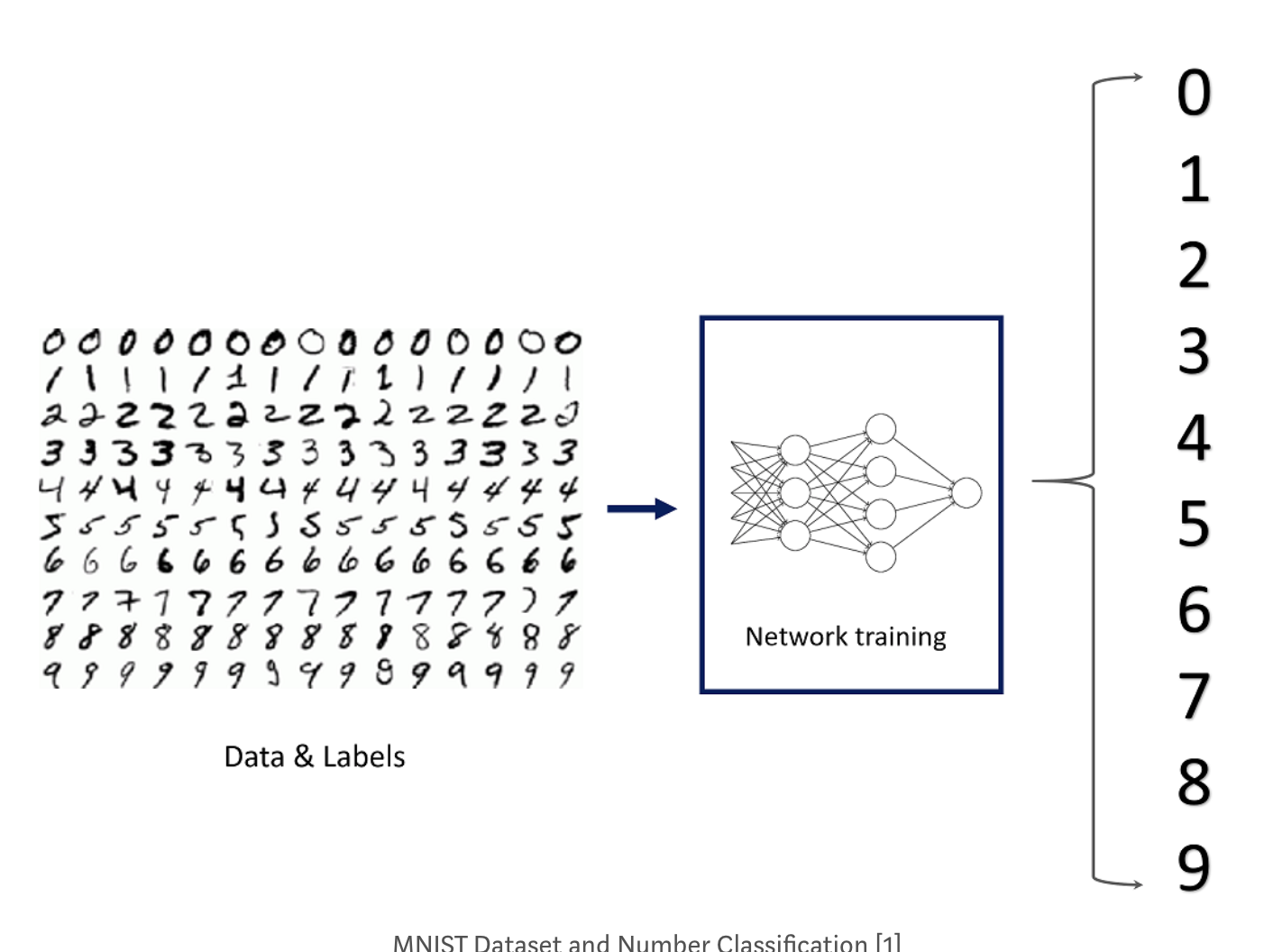
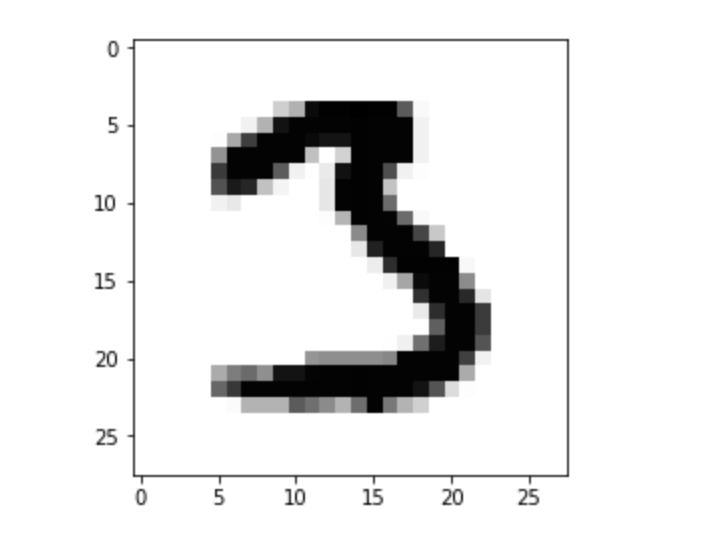
Image Classification using MNIST

Convolutional neural network that you can build for image classification, we can get the most cliche dataset for classification: MNIST dataset, which stands for Modified National Institute of Standards and Technology database. It is a large database of handwritten digits that is commonly used for training various image processing systems.



1. Importing tensorflow and MNIST data set under the keras API.
2. The MNIST database contains 60,000 training images and 10,000 testing images taken from American Census Bureau employees and American high school students. Therefore, in the second line, I have separated these two groups as train and test and also separated the labels and the images. x\_train and x\_test parts contain greyscale RGB codes (from 0 to 255) while y\_train and y\_test parts contains labels from 0 to 9 which represents which number they actually are. To visualize these numbers, we can get help from matplotlib.
3. The grey scale visualization of the code is as below



1. Shape attribute of the numpy array is used to know the shape of the data set to channel it to convolution network.
2. We get (60000, 28, 28) in which 60000 represents the number of images in the train dataset and (28, 28) represents the size of the image: 28 x 28 pixel.
3. Next, we have to reshape and normalize the image for that we need 4-dims numpy arrays. We must normalize the data as needed by the network which can be done by dividing the RGB codes to 255.
4. Building the convolutional network- Import the Sequential Model from Keras and add Conv2D, MaxPooling, Flatten, Dropout, and Dense layers.

In addition, Dropout layers fight with the overfitting by disregarding some of the neurons while training while Flatten layers flatten 2D arrays to 1D array before building the fully connected layers.

1. Compiling the fitting module- use an optimizer with a given loss function which uses a metric. Then, we can fit the model by using our train data.  Adam optimizer is used for that. 98-99% of accuracy is achieved in 10 epochs.
2. Evaluating the model – 98% of accuracy is achieved.
3. Later individual predictions are made when we have trained the model



Refernces-

[1] <http://cs231n.github.io/convolutional-networks/>

[2] MathWorks, Introducing Deep Learning with MATLAB, <https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/80879v00_Deep_Learning_ebook.pdf>

[3] <https://www.kaggle.com/datasets>