

A deep-learning neural network for image recognition

A working model

Filippo Biscarini Senior Scientist CNR, Milan (Italy) Nelson Nazzicari Researcher 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







```
a382(09730051>>15(26388691200706341411/299604405841/399088916120918093
511459302132594327498285494085818498539028206061048112502343423602/
1312543484476/1061214370197853737516963081900369215617313705453027395
936536663063636316435296558140016106811040126013268724030635643924/3
6932379726365001443178221614032161411657514092143092559914767513355
16570280239076644414818202597191350099875251601541010640338786404616
4711157715318992451400380933898206657781236236269214878587×89741577
54173111062948264527806612933359900964013543219783280882263928025840
41127104408>4467)52051M718569627371683222684255894332959993019432980
55365156249670241245185168913358702930791095486205447004205020948063
03148274996624736919134421754\346738728331066481165481617461\1055126
5021631527385918220613261662479082404546921793172763826068723216421:
9831667445893270988003909895578593727247732800223179333351069327636
5707175931D27996942411H488026300328006622439210592870477546594386486
2328082976190042066H3390H7322007H59890131540557083946565901585596584
19068771986521010834771309603818920308082651549894810698966356607606
26102697195870061644862331399498521909418862145005/2299781873791971.
99313419554393358506518268922854892634529173388211237847227878821826
1358488525716183800103624086621794332339118435272035698896060903895
95626953973046294062710391260637194095988305979834496042/9/67014284
9075742399902521382316760720059828713448821675884571111932567125167:
42419848030788394733140872116265818208271827877376862207220652365586
07878692388651132606054910221908829321018038780686908361338473573999
8496/34383669$620/)41+18698024146/04055917492340936/846245=394684700
4569887658228301671514/55563834986/5/9010550246274/075063508957970/
5 | 93 + 3 2 2 3 | 5 / 3 3 5 8 | | 7 9 4 7 6 3 7 | 8 0 7 9 | 8 | 1 | 2 6 0 0 | 1 | 0 9 6 | 5 8 7 7 3 4 7 0 7 6 | 4 / 1 / 8 4 6 8 0 9 2 2 a
7465912265669074611655785952907132371630197778160282472296537791433:
958820310319537412182955205589137740D423768895363898783507800802044
2834864195038596273575945235292894514394809482411192794299413913036
1062629140019525241668951097063000620612798529528081335339294724087
2864221165489968200824128949447310731366549617687691786666301635158
18807591254682761098399058957291108403040/17310864844076829554561190
7 | 5 8 | 6 | 6 | 6 | 5 | 6 | 8 | 9 | 9 | 9 | 9 | 6 | 0 | 9 | 5 | 5 | 6 | 2 | 5 | 7 | 9 | 7 | 7 | 7 | 8 | 9 | 5 | 7 | 8 | 7 | 6 | 7 | 8 | 7 | 6 | 7 | 8 | 7 | 6 | 7 | 8 | 7 | 6 | 7 | 8 | 7 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7 | 8 | 7
35180919081027773774219672505944728621141279386707144190046116784210
604365788626808776518823031802950239948251580184227730001/7994289$32:
83 Credits: http://petr-marek.com/ 55526059519689486707722610776373371200/183986039
```

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
- Step 2: get predictions (recognize images/handwritten digits) on test data
- 3) Step 3: measure the accuracy of prediction

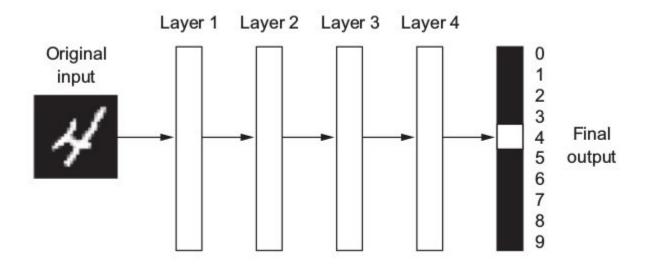






Handwritten digit recognition





From François Chollet

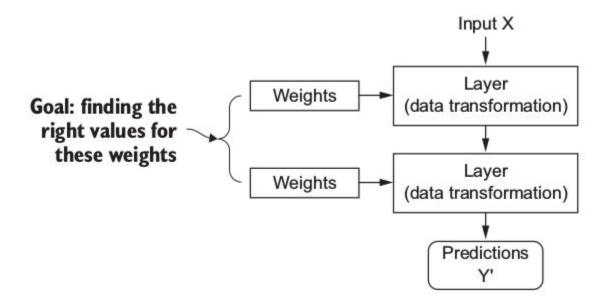


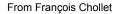




Model diagram













The needed tools



- Python (3)
- Interactive Python Notebook (.ipynb file) → Jupyter notebooks
- Google colab / Docker on server [our working environment]
- Keras (wrapper around Tensorflow) [more on this later]







The needed tools



- Python 3
- Interactive Python Notebook (.ipynb file) → Jupyter notebooks
- Google colab / Docker on server [our working environment]
- Keras (wrapper around Tensorflow) [more on this later]



What is your skill level with these tools?

- feel free to ask questions
- use our slack chat to interact with us (one of the two instructors will always be there)
- if needed we can go back on some of this material







A first working example - components



SETUP

- import libraries
- configure parameters

DATA MANAGEMENT

- load MNIST data
- data (images) preprocessing

MODEL

- build
- compile
- train
- test







A first working example - components



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

The standard part

The boring part

The cool part







A first working example - steps (real world)



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

- go on the server
- code 01 keras MNIST.ipynb





