MNIST Mini Project

Sherif Ahmed Tohamy

First Model:

1- Input layer:
Since MLPs only take as input 1D vectors with dimensions (1, n), they cannot take a raw 2D image matrix with dimensions (x, y). To fit the image in the input layer, we first need to transform our image into one large vector with the dimensions (1, n) that contains all the pixel values in the image. This process is called image flattening
The total number (n) of pixels in MNIST images are 28 × 28 = 784. Then, in order to feed this image to our network, we need to flatten the (28 × 28) matrix into one long vector with dimensions (1, 784)

2- Hidden Layer:
The neural network can have one or more hidden layers (technically, as many as we want). Each layer has one or more neurons. We arbitrarily designed the network to have two hidden layers, each having 512 nodes using the ReLU activation function for each hidden layer. The ReLU function performs best in the hidden layers and for most classification problems where classes are mutually exclusive, softmax is generally a good choice in the output layer. The softmax function gives us the probability that the input image depicts one of the (n) classes.

3- Output layer:
In classification problems, the number of nodes in the output layer should be equal to the number of classes that you are trying to detect. In this problem, we are classifying 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Then we need to add one last Dense layer that contains 10 nodes.

Drawbacks of MLPs for processing images:

1- Flattening a 2D image to a 1D vector input results in losing the spatial features of the image.
2- MLPs are composed of dense layers that are fully connected to each other. Fully connected means every node in one layer is connected to all nodes of the previous layer and all nodes in the next layer. In this scenario, each neuron has parameters (weights) to train per neuron from the previous layer. While this is not a big problem for the MNIST dataset because the images are really small in size (28 × 28), what happens when we try to process larger images? For example, if we have an image with dimensions 1,000 × 1,000, it will yield 1 million parameters for each node in the first hidden layer. So, if the first hidden layer has 1,000 neurons, this will yield 1 billion parameters even in such a small network.
3- CNNs, on the other hand, are locally connected layers, and the nodes are connected to only a small subset of the previous layers' nodes. Locally connected layers use far fewer parameters than densely connected layers

Second Model:

Rules I follow to build the CNN model:

1- The more layers you add, the better (at least theoretically) out network will learn, but this will come at the cost of increasing the computational and memory space complexity, because it increases the number of parameters to optimize.
2- As the input image goes through the network layers, its depth increases, and the dimensions (width, height) shrink, layer by layer. In general, two or three layers of 3 × 3 convolutional layers followed by a 2 × 2 pooling can be a good start for smaller datasets.
3- Add more convolutional and pooling layers until our image has a reasonable size (say, 4 × 4 or 5 × 5), and then add a couple of fully connected layers for classification.
4- We need to set up several hyperparameters (like filter, kernel_size, and padding). Remember that you do not need to reinvent the wheel: instead, look in the literature to see what hyperparameters usually work for others. Choose an architecture that worked well for someone else as a starting point, and then tune these hyperparameters to fit your situation. The next chapter is dedicated to looking at what has worked well for others. So, we build a smaller version of AlexNet Architecture.
5- Cross-entropy is commonly used in classification problems because it quantifies the difference between two probability distributions.
6- We used the ReLU activation function for all the hidden layers. In the last dense layer, we used a softmax activation function with 10 nodes to return an array of 10 probability scores (summing to 1). Each score will be the probability that the current image belongs to our 10 image classes.