Problem 2

1) We come up with an architecture we call **SmallVGGNet**

The learning rate is **0.03** (3x lower than the NN of Problem 1)
MNIST is composed of images of size **1x28x28**
Each batch has size **m**.
All weights are **Xavier** initialized.
Each convolutional layer pass is followed by a **ReLU** non-linearity
Each fully-connected layer pass is followed by a **ReLU** non-linearity except the last one which is followed by a **softmax** non-linearity
Each pooling layer pass is not followed by any non-linearity

Block 1

1st convolutional layer (conv1) :
Properties : **16** kernels of size **3x3x1**, stride **1**, padding **3**
Number of parameters : **16 x (3x3x1 + 1) ⇔ 160**
Input : **m x 1 x 28x28**
Output : **m x 16 x 32x32**

2nd convolutional layer (conv2) :
Properties : **16** kernels of size **3x3x16**, stride **1**, padding **3**
Number of parameters : **16 x (3x3x16 + 1) ⇔ 2.320**
Input : **m x 16 x 32x32**
Output : **m x 16 x 32x32**

1st pooling layer (pool1) :
Properties : kernel of size **2x2**, no overlapping
Number of parameters : **0**
Input : **m x 16 x 32x32**
Output : **m x 16 x 16x16**

Block 2

3rd convolutional layer (conv1) :
Properties : **32** kernels of size **3x3x16**, stride **1**, padding **1**
Number of parameters : **32 x (3x3x16 + 1) ⇔ 4.640**
Input : **m x 16 x 16x16**
Output : **m x 32 x 16x16**

4th convolutional layer (conv2) :
Properties : **32** kernels of size **3x3x32**, stride **1**, padding **1**
Number of parameters : **32 x (3x3x32 + 1) ⇔ 9.248**

Input : **m x 32 x 16x16**
Output : **m x 32 x 16x16**

2nd pooling layer (pool2) :
Properties : kernel of size **2x2**, no overlapping
Number of parameters : **0**
Input : **m x 32 x 16x16**
Output : **m x 32 x 8x8**

Block 3

5th convolutional layer (conv5) :
Properties : **64** kernels of size **3x3x32**, stride **1**, padding **1**
Number of parameters : **64 x (3x3x32 + 1) ⇔ 18.496**
Input : **m x 32 x 8x8**
Output : **m x 64 x 8x8**

6th convolutional layer (conv6) :
Properties : **64** kernels of size **3x3x64**, stride **1**, padding **1**
Number of parameters : **64 x (3x3x64 + 1) ⇔ 36.928**
Input : **m x 64 x 8x8**
Output : **m x 64 x 8x8**

3rd pooling layer (pool3) :
Properties : kernel of size **2x2**, no overlapping
Number of parameters : **0**
Input : **m x 64 x 8x8**
Output : **m x 64 x 4x4**

Flattening of the Output

Input : **m x 64 x 4x4**
Output : **m x 1024**

Block 4

1st fully-connected layer (fc1) :
Size: **1024 x 500**
Number of parameters : **(1024 + 1) x 500 ⇔ 512.500**
Input : **m x 1024**
Output : **m x 500**

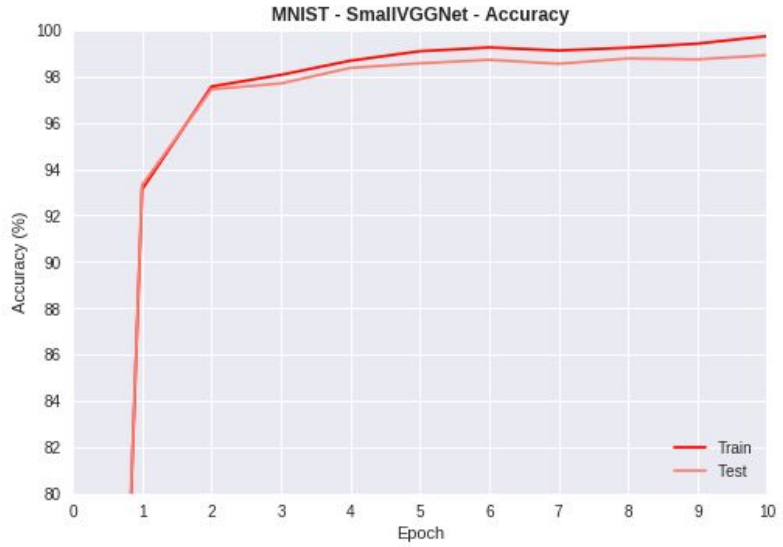
2nd fully-connected layer (fc2) :
Size: **500x500**
Number of parameters : **(500 + 1) x 500 ⇔ 250.500**

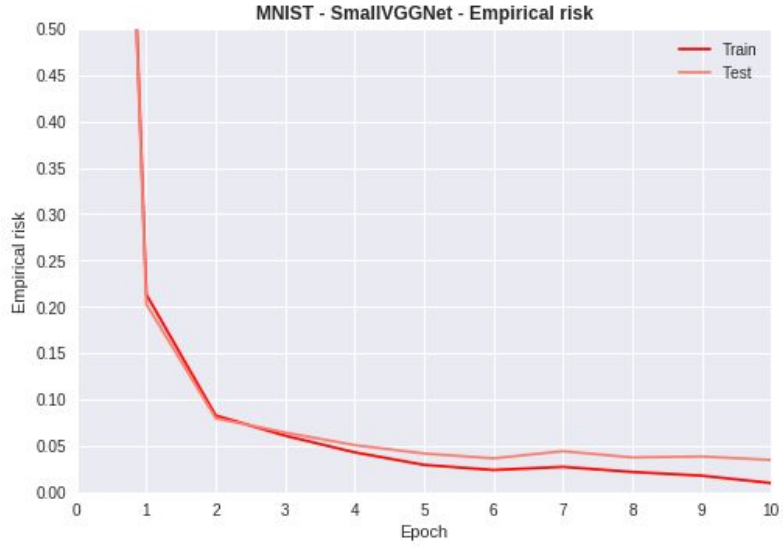
Input : **m x 500**
Output : **m x 500**

3rd fully-connected layer (fc3) :
Size: **500x10**
Number of parameters : **(500 + 1) x 10 ⇔ 5.010**
Input : **m x 500**
Output : **m x 10**

Total number of parameters : 839.802 which is a bit less than the **Problem 1 NN**

2.





Despite having less parameters and a lower learning rate, the CNN performs much better than the MLP.
Test accuracy : CNN ~99% MLP ~98%; test loss : CNN ~0.05 MLP ~0.7
The CNN takes advantage of the spatial information of the image while the MLP doesn't. This loss of information is expressed in the loss and accuracy difference.