8. Deep learning Image Classification

Ishak AYAD , Djahid ABDELMOUMENE et Mai NGUYEN-VERGER
(ishak.ayad@cyu.fr, djahidabdelmoumen@gmail.com, mai.nguyen-verger@cyu.fr)

1 Image classification (Time: 3h)

In this lab exercise, you will learn how to build and train an image classification model using neural networks and deep learning. Image classification is the task of assigning a label or a category to an image based on its content. For example, given an image of a dog, the image classification model should output the label “dog”.

1.1 Dataset

In order to train our classification Neural network, we are going to need some data. The data that we will be using for this exercise is the MNIST dataset, this is a set of 60,000 (+10,000 for testing) images of handwritten digits each with their corresponding digit labels.

We would need to preprocess the data from the MNIST dataset before we can use it. this means flattening the image in the case of the MLP (ie. transforming it from 28x28 to size 784) and transforming the output label from a simple digit to something that can be interpreted better by our network, like One-Hot Encoding, this would transform our single digit label output to a vector of 10 values for each digit.

Another dataset we are going to be using is the CIFAR-10 dataset which contains 50,000 (10,000 for testing) 32x32x3 (the third dimension is the three color channels of the image RGB) images of 10 classes of objects. In the same way we did for the MNIST dataset we would need to pre-process this dataset to be able to train our networks. The same One-Hot encoding for the output would be applied here.

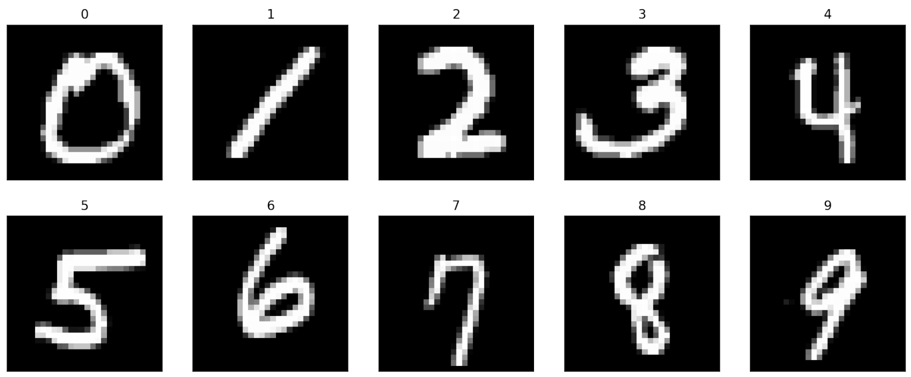

Figure 1: Excerpt from the MNIST dataset

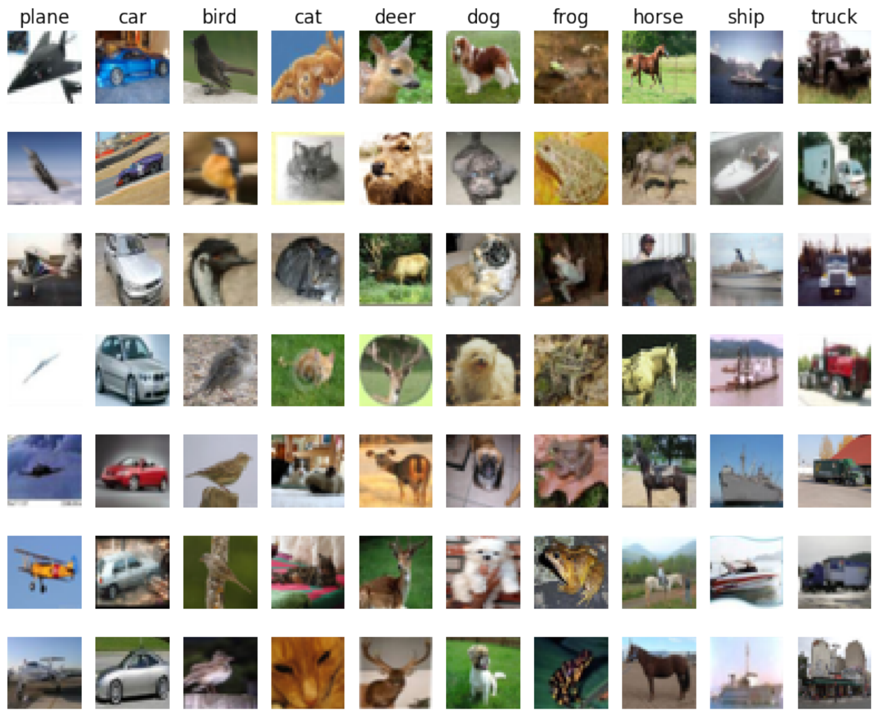

Figure 2: Exerpt from the CIFAR-10 dataset

1.2 Neural networks

A neural network is a model that mimics the structure and function of biological neurons. It consists of nodes that receive inputs and produce outputs based on activation functions. The nodes are connected by weighted links that determine how much each input affects each output.

This can be helpful for our image classification task because it can learn to recognize the patterns and features that distinguish different digits in the images. By adjusting the weights of the links between the nodes, the neural network can improve its accuracy and performance on the task.

So for this we are going to be attempting the digit classification task using two types of neural networks:

1.2.1 Multi-Layer Perceptrons

The simplest form of a multi-layered network is the Multi-Layer Perceptron (MLP). It consists of multiple layers of neurons, the first layer is called the input layer and the last one is the output layer. The rest of the middle layers are called hidden layers. For this classification problem our input layer is going to be a flattened version of the values for all 784 (28x28) pixels of the image or in the case of CIFAR-10 3072 (32x32x3) and the output layer would be size 10 for both, one for each class. for example if the input is an image of a handwritten 0 the output layer should be all zeros except for the first neuron (which represents the 0 digit).

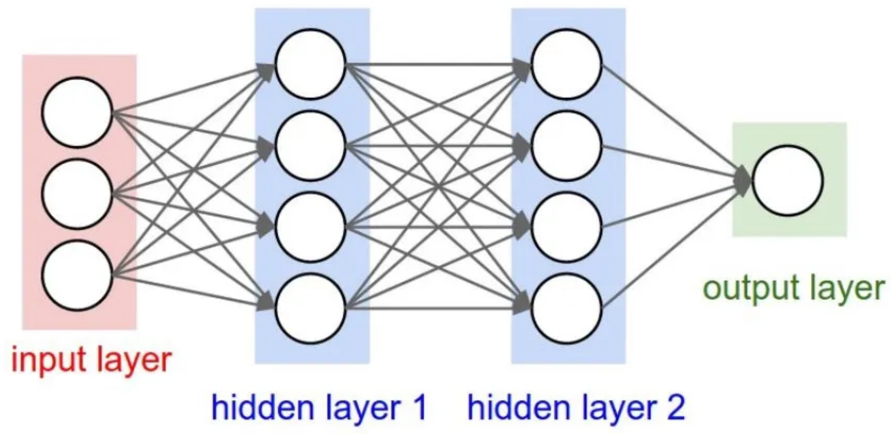

Figure 3: Diagram for a Multi-Layer Perceptron (MLP)

1.2.2 Convolutional Neural network (CNN)

Another type of neural network that we are going to be looking at is the Convolutional Neural Network, it is a class of artificial neural networks most commonly applied to analyze visual imagery. CNNs use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are able to capture local features and patterns in the input images. CNNs typically consist of several types of layers, such as convolutional layers, pooling layers, and fully connected layers.

In this lab we are going to use a specific architecture of CNN called LeNet-5.

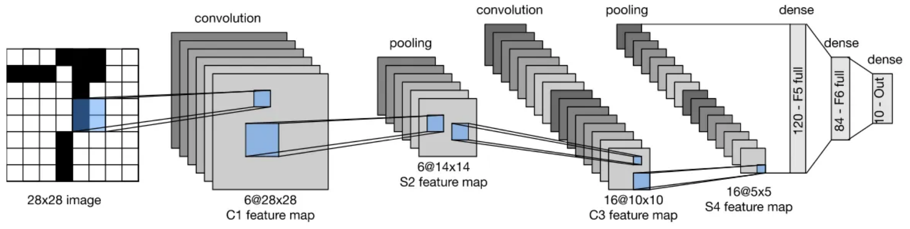

Figure 4: LeNet-5. A Convolutional Neural Network (CNN) architecture

1.2.3 Loss function

Now that we have a general idea for these two types of networks we are going to use, we could calculate the ”loss” on the output layers. A loss function is a function that measures the error or discrepancy between the output of a model and the desired or true value. Loss functions are used to evaluate the performance of a model and to guide the optimization process. They can be different depending on the type and goal of the task. For example in our case we could use something simple like the Total Squared Error or something more appropriate for this task like Categorical Cross-Entropy.

1.2.4 Optimizer

After calculating the loss and building the network, we would also need the main ”engine” for our neural network training. An optimizer is an algorithm or method used to minimize an error function (loss function). A common optimizer is gradient descent, which updates the weights in the opposite direction of the gradient of the loss function.

2 Exercises

See included Jupyter Notebook for practical details.