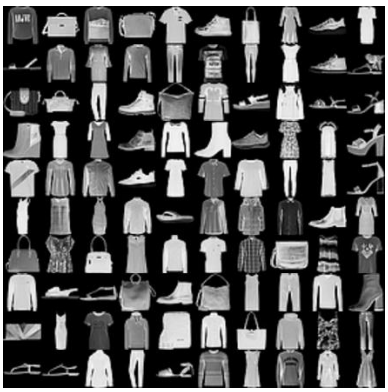
APVC – Exercises – Convolutional Neural Networks

Exercise 1 (CNNs vs. *shallow neural network*)

In this exercise you will compare the results obtained in image classification using a classical neuronal network (*shallow*) and a convolutional neural network.

For this, the FASHION_MNIST dataset of Challenge 1 will be used once again. Remember that this dataset consists of 70,000 mini-images of 28x28 pixels representing clothes, shoes and accessories (10 classes), originally divided into 60,000 images for training and 10,000 for testing.



FASHION_MNIST

1) Consider the multiclass "*shallow" neural network* that was developed in Challenge 1. If you don't have it available, you can use the fashionNet_v2.py script as a base, which implements a simple neural network with 128 neurons in the "hidden" layer and 10 in the output layer. Train using the training and validation sets, and then evaluate the performance on the test set (and write down that performance). If you have the results you obtained in Challenge 1 "at hand", you do not need to complete this point.

2) Now replace the "*shallow*" neural network with a convolutional neural network (CNN). The characteristics of the CNN are as follows: it has two convolutional layers, each followed by a *Max Pooling layer* (2->1), 128 neurons in the dense layer and 10 in the output layer. In the first convolutional layer there should be 16 3x3 filters and in the second 32 3x3 filters.
   a. Based on the convolutional network topology, calculate the number of network parameters (weights) that need to be optimized – do the math using pen and paper!
   b. Implement the convolutional neural network. Confirm the number of network parameters calculated in a) using the `summary()` method.
   c. Train the network and check its performance on the test set. Has it improved compared to what was achieved with the Challenge 1 "*shallow*" neural network?

3) Try to improve the performance of your CNN by using *Dropout* and *Data Augmentation*. You may also try to tune-up your CNN varying the size and number of filters of the convolutional layers, placing additional convolutional layers before each *pooling layer,* changing the number of neurons in the dense layer, etc. How far did you go with the accuracy score?

Exercise 2 (CNN for monument image classification)

In this exercise we intend to classify mini-images that show details of monuments.

To carry out the exercise you need to download the *dataset* `cultural_heritage_images64`[1], available on Moodle. This *dataset* consists of 10 (unbalanced) classes of images corresponding parts of monuments or other buildings with cultural value, such as bell towers, altars, domes, gargoyles, stained glass, etc.

The size of the images in dataset is 64x64 pixels. The original organization of the *dataset* includes a training set (which can be further split into training and validation) and a test set.

1. Upload the *dataset* and process it to get training, validation and testing sets.
2. Develop and train a convolutional neural network to classify the images. The network's topology is up to you.
3. Calculate the accuracy and show the confusion matrix for the test set.

Notes & Tips:

- Since the images are loaded from the file system, it is suggested that you use the available `flowerNet.py` script (or notebook) as a basis, where the automatic construction procedure for the *dataset* is quite similar to what needs to be done for this case.

- The test set should be loaded with the `shuffle=False` option, and without setting the `validation_split`, `seed`, and `subset` options.

- This dataset is more difficult to classify than the previous ones, so don't set your expectations too high (if you reach an accuracy in the test set close to 75% that's not bad at all!).

[1] https://www.kaggle.com/datasets/ikobzev/architectural-heritage-elements-image64-dataset