

Questions

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 loss function is being used ?
 - Please, give the equation(s)
2. For each experiment excepted the last one (shallow network learning from raw data, shallow network learning from features and CNN):
 - 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.
 - Test 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?
3. The CNNs models are deeper (have more layers), do they have more weights than the shallow ones? explain with one example.
4. Train a CNN for the chest x-ray pneumonia recognition. In order to do so, complete the code to reproduce the architecture plotted in the notebook. Present the confusion matrix, accuracy and F1-score of the validation and test datasets and discuss your results.

ARN - Practical work 4

Group information:

- First names1 Last names1 **Rodrigo LOPEZ DOS SANTOS**
- First names2 Last names2 **Urs BEHRMANN**

Learning algorithm

We will use the same learning algorithm for all the experiments: RMSprop. We will also use the same parameters for all the experiments.

Optimizer:	RMSprop
learning_rate:	0.001
momentum:	0.1
loss:	categorical_crossentropy
batch_size:	128
n_epoch:	50

The equation for the loss function is:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) \left(\frac{\delta C}{\delta w} \right)^2$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t}} \frac{\delta C}{\delta w}$$

$E[g]$ — moving average of squared gradients. dC/dw — gradient of the cost function with respect to the weight.
 η — learning rate. β — moving average parameter (good default value — 0.9)

How the number of weights is calculated

The number of weights between two layers is calculated by multiplying the number of neurons in the first layer by the number of neurons in the second layer. We also add the number of biases in the second layer.

$$P = \sum_{l=1}^{l-1} (N_l \times N_{l+1}) + N_{l+1}$$

Example

The input layer has 784 neurons and the output layer has 10 neurons.

If we have a hidden layer with 16 neurons, the number of weights is calculated as follows:

Layer 1: $784 \times 16 + 16 = 12560$

Layer 2: $16 \times 10 + 10 = 170$

Total: $12560 + 170 = 12730$

MLP from raw data

We will test three different models with different number of neurons in the hidden layers.

For the first model, we chose a low number of neurons in the hidden layers to see how the model would perform with insufficient capacity. We hope to see the highest error rate in this model.

For the second model, we chose a number of neurons in the hidden layers that is higher than the number of neurons in the input layer. We hope to see a better performance than the first model.

For the third model, we chose a high number of neurons in the hidden layers to see how the model would perform with a high capacity. We hope to see the lowest error rate in this model, but we also expect to see overfitting.

We have the same number of inputs and outputs for all models. The input layer has 784 neurons and the output layer has 10 neurons.

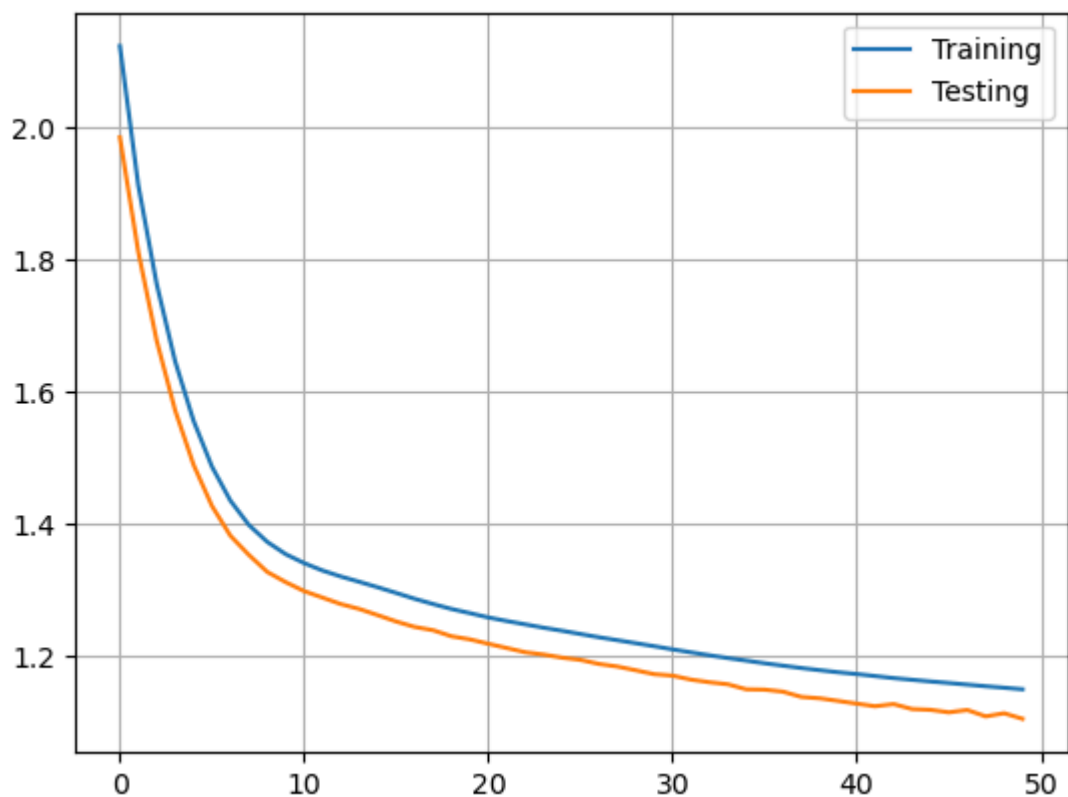
First model

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	1,570
dense_74 (Dense)	(None, 10)	30

Total params:	1,600 (6.25 KB)
Trainable params:	1,600 (6.25 KB)
Non-trainable params:	0 (0.00 B)

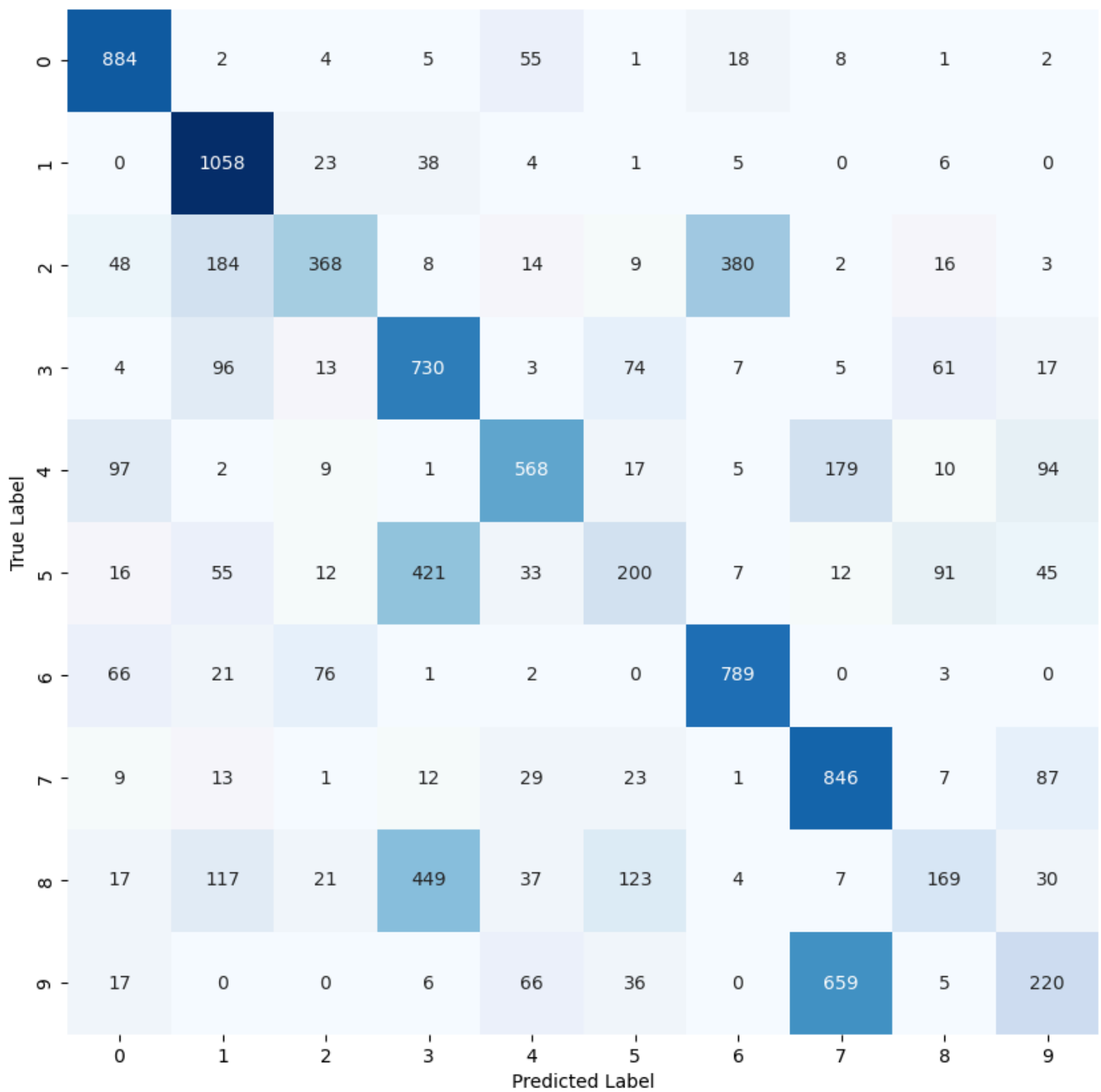
Performance

Test score:	1.1609300374984741
Test accuracy:	0.5831999778747559



Confusion matrix

F1 macro Score:	0.5442219669271385
F1 weighted Score:	0.5492755535704517
F1 micro Score:	0.5832



Second model

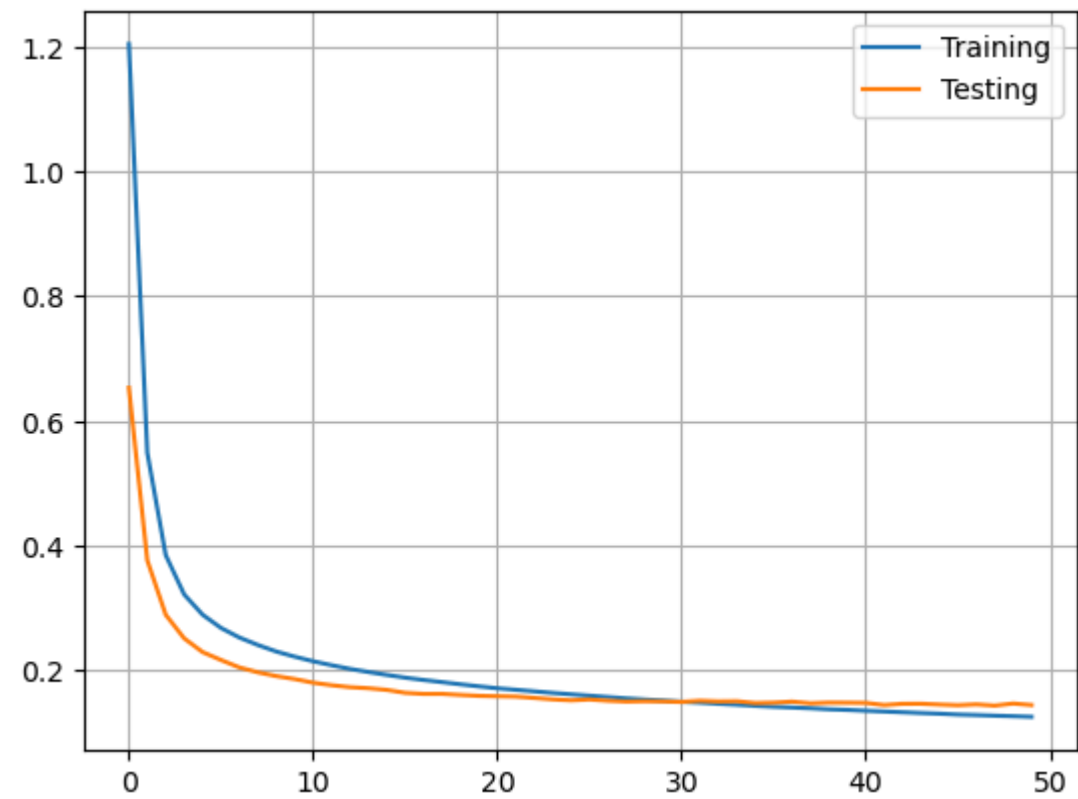
Layer (type)	Output Shape	Param #
dense_72 (Dense)	(None, 16)	12,560
dense_74 (Dense)	(None, 10)	170

Total params:	12,730 (49.73 KB)
---------------	-------------------

Trainable params:	12,730 (49.73 KB)
Non-trainable params:	0 (0.00 B)

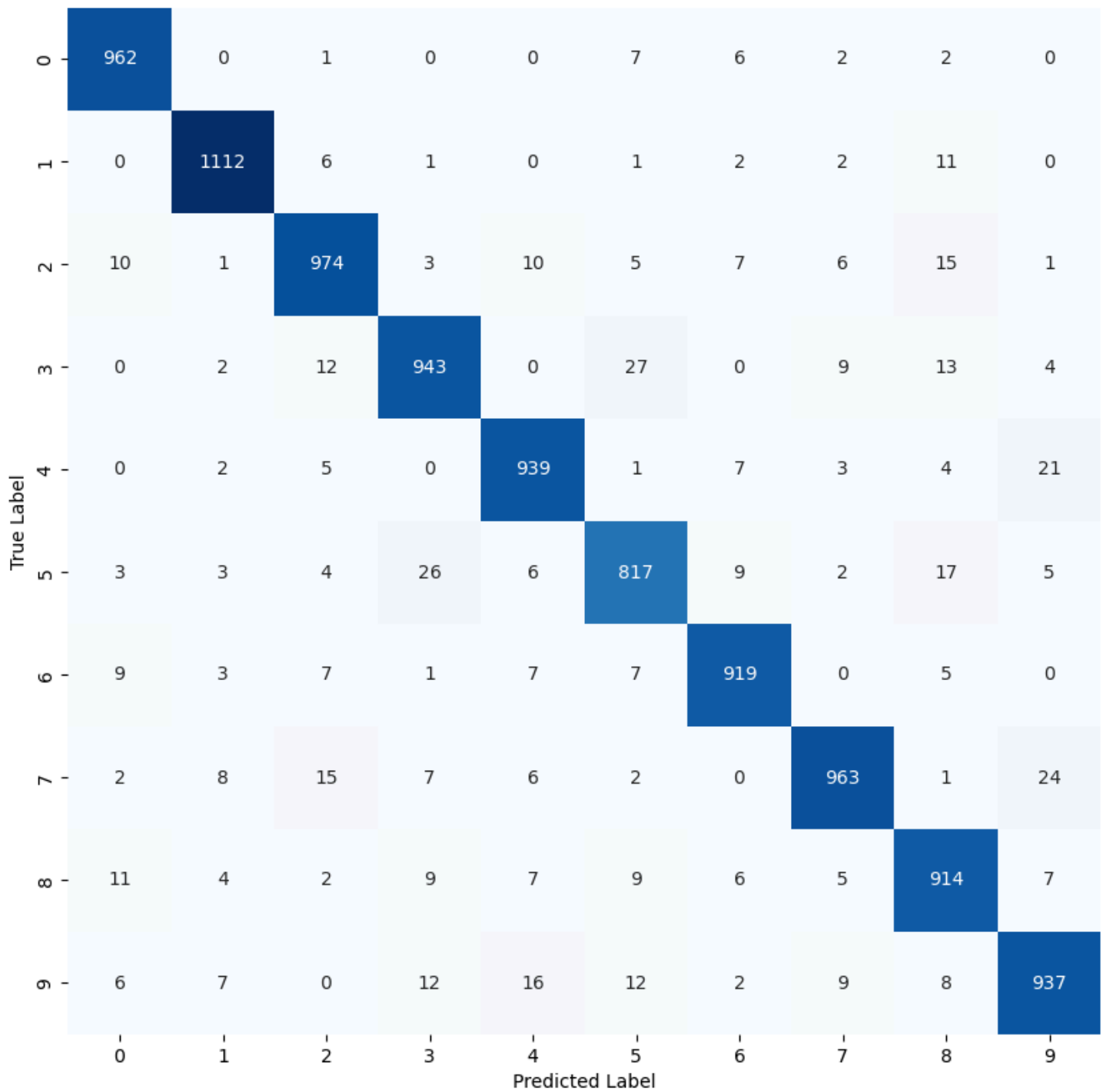
Performance

Test score:	0.17095278203487396
Test accuracy:	0.9480000138282776



Confusion matrix

F1 macro Score:	0.9473270917753236
F1 weighted Score:	0.9479600401672442
F1 micro Score:	0.948



Third model

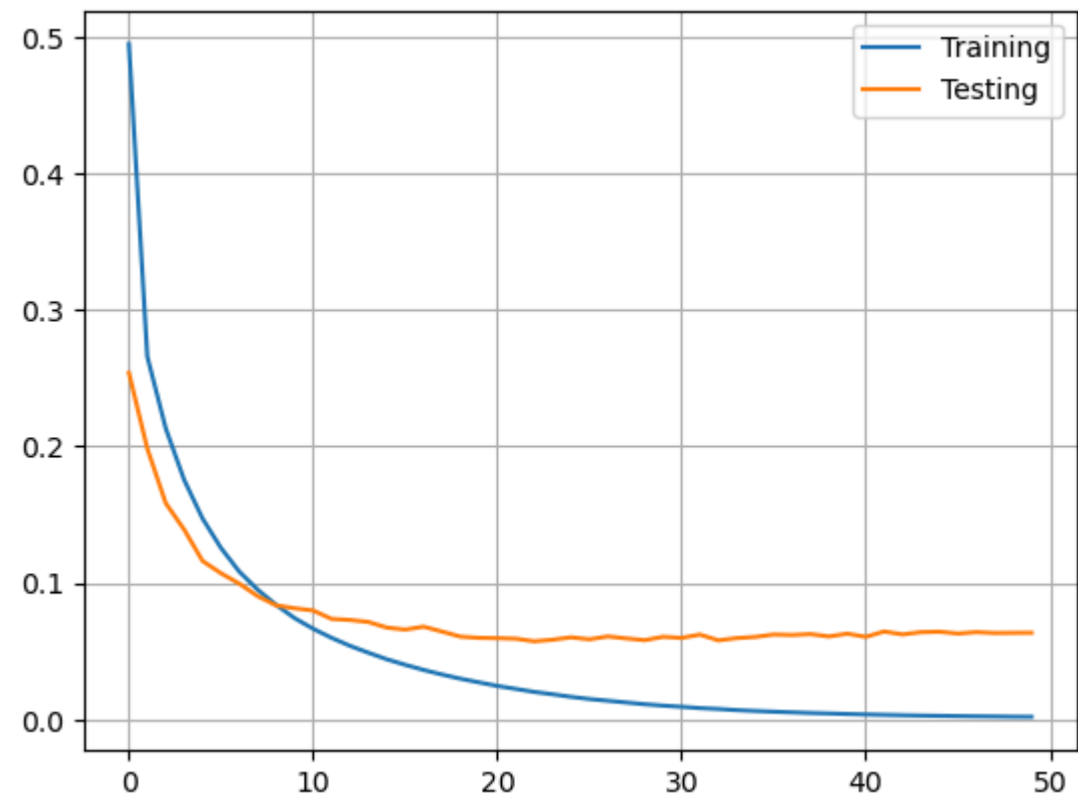
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 254)	199,390
dense (Dense)	(None, 10)	2,550

Total params:	201,940 (788.83 KB)
---------------	---------------------

Trainable params:	201,940 (788.83 KB)
Non-trainable params:	0 (0.00 B)

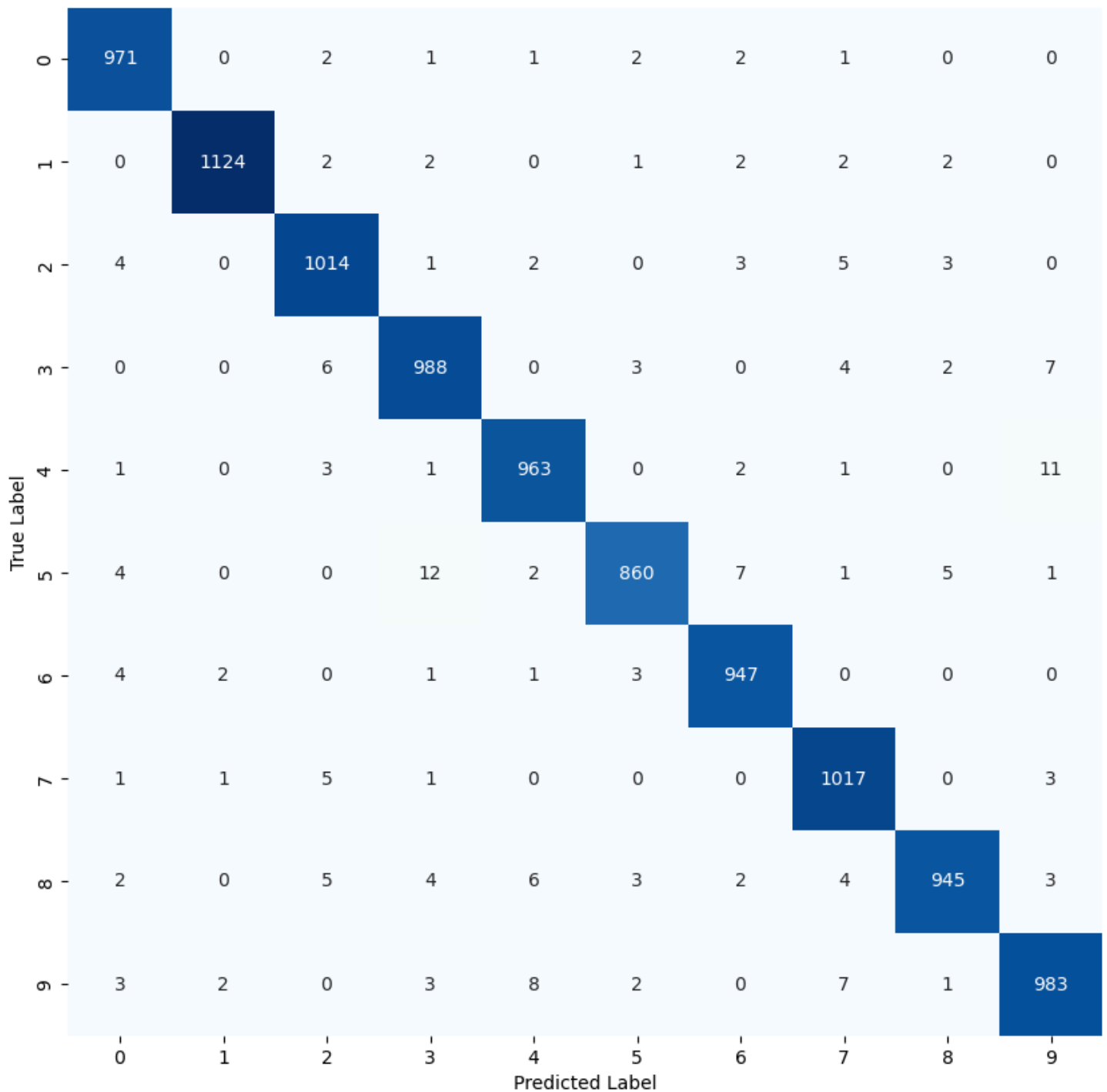
Performance

Test score:	0.06881905347108841
Test accuracy:	0.9811999797821045



Confusion matrix

F1 macro Score:	0.9809796904590385
F1 weighted Score:	0.9811910297485315
F1 micro Score:	0.9812



Conclusion of the first exercise

First model has the highest error rate, as expected, because it has the lowest capacity.

The second model has a better performance than the first model, but it is still not as good as the third model.

The third model has the lowest error rate, but it is also the model with the highest capacity. We can see in the performance plots that the third model is overfitting the training data.

MLP form HOG


First model

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	1,570
dense_74 (Dense)	(None, 10)	30

Total params:	1,600 (6.25 KB)
Trainable params:	1,600 (6.25 KB)
Non-trainable params:	0 (0.00 B)

Performance

Test score:	1.1609300374984741
Test accuracy:	0.5831999778747559

Performance first model

Confusion matrix

F1 macro Score:	0.5442219669271385
F1 weighted Score:	0.5492755535704517
F1 micro Score:	0.5832

Confusion matrix 1


Second model

Layer (type)	Output Shape	Param #
dense_72 (Dense)	(None, 16)	12,560
dense_74 (Dense)	(None, 10)	170

Total params:	12,730 (49.73 KB)
Trainable params:	12,730 (49.73 KB)
Non-trainable params:	0 (0.00 B)

Performance

Test score:	0.17095278203487396
Test accuracy:	0.9480000138282776

Performance second model

Confusion matrix

F1 macro Score:	0.9473270917753236
F1 weighted Score:	0.9479600401672442
F1 micro Score:	0.948

Confusion matrix 2

Third model

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 254)	199,390
dense (Dense)	(None, 10)	2,550

Total params:	201,940 (788.83 KB)
Trainable params:	201,940 (788.83 KB)
Non-trainable params:	0 (0.00 B)

Performance

Test score:	0.06881905347108841
Test accuracy:	0.9811999797821045

Performance third model

Confusion matrix

F1 macro Score:	0.9809796904590385
F1 weighted Score:	0.9811910297485315
F1 micro Score:	0.9812

Confusion matrix 3