
ARN - Report - Labo04

Anthony Coke, Guilain Mbayo, Mehdi Salhi



May 11, 2022

Contents

| | |
|-----------------------------|----------|
| Learning algorithm | 3 |
| Model Complexity | 3 |
| Deep Neural Networks | 4 |
| Tests | 4 |

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:

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

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: Outputs: Activation function: tanh Activation function for output layer: softmax Neurons in hidden layer: 250 Batch size: 4096 Dropout: 0.5 Number of epoch: 150

MLP_from_HOG.ipynb

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

Test score: 0.11751019209623337
Test accuracy: 0.9818999767303467

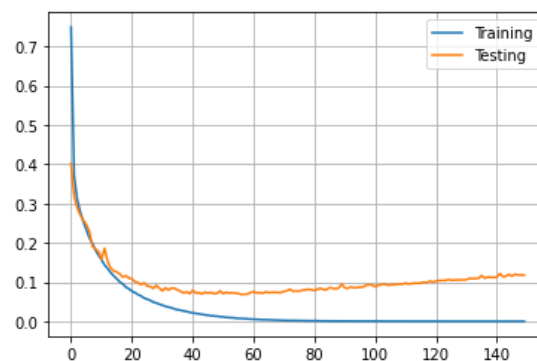


Figure 2: ARN-RAW-Plot-tanh-softmax_Batch2048_NoDropout_Epoch150

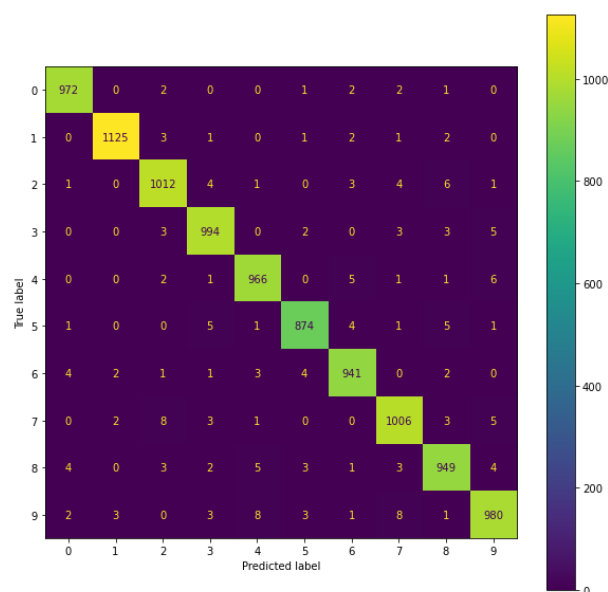


Figure 3: ARN-RAW-ConfMat-tanh-softmax_Batch2048_NoDropout_Epoch150

We can see in this experiment that there's clearly an overfitting.

Test score: 0.0734260305762291
Test accuracy: 0.9796000123023987

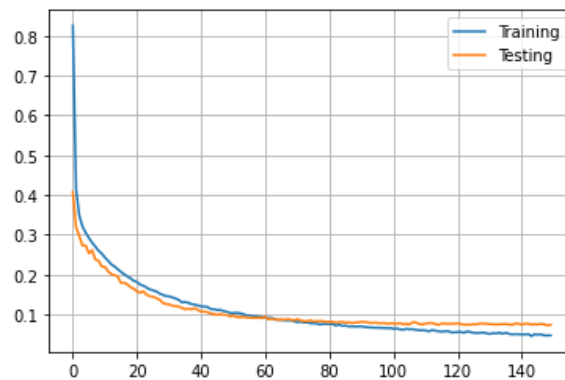


Figure 4: ARN-RAW-Plot-tanh-softmax_Batch2048_Dropout_Epoch150

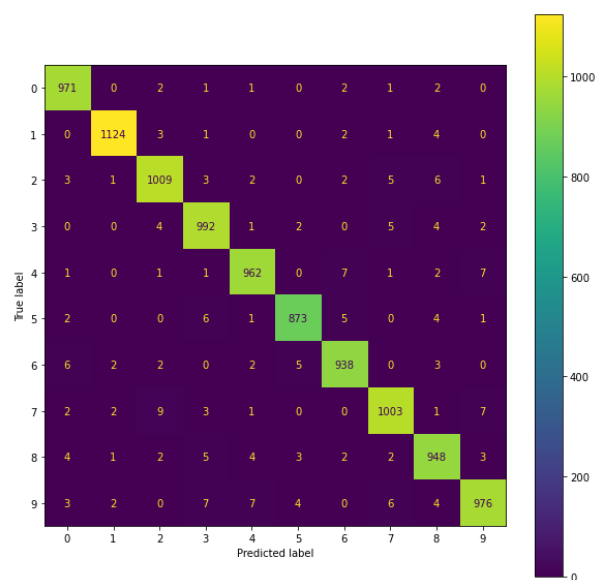


Figure 5: ARN-RAW-ConfMat-tanh-softmax_Batch2048_Dropout_Epoch150

Test score: 0.06564721465110779
Test accuracy: 0.9828000068664551

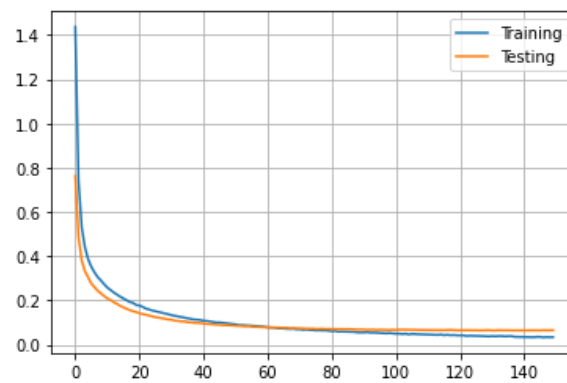


Figure 6: ARN-RAW-Plot-sigmoid-softmax_Batch2048_Dropout_Epoch150

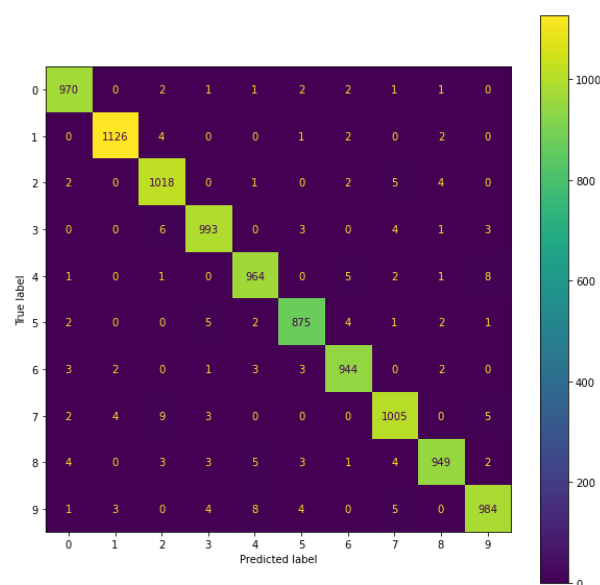


Figure 7: ARN-RAW-ConfMat-sigmoid-softmax_Batch2048_Droptout_Epoch150

Test score: 0.08130748569965363
Test accuracy: 0.9761999845504761

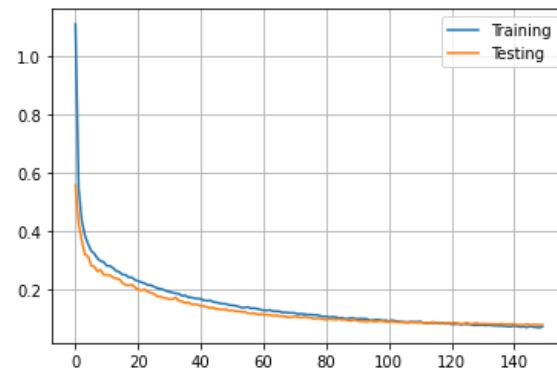


Figure 8: ARN-RAW-Plot-tanh-softmax-Neur250_Batch4096_Dropout_Epoch150

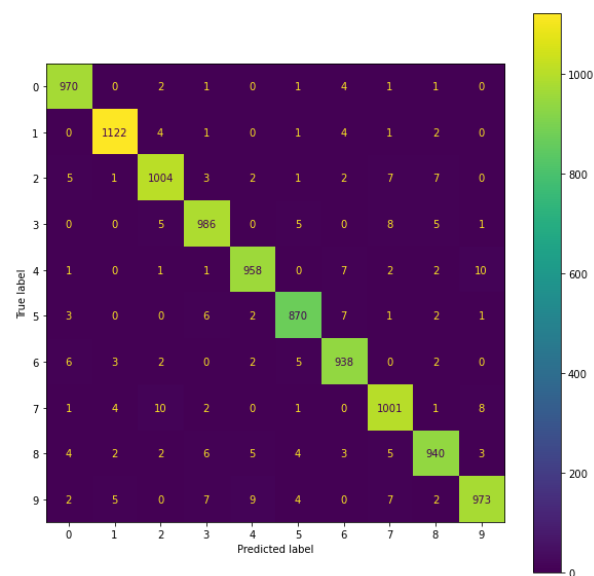


Figure 9: ARN-RAW-ConfMat-tanh-softmax-Neur250_Batch4096_Dropout_Epoch150

Test score: 0.06945059448480606
Test accuracy: 0.9797000288963318

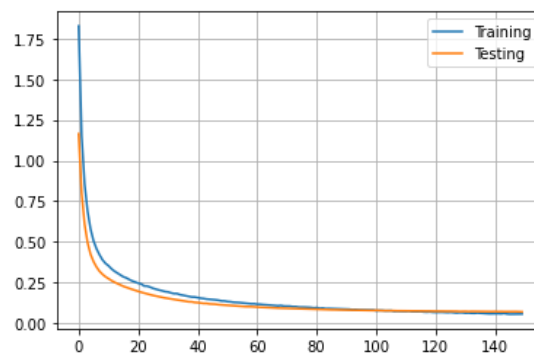


Figure 10: ARN-RAW-Plot-sigmoid-softmax-Neur250_Batch4096_Dropout_Epoch150

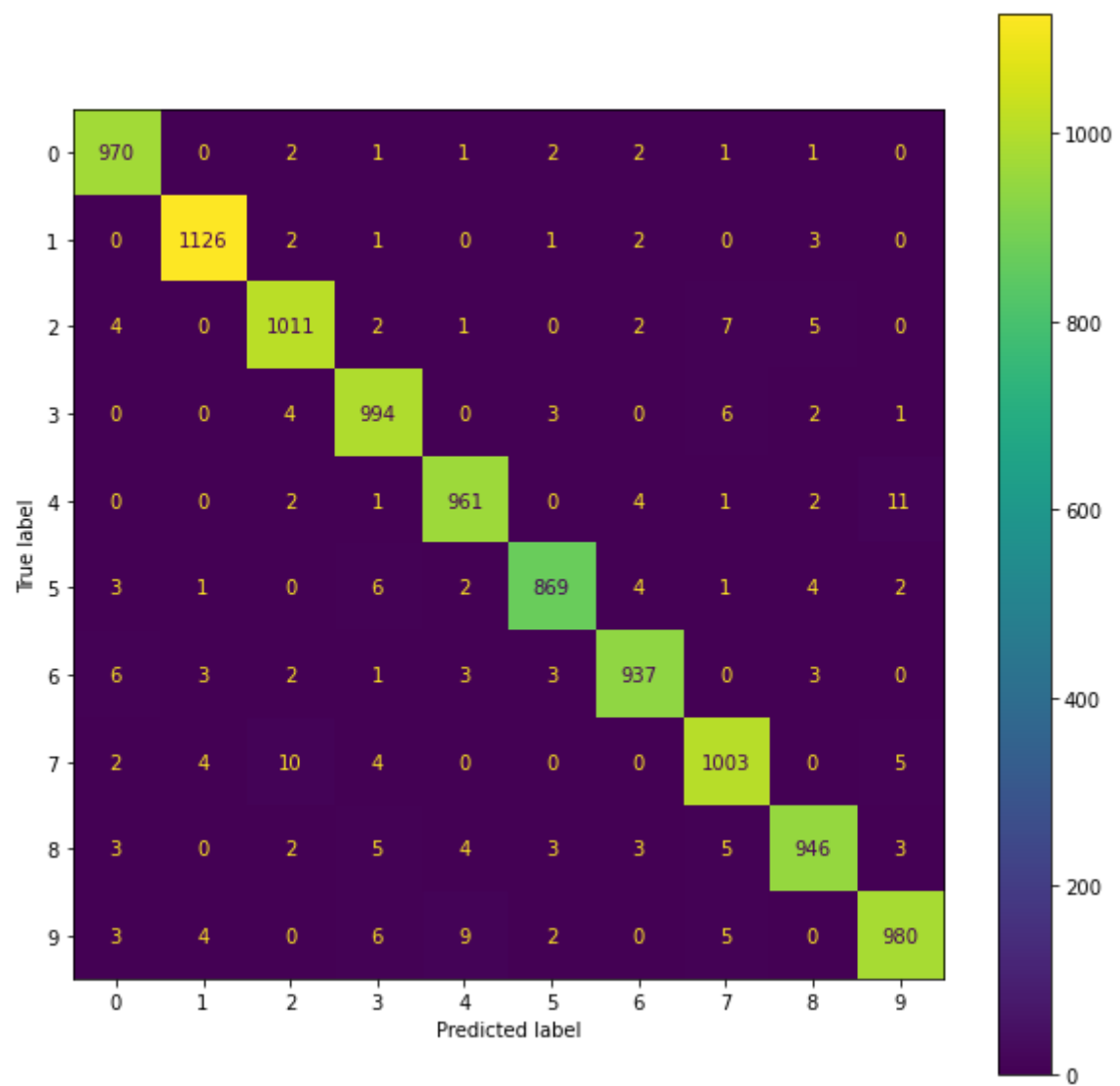


Figure 11: ARN-RAW-ConfMax-sigmoid-softmax-Neur250_Batch4096_Dropout_Epoch150

Test score: 0.06911627948284149
Test accuracy: 0.983299970626831

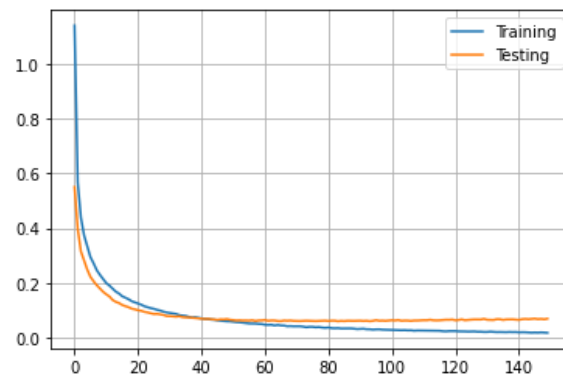


Figure 12: ARN-RAW-Plot-relu_softmax-Neur250_Batch4096_Dropout_Epoch150

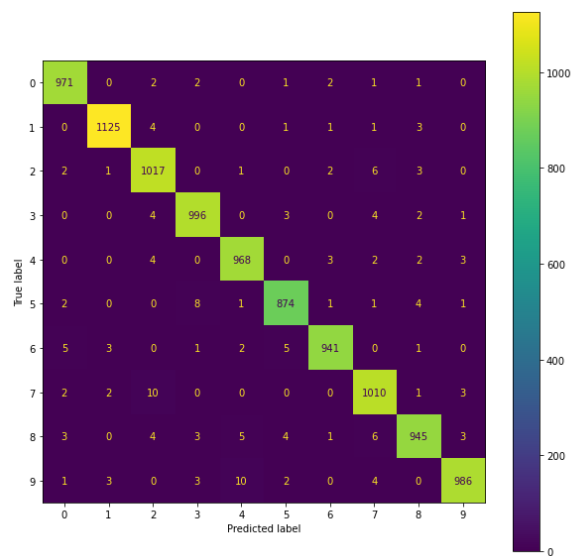


Figure 13: ARN-RAW-ConfMat-relu_softmax-Neur250_Batch4096_Dropout_Epoch150

Test score: 0.07871479541063309
Test accuracy: 0.9796000123023987

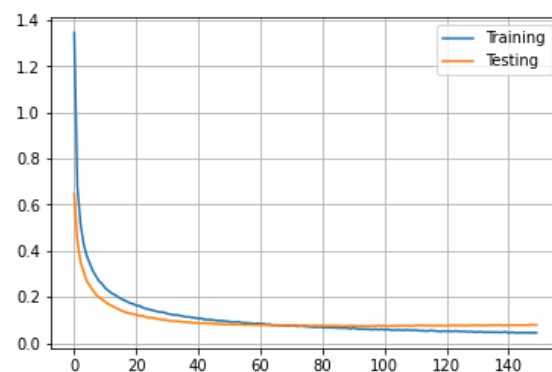


Figure 14: ARN-RAW-Plot-relu-softmax-Neur150_Batch4096_Dropout_Epoch150

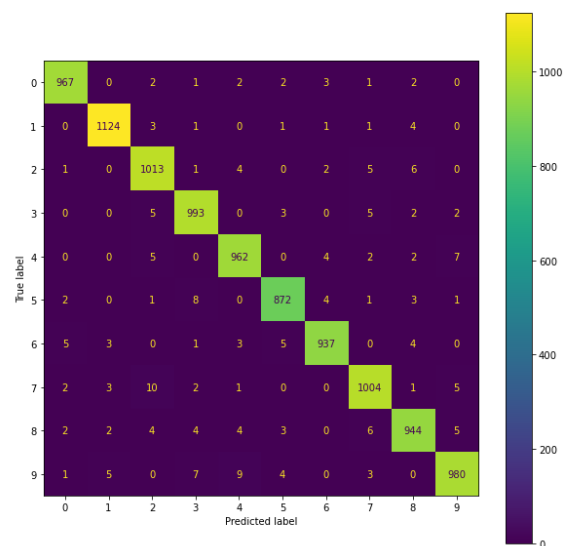


Figure 15: ARN-RAW-ConfMat-relu-softmax-Neur150_Batch4096_Dropout_Epoch150

Test score: 0.09491664171218872
Test accuracy: 0.9702000021934509

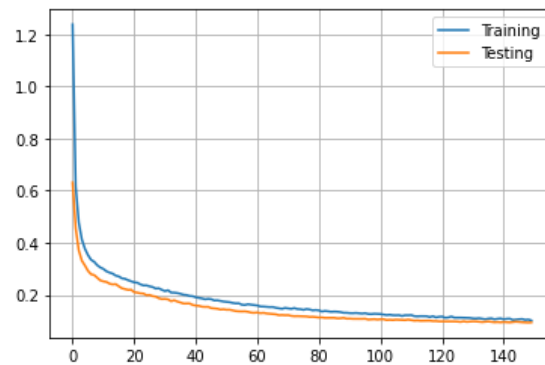


Figure 16: ARN-RAW-Plot-tanh-softmax-Neur150_Batch4096_Dropout_Epoch150

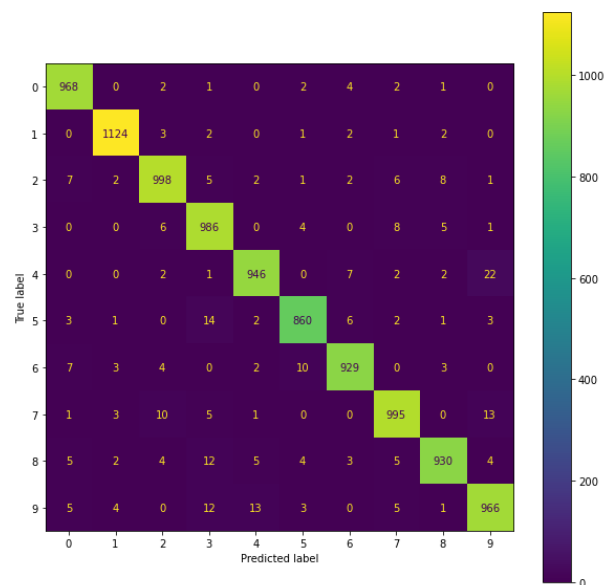


Figure 17: ARN-RAW-ConfMat-tanh-softmax-Neur150_Batch4096_Dropout_Epoch150

Test score: 0.08042246848344803
Test accuracy: 0.9757999777793884

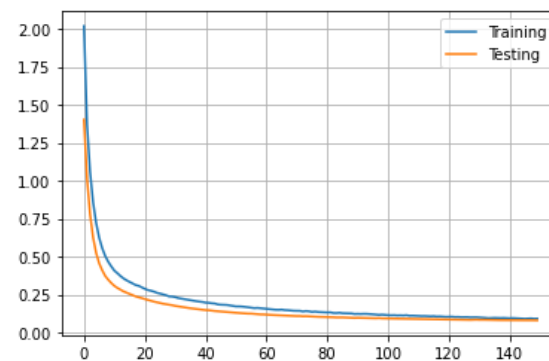


Figure 18: ARN-RAW-Plot-sigmoid-softmax-Neur150_Batch4096_Dropout_Epoch150

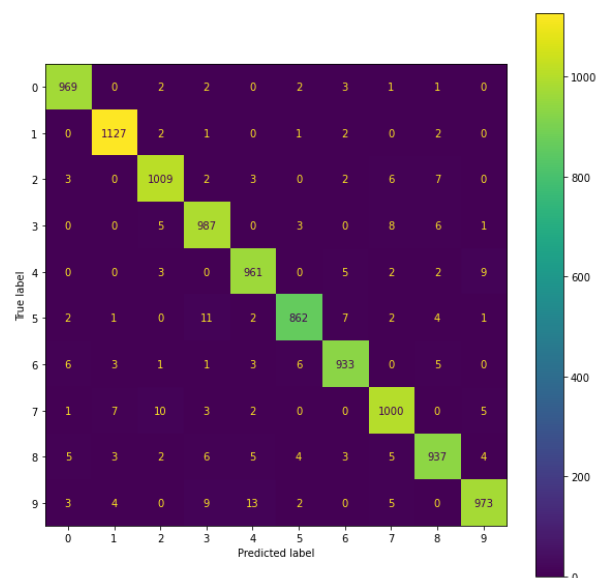


Figure 19: ARN-RAW-ConfMat-sigmoid-softmax-Neur150_Batch4096_Dropout_Epoch150

MLP_from_HOG.ipynb

Test score: 0.08003607392311096
Test accuracy: 0.9833999872207642

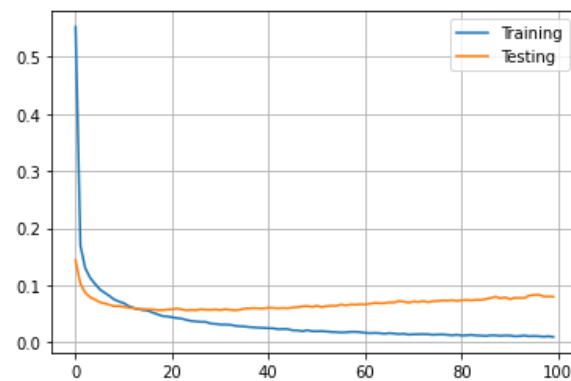


Figure 20: ARN-HOG-Plot-relu-softmax-Neur200_Batch512_Dropout_Epoch100

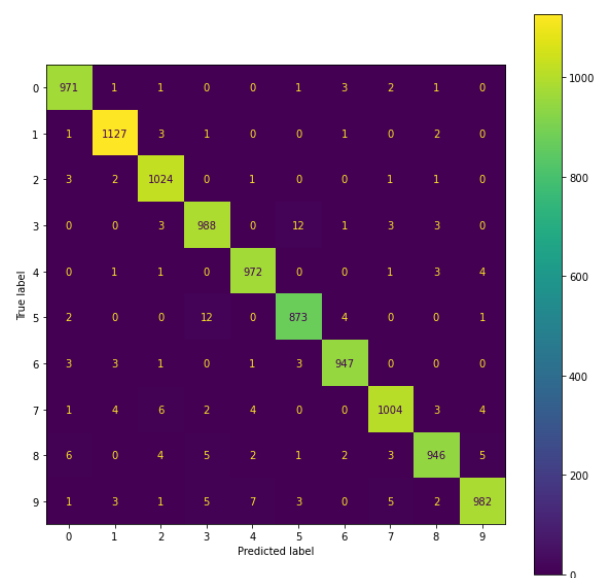


Figure 21: ARN-HOG-ConfMat-relu-softmax-Neur200_Batch512_Dropout_Epoch100

Test score: 0.08840184658765793
Test accuracy: 0.97560004196167

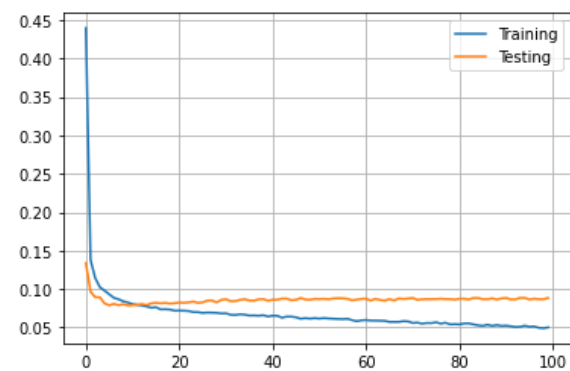


Figure 22: ARN-HOG-Plot-tanh-softmax-Neur200-Batch512_Dropout_Epoch100

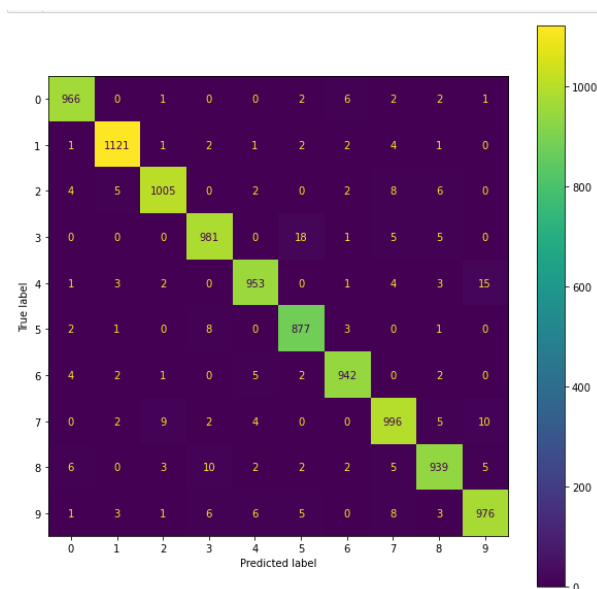


Figure 23: ARN-HOG-ConfMat-tanh-softmax-Neur200_Batch512_Dropout_Epoch100

Test score: 0.05404368415474892
Test accuracy: 0.9840999841690063

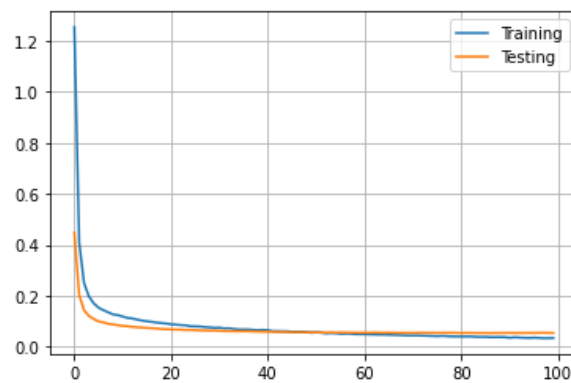


Figure 24: ARN-HOG-Plot-sigmoid-softmax-Neur200_Batch512_Dropout_Epoch100

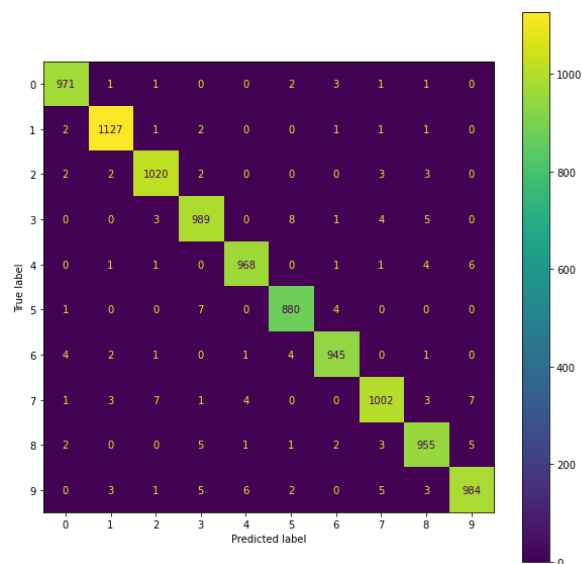


Figure 25: ARN-HOG-ConfMat-sigmoid-softmax-Neur200_Batch512_Dropout_Epoch100

Test score: 0.09604205936193466
Test accuracy: 0.9696999788284302

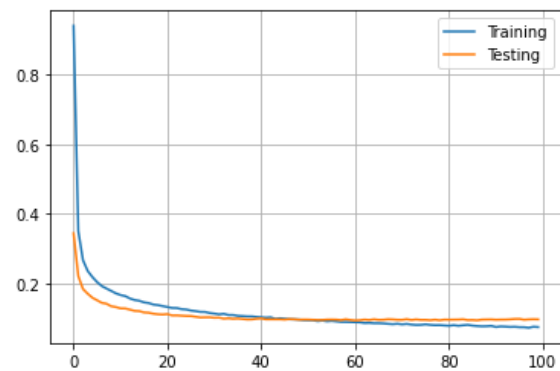


Figure 26: ARN-HOG-Plot-relu-Pixel7_Neur200_Batch512_Dropout_Epoch100

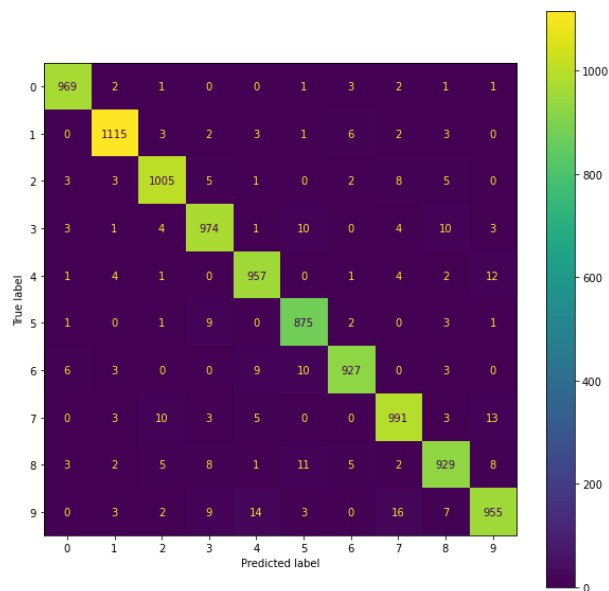


Figure 27: ARN-HOG-ConfMat-relu-Pixel7_Neur200_Batch512_Dropout_Epoch100

Test score: 0.13775815069675446
Test accuracy: 0.9538999795913696

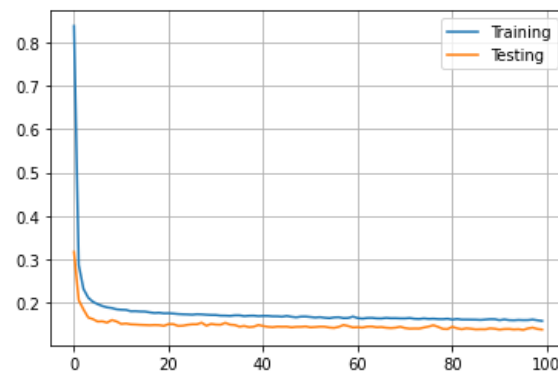


Figure 28: ARN-HOG-Plot-tanh-Pixel7_Neur200_Batch512_Dropout_Epoch100

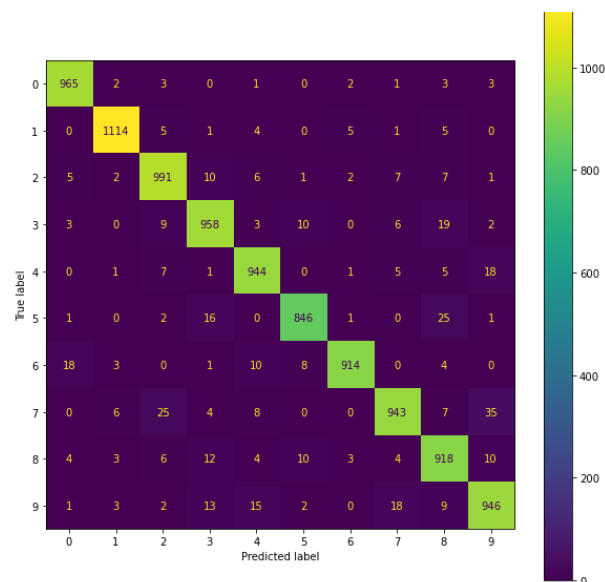


Figure 29: ARN-HOG-ConfMat-tanh-Pixel7_Neur200_Batch512_Dropout_Epoch100

Test score: 0.09379129111766815
Test accuracy: 0.9699000120162964

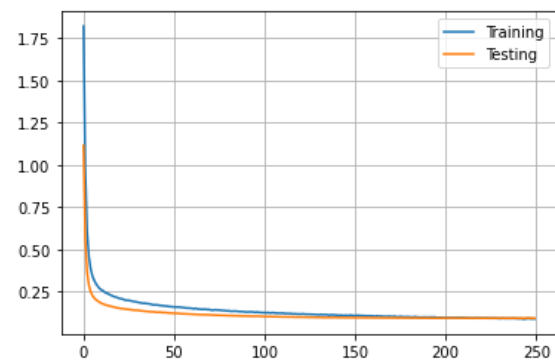


Figure 30: ARN-HOG-Plot-sigmoid-Pixel7_Neur200_Batch512_Dropout_Epoch250

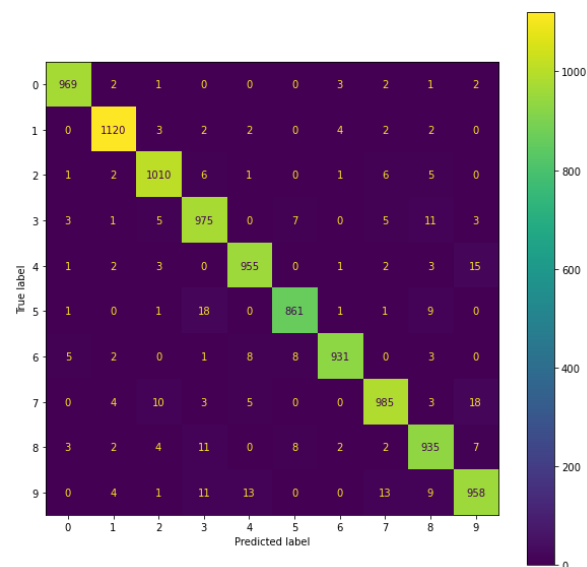


Figure 31: ARN-HOG-ConfMat-sigmoid-Pixel7_Neur200_Batch512_Dropout_Epoch250

Test score: 0.12630851566791534
Test accuracy: 0.9679999947547913

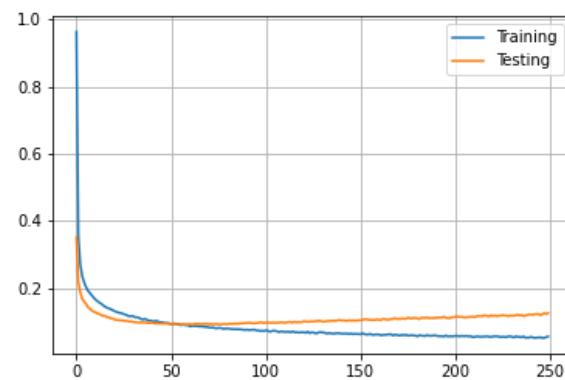


Figure 32: ARN-HOG-Plot-relu-Pixel7_Neur200_Batch512_Dropout_Epoch250

Test score: 0.10599838942289352
Test accuracy: 0.9700999855995178

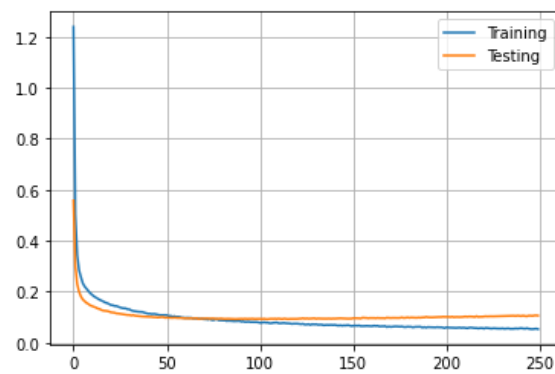


Figure 33: ARN-HOG-Plot-relu-Pixel7_Neur200_Batch1024_Dropout_Epoch250

Test score: 0.1260688304901123
Test accuracy: 0.957599975204468

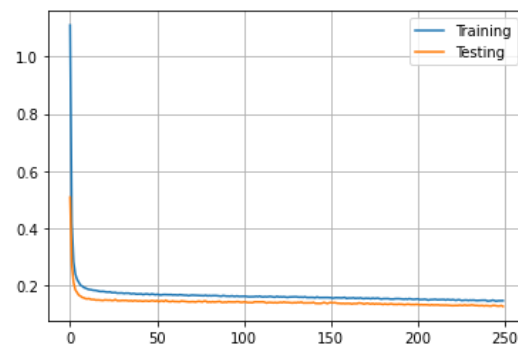


Figure 34: ARN-HOG-Plot-tanh-Pixel7_Neur200_Batch1024_Dropout_Epoch250

Test score: 0.0942075327038765
Test accuracy: 0.9684000015258789

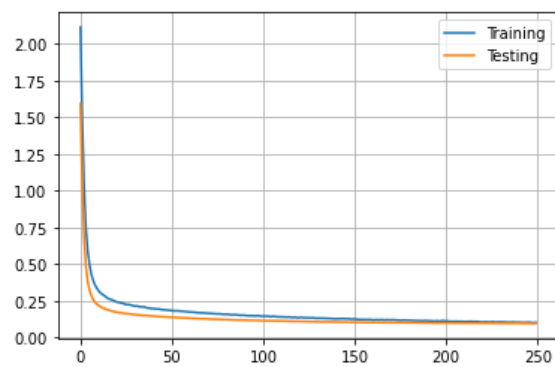


Figure 35: ARN-HOG-Plot-sigmoid-Pixel7_Neur200_Batch1024_Dropout_Epoch250

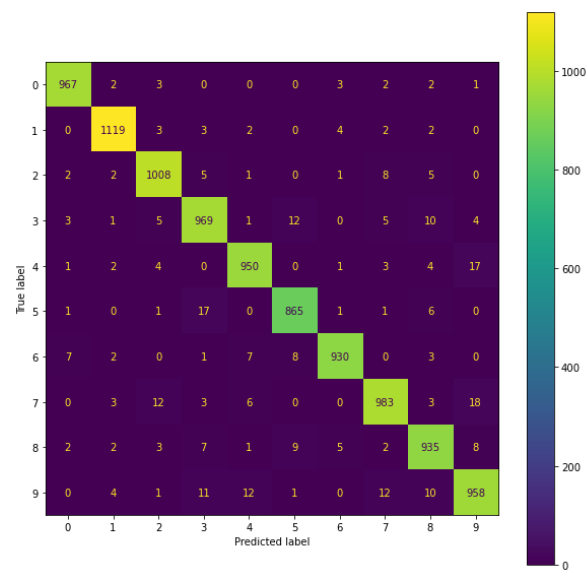


Figure 36: ARN-HOG-ConfMat-sigmoid-Pixel7_Neur200_Batch1024_Dropout_Epoch250

Test score: 0.0766327977180481
Test accuracy: 0.9750999808311462

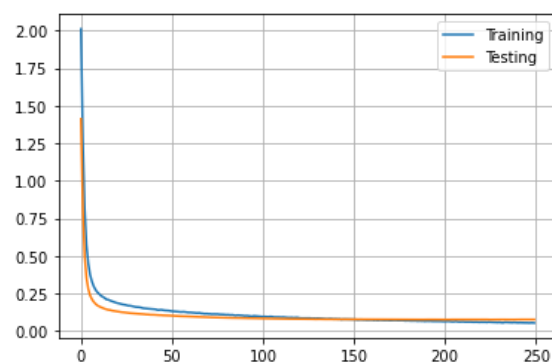


Figure 37: ARN-HOG-Plot-sigmoid-Pixel7_Or16_Neur200_Batch1024_Dropout_Epoch250

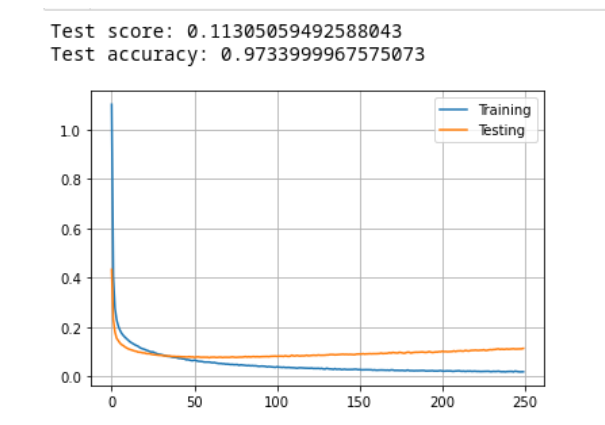


Figure 38: ARN-HOG-Plot-relu-Pixel7_Or16_Neur200_Batch1024_Dropout_Epoch250

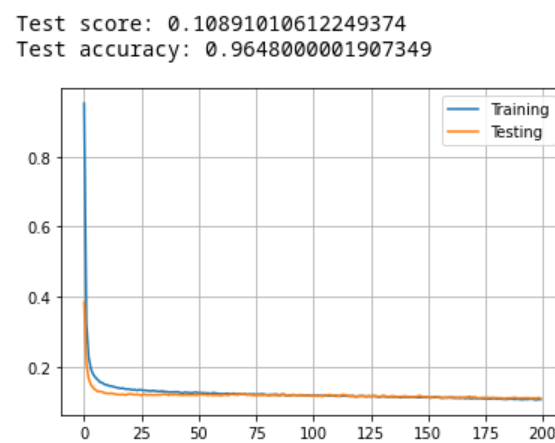


Figure 39: ARN-HOG-Plot-tanh-Pixel7_Or16_Neur200_Batch1024_Dropout_Epoch200

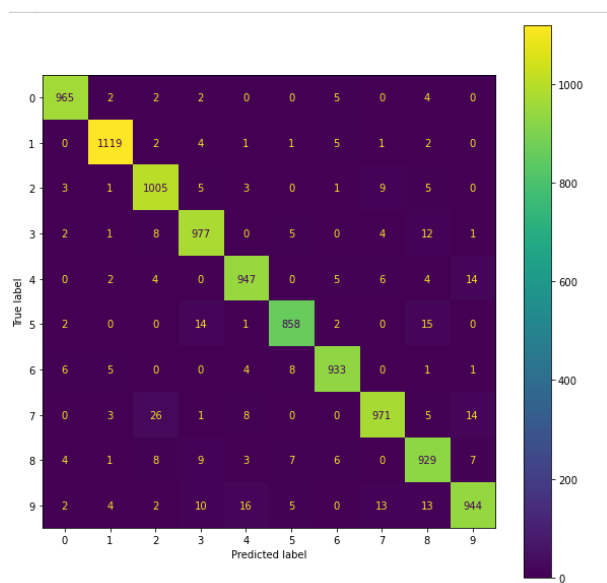


Figure 40: ARN-HOG-ConfMat-tanh-Pixel7_Or16_Neur200_Batch1024_Dropout_Epoch200

Test score: 0.09441999346017838
Test accuracy: 0.9700999855995178

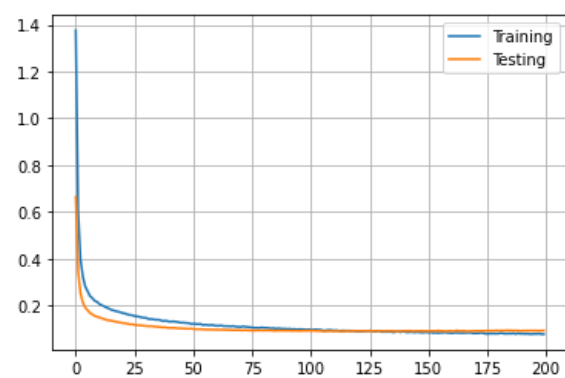


Figure 41: ARN-HOG-Plot-relu-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

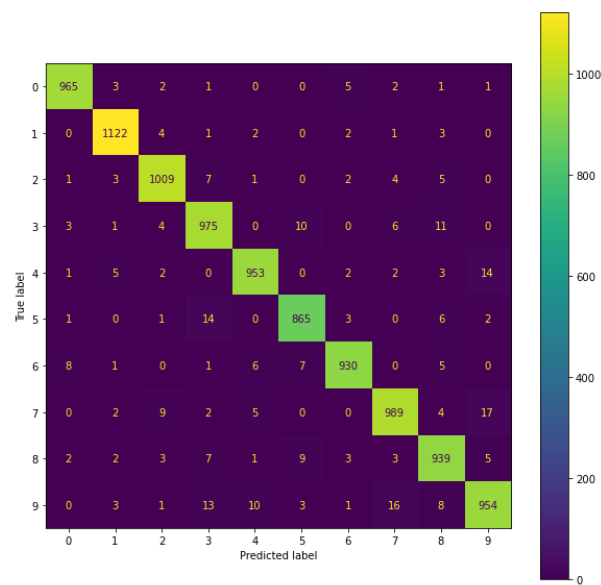


Figure 42: ARN-HOG-ConfMat-relu-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

Test score: 0.1024104580283165
 Test accuracy: 0.9660999774932861

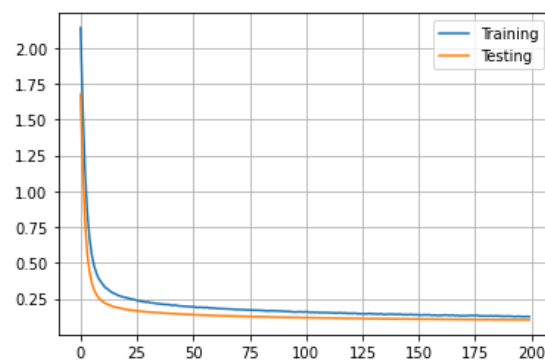


Figure 43: ARN-HOG-Plot-sigmoid-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

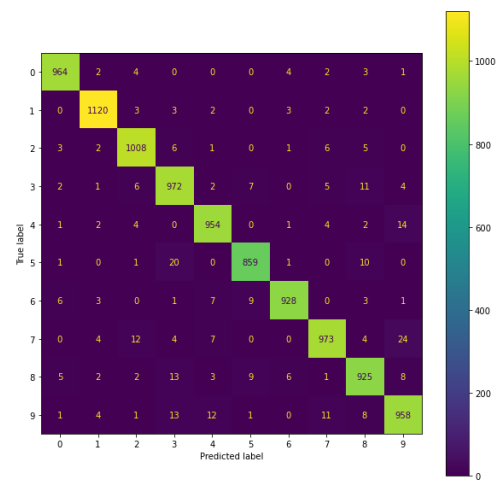


Figure 44: ARN-HOG-ConfMat-sigmoid-Pixel7_Or8_Neur150_Batch1024_Dropout_Epoch200

Test score: 0.1345009058713913
Test accuracy: 0.9556000232696533

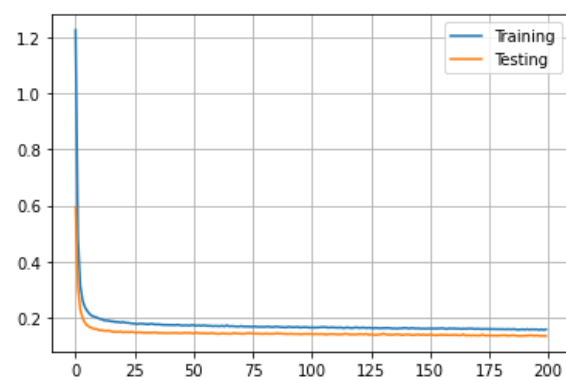


Figure 45: ARN-HOG-Plot-tanh-Pixel7_Or8_Neur150-Batch1024_Dropout_Epoch200

Test score: 0.10055689513683319
Test accuracy: 0.9699000120162964

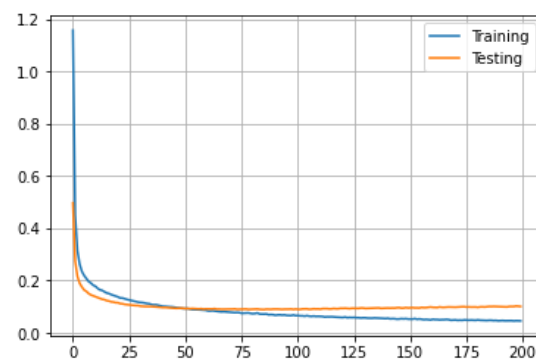


Figure 46: ARN-HOG-Plot-relu-Pixel7_Or8_Neur250_Batch1024_Dropout_Epoch200

Test score: 0.09292542189359665
Test accuracy: 0.9690999984741211

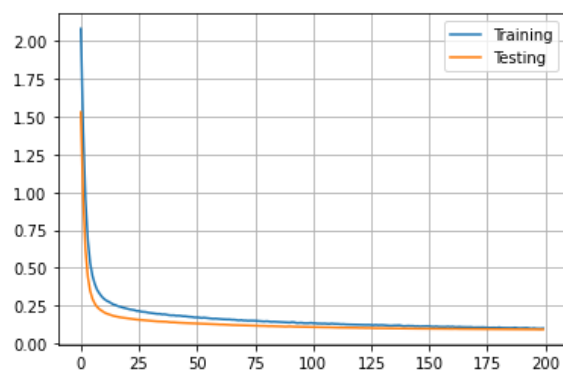


Figure 47: ARN-HOG-Plot-sigmoid-Pixel7_Or8_Neur250_Batch1024_Dropout_Epoch200

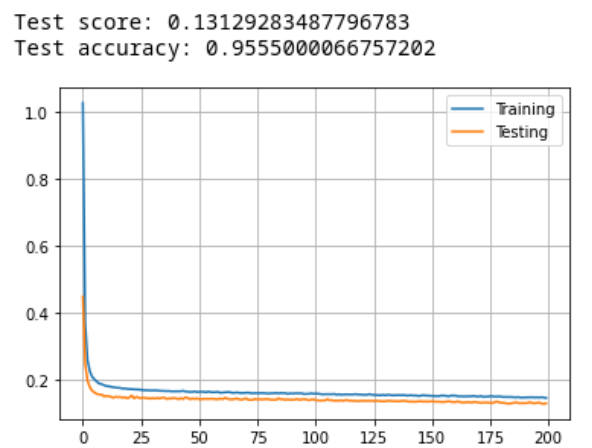


Figure 48: ARN-HOG-Plot-tanh-Pixel7_Or8_Neur250_Batch1024_Dropout_Epoch200

CNN.ipynb

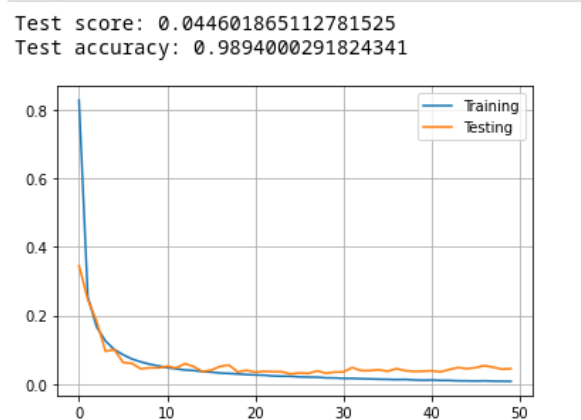
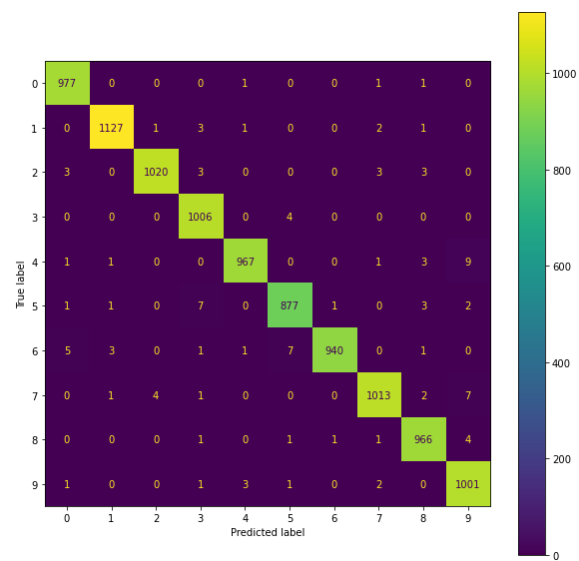
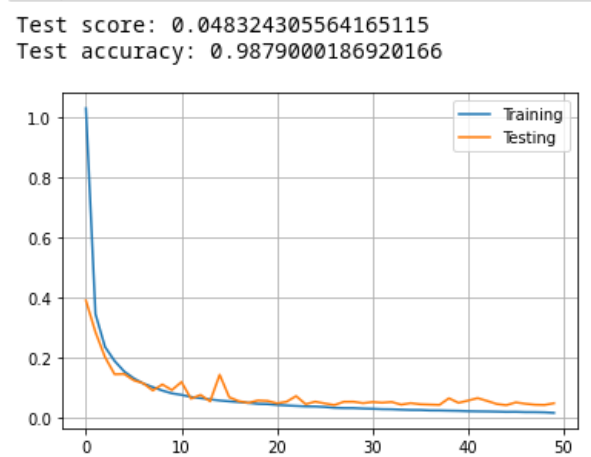
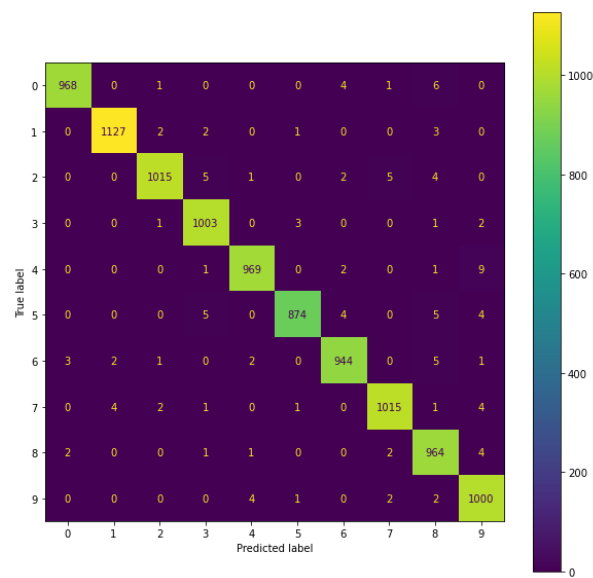
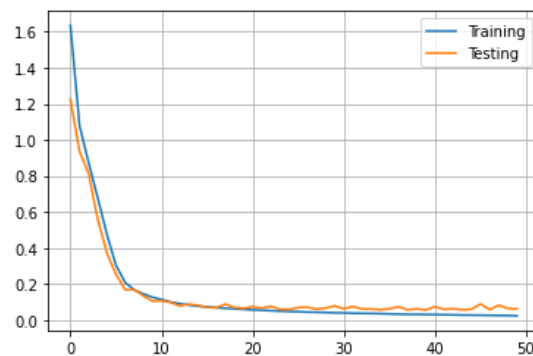


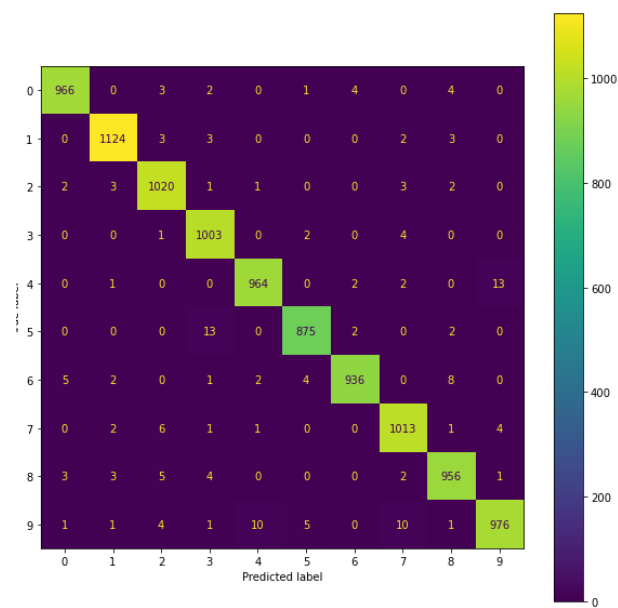
Figure 49: ARN-CNN-Plot-relu-Batch256_25L4_Epoch50

**Figure 50:** ARN-CNN-ConfMat-relu-Batch256_25L4_Epoch50**Figure 51:** ARN-CNN-Plot-relu-Batch256_10L4_Epoch50

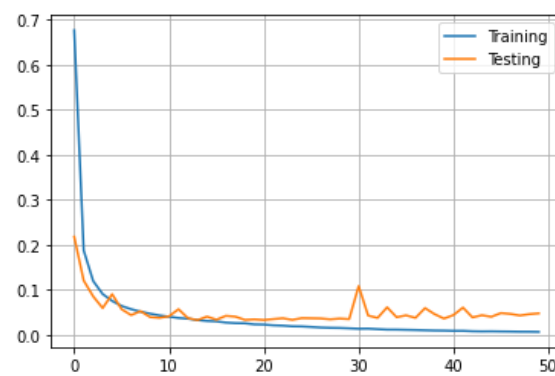
**Figure 52:** ARN-CNN-ConfMat-relu-Batch256_10L4_Epoch50

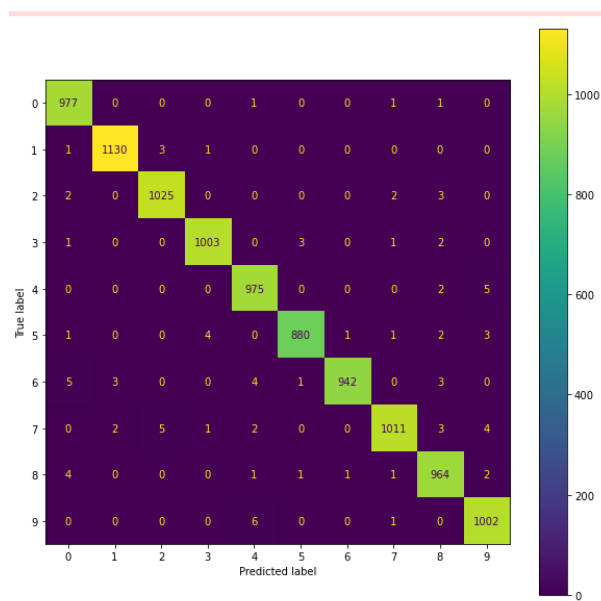
Test score: 0.06122294440865517
Test accuracy: 0.983299970626831

**Figure 53:** ARN-CNN-Plot-relu-Batch256_5L4_Epoch50

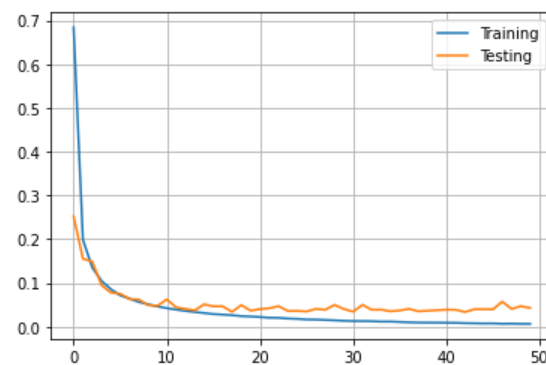
**Figure 54:** ARN-CNN-ConfMat-relu-Batch256_5L4_Epoch50

Test score: 0.0466320738196373
Test accuracy: 0.9908999800682068

**Figure 55:** ARN-CNN-Plot-relu-Batch256_35L4_Epoch50

**Figure 56:** ARN-CNN-ConfMat-relu-Batch256_35L4_Epoch50

Test score: 0.04271617904305458
Test accuracy: 0.9890000224113464

**Figure 57:** ARN-CNN-Plot-tanh-Batch256_25L4_Epoch50

Test score: 0.036155108362436295
Test accuracy: 0.9898999929428101

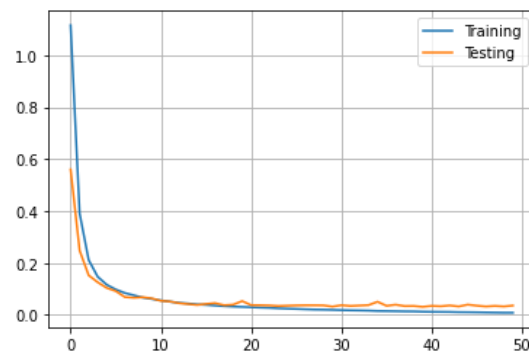


Figure 58: ARN-CNN-Plot-sigmoid-Batch256_25L4_Epoch50

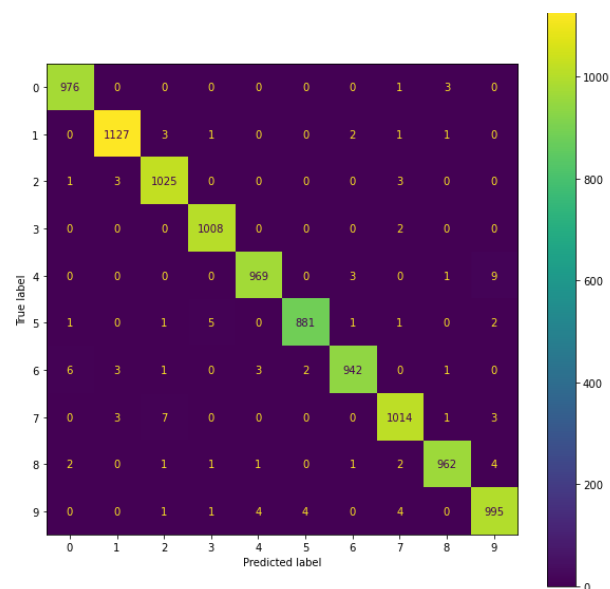


Figure 59: ARN-CNN-ConfMat-sigmoid-Batch256_25L4_Epoch50