-07 ... 1.92583975e-04
  1.82772157e-04 4.21503466e-03]
 [2.74540548e-06 5.57959866e-05 4.68355069e-07 ... 3.29460909e-06
  5.54329483e-03 1.57847660e-06]
 [2.11494735e-05 1.39356416e-06 7.07783474e-05 ... 2.89122028e-07
  5.49738729e-07 9.28026580e-08]]
```

Live Demo:
Get started with Deep Learning basics

1. Open Colab
2. Upload **https://github.com/jorditorresBCN/HPC4AIbook/blob/main/Chapter.07/PracticalIntroductionToDeepLearningBasics.ipynb**

To get started, import the TensorFlow library into your program:

```
[1]
    import tensorflow as tf
    from tensorflow import keras

    import numpy as np
    import matplotlib.pyplot as plt

    print(tf.__version__)
```

```
2.19.0
```

Steps to create a model with TensorFlow

1	2	3	4	5	6
Load Data & Prepare Data	Define Model	Compile Model	Fit Model	Evaluate Model	Use the Model

1. Load and prepare data

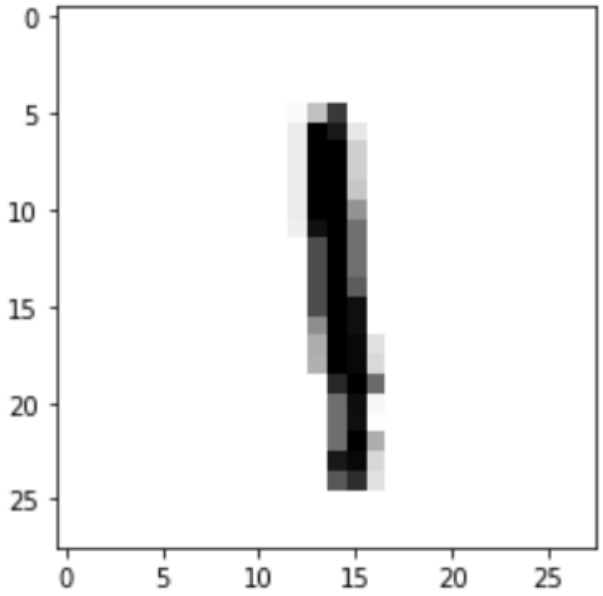
- Case study: “The MNIST database”
  - Dataset of handwritten digits classification
  - 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images.

```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data(path='/gpfs/projects/nct00/nct00(
```

MN5: path='/gpfs/.../mnist.npz'

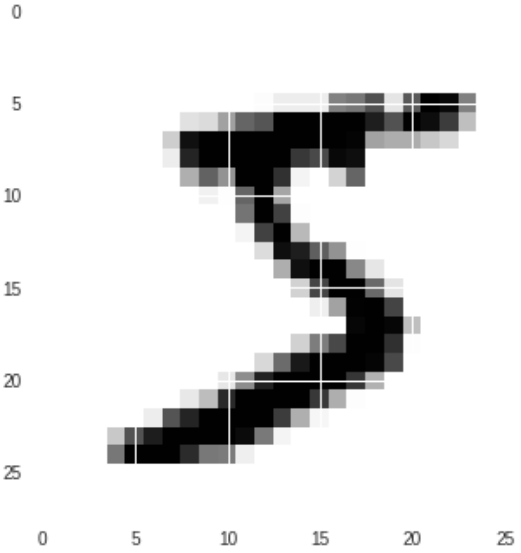
In [3]:
```
import matplotlib.pyplot as plt
plt.imshow(x_train[8], cmap=plt.cm.binary)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x7fff30123e10>
```



```
[9] import matplotlib.pyplot as plt
    plt.imshow(x_train[0], cmap=plt.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x7fbd96975b70>
```

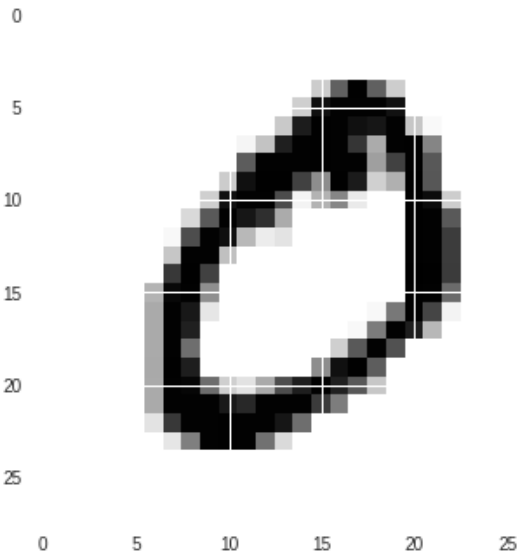



```
[11] print (y_train[0])
```

```
5
```

```
[12] import matplotlib.pyplot as plt
     plt.imshow(x_train[1], cmap=plt.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x7fbd96951630>
```




```
print (y_train[1])
```

```
0
```

1. Load and prepare data

- Normalizing the data
  - Original images of 28×28 pixels whose values range from [0, 255] of type uint8.

```
In [4]: print(y_train[8])
```

```
1
```

```
In [5]: print(x_train.ndim)
```

```
3
```

```
In [6]: print(x_train.shape)
```

```
(60000, 28, 28)
```

```
In [7]: print(x_train.dtype)
```

```
uint8
```

1. Load and prepare data

- Normalizing the data
  - But it is usual to scale : in the example to [0, 1].

```
[8] x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')

    x_train /= 255
    x_test /= 255
```

1. Load and prepare data

- reshape 2D → 1D

```
[9] print(x_train.shape)
    print(x_test.shape)
```

```
(60000, 28, 28)
(10000, 28, 28)
```

```
[10] x_train = x_train.reshape(60000, 784)
     x_test = x_test.reshape(10000, 784)
```

```
[11] print(x_train.shape)
     print(x_test.shape)
```

```
(60000, 784)
(10000, 784)
```

1. Load and prepare data

- categorize data (one-shot):

```
[13] print(y_train[0])

     5
```

```
[14] print(y_train.shape)

     (60000,)
```

```
[16] from tensorflow.keras.utils import to_categorical
```

1. Load and prepare data

- categorize data (one-shot):

```
[17] y_train = to_categorical(y_train, num_classes=10)
     y_test = to_categorical(y_test, num_classes=10)
```

```
[18] print(y_test[0])
     print(y_train[0])

     [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
     [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

```
[19] print(y_train.shape)
     print(y_test.shape)

     (60000, 10)
     (10000, 10)
```

2. Define model

```
In [24]: from tensorflow.keras import Sequential
         from tensorflow.keras.layers import Dense

         model = Sequential()
         model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
         model.add(Dense(10, activation='softmax'))
```

Model: core data structure

- keras.models.Sequential class is a wrapper for the neural network model
- Models in Keras are defined as a sequence of layers.

2. Define model

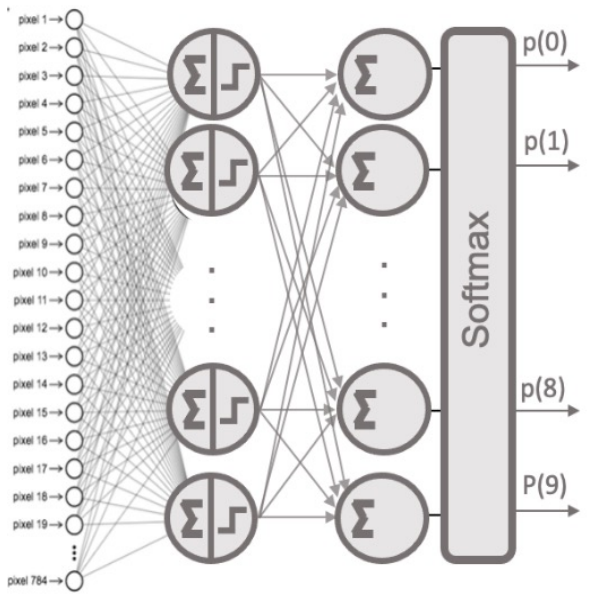
```
In [25]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                7850
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
=================================================================
Total params: 7,960
Trainable params: 7,960
Non-trainable params: 0
_________________________________________________________________
```

- Keras will automatically infer the shape of all layers after the first layer.
- This means you only have to set the input dimensions for the first layer.

2. Define model

```
In [24]: from tensorflow.keras import Sequential
         from tensorflow.keras.layers import Dense

         model = Sequential()
         model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
         model.add(Dense(10, activation='softmax'))
```

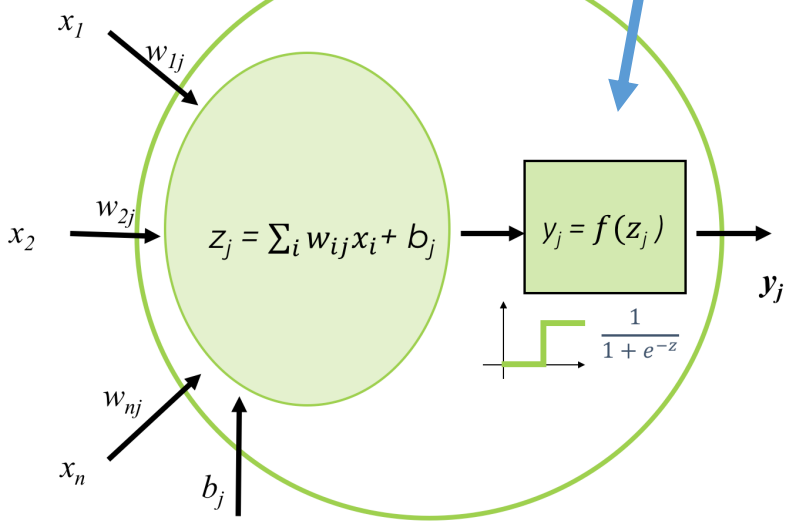


2. Define model

```
In [24]: from tensorflow.keras import Sequential
         from tensorflow.keras.layers import Dense

         model = Sequential()
         model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
         model.add(Dense(10, activation='softmax'))
```

Number of neurons per layer

Type of activation function



2. Define model

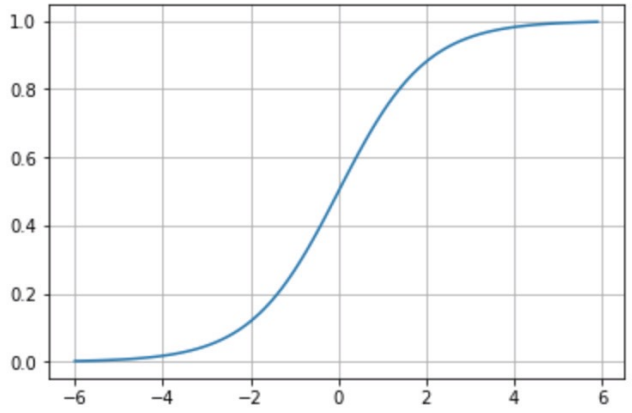
```
In [24]: from tensorflow.keras import Sequential
         from tensorflow.keras.layers import Dense

         model = Sequential()
         model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
         model.add(Dense(10, activation='softmax'))
```

$$y = \frac{1}{1+e^{-z}}$$

The sigmoid function allows us to reduce extreme or atypical values without eliminating them.

converts independent variables of almost infinite range into simple probabilities between 0 and 1.

Most of the output will be very close to the extremes of 0 or 1, as we have already commented.



2. Define model

In [24]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(10, activation='sigmoid', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

Softmax?

2. Define model

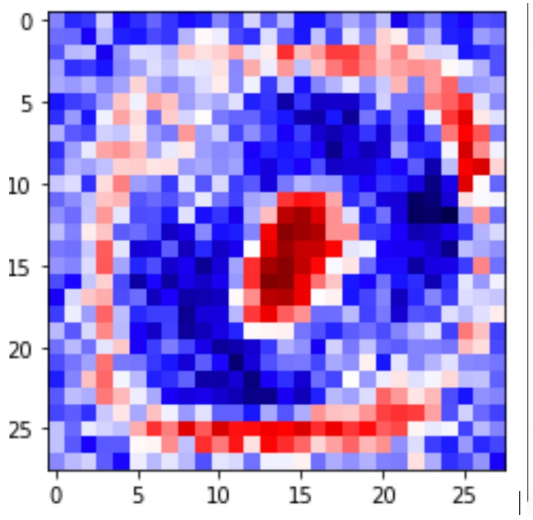
- The *softmax* function has two main steps:
  - first, the "**evidences**" for an image belonging to a certain label are computed,
  - and later the evidences are converted into **probabilities** for each possible label.

- Example: MNIST database

Evidence of belonging

- An approach to measure the evidence that a certain image belongs to a particular class is to make a weighted sum of the evidence of belonging to each of its pixels to that class.

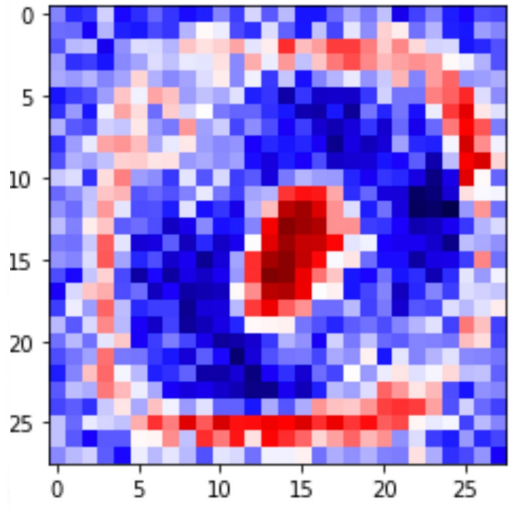
To explain the idea I will use a visual example ->



Evidence of belonging

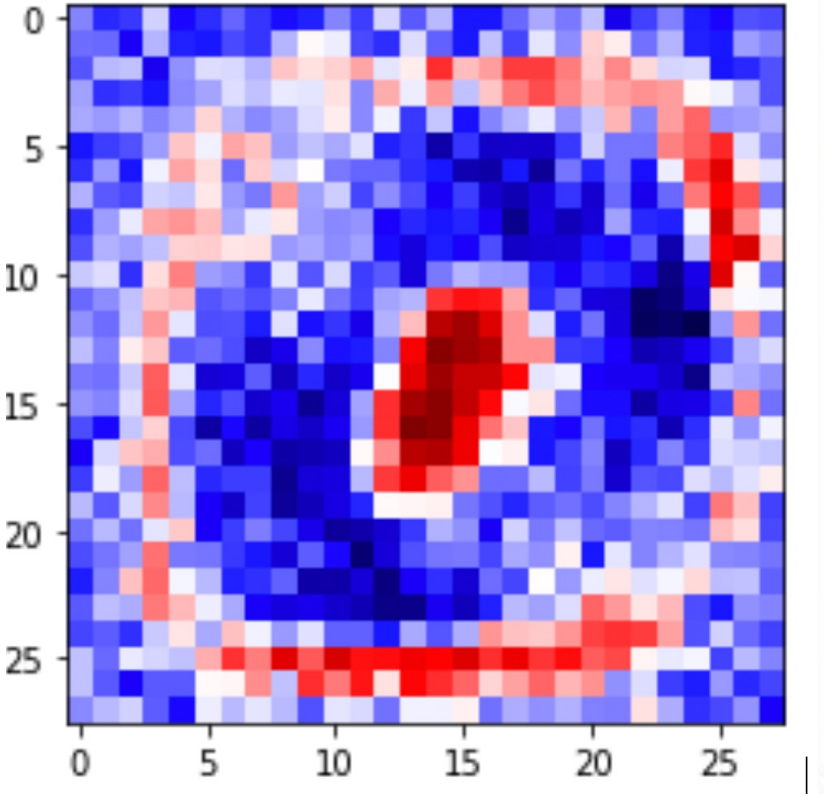
Let’s suppose that we already have the model learned for the number zero (28x28):

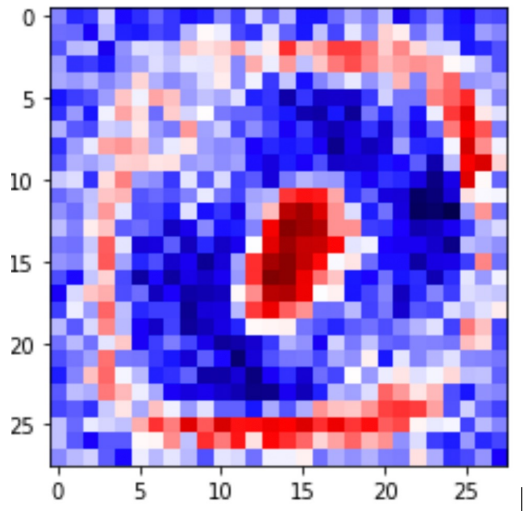
- Pixels in **red** represent negative weights (i.e., reduce the evidence that it belongs),
- Pixels in **blue** represent positive weights (the evidence of which is greater increases).
- The **black** color represents the neutral value.



Evidence of belonging

- Let's imagine that we **trace a zero** over it.
  - **In general, the trace of our zero would fall on the blue zone**
  - It is quite evident that if our stroke goes over the red zone, it is most likely that we are not writing a zero;
  - therefore, using a metric based on adding if we pass through the **blue** zone and subtracting if we pass through the **red** zone seems reasonable.



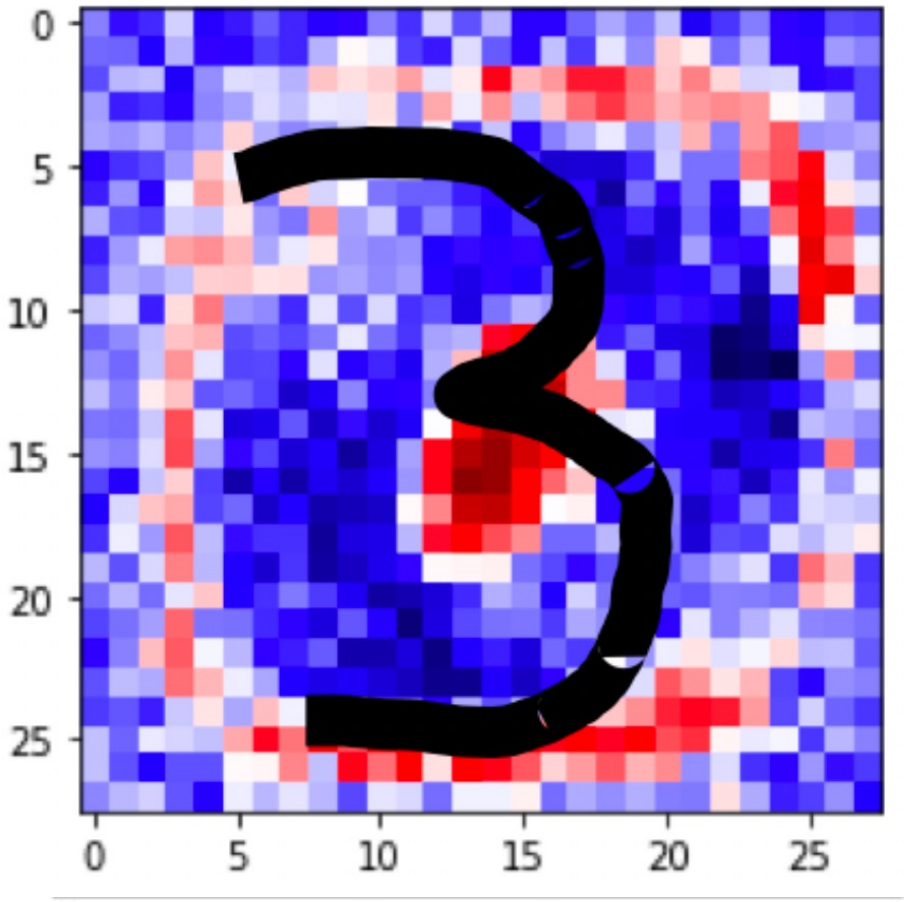



```
[[ 22. -21.  11.  -8.   0. -17.  -4. -19.  12.  21. -10.  12. -20. -19.  11. -21.  14. -13. -21. -10.  17.  -5. -11.  15. -16. -19. -21.   4.]
 [ 10.   1.   2.   5.  20.  -5.  16. -19.  -1. -16.  -9.  25.  -9. -11. -48.   7. -10. -14. -15. -11.   6. -11.  17. -21.  20. -17. -20. -18.]
 [  1.  17.  16.   3. -19. -16.   3. -15. -33. -46. -56. -53. -42. -56. -99. -88. -104. -98. -86. -79. -42. -54. -51. -37.  -4.  13.  21.  17.]
 [ -4.  -7.   5.   5. -27.  -9. -17. -22. -12. -29. -46. -33. -50. -41. -31. -50. -13. -44. -82. -49. -66. -55. -78. -52. -34.  -4. -19. -14.]
 [ 11.  -4.  -7.  13. -60. -44. -68. -50. -38. -30. -23. -23.  -1. -10. -21.  13. -24. -13.  -6. -19. -21. -15. -21. -58. -61. -34. -63. -33.]
 [ 16.  -8. -30. -21. -72. -46. -45. -38. -28. -22. -23. -21. -13.   3.  43.  32.  15.  19.  21.  -1. -16.   7. -21. -44. -81. -92. -66. -14.]
 [-21.  10. -31. -49. -46. -33. -36. -36. -10. -36. -15.   4.  -6.   9.  32.  26.  24.  55.  44.  16.  29.   6.  33.  -2. -82. -142. -58. -24.]
 [ -7. -26. -23. -46. -89. -16.  -3. -37. -37. -31.  -1.  18.  17.  24.   9.  16.  28.  42.  36.  38.  22.  16.  16.  19. -50. -107. -89. -12.]
 [ 29. -38. -21. -23. -62. -50. -33. -27. -33.  -6. -14.  -2.   7.  -2.  10.  28.  42.  62.  60.  23.   1.  18.  32.  26. -43. -155. -98. -39.]
 [ -8. -13.  -1. -42. -50. -24. -43. -41. -24.   8.   8.   1.  19.  34.  18.   7.  21.  42.  47.  43.  19.   7.  34.  46.   6. -145. -89. -33.]
 [  2. -10.   9. -27. -32. -16. -10. -11. -10.  -9. -24. -15.  16.   7.   2. -28. -24.  27.  44.  28.  11.  42.  22.  40.  23. -96. -94. -34.]
 [ -9.  -1.   1. -53. -62. -39. -14. -18. -11. -23. -31.  -1. -15. -22. -71. -92. -100. -47.  -7.   9.   8.  19.  54.  40.  60. -71. -95. -49.]
 [ 10.   8. -32. -22. -42. -13.  21.   4.  -7.   5.  17.  13. -14. -65. -112. -156. -118. -64. -31.   8.  23.  25.  57.  68.  81. -17. -39. -27.]
 [  4.   1.   5. -22. -14. -12.  27.  11.  -3.  15.  11.  15.  -4. -68. -132. -165. -133. -56.  -9. -20. -17.  15.  46.  47.  62. -12. -61. -38.]
 [ 12. -24. -12. -38. -38.  29.  24.  57.  27.  12.  21.   5. -36. -103. -152. -139. -112. -70. -13. -22.  -7.  12.  49.  67.  63. -15. -40.  -6.]
 [-18.  -5. -51. -44.   8.  34.  25.  44.  36.  53.  54.   1. -45. -118. -181. -130. -104. -55. -14. -22.  14.  11.  22.  39.  32. -27. -43. -18.]
 [-17. -20. -20. -54.   0.  30.  30.  47.  26.  60.  45.  -9. -72. -130. -129.  -99.  -56.  -3.   1.  19.  -8.  18.   5.   7.   9. -49. -44. -20.]
 [ 20. -19. -33. -68.   0.  13.  25.  45.  13.  47.  41.  -7. -74. -144. -119.  -89.  -48.  -7.   2.  -6.  20.   6.  26.   2.   9. -39. -36. -10.]
 [-23.  14. -40. -85. -11.  -1.   4.  34.  17.  62.  73.   8. -73.  -92.  -92.  -38.  -12.   2. -18.   3.  -2.  20.  22.   1. -38. -30. -18. -29.]
 [  9.  17. -11. -74. -31.   3.  27.   2.  34.  48.  37.  31.   9. -51. -12. -19.  -8.   7. -33.  -5. -30.  -7.   8. -11. -37. -47. -47. -37.]
 [-22.  -4. -32. -54. -19.  12.  21.  -8.  23.  16.  39.  55.  46.  -3.  11. -16.  -0. -28. -19. -14.  -1.  14. -18. -10. -57. -30.   5.   1.]
 [-12. -25. -30. -42. -11.  -7.  13.   5.  21.  10.  50.  54.  18.  34.   4.  20.  -7. -22. -15. -49.  -7. -30.  -4. -19. -52. -34.   9.  10.]
 [-17.   2.  -5. -31. -75. -15.  20.  -3.  19.  11.  49.  49.  38.  48.  21.  10.  -7. -12. -20. -16. -41. -27.  -7. -17. -18. -21.  -4.  10.]
 [ -3. -13. -26. -56. -77. -46. -24.  -1.  12.  36.  49.  47.  44.  51.  61.  24.   8.   4. -36. -51. -52. -52. -27. -75. -49. -15.   1.  -9.]
 [ 14. -16. -45. -28. -31. -45. -64. -52. -30. -18.   4.  16.  24.   7.   5. -13. -18. -64. -84. -71. -99. -99. -66. -56. -42.  -5.   4.   9.]
 [-22. -14.   3. -15. -29. -46. -51. -75. -103. -105. -129. -119. -125. -121. -142. -119. -134. -117. -102. -86. -76. -80. -30. -65. -34.  -0. -20.  -7.]
 [-13.   6.   3. -11.   3. -52. -21. -32. -73. -100.  -87. -98. -89. -85. -98. -72. -93. -69. -82. -39. -53. -62.  -6.   4. -22.   0.  18.  10.]
 [ -9.  15.   3.  15.  -8. -33.  12. -24. -25. -14. -40. -24. -46. -45. -38. -51. -13. -19. -41. -10. -45. -25. -14.  -6.  12.  21. -13. -19.]]
```

Evidence of belonging

- To confirm that it is a good metric, let's imagine now that **we draw a three**
- it is clear that the red zone of the center of the previous model that we used for the zero will penalize the aforementioned metric since,

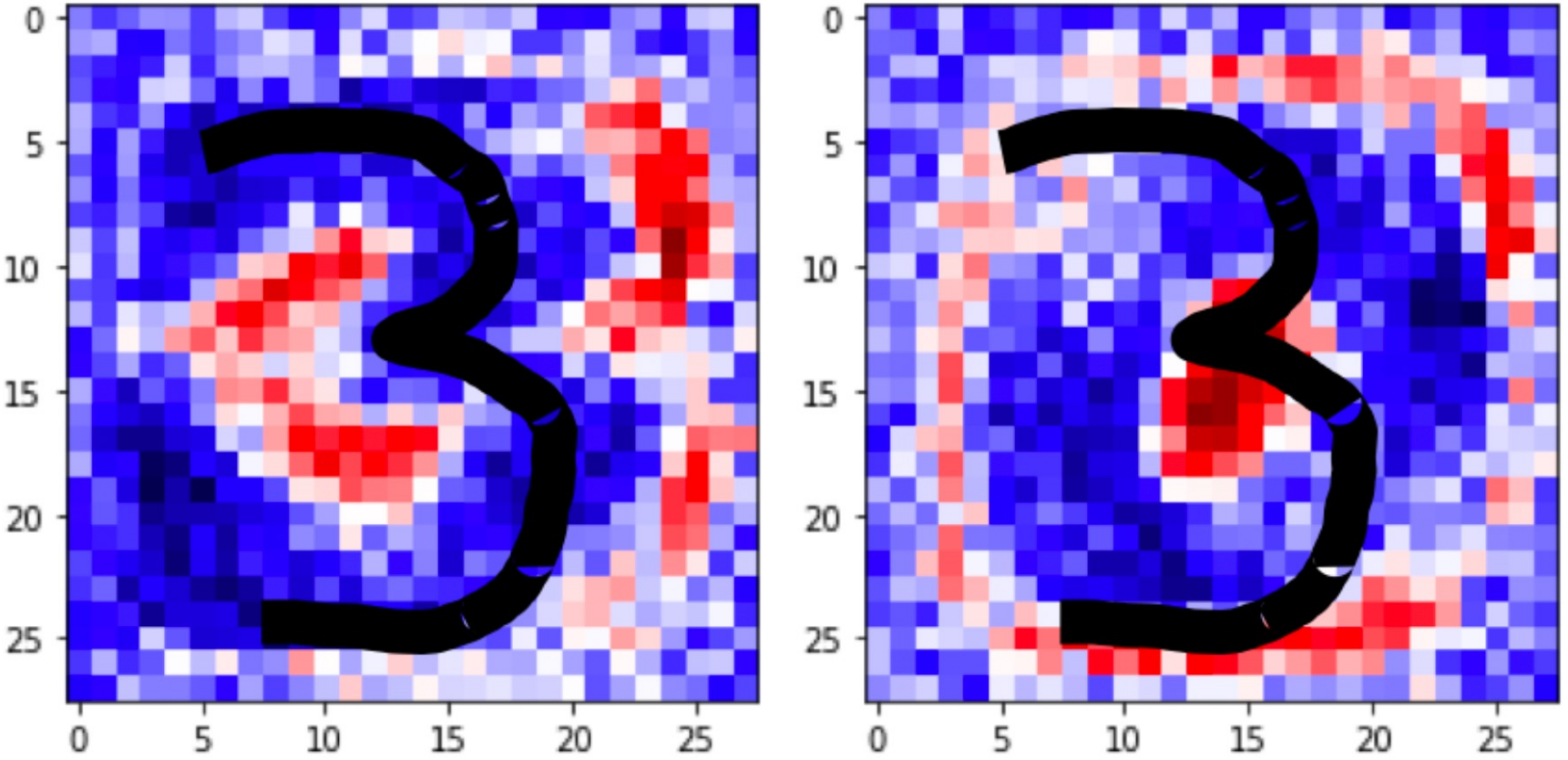
- as we can see in the figure, when writing a three we pass over



Evidence of belonging

But, on the other hand, if the reference model is the one corresponding to number 3

we can see that the different possible traces that represent a three are mostly maintained in the blue zone.



Probability of belonging

- The **second step** involves computing probabilities.
- Specifically **we turn the sum of evidences into predicted probabilities** using this function:

$$Softmax_i = \frac{e^{evidence_i}}{\sum_j e^{evidence_j}}$$

- softmax uses the exponential value of the calculated evidence and then normalizes them so that the sum equates to one, forming a probability distribution.

Probability of belonging

$$e^{evidence_i}$$

- Intuitively, the effect obtained with the use of exponentials is that one more unit of evidence has a multiplier effect and one unit less has the inverse effect.
- The interesting thing about this function is that
  - **a good prediction** will have a single entry in the vector with a value close to 1, while the remaining entries will be close to 0.
  - in a **weak prediction**, there will be several possible labels, which will have more or less the same probability.

Softmax

- A function that provides probabilities for each possible class in a **multi-class classification model**. The probabilities add up to exactly 1.0.
  - For example, softmax might determine that the probability of a particular image being a “7” at 0.8, a “9” at 0.17, ...

```
predictions = model.predict(x_test)
```

```
print(predictions[0])
```

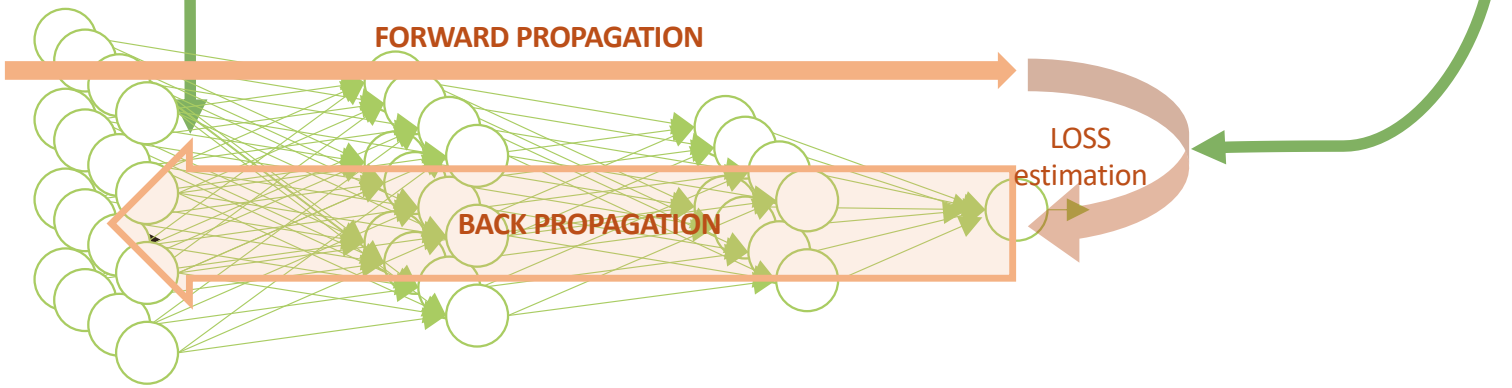
```
[0.   0.   0.   0.01 0.01 0.   0.   0.8  0.01 0.17]
```

```
np.sum(predictions[0])
```

```
1.0
```

3. Compile model

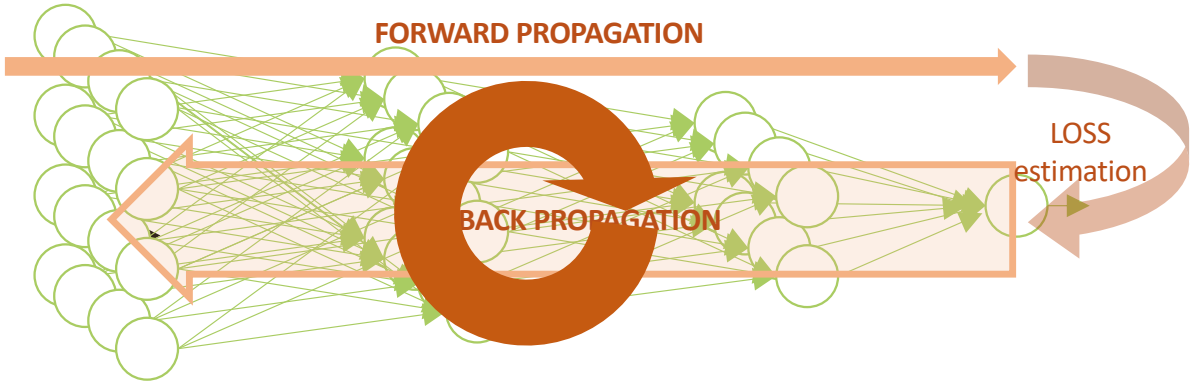
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```



4. Fit model

```
[24] model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
60000/60000 [==============================] - 6s 98us/sample - loss: 0.4039 - accuracy: 0.8930
Epoch 2/5
60000/60000 [==============================] - 6s 94us/sample - loss: 0.1958 - accuracy: 0.9441
Epoch 3/5
60000/60000 [==============================] - 5s 88us/sample - loss: 0.1421 - accuracy: 0.9590
Epoch 4/5
60000/60000 [==============================] - 5s 86us/sample - loss: 0.1101 - accuracy: 0.9688
```



5. Evaluate model

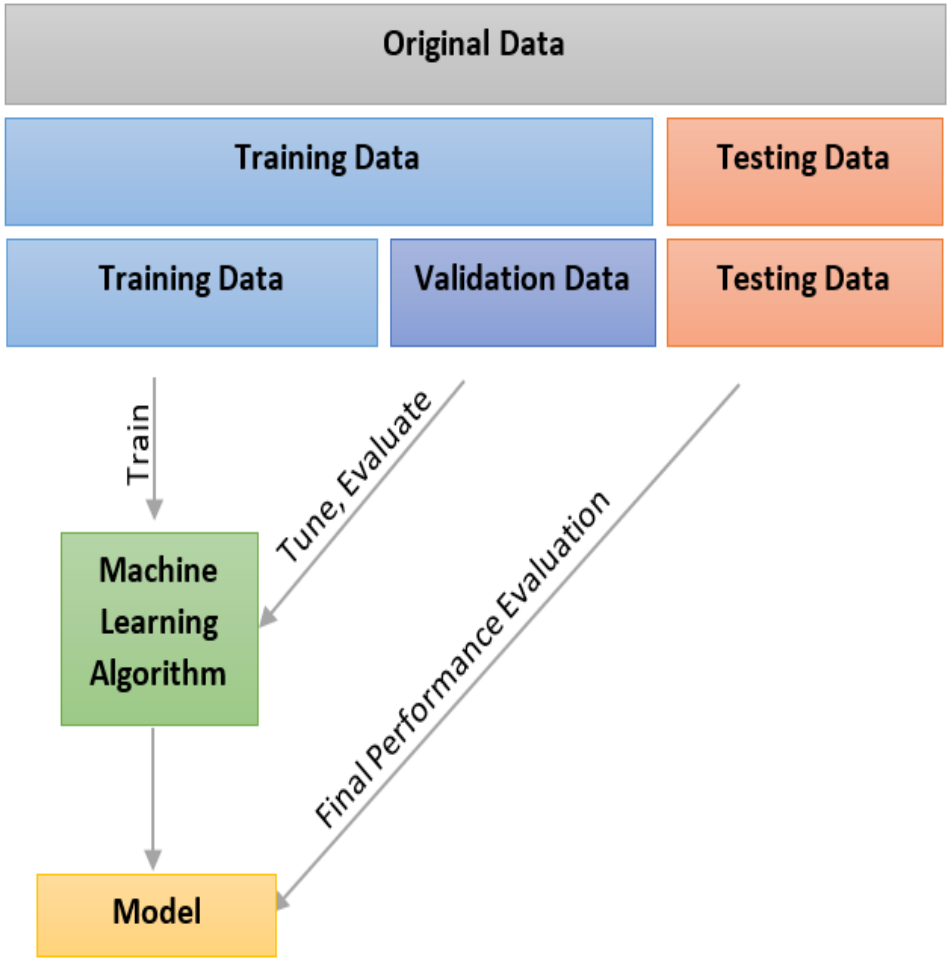
```
[25] test_loss, test_acc = model.evaluate(x_test, y_test)
     print ('Test accuracy:', test_acc)
```

```
10000/10000 [==============================] - 0s 48us/sample - loss: 0.0955 - accuracy: 0.9716
Test accuracy: 0.9716
```

5. Evaluate model

Data

- Training set
  - → **For training**
- Validation set
  - → **For hyperparameter tu**
- Test set
  - → **Test model performan**



5. Evaluate model

- Confusion matrix

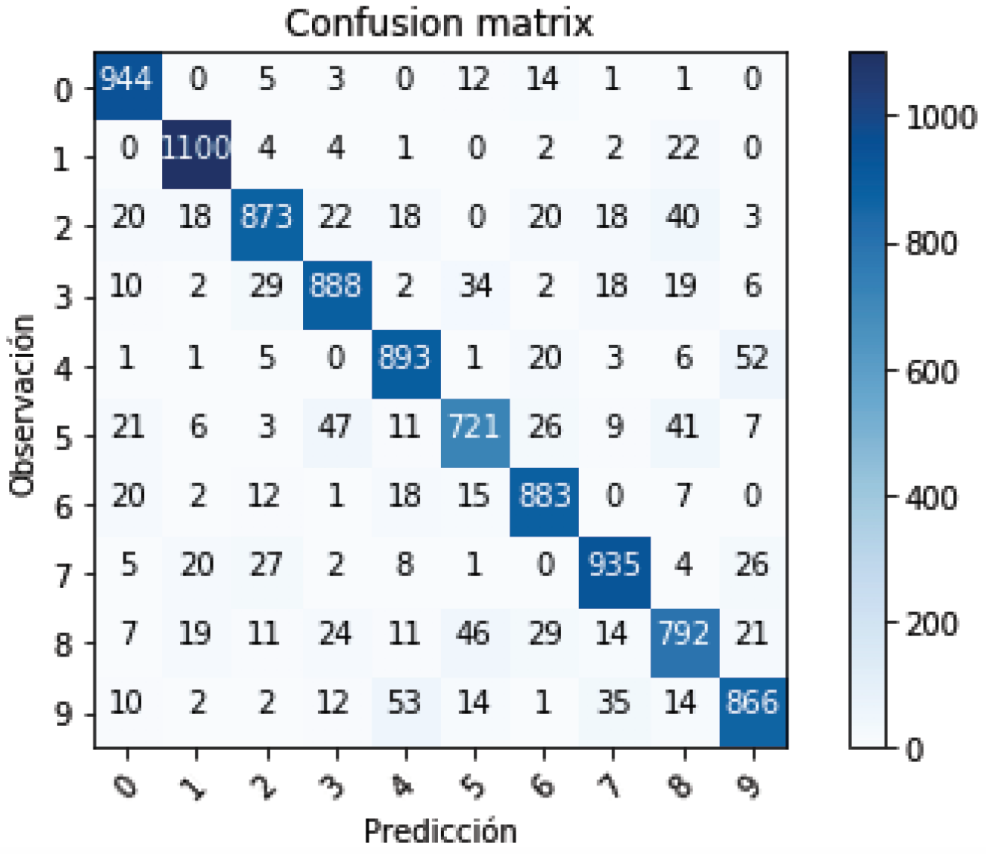
		Predicted class	
		positive	negative
Actual class	positive	TP	FN
	negative	FP	TN

- Accuracy

$$Acc = (TP + TN) / (TP + TP + FN + FP)$$

5. Evaluate model

Confusion matrix for this model



6. Use the model

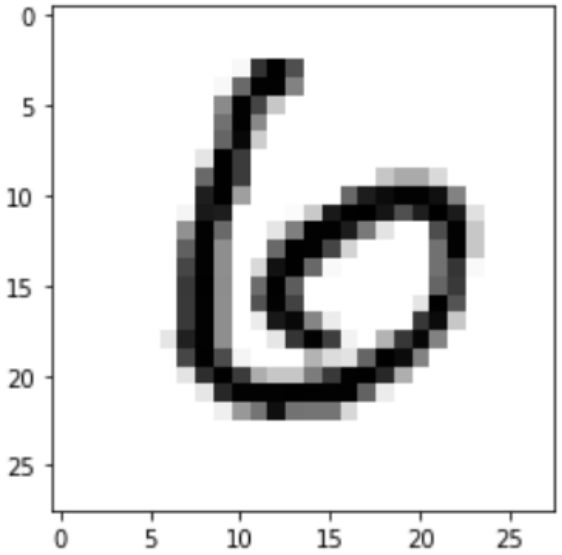
```
[31] predictions = model.predict(x_test)
     print (predictions)
```

```
[[1.26404475e-05 2.49308573e-06 4.45205660e-04 ... 9.93939281e-01
  2.26089851e-05 3.87840439e-04]
 [1.40489414e-04 5.95517922e-03 9.92044926e-01 ... 7.36168673e-08
  8.24517483e-05 5.90428044e-08]
 [4.98437309e-08 9.97506320e-01 1.01395021e-03 ... 9.72060545e-04
  2.70858494e-04 1.64484100e-05]
 ...
 [1.58638017e-07 4.75961940e-07 7.93745471e-07 ... 1.92583975e-04
  1.82772157e-04 4.21503466e-03]
 [2.74540548e-06 5.57959866e-05 4.68355069e-07 ... 3.29460909e-06
  5.54329483e-03 1.57847660e-06]
 [2.11494735e-05 1.39356416e-06 7.07783474e-05 ... 2.89122028e-07
  5.49738729e-07 9.28026580e-08]]
```

6. Use the model

In [32]:
```
x_test_old = x_test.reshape(10000, 28,28)
plt.imshow(x_test_old[11], cmap=plt.cm.binary)
```

Out[32]:
```
<matplotlib.image.AxesImage at 0x7ffec02fcb10>
```



6. Use the model

```
In [33]: predictions = model.predict(x_test)

In [34]: np.argmax(predictions[11])
Out[34]: 6

In [35]: print(predictions[11])
[0.0705287  0.01105603 0.19185546 0.00774726 0.05123473 0.03251035
 0.536685   0.00300946 0.08143999 0.01393303]

In [36]: np.sum(predictions[11])
Out[36]: 1.0
```

Pf: Deep Learning Basics (using Colab)

- **Tasks included:**
  - task 7.1 – Set up your google colab environment
    task 7.2 – Execute the provided notebook step-by-step
    task 7.3 – Improve the accuracy of your model

TODAY

part 1

  - task 8.1 – Improving a basic cnn model
    task 8.2 – Exporting the python script
    task 8.3 – Running your first natural network on a login node
    task 8.4 – Submitting your first gpu dl training with slurm

part 2

  - task 9.1 – Comparative implementation in pytorch and tensorflow

Pf: Deep Learning Basics (using Colab)

- **In class:**
  - This lab is mainly **pedagogical**, aimed at helping you understand the fundamental concepts of deep learning.
  - During lab class, the instructor will **assist each student individually** to consolidate understanding.
- **Deliverable & Evaluation:**
  - Upload a single PDF (per group) to the **racó@FIB** intranet.
  - This PDF can simply include **screenshots or short notes** showing that you have completed each task.
    The goal is to confirm that you worked through the exercises, not to produce a polished "report".
- **Evaluation day:**
  - A group will give a **brief and informal explanation** of their results.
  - "No preparation is required" — **a few comments or screenshots are enough**.
    The focus is on learning, not on presentation style or timing.