Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

3^{rd} Lab Assignment

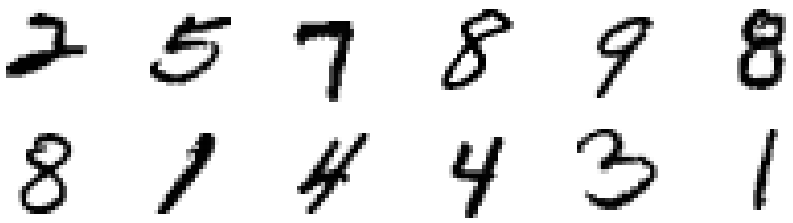
Neural Networks

This assignment aims at illustrating the applications of neural networks to image classification. We'll train a multilayer perceptron (MLP) and a convolutional neural network (CNN) for classification and compare the two approaches.

`Note` In this assignment, you'll often find between brackets, as in {`command`}, suggestions of Python commands that may be useful to perform the requested tasks. You should search Python documentation, when necessary, to obtain a description of how to use these commands.

1 Digit Recognition

Our classification problem is a pattern recognition one, using supervised learning. Our goal is to classify binary images of digits (from 0 to 9), with 28x28 pixels each. The following figure illustrates some of the digits (black and white are reversed).



1.1 Data

We will use a smaller version of the well known MNIST data set containing gray scale images of digits. The original data set contains 60000 training images and 10000 test images. In our version of this data set there are only 3000 training images and 500 test images.

1. Load the data files and check the size of inputs X and labels y.{`load, shape from numpy`}
2. Display some of the digits in the train and test data. {`imshow, show from matplotlib`}
3. Divide data by 255 to get floating point values in the 0–1 range.
4. Convert labels to one-hot encoding. In this representation, the label matrix will have 10 elements, with the component that corresponds to the pattern's class equal to 1, and all the other components equal to 0. {`to_categorical from keras`}

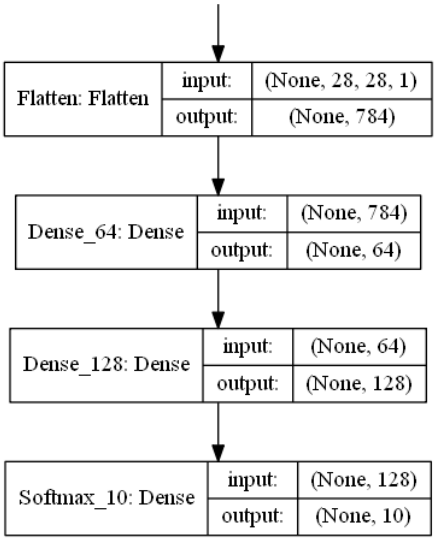


Figure 1: Graph plot of the MLP model.

5. Split train data into two subsets, one for actual training and the other for validation. Use 30% for validation.{`train_test_split from sklearn`}

1.2 MLP

Figure 1 shows a graph plot of the multi layer perceptron (MLP) you will create.

1. Create a sequential model and start by adding a flatten layer to convert the 2D images to 1D. Since this is the first layer in your model, you'll need to to specify the input shape which, for 28x28 pixels with only one color channel (grayscale), is (28,28,1). {`Sequential, Add, Flatten from keras`}
2. Add two hidden layers to your MLP, the first with 64 and the second with 128 neurons. Use 'relu' as activation function for all neurons in the hidden layers.{`Add, Dense from keras`}
3. End the network with a softmax layer. This is a dense layer that will return an array of 10 probability scores, one per class, summing to 1. {`Dense from keras`}
4. Get the summary of your network to check it is correct.{`summary from keras`}
5. Create an Early stopping monitor that will stop training when the validation loss is not improving (use patience=15 and restore_best_weights=True). {`EarlyStopping from keras.callbacks`}

6. Fit the MLP to your training and validation data using 'categorical_crossentropy' as the loss function, a batch size of 300 and Adam as the optimizer (learning rate=0.01, clipnorm=1). Choose, as stopping criterion, the number of iterations reaching 400. Don't forget the Early Stopping callback.{`Compile, Fit from keras`}

7. Plot the evolution of the training loss and the validation loss. Note that the call to fit() returns a History object where metrics monitored during training are kept. {`Plot from matplotlib`}

8. To get an idea of how well the model generalizes to new, unseen data, evaluate performance (accuracy and confusion matrix) on the test data. {`predict from keras, accuracy_score, confusion_matrix from sklearn.metrics`}

9. Repeat the previous items without Early Stopping.

1.3 CNN

Figure 2 shows a graph plot of the Convolutional neural network you will create.

1. Create a Convolutional Neural Network (CNN) with two alternated Convolutional (with relu activation and 3x3 filters) and MaxPooling2D layers (2x2). Use 16 filters in the first conv layer and 32 in the second. Add a flatten layer and then a dense layer with 64 units (with relu activation). End the network with a softmax layer. {`Sequential, Add, Conv2D, MaxPooling2D, Flatten, Dense from keras`}

2. Get the summary of your network to check it is correct.{`summary from keras`}

3. Fit the CNN to your training and validation data. Use the same loss function, batch size, optimizer and Early Stopping callback that were used for the MLP.

4. Plot the evolution of the training loss and the validation loss

5. Evaluate performance (accuracy and confusion matrix) on the test data.

1.4 Comments

1. Explain in your own words how Early Stopping works and what is the goal.

2. Comment on the differences in loss evolution, execution time and test set performance between the MLP model with and without Early Stopping.

3. Comment on the differences between MLP and CNN with Early Stopping, in what regards performance and number of parameters.

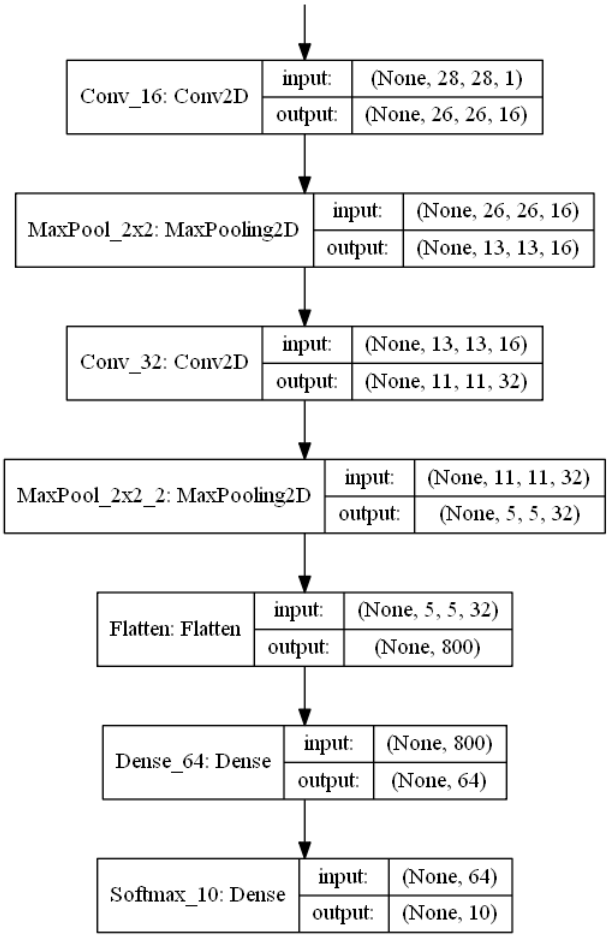


Figure 2: Graph plot of the CNN model.