Lab I: Train neural networks for MCU

Polytech Nice Sophia

During this lab, we will train our own CNN using a Tensorflow Notebook.

Part I. Train a network for Manual Digits classification (Image classification, 2D convolutions)

In this part, your goal is to define a Convolutional Neural Network (CNN) reaching **between 98,5 and 99%** of good recognition (accuracy) on the MNIST Dataset. The accuracy of your model must be confirmed by an average **on 3 learning** and the test **on the entire test dataset**.

To reach better accuracy you can change the following hyper-parameters of your CNN in the Build model and Train model parts of TF script:

- The number of **filters** in each convolution (*Conv2D*) layer
- The size of the **kernels** in each convolution layer (by default set to 3x3)
- The number of **Convolution layers**
- The use of ***MaxPool2D*** layers between *Conv2D* layers
- The number of ***Dense* layers**
- The number of **neurons** in each dense layer (except the output layer)
- The **activation function** in each layer
- The number of **epochs** of learning (how many time the network will learn the entire dataset)

When you get the expected accuracy, fill the following table as your result.

Layer	Output shape	Number of parameters	Kernel
Input			
Total trainable parameters			
Number of Epochs			

Final model	Accuracy on train	Loss on train	Accuracy on test
Learn 1			

Learn 2						
Learn 3						
Results	**Average**	**Std deviation**	**Average**	**Std deviation**	**Average**	**Std deviation**

Part II. Train a network for Human Activity Recognition (time series classification, 1D convolutions)

In this part, your goal is to define a CNN reaching **at least 90%** of good recognition (accuracy) on the **UCI HAR** Dataset (available here). The accuracy of your model must be confirmed by an average **on 3 learning** and the test on the test dataset (composed of 2793 vectors).
First, change directory in the education materials. Note that data are time series. Consequently, the shape of data is 1D, just as the convolution kernels.
First, **modify the script in order to plot the accuracy and loss (see the lecture notes)** during the training according to epochs. It will help you to interpret the good or bad behavior of your tuning.
To reach better accuracy you can change the following hyper-parameters of your CNN in the Build model and Train model parts of notebook:

- The **Learning rate**
- The number of **epochs** of learning (how many time the network will learn the entire dataset)
- The number of **filters** in each convolution (*Conv1D*) layer
- The size of the **kernels** in each convolution layer
- The number of **Convolution layers**
- The use of ***MaxPool1D*** layers between *Conv1D* layers
- The number of ***Dense* layers**
- The number of **neurons** in each dense layer (except the output layer)

When you get the expected accuracy, fill the following table as your result.

Layer	Output shape	Number of parameters	Kernel
Input			
Total trainable parameters			
Number of Epochs			

Final model	Accuracy on train	Loss on train	Accuracy on test
Learn 1			

Learn 2						
Learn 3						
Results	**Average**	**Std deviation**	**Average**	**Std deviation**	**Average**	**Std deviation**