Physalia Courses

A deep-learning neural network for image recognition

A working model

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

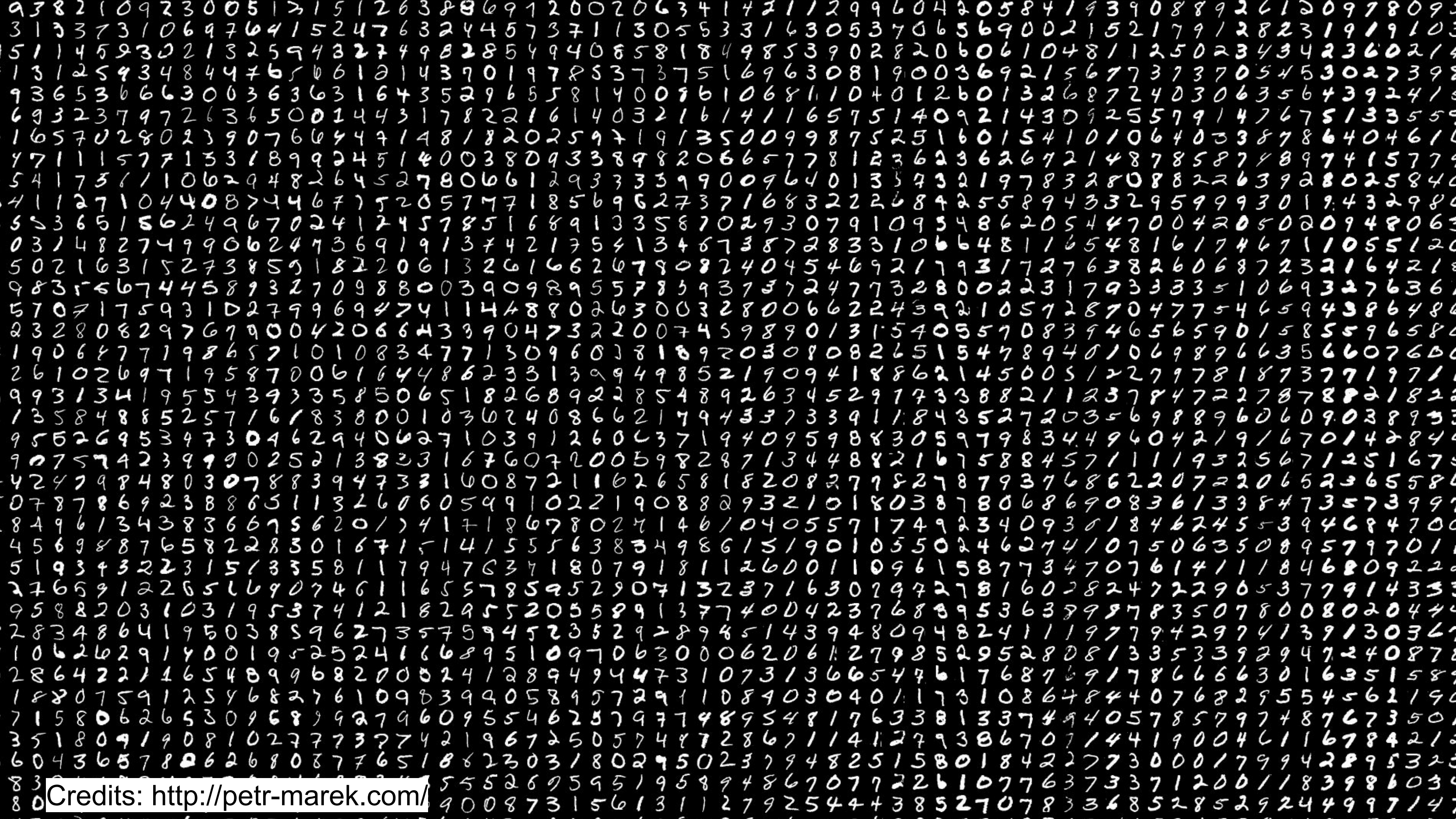
Nelson Nazzicari
Senior Scientist
CREA, Lodi (Italy)

Objectives

- we start by **showing a working deep learning model for image recognition** (a task at which DL is very good!)
- the objective is to give you some ideas of **what DL is about**
  - no worries if you don’t understand everything
  - we’ll delve in details in later sessions
- you’ll get some **basic intuition** of **what DL is** and **how it is structured**

A first working example

- **MNIST** (Modified National Institute of Standards and Technology) database → large collection of **handwritten digits** [more info here]
- Commonly used to train machine learning models for image recognition
- The aim is to use this database to build a **first deep learning model** for **image recognition**



Credits: http://petr-marek.com/

A first working example

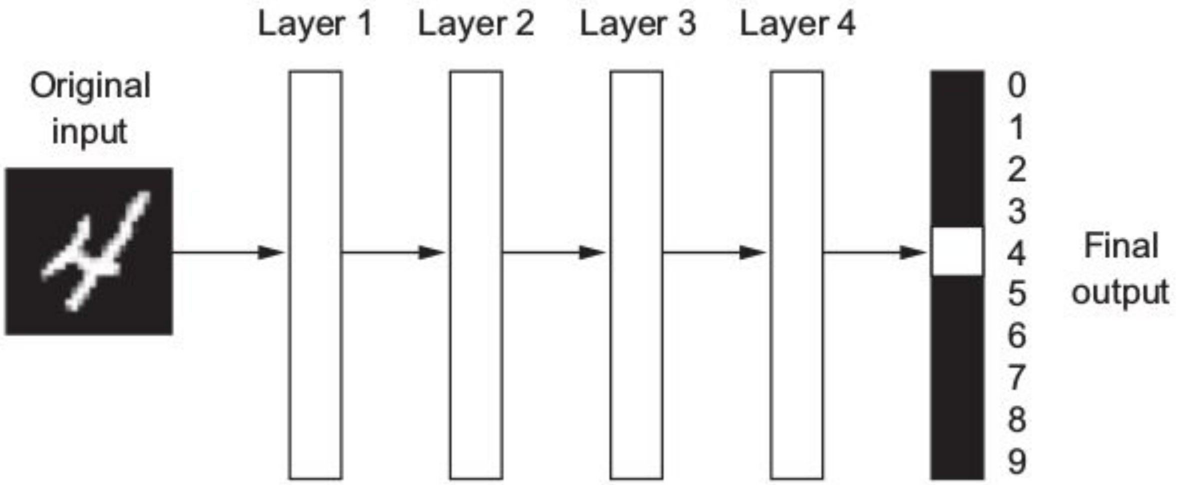
- From the MNIST dataset
  - **60,000 images for training**
  - **10,000 images for testing**
  - Reference: http://yann.lecun.com/exdb/mnist/

A first working example

- From MNIST
  - **60,000 images for training**
  - **10,000 images for testing**
  - Reference: http://yann.lecun.com/exdb/mnist/

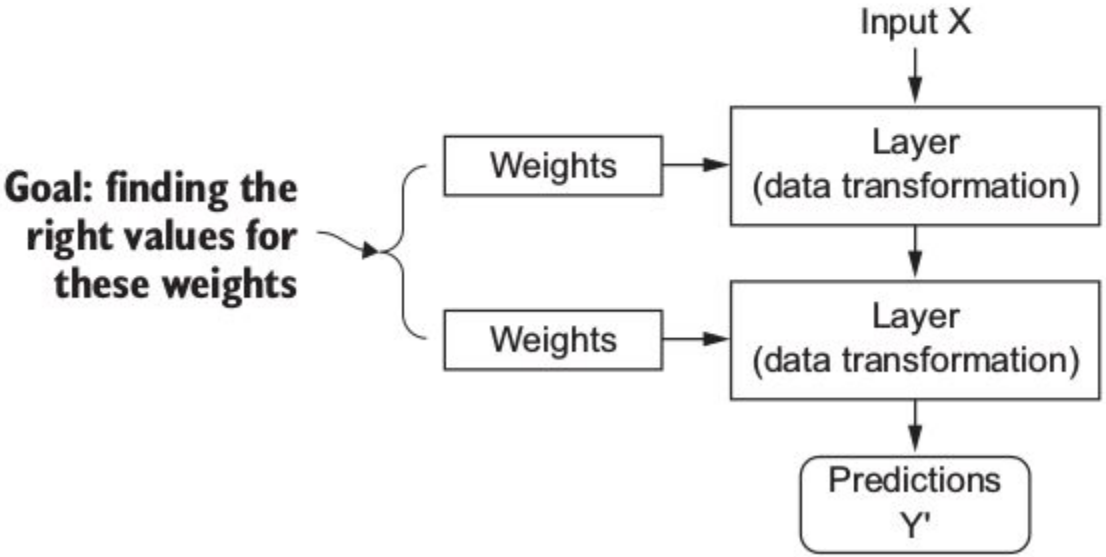
1) Step 1: **train the deep learning model**
2) Step 2: get **predictions** (recognize images/handwritten digits) on **test data**
3) Step 3: measure the **accuracy of prediction**

Handwritten digit recognition



From François Chollet

Model diagram



Input X

Goal: finding the right values for these weights

Weights

Layer (data transformation)

Weights

Layer (data transformation)

Predictions Y'

From François Chollet

The needed tools

- **Python (3)**
- **Interactive Python Notebook** (.ipynb file) → **Jupyter** notebooks
- [Google colab]
- **Keras** (wrapper around Tensorflow) [more on this later]

A first working example - components

1. SETUP
   - import libraries
   - configure parameters
2. DATA MANAGEMENT
   - load MNIST data
   - data (images) preprocessing
3. MODEL
   - build
   - compile
   - train
   - test

A first working example - components

1. SETUP
   - import libraries
   - configure parameters

The standard part

2. DATA MANAGEMENT
   - load MNIST data
   - data (images) preprocessing

The boring part

3. MODEL
   - build
   - compile
   - train
   - test

The cool part

A first working example - steps (real world)

1. SETUP
   - import libraries
   - configure parameters
2. DATA MANAGEMENT
   - load MNIST data
   - data (images) preprocessing
3. MODEL
   - build
   - compile
   - train
   - test
4. RINSE AND REPEAT

The standard part

The boring part

The cool part

The professional part

Let’s do it!

1. “Black box”
2. Decomposing the model

1- the Black Box

- Training the model
- Getting prediction accuracy on test data

- go on your computer/server
- open a terminal
- run **keras.mnist_train.py**
- run **keras.mnist_test.py**

2- decomposing the model

- chunk-by-chunk training and testing
- interactive Jupyter notebook

- **`day1_code01_keras_MNIST.ipynb`**
- `day2_code00_keras_MNIST_detailed.ipynb [tomorrow]`