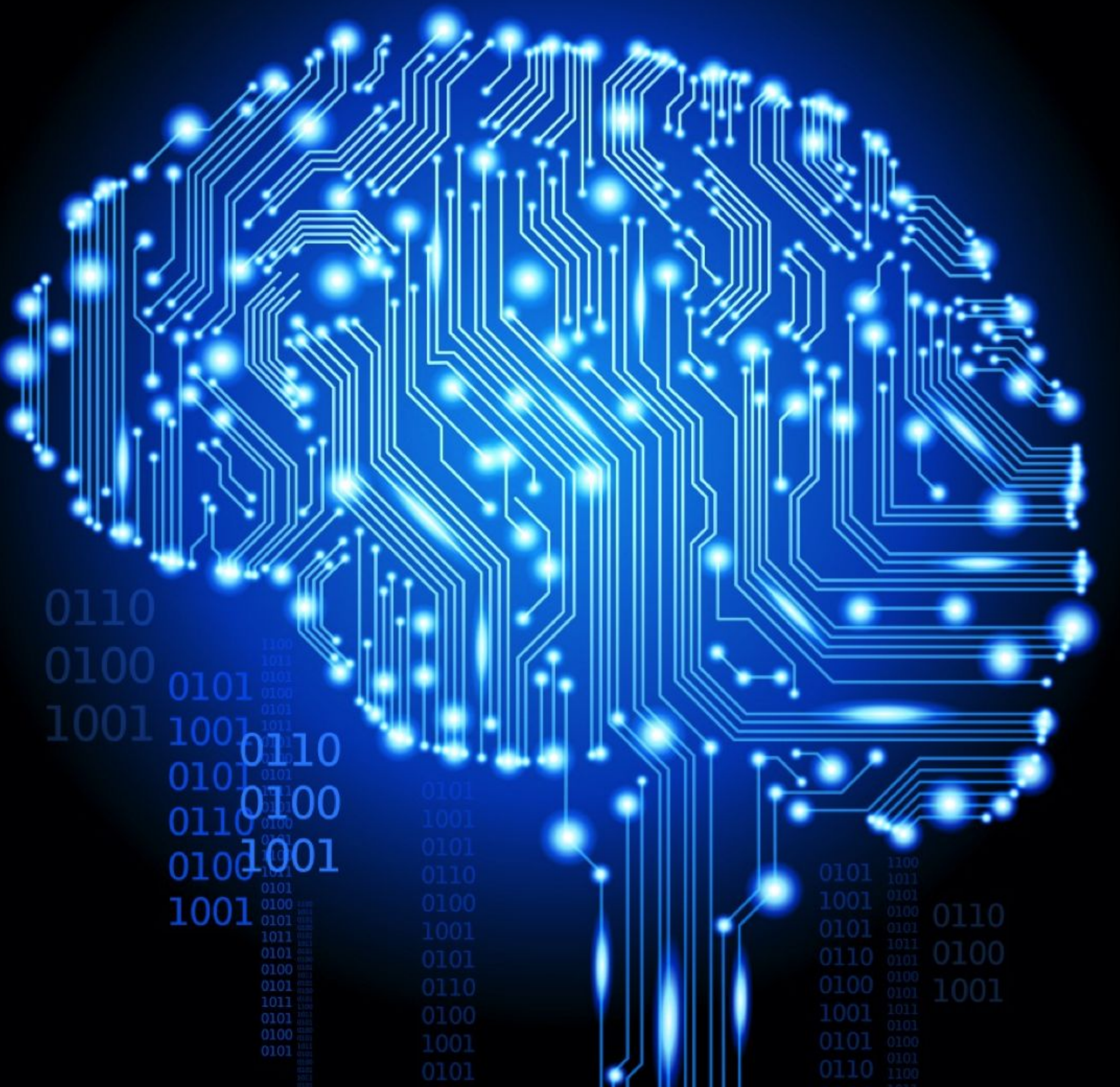
# Keras basics

## ESADE - MIBA (FALL 2017)

JORDI TORRES | FRANCESC SASTRE



# Keras

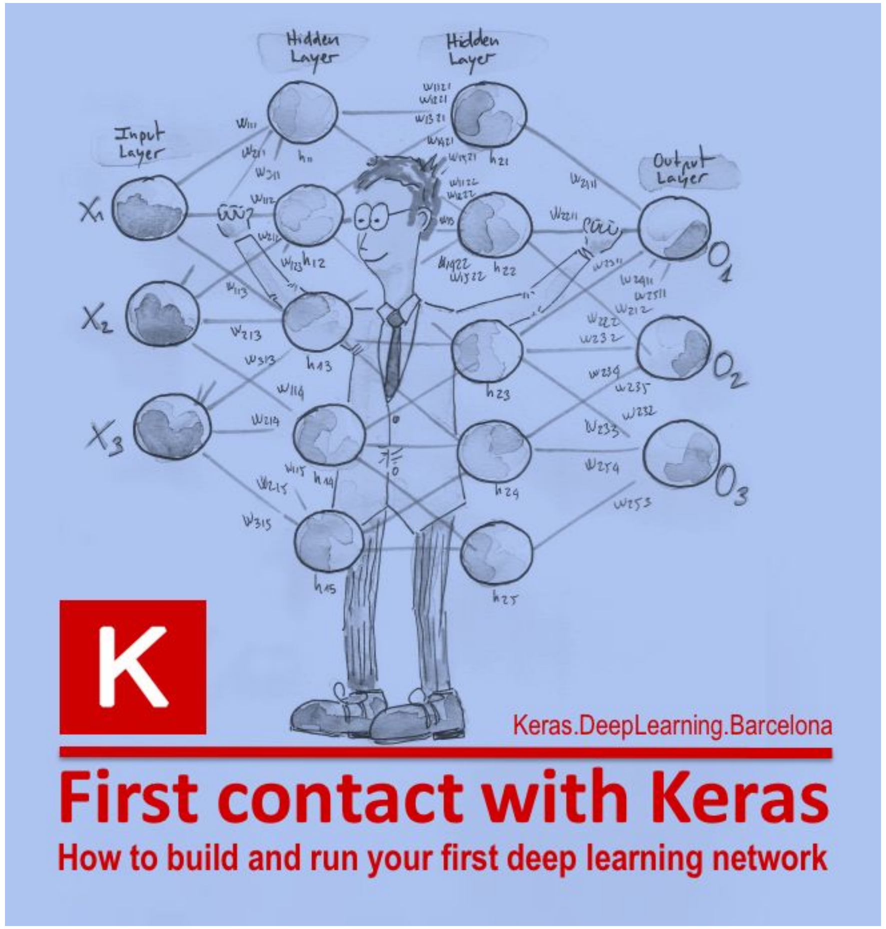
- was developed and maintained by François Chollet

- https://keras.io

- Python library  on top of *TensorFlow, Theano or CNTK*.

# Quick start:

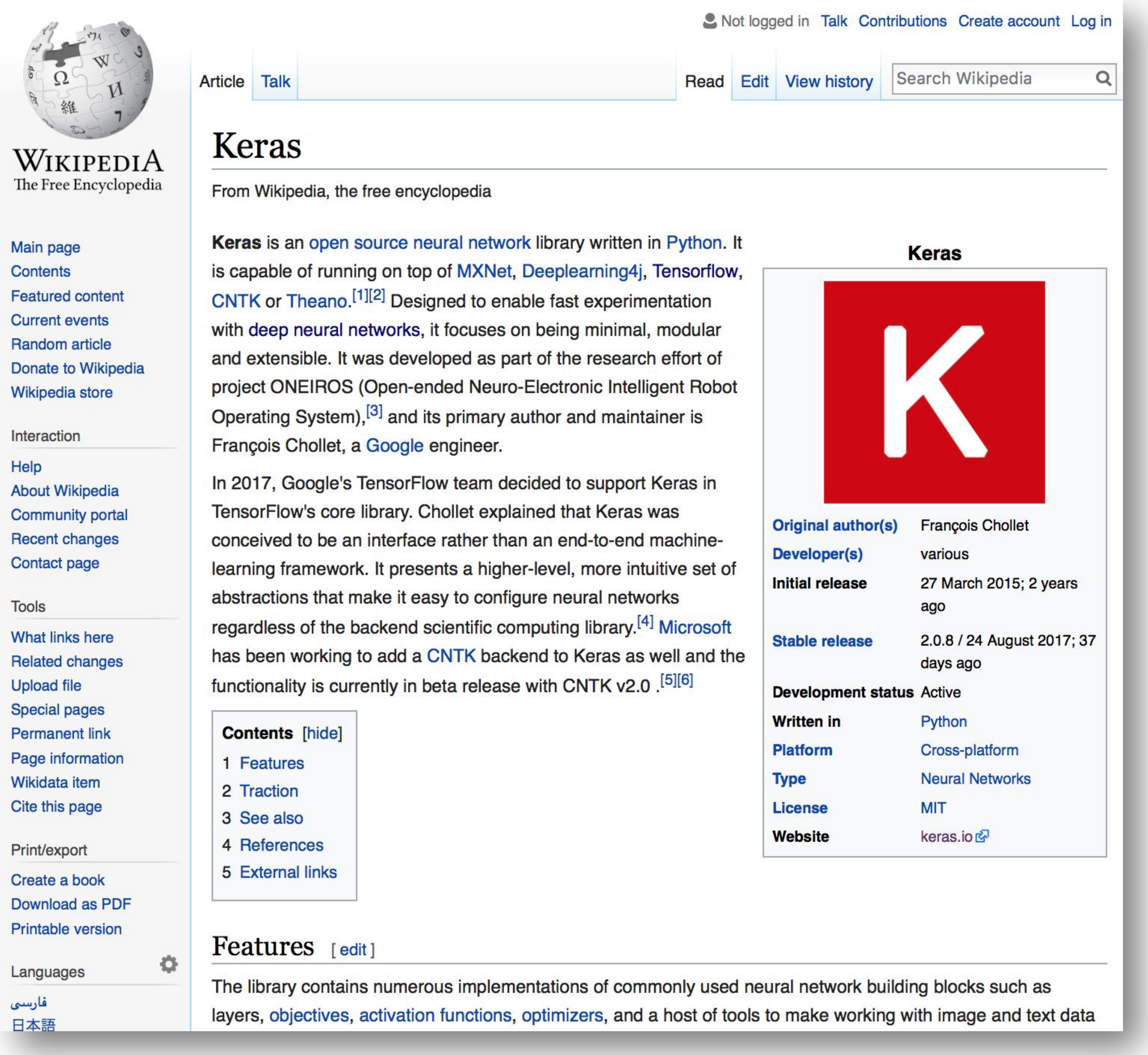
http://keras.deeplearning.barcelona



Keras.DeepLearning.Barcelona

First contact with Keras

How to build and run your first deep learning network



# Keras

From Wikipedia, the free encyclopedia

**Keras** is an open source neural network library written in Python. It is capable of running on top of MXNet, Deeplearning4j, Tensorflow, CNTK or Theano.[1][2] Designed to enable fast experimentation with deep neural networks, it focuses on being minimal, modular and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System),[3] and its primary author and maintainer is François Chollet, a Google engineer.

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than an end-to-end machine-learning framework. It presents a higher-level, more intuitive set of abstractions that make it easy to configure neural networks regardless of the backend scientific computing library.[4] Microsoft has been working to add a CNTK backend to Keras as well and the functionality is currently in beta release with CNTK v2.0 .[5][6]

**Keras**

| | |
|---|---|
| **Original author(s)** | François Chollet |
| **Developer(s)** | various |
| **Initial release** | 27 March 2015; 2 years ago |
| **Stable release** | 2.0.8 / 24 August 2017; 37 days ago |
| **Development status** | Active |
| **Written in** | Python |
| **Platform** | Cross-platform |
| **Type** | Neural Networks |
| **License** | MIT |
| **Website** | keras.io |

## Features [ edit ]

The library contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data

- Good for beginners
- Austerity and simplicity
- Keras model are a combination of discrete elements

# Model: core data structure

- keras.models.Sequential class is a wrapper for the neural network model:

```
from keras.models import Sequential
model = Sequential()
```

# Layers in Keras

- Models in Keras are defined as a sequence of layers.
- There are
  - fully connected layers,
  - max pool layers,
  - activation layers,
  - etc.

- You can add a layer to the model using the model’s **`add()`** function.

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation

#Create the Sequential model
model = Sequential()

#1st Layer - Add an input layer of 32 nodes
model.add(Dense, input_dim=32)

#2nd Layer - Add a fully connected layer of 128 nodes
model.add(Dense(units=128))

#3rd Layer - Add a softmax activation layer
model.add(Activation('softmax'))

#4th Layer - Add a fully connected
layer     model.add(Dense(10))

#5th Layer - Add a Sigmoid activation
layer     model.add(Activation('sigmoid'))
```

# Layer shape

- 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.
  - Example:

    The first layer sets the input dimension to 32.

    The second layer takes in the output of the first layer and sets the output to 128.

    . . .

    We can see that the output has dimension 10.

# Learning & Training

- Configure the learning process: `compile()`

  Compiling the model uses the efficient numerical libraries of the backend used.

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

- Training the Model: `fit()`

```
model.fit(x_train,y_train, epochs=1000, batch_size=32)
```

# Evaluate & Predictions

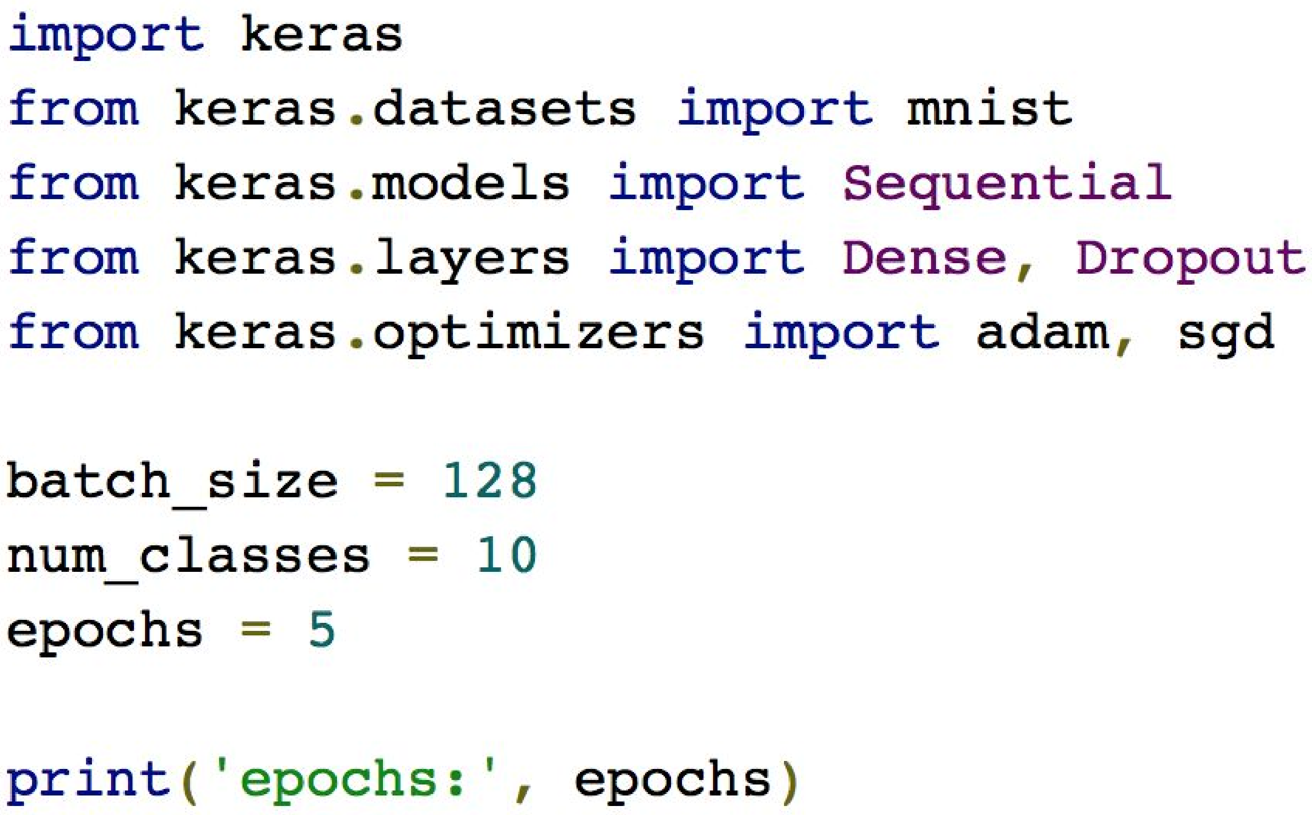
- Evaluate: `evaluate()`

```
loss_and_metrics = model.evaluate(x_test, y_test)
```

- Generate predictions: `predict()`

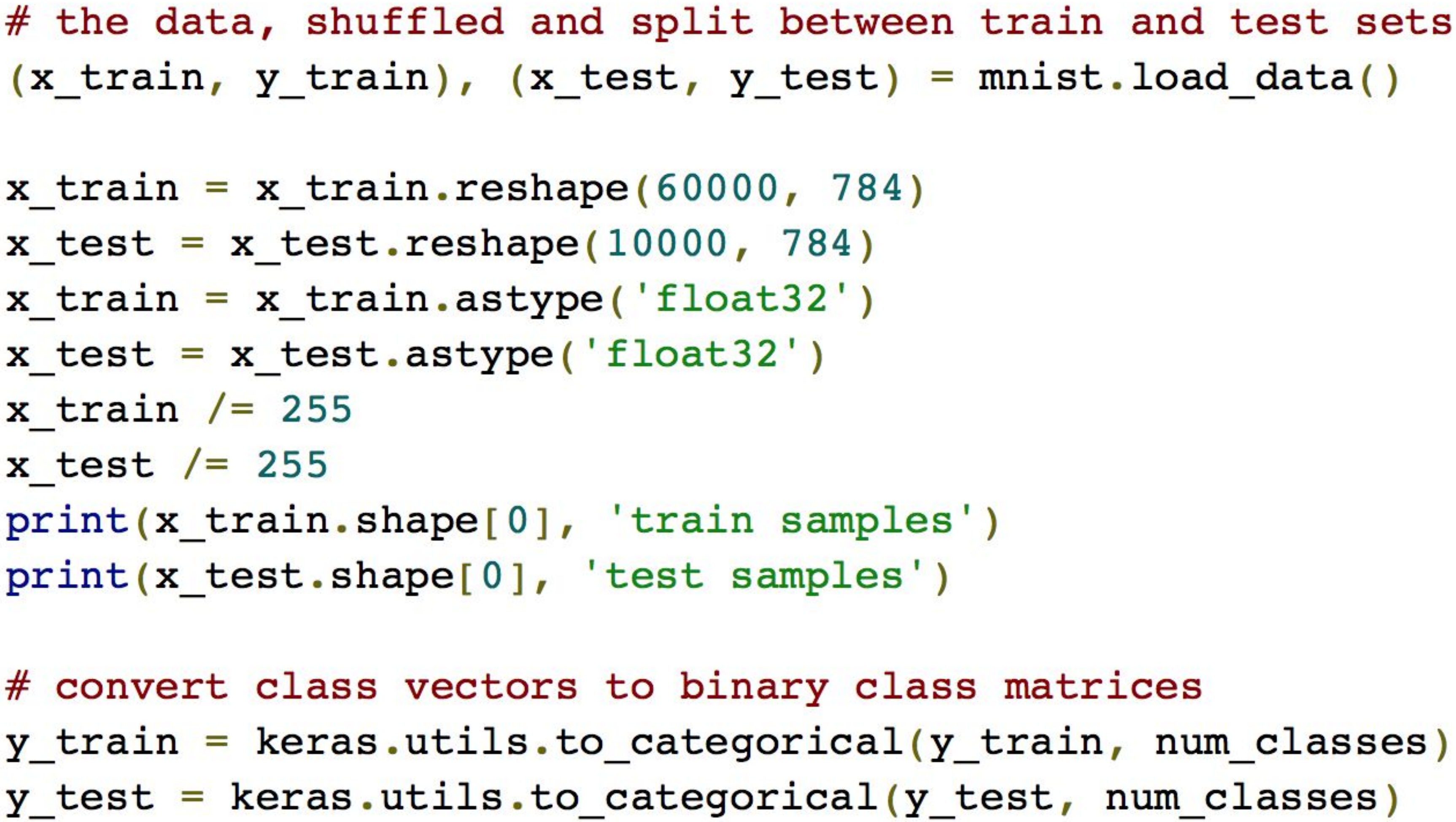
```
classes = model.predict(x_test, batch_size=128)
```

# Example: MNIST



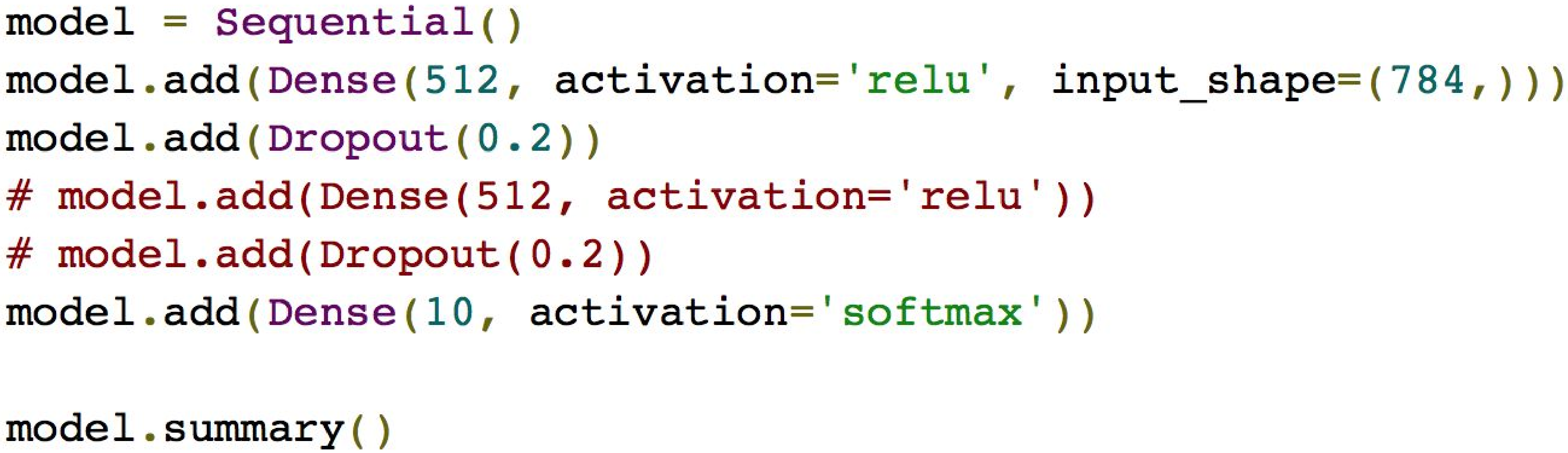
```python
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import adam, sgd

batch_size = 128
num_classes = 10
epochs = 5

print('epochs:', epochs)
```

# Example: MNIST



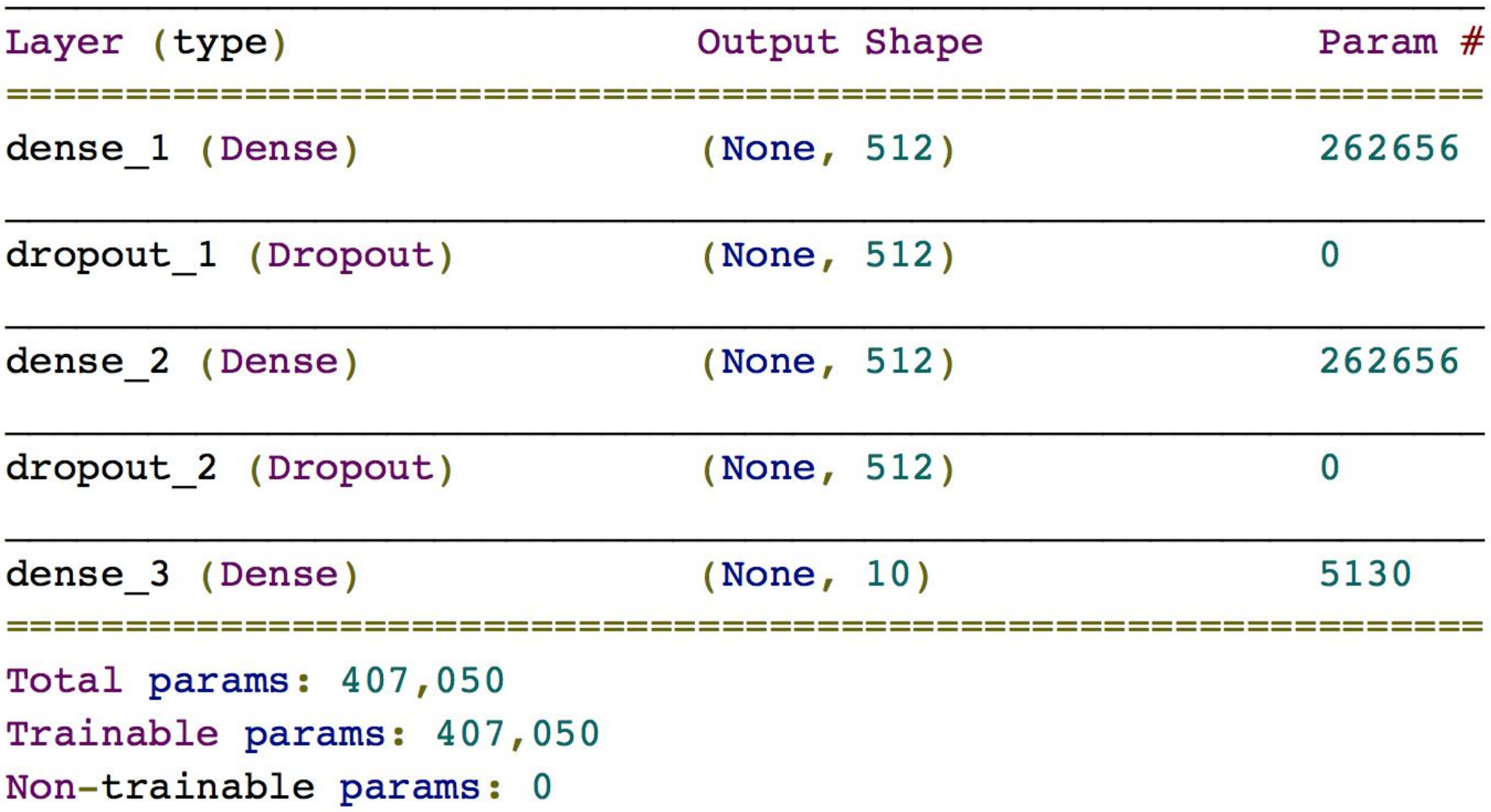
```
# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

# Example: MNIST



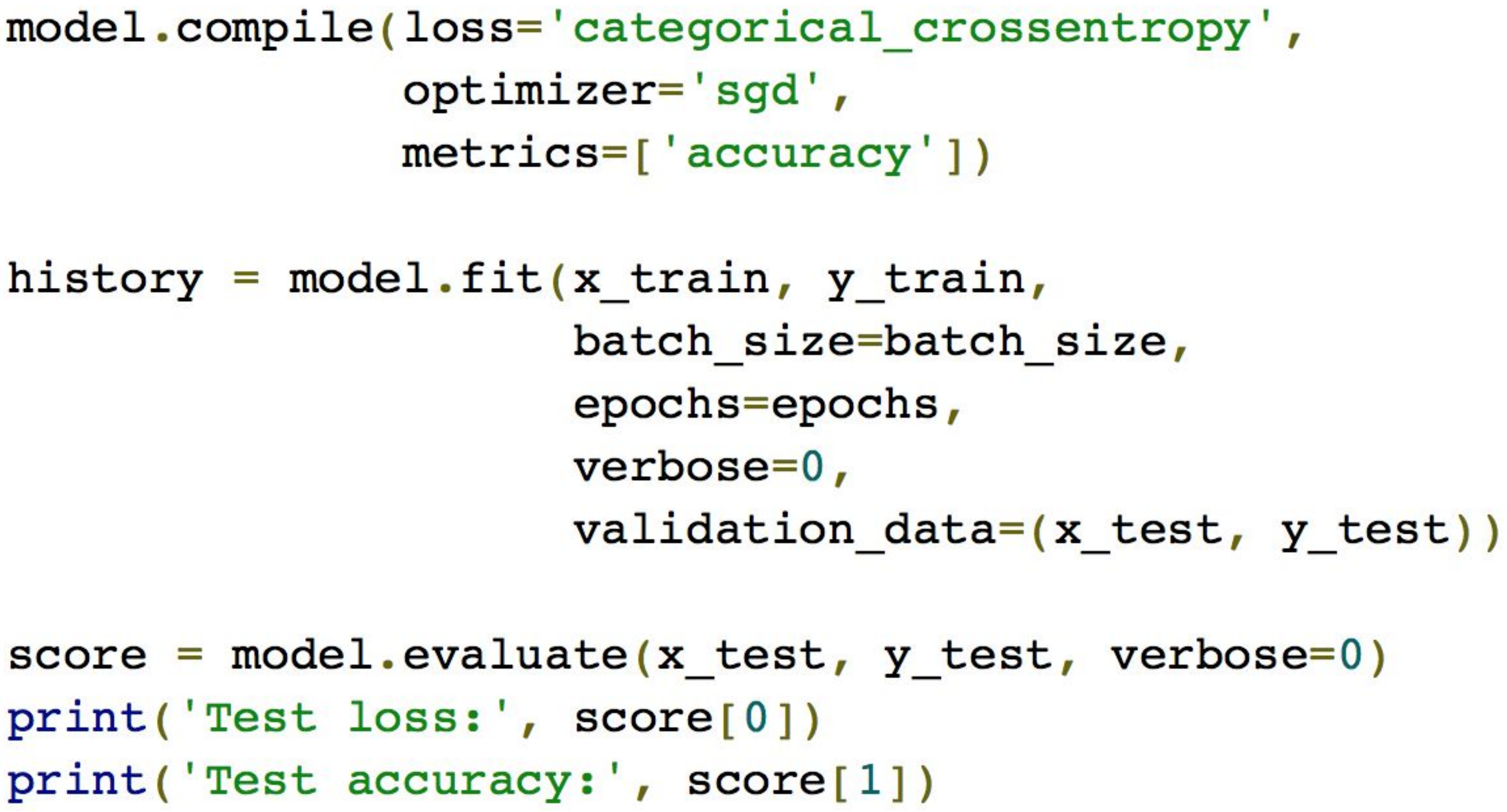
```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
# model.add(Dense(512, activation='relu'))
# model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()
```

# model.summary() output



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
```

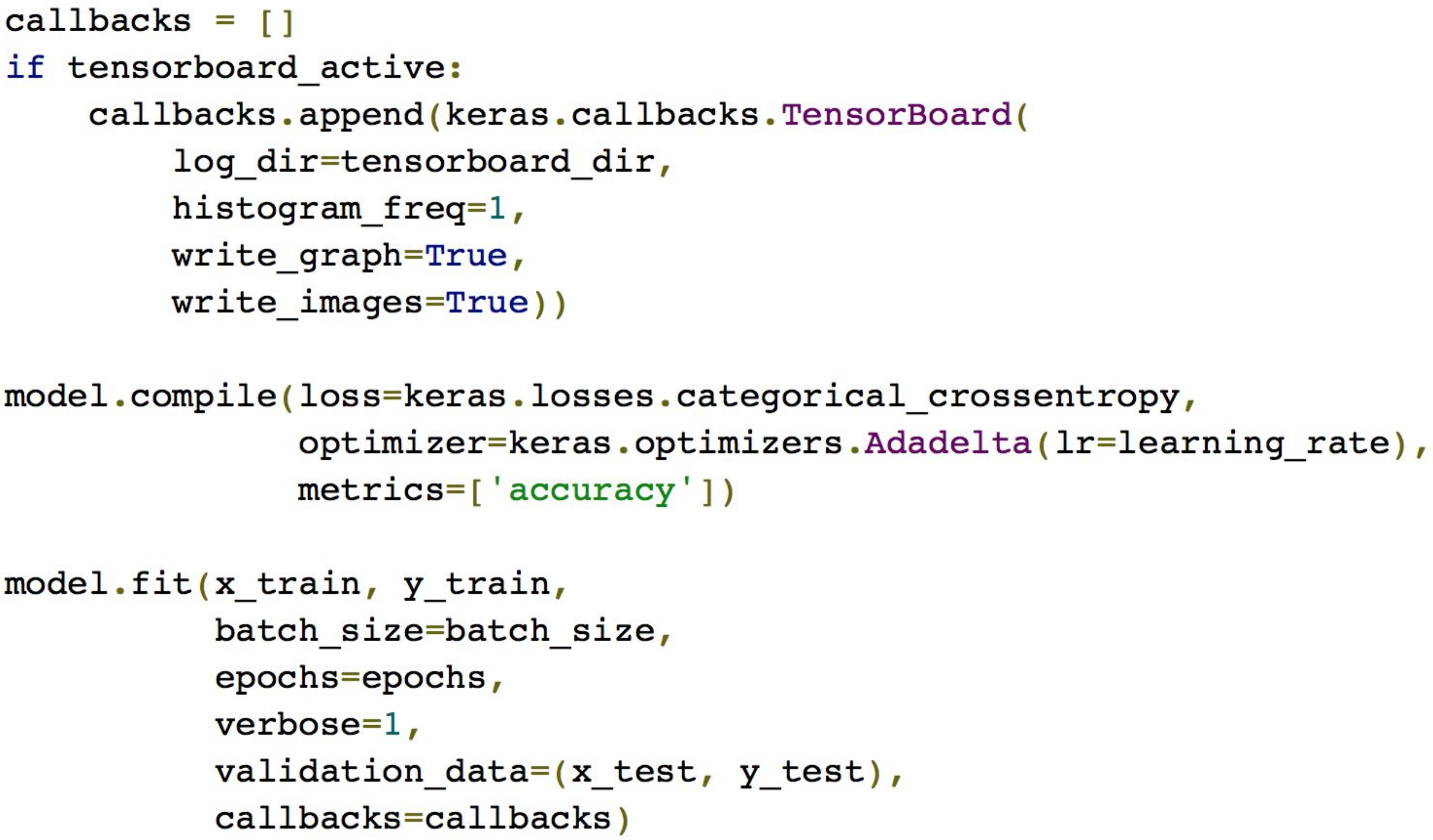
# Example: MNIST



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

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=0,
                    validation_data=(x_test, y_test))

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

# Statistics visualization with TensorBoard

- pass a list of callbacks to the .fit() (training process)



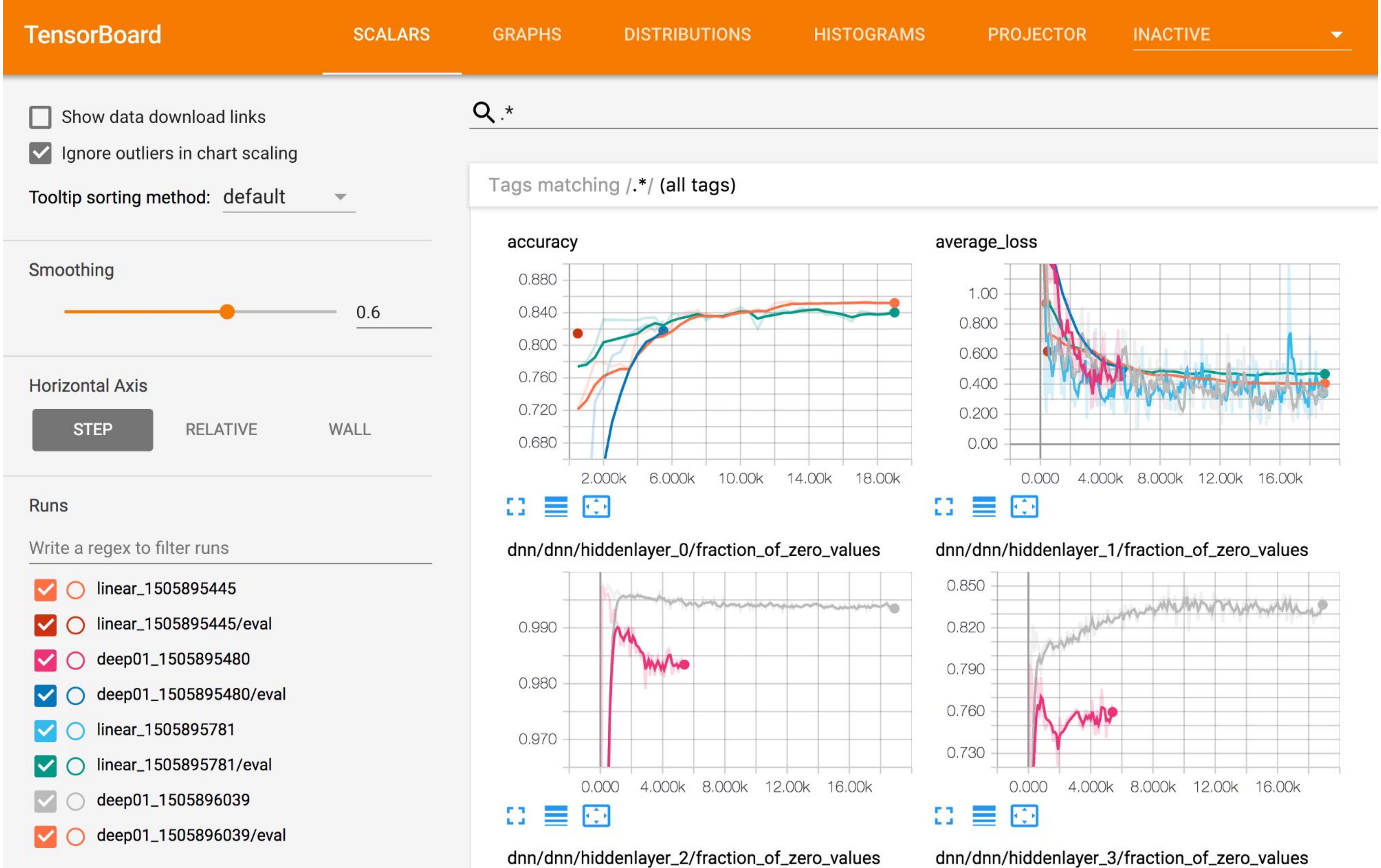
```
callbacks = []
if tensorboard_active:
    callbacks.append(keras.callbacks.TensorBoard(
        log_dir=tensorboard_dir,
        histogram_freq=1,
        write_graph=True,
        write_images=True))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(lr=learning_rate),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test),
          callbacks=callbacks)
```

# Statistics visualization with TensorBoard (cont.)

- run (assuming TensorFlow installed):

```
tensorboard --logdir=/tensorboard_dir
```

- go to `http://localhost:6006` (**Google Chrome recommended**)
  - you can visualize the graph
  - measure performance metrics
  - …



☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

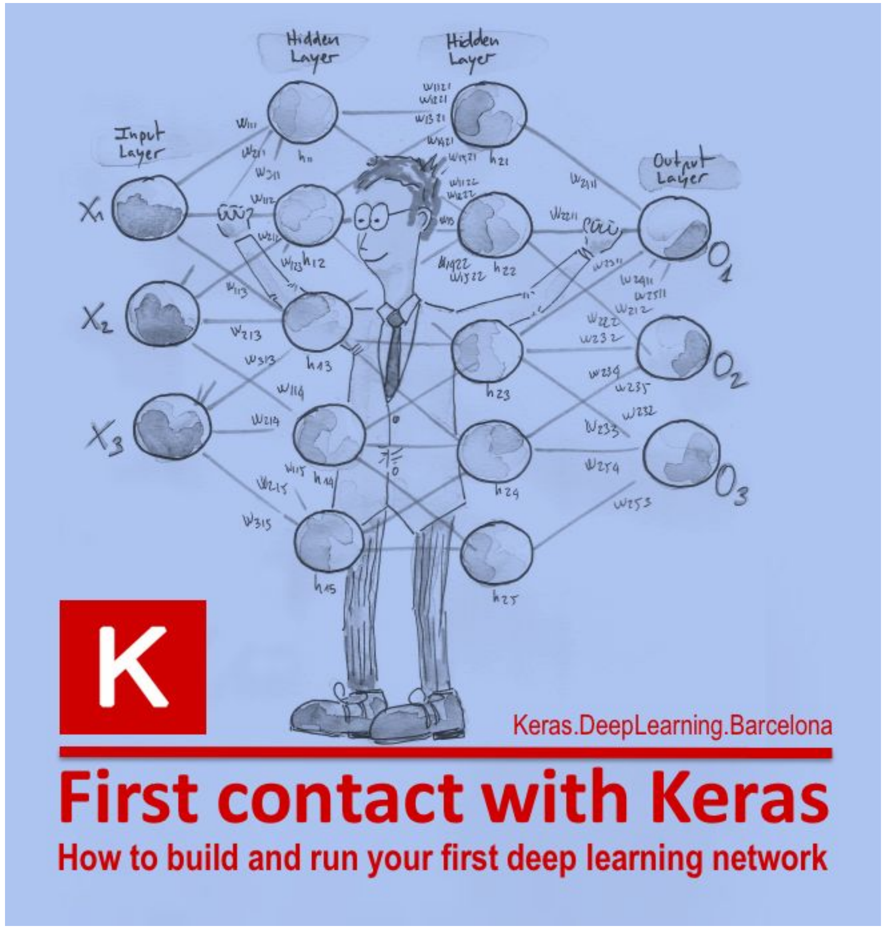
☑ ○ linear_1505895445
☑ ○ linear_1505895445/eval
☑ ○ deep01_1505895480
☑ ○ deep01_1505895480/eval
☑ ○ linear_1505895781
☑ ○ linear_1505895781/eval
☑ ○ deep01_1505896039
☑ ○ deep01_1505896039/eval

.*

Tags matching /.*/ (all tags)

# More information

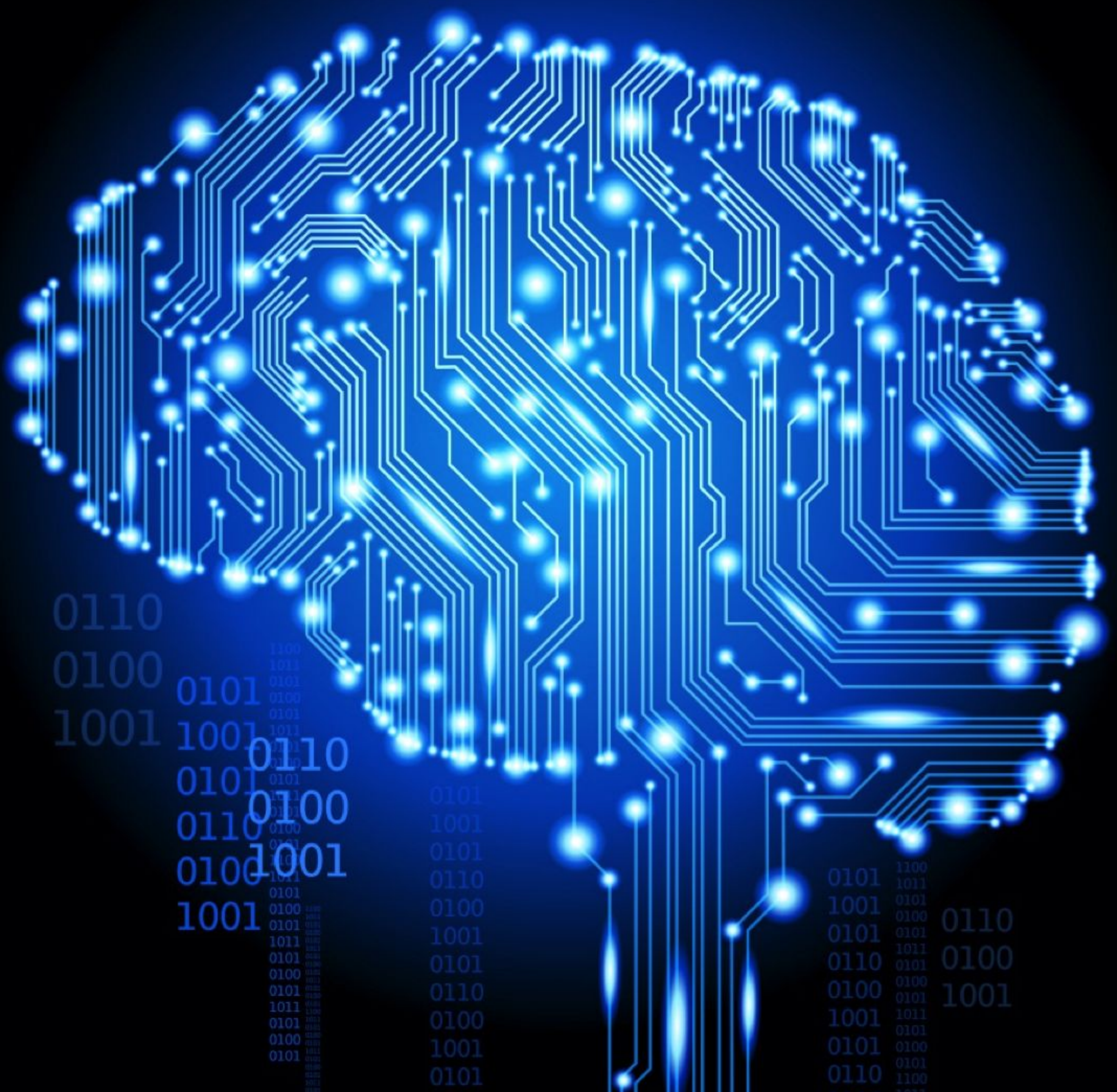
http://keras.deeplearning.barcelona



Keras.DeepLearning.Barcelona

First contact with Keras

How to build and run your first deep learning network



JORDI **TORRES** | FRANCESC **SASTRE**