Use mnist_dataset in local folder

In [1]:
```
print("Jefferson, 1234567")
```

```
Jefferson, 1234567
```

In [2]:
```
import tensorflow as tf
tf.config.list_physical_devices('GPU')
tf.__version__
```

Out[2]: '2.1.0'

In [3]:
```
#import keras
import tensorflow.keras as keras
keras.__version__
```

Out[3]: '2.2.4-tf'

A first look at a neural network

This notebook contains the code samples found in Chapter 2, Section 1 of Deep Learning with Python (https://www.manning.com/books/deep-learning-with-python?a_aid=keras&a_bid=76564dff). Note that the original text features far more content, in particular further explanations and figures: in this notebook, you will only find source code and related comments.

We will now take a look at a first concrete example of a neural network, which makes use of the Python library Keras to learn to classify hand-written digits. Unless you already have experience with Keras or similar libraries, you will not understand everything about this first example right away. You probably haven't even installed Keras yet. Don't worry, that is perfectly fine. In the next chapter, we will review each element in our example and explain them in detail. So don't worry if some steps seem arbitrary or look like magic to you! We've got to start somewhere.

The problem we are trying to solve here is to classify grayscale images of handwritten digits (28 pixels by 28 pixels), into their 10 categories (0 to 9). The dataset we will use is the MNIST dataset, a classic dataset in the machine learning community, which has been around for almost as long as the field itself and has been very intensively studied. It's a set of 60,000 training images, plus 10,000 test images, assembled by the National Institute of Standards and Technology (the NIST in MNIST) in the 1980s. You can think of "solving" MNIST as the "Hello World" of deep learning -- it's what you do to verify that your algorithms are working as expected. As you become a machine learning practitioner, you will see MNIST come up over and over again, in scientific papers, blog posts, and so on.

The MNIST dataset comes pre-loaded in Keras, in the form of a set of four Numpy arrays:

In [4]:

```
import numpy as np
import os
import gzip

def load_data(data_folder):

    files = [
        'train-labels-idx1-ubyte.gz', 'train-images-idx3-ubyte.gz',
        't10k-labels-idx1-ubyte.gz',  't10k-images-idx3-ubyte.gz'
    ]

    paths = []
    for fname in files:
        paths.append(os.path.join(data_folder,fname))

    print("Opening y_train path ", paths[0])
    with gzip.open(paths[0], 'rb') as lbpath:
        y_train = np.frombuffer(lbpath.read(), np.uint8, offset=8)

    print("Opening x_train path ", paths[1])
    with gzip.open(paths[1], 'rb') as imgpath:
        x_train = np.frombuffer(
            imgpath.read(), np.uint8, offset=16).reshape(len(y_train),
            28, 28)

    print("Opening y_test path ", paths[2])
    with gzip.open(paths[2], 'rb') as lbpath:
        y_test = np.frombuffer(lbpath.read(), np.uint8, offset=8)

    print("Opening x_test path ", paths[3])
    with gzip.open(paths[3], 'rb') as imgpath:
        x_test = np.frombuffer(
            imgpath.read(), np.uint8, offset=16).reshape(len(y_test),
            28, 28)

    return (x_train, y_train), (x_test, y_test)
```

In [5]:

```
#from keras.datasets import mnist
# (train_images, train_labels), (test_images, test_labels) = mnist.load_

local_folder = "mnist_dataset"
(train_images, train_labels), (test_images, test_labels) \
    = load_data(local_folder)
```

```
Opening y_train path  mnist_dataset\train-labels-idx1-ubyte.gz
Opening x_train path  mnist_dataset\train-images-idx3-ubyte.gz
Opening y_test path  mnist_dataset\t10k-labels-idx1-ubyte.gz
Opening x_test path  mnist_dataset\t10k-images-idx3-ubyte.gz
```

`train_images` and `train_labels` form the "training set", the data that the model will learn from. The model will then be tested on the "test set", `test_images` and `test_labels` . Our images are encoded as Numpy arrays, and the labels are simply an array of digits, ranging from 0 to 9. There is a one-to-one correspondence between the images and the labels.