Our First Code

TIN0255/09P9M80/D82 - Aprendizagem Profunda

2025.1 - Prof. Pedro Moura

Our First Code

- Python is comfortably the most popular software language in the **data science community**.
  - It's the language we adopt for our example code throughout the semester.
- Python's prevalence extends across the composition of **stand-alone scripts** through to the deployment of **machine learning models into production systems**.
- However, one shall be concerned about its performance issues in real case scenarios.
  - These are not going to be an issue for us in our classes.

Our First Code

- There are step-by-step installation instructions available in the GitHub repository associated with the book “Deep Learning Illustrated”
  - Through this link
  - These instructions suggest running Jupyter from within a Docker container.
- You can view the notebooks in the GitHub repo as well, instead of running them on your own machine.
- Jupyter is a common option today for writing and sharing scripts, particularly during exploratory phases in which a data scientist is experimenting with preprocessing, visualizing, and modeling her data.

A Shallow Network

- In the following, we will:
- Detail a revered dataset of handwritten digits
- Load these data into a Jupyter notebook

- Use Python to prepare the data for modeling
- Write a few lines of code in Keras to construct an artificial neural network (in TensorFlow, behind the scenes) that predicts what digit a given handwritten sample represents.

The MNIST Handwritten Digits

- In one of the last classes, we saw that one of the advantages Yann LeCun and his colleagues had over previous deep learning practitioners was a **superior dataset for training the model**.

- It was in this context that the dataset MNIST emerged.
  - This dataset contains handwritten digits.

- It is ubiquitous across deep learning tutorials (for good reason).
  - It is used in CNN, GAN and etc.

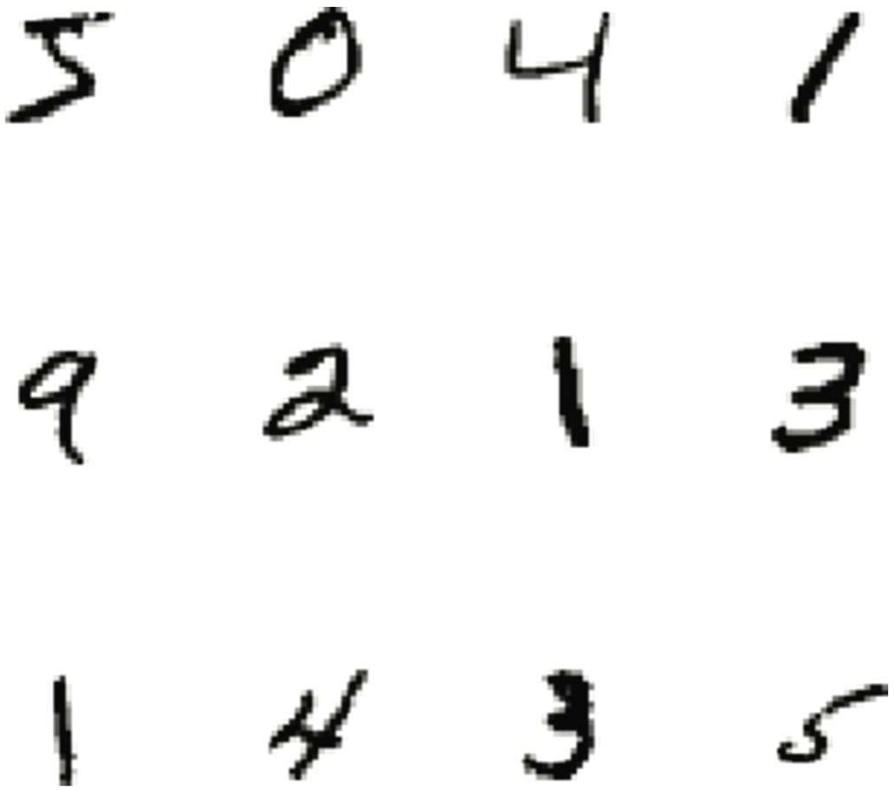
The MNIST Handwritten Digits

- It is small enough (by modern standards) that it can be modeled rapidly, even on a laptop computer processor.

- The MNIST digits are handy because they occupy a sweet spot with respect to how challenging they are to classify:
  - handwriting samples are sufficiently diverse and contain complex enough details → **not easy** for a machine-learning algorithm to identify with high accuracy.
  - By no means do they pose an insurmountable problem.

- A **well-designed deep-learning** model can **nearly faultlessly classify** the handwriting as the appropriate digit.

Biological Neuroanatomy



A sample of a dozen images from the MNIST dataset. Each image contains a single digit handwritten by either a high school student or a U.S. census worker.

The MNIST Handwritten Digits

- This dataset was curated by LeCun, Corinna Cortes and the Microsoft-AI-researcher-turned-musician Chris Burges in the 1990s.
- It consists of 60,000 handwritten digits for training an algorithm, and 10,000 more for validating the algorithm's performance on previously unseen data.
- The data are a subset of a larger body of handwriting samples collected from **high school students** and **census workers** by the U.S. National Institute of Standards and Technology (NIST).

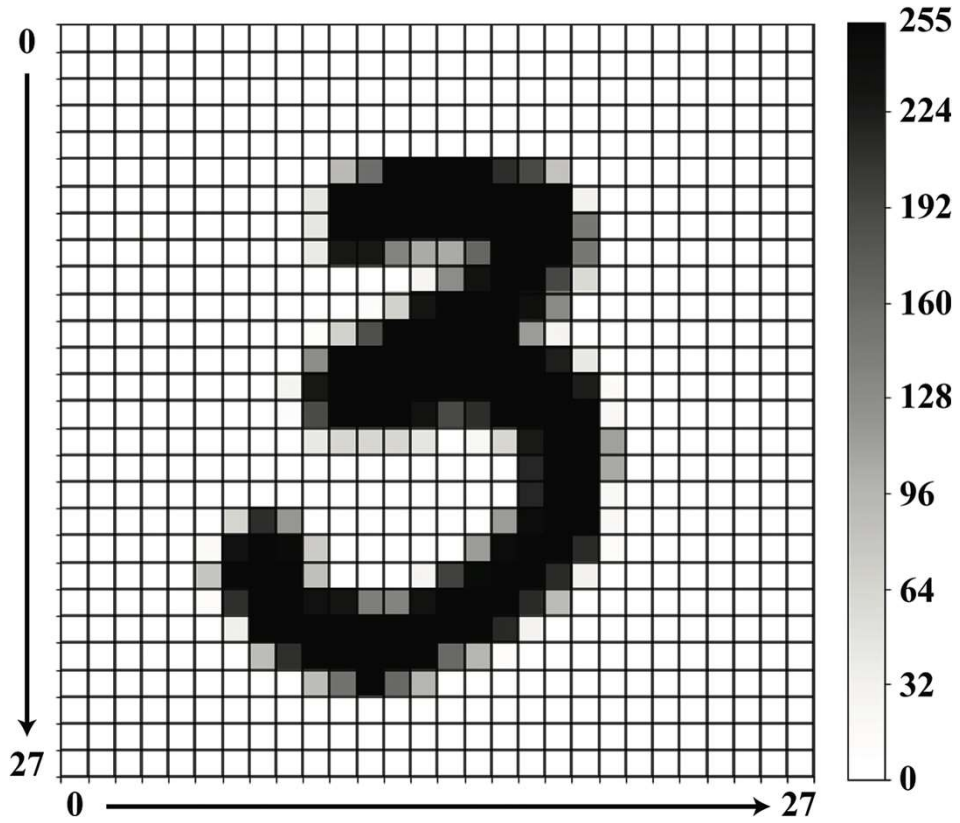
The MNIST Handwritten Digits

The Danish computer scientist Corinna Cortes is head of research at Google's New York office. Cortes curated (with Chris Burges and Yann LeCun) the widely used MNIST dataset.

The MNIST Handwritten Digits

- Every MNIST digit is a 28x28-pixel image.
- Each pixel is 8-bit (1 byte) → pixel darkness can vary from 0 (white) to 255 (black).
  - Range of integers represents gradually darker shades of gray.

The MNIST Handwritten Digits

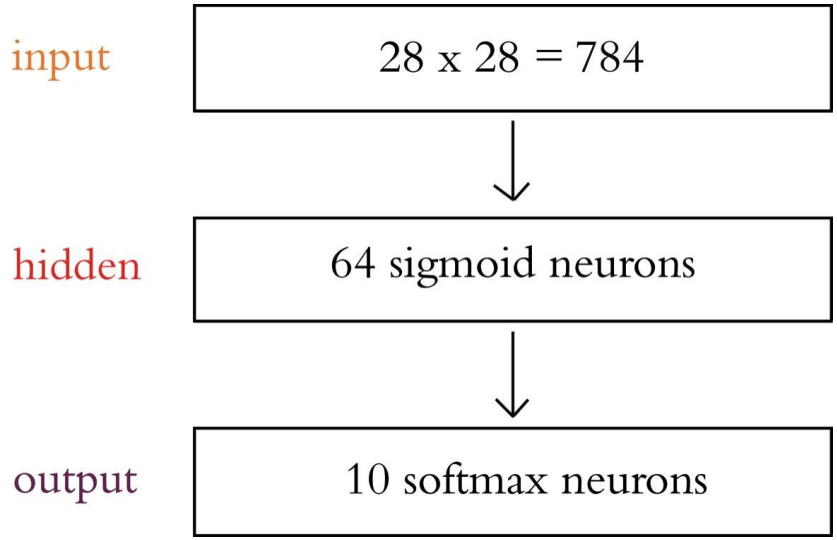


Each handwritten MNIST digit is stored as a 28×28-pixel grayscale image. See the Jupyter notebook titled *MNIST Digit Pixel by Pixel* for the code used to create this figure.

A Schematic Diagram of the Network

- We are going to use the *Shallow Net in Keras* Jupyter notebook.
- We are going to create an **Artificial Neural Network** to predict what digit a given handwritten MNIST image represents.
- This artificial neural network has one **hidden layer** of artificial neurons, for a total of **three layers**.
- With so few layers this ANN would not generally be considered a deep learning architecture → **shallow network**.

A Schematic Diagram of the Network



A rough schematic of the shallow artificial-neural-network architecture we're whipping up in this chapter.

A Schematic Diagram of the Network

- The first layer of the network is reserved for inputting our MNIST digits.
- They are 28x28 pixel images → each one has a total of 784 values.
- After we load in images, we'll **flatten** them from **two-dimensional** 28x28 shape to a **one dimensional** array of 784 elements.

A Schematic Diagram of the Network

- Does collapsing the images from two dimensions to one cause us to **lose meaningful structure** of the handwritten digits? Yes!
- However, working with one-dimensional data means we can use relatively unsophisticated neural network models.
- This is appropriate at this early stage in our classes.
- Later we'll be in a position to appreciate more-complex models that can handle multidimensional inputs.

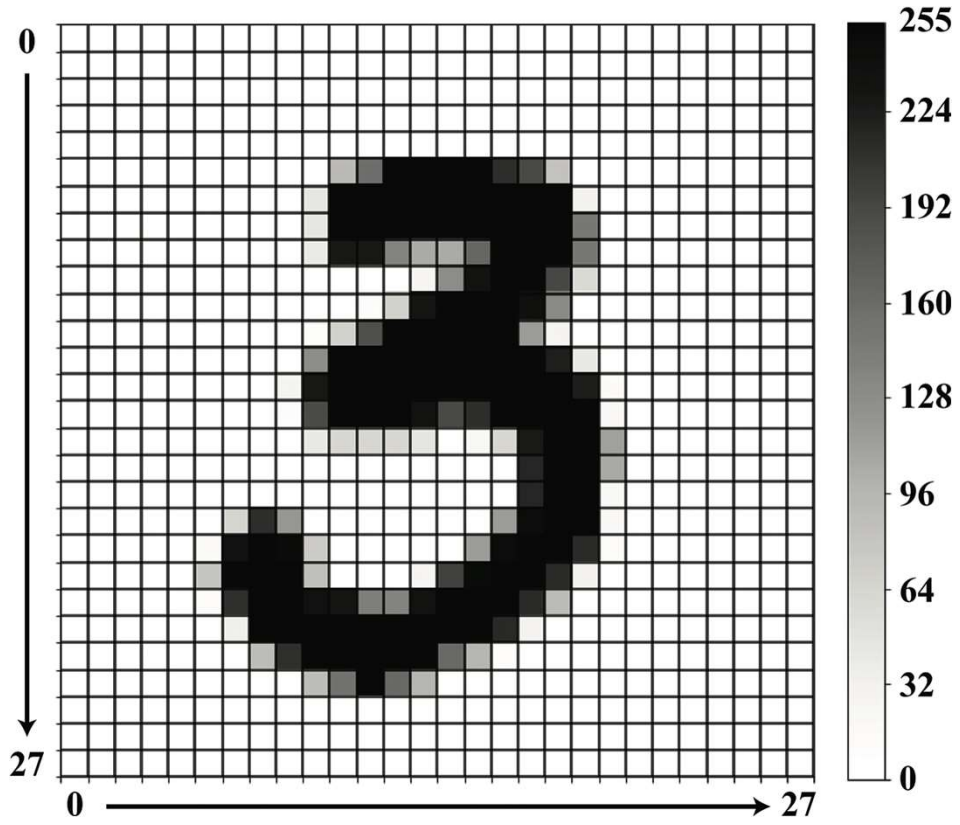
A Schematic Diagram of the Network

- These pixel-data inputs will be passed through a single **hidden layer** of 64 neurons.
  - **Hidden layer** → they are not exposed and impacted by data only indirectly via the input or ouput layer.
- This hidden layer has 64 neurons using the **sigmoid** activation function.
- Neurons in this hidden layer are responsible for **learning representations** of the input data so that the network can predict what digit a given image represents.
- This constitutes a **dense layer** (algo called **fully-connected layer**).

A Schematic Diagram of the Network

- The information that is produced by the hidden layer will be passed to 10 neurons in the output layer.
- We are going to learn in detail how the **softmax layer** of neurons works later.
  - In essence: we have 10 neurons because we have 10 categories (classes) of digit to classify.
- Each of these 10 neurons output a probability: one for each of the 10 possible digits that a given MNIST image could represent.

A Schematic Diagram of the Network



- Suppose we feed this image to a fairly well-trained network.

A Schematic Diagram of the Network

- This fairly well-trained network might output that there is a 0.92 probability that the image is of a **three**, a 0.06 probability that it's a **two**, a 0.02 probability that it's an **eight**, and a probability of 0 for the other seven digits.

- We can simply ouput the value associated with the highest probability → **most used strategy**.

- Or we can make use of the fact that the softmax layer outputs a **probability distribution** to draw a number according to this distribution → this depends on the application.

Training a Deep Learning Model

- Critical aspects are:
  - The `fit()` method enables us to train our artificial neural network with training images `X_train` as inputs and `y_train` as output.
  - As the network trains, the `fit()` method also provides us with the option to evaluate the performance of our network: `validation_data` argument.
  - With deep learning, it is commonplace to train our model on the same data multiple times → one pass through all of our training data is called one **epoch** (parameter `epochs`).
  - By setting `verbose` to `1`, the `model.fit()` method will provide us with plenty of feedback as we train.

Training a Deep Learning Model

- We will focus on *Validation accuracy* that is the output following each epoch of training.

- After the first epoch of training: `val_acc` equals 0.1010 (one should note that **neural networks are stochastic**).
  - This means that 10.1% of the images from the validation dataset were correctly classified by ou shallow architecture.
  - There are 10 classes of handwritten digits → that's what we would expect by a **random classifier**.

- As the network continues training, the results improve:
  - After 10 epochs, it is correctly classifying 36.5% of validation images.
  - After 200 epochs, its improvement appears to be plateauing as it approaches 86% → **That's pretty good for a shallow architecture!**

Referências

1. Capítulo 5 do livro *“Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence”* de Jon Krohn, Grant Beyleveld e Aglaé Bassens.