ARN - Report - Labo04

Anthony Coke, Guilain Mbayo, Mehdi Salhi

May 11, 2022

Contents

Learning algorithm

> 1. What is the learning algorithm being used to optimize the weights of the neural networks? What are the parameters (arguments) being used by that algorithm? What cost function is being used ? please, give the equation(s)

> MLP_from_raw_data.ipynb

The algorithm used is RMSprop.

The arguments used by this algorithm are: - Learning rate A Tensor, floating point value, or a schedule that is a tf.keras.optimizers.schedules.LearningRateSchedule, or a callable that takes no arguments and returns the actual value to use. The learning rate. Defaults to 0.001. - rho: Discounting factor for the history/coming gradient. Defaults to 0.9. - momentum: A scalar or a scalar Tensor. Defaults to 0.0. - epsilon: A small constant for numerical stability. This epsilon is “epsilon hat” in the Kingma and Ba paper (in the formula just before Section 2.1), not the epsilon in Algorithm 1 of the paper. Defaults to 1e-7. - centered: Boolean. If True, gradients are normalized by the estimated variance of the gradient; if False, by the uncentered second moment. Setting this to True may help with training, but is slightly more expensive in terms of computation and memory. Defaults to False. - name: Optional name prefix for the operations created when applying gradients. Defaults to “RMSprop”. - **kwargs: keyword arguments. Allowed arguments are clipvalue, clipnorm, global_clipnorm. If clipvalue (float) is set, the gradient of each weight is clipped to be no higher than this value. If clipnorm (float) is set, the gradient of each weight is individually clipped so that its norm is no higher than this value. If global_clipnorm (float) is set the gradient of all weights is clipped so that their global norm is no higher than this value.

The used cost function is the categorical crossentropy function. It’s equation is:

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

Figure 1: ARN-Labo04-CrossEntrEquation

Model Complexity

> 2. Model complexity: for each experiment (shallow network learning from raw data, shallow network learning from features, CNN, and Fashion MNIST), select a neural network topology and describe the inputs, indicate how many are they, and how many outputs. Compute

> the number of weights of each model (e.g., how many weights between the input and the hidden layer, how many weights between each pair of layers, biases, etc..) and explain how do you get to the total number of weights.

> MLP_from_raw_data.ipynb Inputs: 784, which are each pixels in a picture Outputs: 10 classes (numbers between 0 and 9) Activation function: tanh Activation function for output layer: softmax Neurons in hidden layer: 250 Batch size: 4096 Dropout: 0.5 Number of epoch: 150 The model has 784 inputs, 1 hidden layer that contains 250 neurons and 10 outputs. The number of weights between the inputs and the hidden layer is 784 *750 = 196000. The number of weights between the hidden layer and the outputs is 250* 10 = 2500. The total number of weights is 198500.

> MLP_from_HOG.ipynb Inputs: 392 Outputs: 10 classes (numbers between 0 and 9) Activation function: sigmoïd Activation function for output layer: softmax Neurons in hidden layer: 200 Batch size: 1024 pixel per cell: 7 n_orientation: 16 number of epoch: 250 (but we could see that 150 is enough) Dropout: 0.5 The model has 392 inputs, 1 hidden layer that contains 200 neurons and 10 outputs. The number of weights between the inputs and the hidden layer is 392 *200 = 78400. The number of weights between the hidden layer and the outputs is 200* 10 = 2000. The total number of weights is 80400.

> CNN.ipynb

> Fashion_MNIST.ipynb

Deep Neural Networks

> 3. Do the deep neural networks have much more “capacity” (i.e., do they have more weights?) than the shallow ones? explain with one example

The deep neural network have more hidden layer than the shallow ones, but it doesn’t necessary mean that it has more neurons in it. For exemple, in this lab we use 300 neurons in the hidden layer for the shallows network (raw_data and HOG), against only 25 neurons for the deep one (CNN). The deep neural networks have more capacity, because they usually need less components to achieve the same goal or better than a shallow neural network. If we compare the weights of each model, the shallow one will have more weight than the deep one. For exemple, a model with 2 entries, 6 neurons in one hidden layer and 2 output, we get 2 * 6 + 6 * 2 = 24 links that have each their weight. For the same model but with 3 hidden layers, we got 2 * 2 + 2 * 2 + 2 * 2 + 2 * 2 = 16 links, and so 16 weights.

Tests

> 4. Test every notebook for at least three different meaningful cases (e.g., for the MLP exploiting raw data, test different models varying the number of hidden neurons, for the feature-based model, test pix_p_cell 4 and 7, and number of orientations or number of hidden neurons, for the CNN, try different number of neurons in the feed-forward part) describe the model and present the performance of the system (e.g., plot of the evolution of the error, final evaluation scores and confusion matrices). Comment the differences in results. Are there particular digits that are frequently confused?

> MLP_from_raw_data.ipynb

Model:

- Activation function: tanh
- Neurons: 300
- Dropout: -
- Batch size: 2048
- Epochs: 150

```
Test score: 0.11751019209623337
Test accuracy: 0.9818999767303467
```

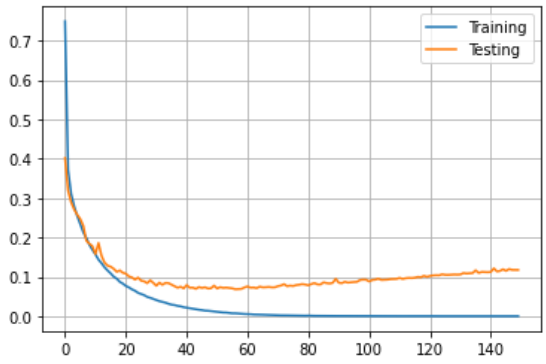

Figure 2: ARN-RAW-Plot-tanh-softmax_Batch2048_NoDropout_Epoch150

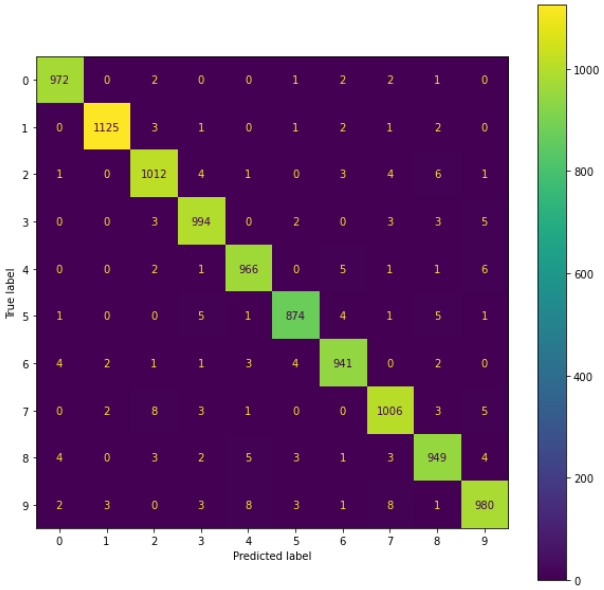

Figure 3: ARN-RAW-ConfMat-tanh-softmax_Batch2048_NoDropout_Epoch150

Model:

- Activation function: tanh
- Neurons: 300

- Dropout: 0.5
- Batch size: 2048
- Epochs: 150

```
Test score: 0.0734260305762291
Test accuracy: 0.9796000123023987
```

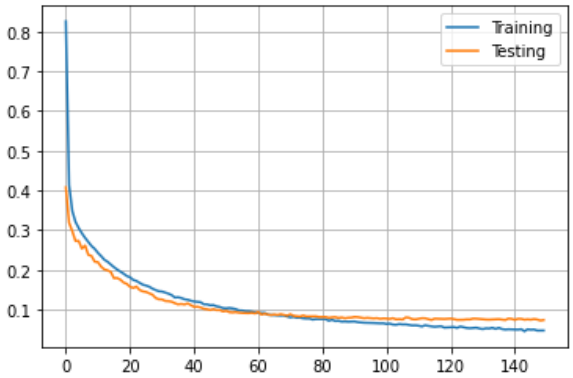

Figure 4: ARN-RAW-Plot-tanh-softmax_Batch2048_Dropout_Epoch150

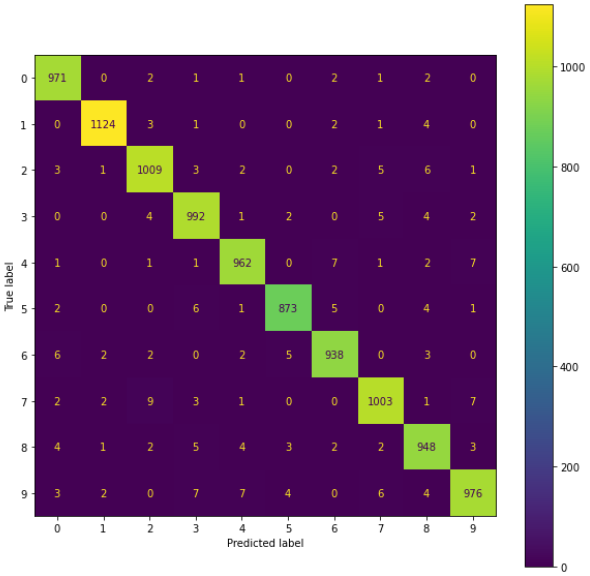

Figure 5: ARN-RAW-ConfMat-tanh-softmax_Batch2048_Dropout_Epoch150

Model:

- Activation function: tanh
- Neurons: 250

- Dropout: 0.5
- Batch size: 4096
- Epochs: 150

```
Test score: 0.08130748569965363
Test accuracy: 0.9761999845504761
```

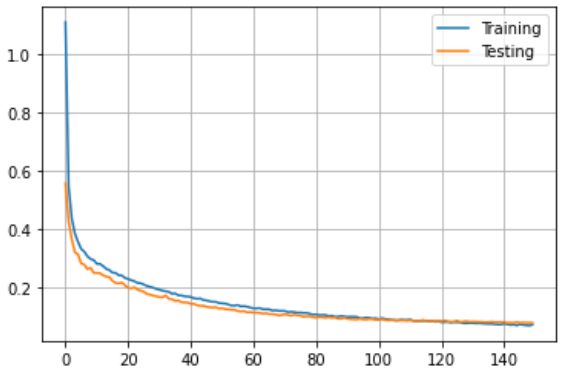

Figure 6: ARN-RAW-Plot-tanh-softmax-Neur250_Batch4096_Dropout_Epoch150

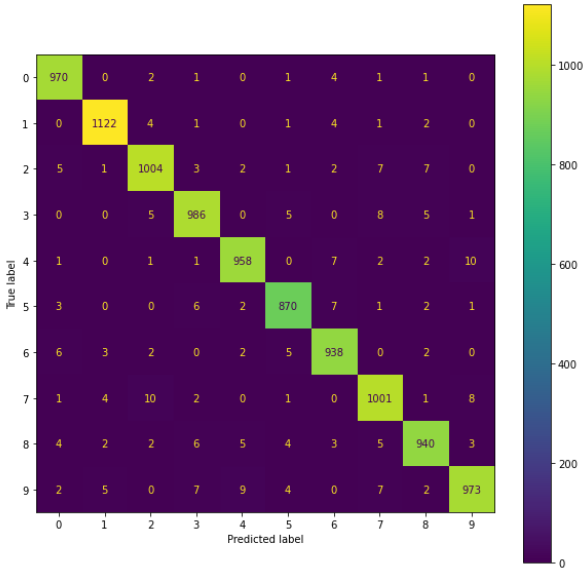

Figure 7: ARN-RAW-ConfMat-tanh-softmax-Neur250_Batch4096_Dropout_Epoch150

Model:

- Activation function: sigmoid
- Neurons: 250

- Dropout: 0.5
- Batch size: 4096
- Epochs: 150

```
Test score: 0.06945059448480606
Test accuracy: 0.9797000288963318
```

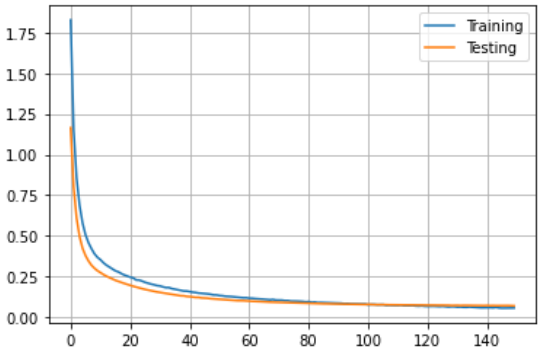

Figure 8: ARN-RAW-Plot-sigmoid-softmax-Neur250_Batch4096_Dropout_Epoch150

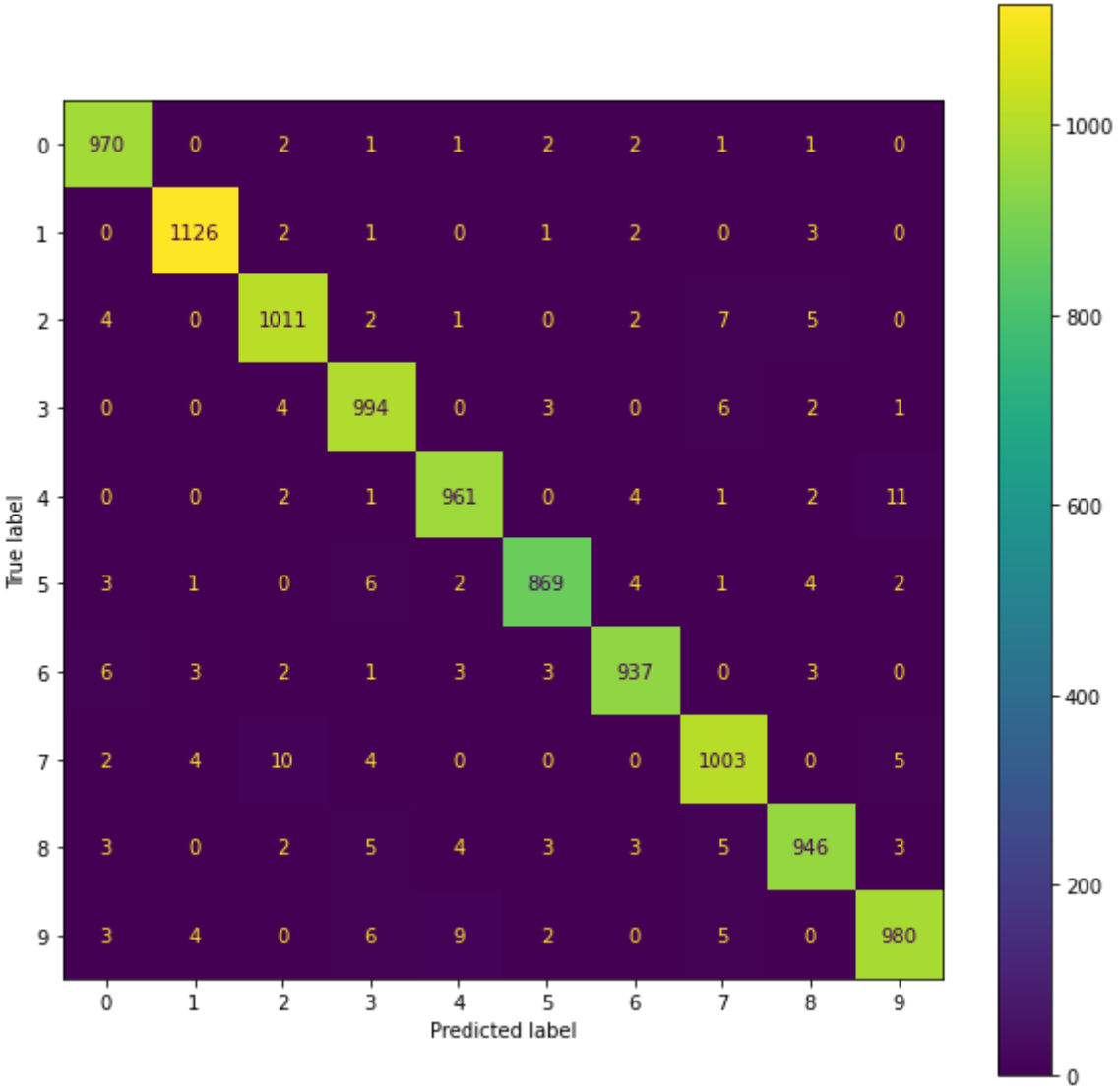

Figure 9: ARN-RAW-ConfMax-sigmoid-softmax-Neur250_Batch4096_Dropout_Epoch150

Model:

- Activation function: tanh
- Neurons: 150

- Dropout: 0.5
- Batch size: 4096
- Epochs: 150

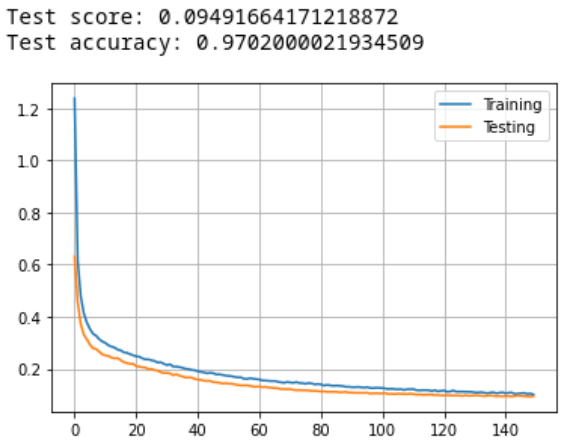

Figure 10: ARN-RAW-Plot-tanh-softmax-Neur150_Batch4096_Dropout_Epoch150

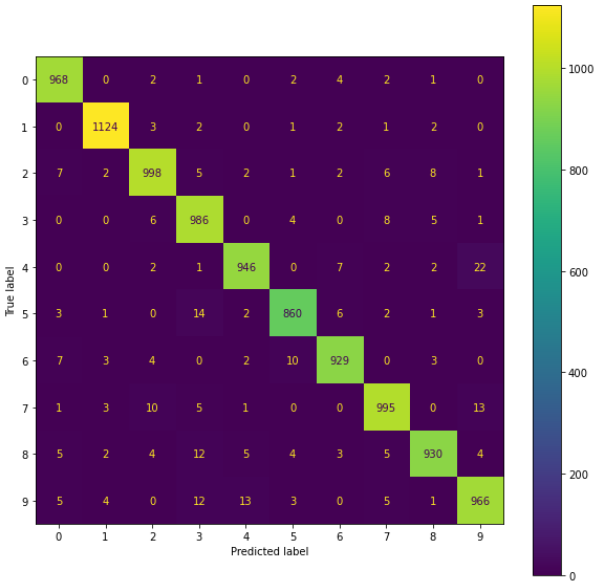

Figure 11: ARN-RAW-ConfMat-tanh-softmax-Neur150_Batch4096_Dropout_Epoch150

Model:

- Activation function: sigmoid
- Neurons: 150

- Dropout: 0.5
- Batch size: 4096
- Epochs: 150

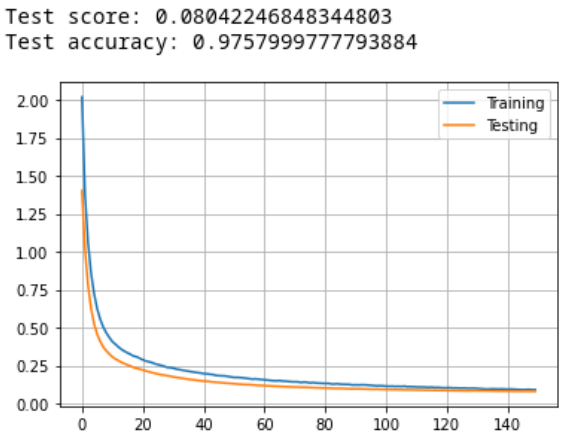

Figure 12: ARN-RAW-Plot-sigmoid-softmax-Neur150_Batch4096_Dropout_Epoch150

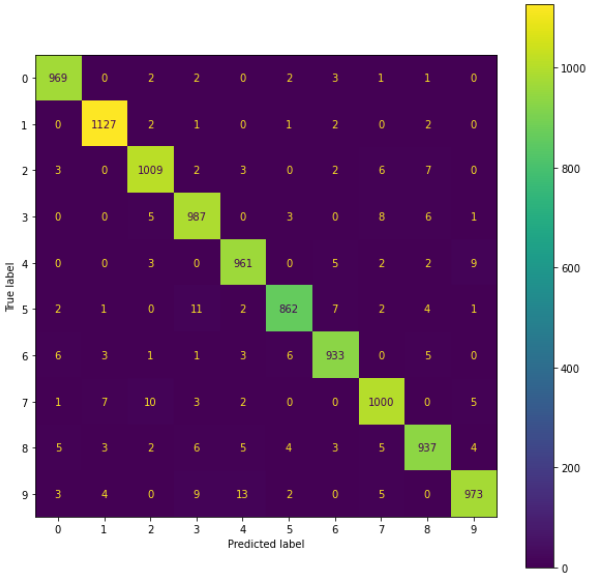

Figure 13: ARN-RAW-ConfMat-sigmoid-softmax-Neur150_Batch4096_Dropout_Epoch150

> MLP_from_HOG.ipynb

Model:

- Activation function: relu
- Neurons: 200
- Batch size: 512
- Dropout: 0.5

- Epochs: 100
- Pixels: 4
- Orientations: 8

```
Test score: 0.080036073923110 96
Test accuracy: 0.9833999872207642
```

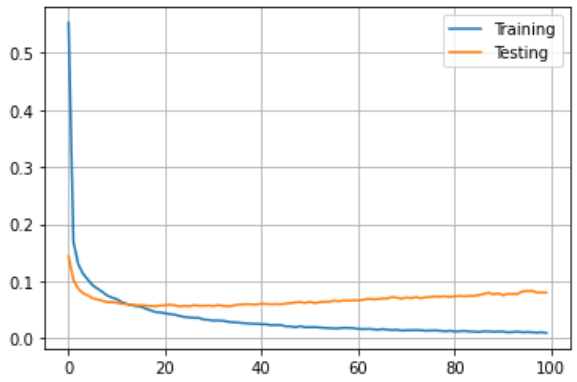

Figure 14: ARN-HOG-Plot-relu-softmax-Neur200_Batch512_Dropout_Epoch100

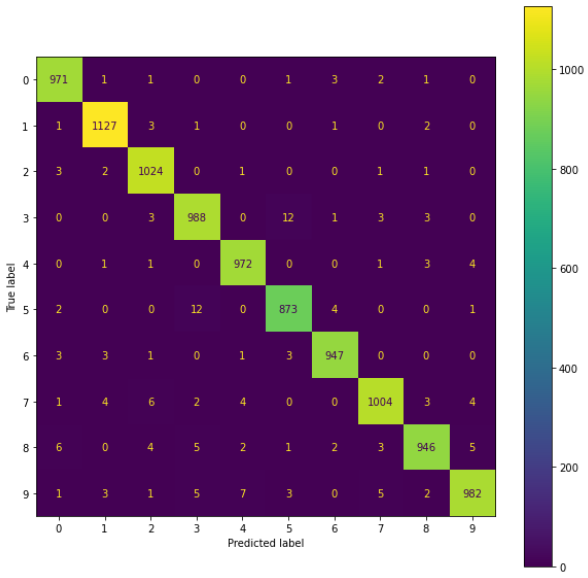

Figure 15: ARN-HOG-ConfMat-relu-softmax-Neur200_Batch512_Dropout_Epoch100

Model:

- Activation function: sigmoid
- Neurons: 200

- Batch size: 512
- Dropout: 0.5
- Epochs: 100
- Pixels: 4
- Orientations: 8

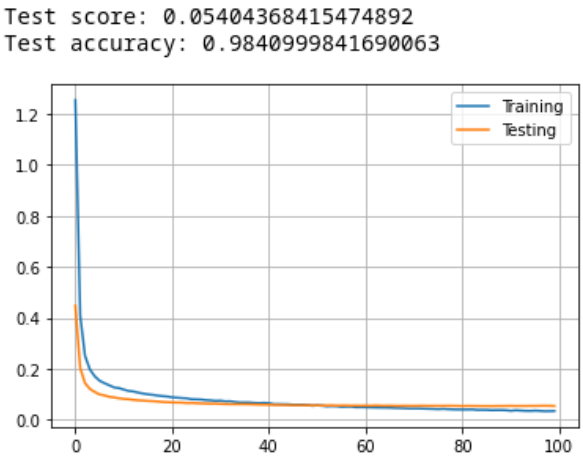

Figure 16: ARN-HOG-Plot-sigmoid-softmax-Neur200_Batch512_Dropout_Epoch100

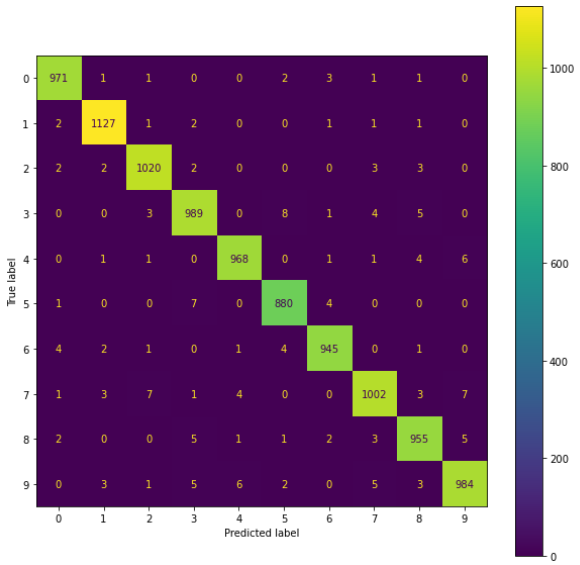

Figure 17: ARN-HOG-ConfMat-sigmoid-softmax-Neur200_Batch512_Dropout_Epoch100

Model:

- Activation function: tanh

- Neurons: 200
- Batch size: 512
- Dropout: 0.5
- Epochs: 100
- Pixels: 7
- Orientations: 8

```
Test score: 0.13775815069675446
Test accuracy: 0.9538999795913696
```

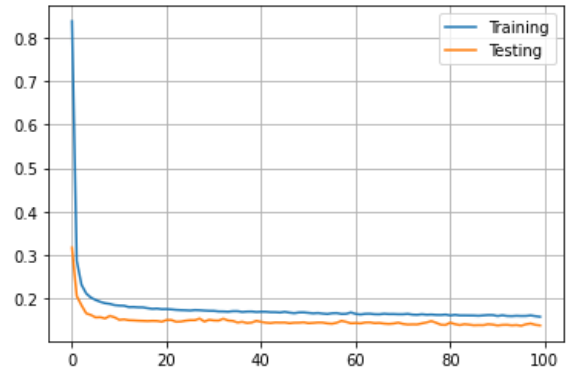

Figure 18: ARN-HOG-Plot-tanh-Pixel7_Neur200_Batch512_Dropout_Epoch100

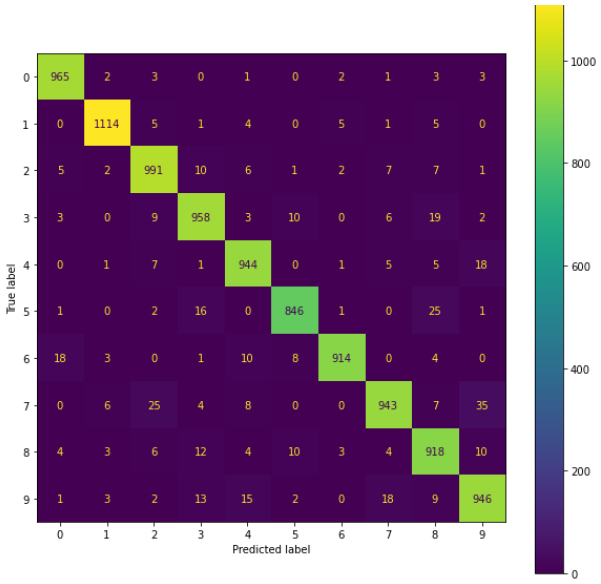

Figure 19: ARN-HOG-ConfMat-tanh-Pixel7_Neur200_Batch512_Dropout_Epoch100

Model:

- Activation function: sigmoid
- Neurons: 200
- Batch size: 512
- Dropout: 0.5
- Epochs: 250
- Pixels: 7
- Orientations: 8

```
Test score: 0.09379129111766815
Test accuracy: 0.9699000120162964
```

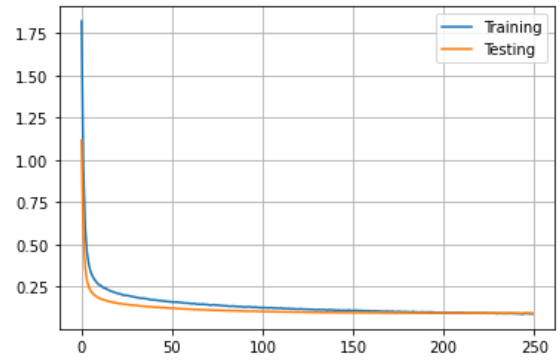

Figure 20: ARN-HOG-Plot-sigmoid-Pixel7_Neur200_Batch512_Dropout_Epoch250

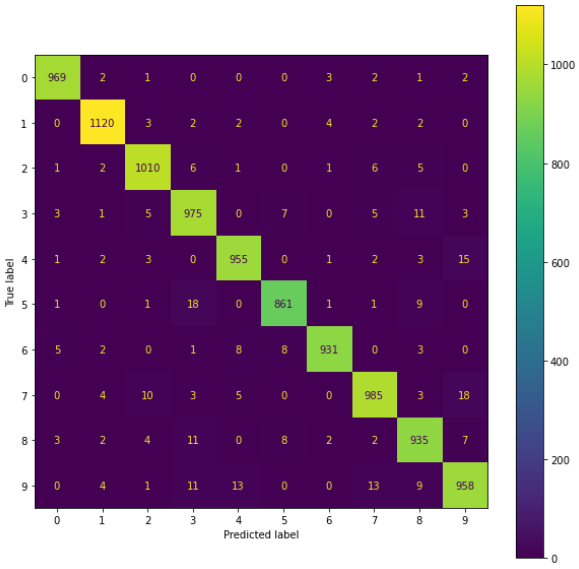

Figure 21: ARN-HOG-ConfMat-sigmoid-Pixel7_Neur200_Batch512_Dropout_Epoch250

Model:

- Activation function: tanh
- Neurons: 200
- Batch size: 1024
- Dropout: 0.5
- Epochs: 250
- Pixels: 7
- Orientations: 8

```
Test score: 0.1260688304901123
Test accuracy: 0.9575999975204468
```

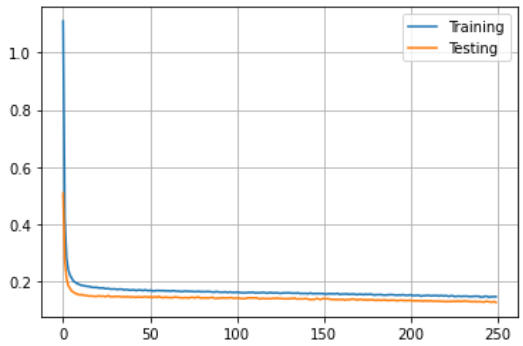

Figure 22: ARN-HOG-Plot-tanh-Pixel7_Neur200_Batch1024_Dropout_Epoch250

Manque confmat

Model:

- Activation function: sigmoid
- Neurons: 200
- Batch size: 1024
- Dropout: 0.5
- Epochs: 250
- Pixels: 7
- Orientations: 8

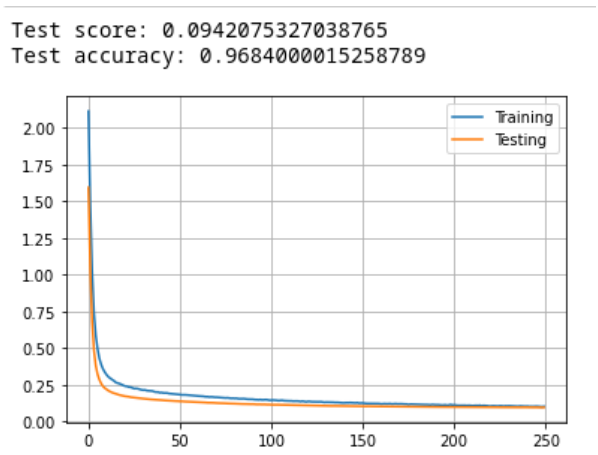

Figure 23: ARN-HOG-Plot-sigmoid-Pixel7_Neur200_Batch1024_Dropout_Epoch250

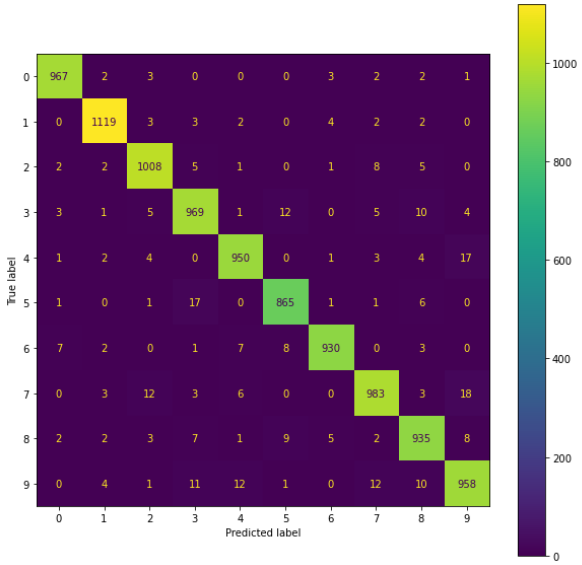

Figure 24: ARN-HOG-ConfMat-sigmoid-Pixel7_Neur200_Batch1024_Dropout_Epoch250

Model:

- Activation function: tanh
- Neurons: 200
- Batch size: 1024
- Dropout: 0.5
- Epochs: 200
- Pixels: 7

- Orientations: 16

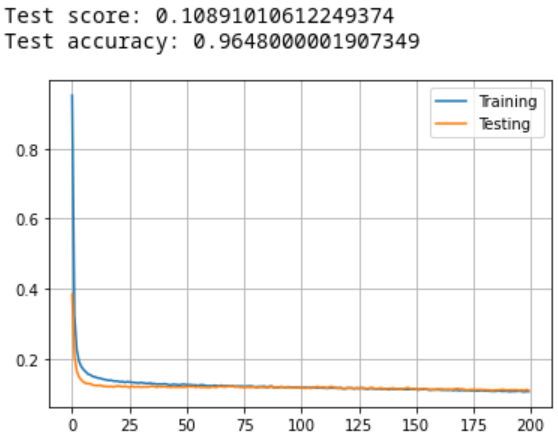

Figure 25: ARN-HOG-Plot-tanh-Pixel7_Or16_Neur200_Batch1024_Dropout_Epoch200

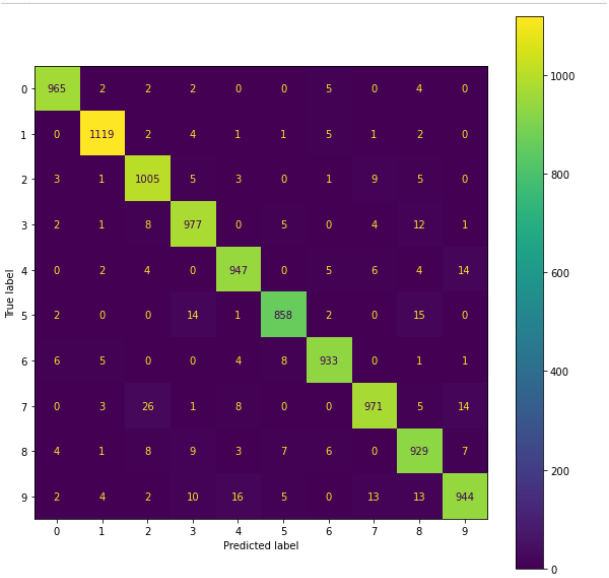

Figure 26: ARN-HOG-ConfMat-tanh-Pixel7_Or16_Neur200_Batch1024_Dropout_Epoch200

Model:

- Activation function: relu
- Neurons: 150
- Batch size: 1024
- Dropout: 0.5

- Epochs: 200
- Pixels: 7
- Orientations: 8

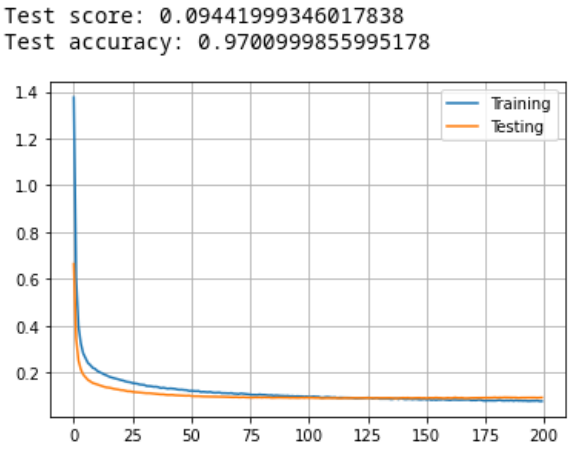

Figure 27: ARN-HOG-Plot-relu-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

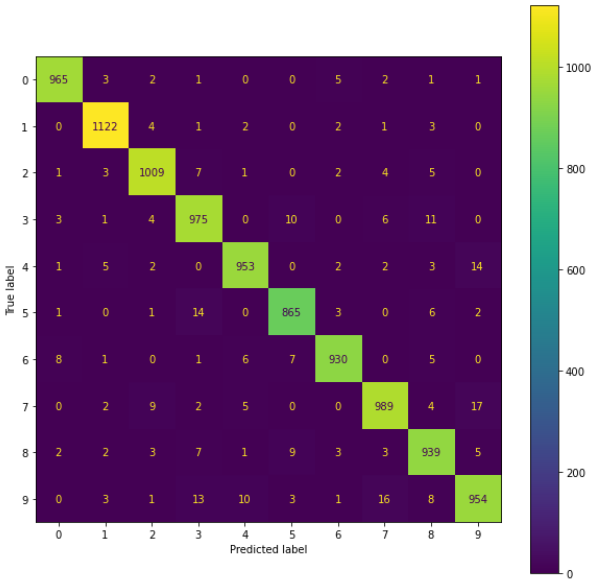

Figure 28: ARN-HOG-ConfMat-relu-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

Model:

- Activation function: sigmoid
- Neurons: 150

- Batch size: 1024
- Dropout: 0.5
- Epochs: 200
- Pixels: 7
- Orientations: 8

```
Test score: 0.1024104580283165
Test accuracy: 0.9660999774932861
```

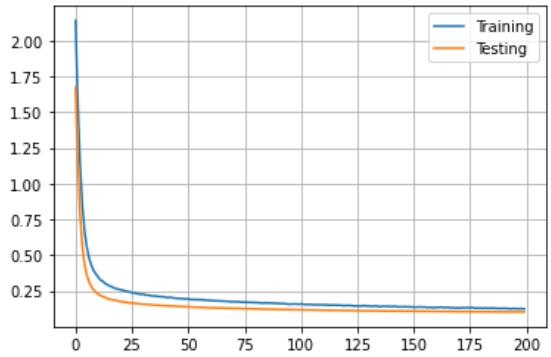

Figure 29: ARN-HOG-Plot-sigmoid-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

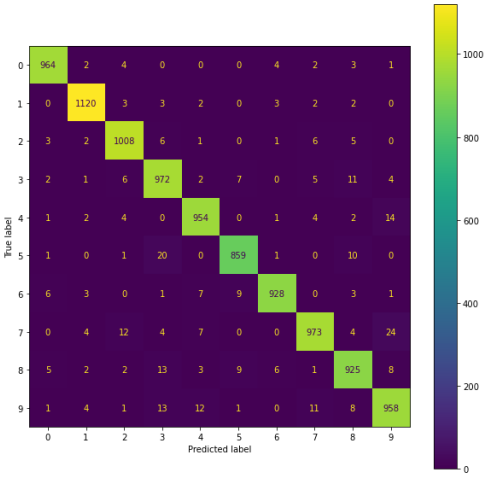

Figure 30: ARN-HOG-ConfMat-sigmoid-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

Model:

- Activation function: sigmoid
- Neurons: 250
- Batch size: 1024

- Dropout: 0.5
- Epochs: 200
- Pixels: 7
- Orientations: 8

```
Test score: 0.09292542189359665
Test accuracy: 0.9690999984741211
```

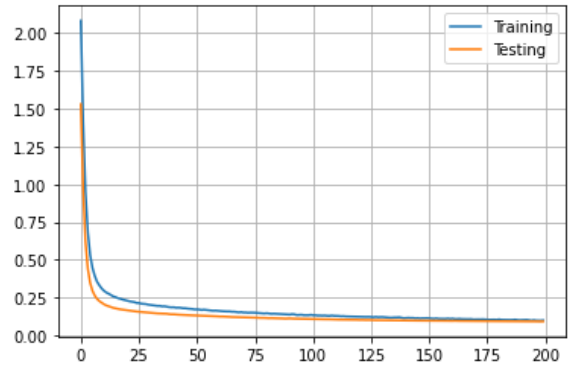

Figure 31: ARN-HOG-Plot-sigmoid-Pixel7_Or8_Neur250_Batch1024_Dropout_Epoch200

> CNN.ipynb

Model:

- L4 neurons: 25
- L4 activation function: Relu
- Batch size: 2048
- Epochs: 50

```
Test score: 0.044601865112781525
Test accuracy: 0.9894000291824341
```

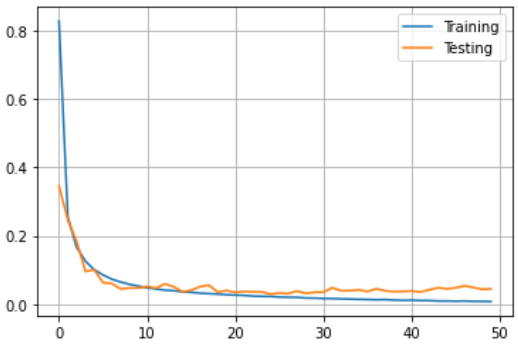

Figure 32: ARN-CNN-Plot-relu-Batch256_25L4_Epoch50

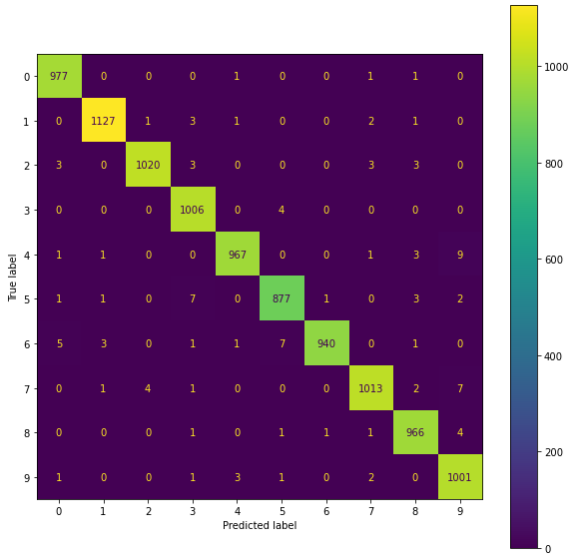

Figure 33: ARN-CNN-ConfMat-relu-Batch256_25L4_Epoch50

As we can see, the results with this preconfigured model are not that bad. The score is low and the accuracy is close to 1.0. However, from 10 epochs this model slowly starts to overfit. We can clearly see the gap between the training error and the testing error increases as we continue to iterate through the epochs.

Model:

- L4 neurons: 10

- L4 activation function: Relu
- Batch size: 2048
- Epochs: 50

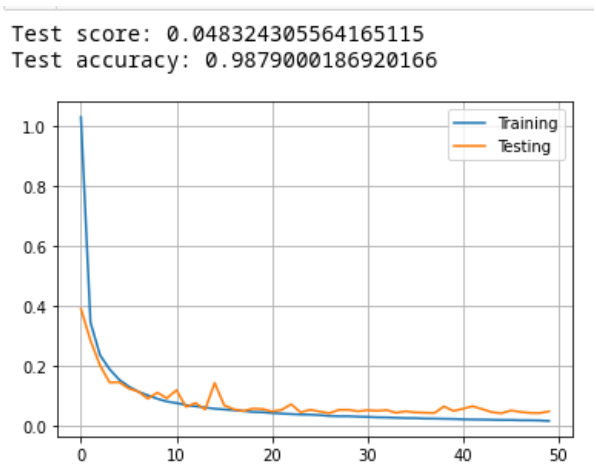

Figure 34: ARN-CNN-Plot-relu-Batch256_10L4_Epoch50

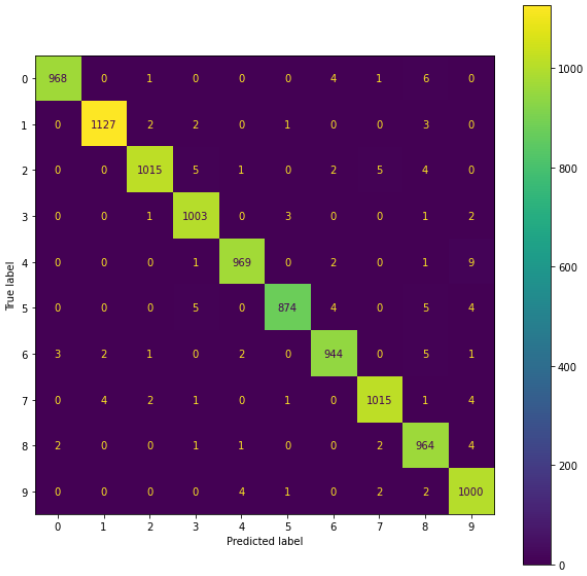

Figure 35: ARN-CNN-ConfMat-relu-Batch256_10L4_Epoch50

We tried to decrease the number of neurons in the L4 layer. The scores are quiet similar to the previous model, but the testing error fluctuates more especially from the start to around 25 epochs. There's still a small overfitting but the curves seem to be stable.

Model:

- L4 neurons: 5
- L4 activation function: Relu

- Batch size: 2048
- Epochs: 50

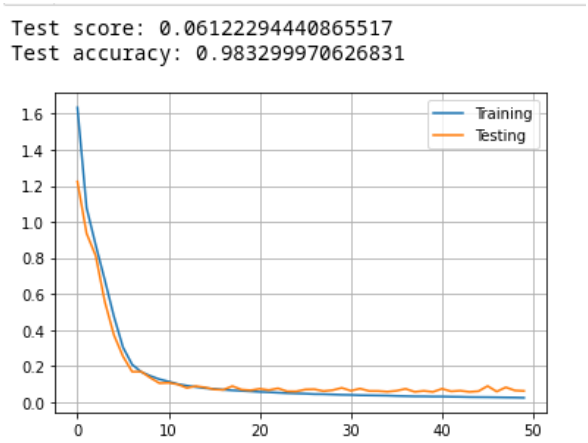

Figure 36: ARN-CNN-Plot-relu-Batch256_5L4_Epoch50

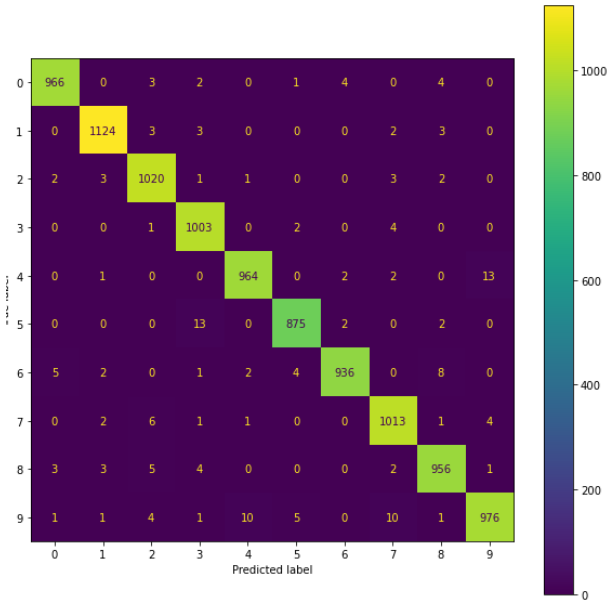

Figure 37: ARN-CNN-ConfMat-relu-Batch256_5L4_Epoch50

Here we tried an extremly low number of neurons in the L4 layer. The result is pretty good. The score is a little bit higher than the first model tested, but the two curves are almost overlapping the entire time. It looks like a good model. The confusion matrix shows that the number 4, 5 and 9 are quiet often wrongly classified.

Model:

- L4 neurons: 35

- L4 activation function: Relu
- Batch size: 2048
- Epochs: 50

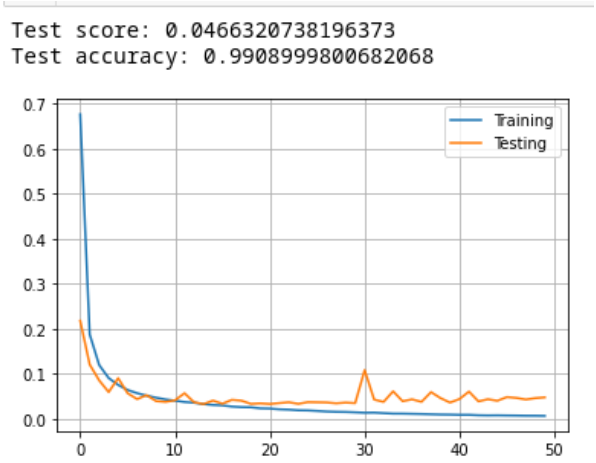

Figure 38: ARN-CNN-Plot-relu-Batch256_35L4_Epoch50

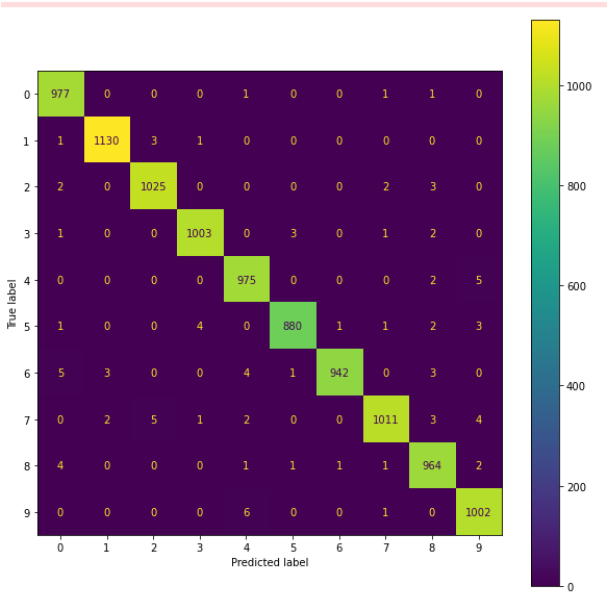

Figure 39: ARN-CNN-ConfMat-relu-Batch256_35L4_Epoch50

We then chose a high number of neurons to see how the model reacts. It's definitely not a good model and 35 neurons is probably a bit to much for this task. There's an overfitting starting at 20 epochs. Surprisingly (or not), despite our bad curves the accuracy is the best we've had. The confusion matrix doesn't show a lot, the classification is not so bad.

Model:

- L4 neurons: 25

- L4 activation function: tanh
- Batch size: 2048
- Epochs: 50

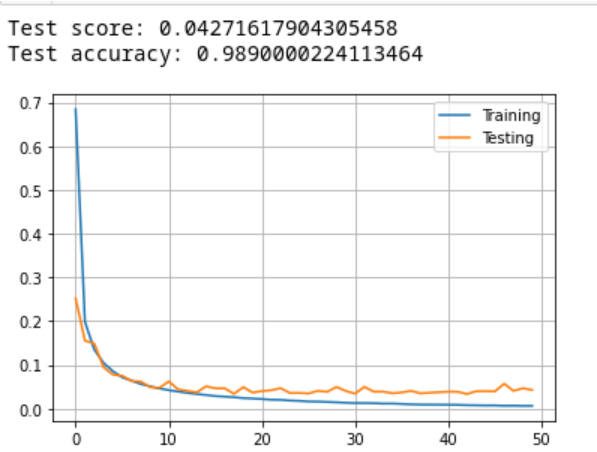

Figure 40: ARN-CNN-Plot-tanh-Batch256_25L4_Epoch50

We also wanted to see how the preconfigured model would behave when changing the activation function in layer 4. This one uses tanh and we can clearly see it overfits directly at 10 epochs, maybe it's just bad luck and after tweaking some parameters it will probably show something good, but we didn't try since we were told just to change the number of neurons in the feed-forward part.

Model:

- L4 neurons: 25

- L4 activation function: sigmoid
- Batch size: 2048
- Epochs: 50

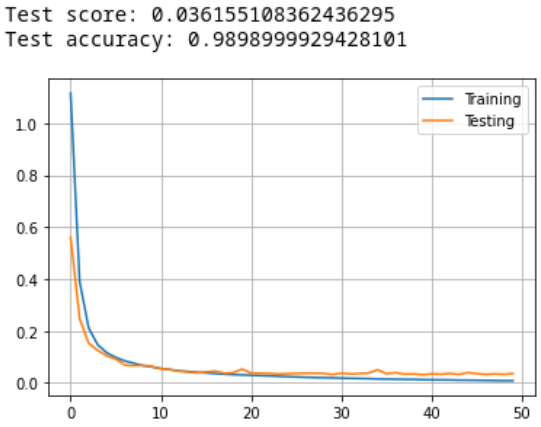

Figure 41: ARN-CNN-Plot-sigmoid-Batch256_25L4_Epoch50

With sigmoid the score is really low, the accuracy close to 1.0 and the curves are overlapping. After 20 epochs the testing error looks like it is going away a bit, but the gap is still really small after 50 epochs. The confusion matrix doesn't show abnormalities, there are few errors here and there, but nothing out of the ordinary.

Conclusion - CNN: As we can see lowering the number of neurons in L4 helped us reduce the gap between the testing and training curves with relu activation function. However, when reaching really low numbers like 5, the really low numbers of neurons, the confusion matrix shows something pretty different that the plot. Some numbers are not classified properly such as 4,5 and 9. Sometimes it represents 1-2% of the predictions for one particular numbers. It's not a big deal, but if we need to be really precise, this result is note enough. Other than that the digits are globally correctly classified even when changing the activation function.