Deep Learning with Python Ch. 2

MNIST Example

- MNIST is a dataset of grayscale images of handwritten digits (28 x 28 pixels) classified according to their 10 categories (0 through 9).
- It was gathered in the 1980s by the National Institute of Standards and Technology. More or less the “hello world” of deep learning.
- We will start out by looking at a *keras* implementation using a *Jupyter Notebook*, so we can run the code. We will sort of explain the code now, then go into more detail later.

Data representations for neural networks

- The MNIST example started with data stored in multidimensional Numpy arrays, also called *tensors*. Pretty much all current machine-learning systems use tensors as their basic data structure.
- What is a tensor? Essentially it is a container for numbers. A 2D matrix is an example of a tensor.
- Scalars (0D tensors): a tensor containing only one number
- Vectors (1D tensors): an array of numbers. A 1D tensor has exactly one axis. A vector with 5 elements is called 5-dimensional vector, but it is a 1D tensor.
- Matrices (2D tensors): an array of vectors. A matrix has two axes (often usefully called *rows* and *columns*.

Tensor attributes

- If you pack matrices into a new array you get a 3D tensor. Could be visualized as a cube or as an array of matrices. Of course, if you put these 3D tensors into an array, you get a 4D tensor, etc. . .
- The three key attributes that define a tensor are:
  - **numbers of axes** (rank): called *ndim* in Python libraries such as Numpy.
  - **shape**: this is a tuple of integers that describes how many dimensions the tensor has along each axis.
  - **data type**: sometimes called dtype, The type of the data contained in the tensor. Typically one of *float32, uint8, float64, . . .* Maybe this will occasionally be *char*, but never string, because it has to be of a fixed, pre-allocatable size.

Tensors in the MNIST example

- At the beginning of the example we loaded the training and testing images and labels. *train-images* is the training data and it has an ndim of 3. The shape of it is $(60000, 28, 28)$ because it has 60,000 images, each of size 28x28. Its dtype attribute is *uint8*. Each matrix is a grayscale image with data values between 0 and 255.

More on tensors

- You probably recall slices from Python. Well, they can be used with tensors as well. So train-images[10:100] would give you 90 images from the training data, so its shape would be $(90, 28, 28)$.
- Generally the 0th axis in a tensor will be the *sample axis*. For example, in the MNIST example, each sample is the image of a digit. Deep learning models usually break the data into what are called *batches*. Again Python slices can be used to do this.

Real-world examples of data tensors

- **Vector data**: 2D tensors of shape (samples, features)
- **Timeseries or sequence data**: 3D tensors of shape (samples, timesteps, features)
- **Images**: 4D tensors of shape (samples, height, width, channels)
- **Video**: 5D tensors of shape (samples, frames, height, width, channels)

Vector Data

- This is the most common kind of data. In such datasets a single data point can be encoded as a vector, so a batch of such data is a 2D tensor, i.e., an array of vectors.
- For example:
    - An actuarial dataset of people, where we consider factors such as age, zip code, income and such.
    - A dataset of text documents, where we represent each document by the counts of how many times each word appears in it (out of dictionary of, say, the 20,000 most common words).

Timeseries or sequence data

- Whenever time matters in the data (or the notion of a sequence order) we can use a 3D tensor with an explicit time axis.
- For example:
  - A dataset of stock prices. Every minute (say) we store the current price of the stock, as well as the highest and lowest price in the past minute. Thus every minute is a 3D vector, an entire day of trading is a 2D tensor of shape (390, 3) (there are 390 minutes in a trading day) and 250 days worth of data could be a 3D tensor of shape (250, 390, 3).
  - A dataset of tweets, where we encode each tweet as a sequence of 280 characters out of an alphabet of 128 unique characters. Thus each character is encoded as a binary vector of size 128 (all zeros except for a 1 at the index corresponding to the character). Then each tweet can be encoded as a 2D tensor of shape (280, 128) and a dataset of a million tweets would have the shape (1000000, 280, 128).

Image data

- Images typically have three dimensions height, width, and color depth. Although grayscale images (such as our MNIST data) only have a single color channel, by convention images are usually 3D tensors with a 1 dimensional color channel for grayscale images. A batch of 128 grayscale images of size 256x256 could be stored in a tensor of shape (128, 256, 256, 1), while the same images in color would have the shape (128, 256, 256, 3).

Video data

- Videos are one of the few types of real-world data that require 5D tensors. A video can be understood as a sequence of frames with each frame being a color image. Each frame can be stored in a 3D tensor (height, width, color-depth), a sequence of frames can be stored in a 4D tensor (frames, height, width, color-depth) and thus a batch of different videos can be stored in a 5D tensor of shape (samples, frames, height, width, color-depth). In practice this is unwieldy and formats such as MPEG do significant compression.