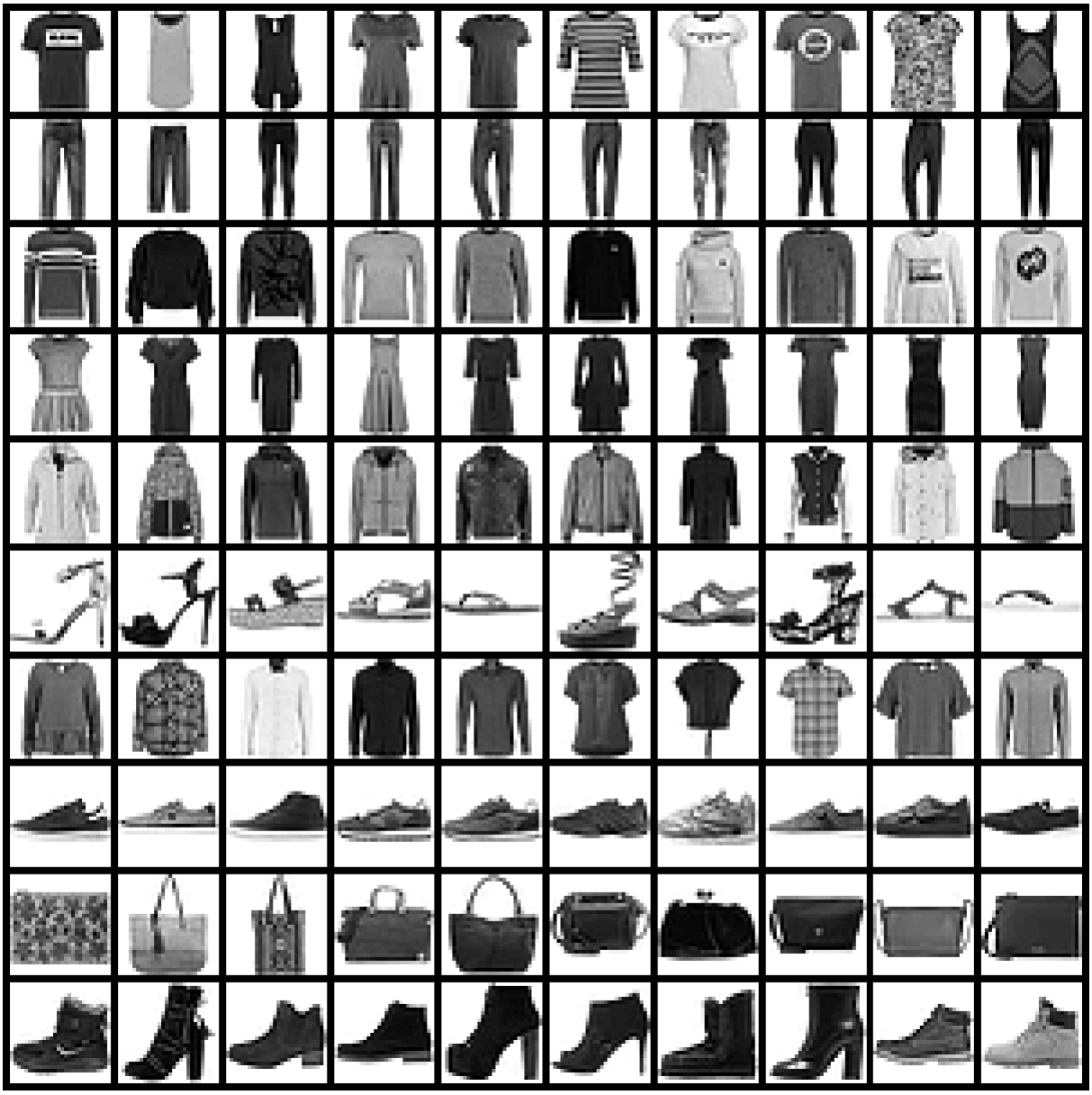
Pattern Recognition Laboratory – Sessions #5 & #6
Fashion items classification with neural networks

Due date: **19.12.2024 (A – LAB/101), 16.01.2025 (B – LAB/102), 8.01.2024 (C – LAB/103)**

In this assignment, we will use the fashion-MNIST collection –images of fashion items of 10 types prepared by Zalando company in a format exactly the same as the handwritten digits that you classified in the previous assignment. More information about this dataset can be found in the GitHub repository (https://github.com/zalandoresearch/fashion-mnist). The dataset is also available on the Galera server (http://galera.ii.pw.edu.pl/~rkz/epart/fashion-mnist.zip).

Ten classes which you'll have to classify are: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.



The reference solution with which you can compare the performance of your solution is a neural network with one hidden layer (100 neurons) trained incrementally (i.e. with weights' update after each sample) with a constant learning rate of 0.001 for 50 epochs.

	fashion-MNIST train set			fashion-MNIST test set		
	OK.	Error	Rejection	OK.	Error	Rejection
Classification coefficients	90.38%	9.62%	0.00%	87.42%	12.58%	0.00%

The results of this trial are not very impressive, so there is a lot of room for improvement. In the reference solution I adopted the simplest way to determine the classification result: max of the network outputs - hence no rejections. You can consider introducing additional conditions when making decisions (to transfer some errors into a rejection basket), but this is not a necessary element. It is important to use the same decision making schema in the reference and refined solution.

Let's assume that we will stick to the classification by fully connected multilayer neural networks trained with a backpropagation algorithm. The two basic elements of this assignment are:

1. Preparation of a reference implementation of network learning with backward error propagation (better quality than the reference classifier above is needed to achieve max grade).

   The implementation of this point will probably require you to experiment a bit with network architecture (number of layers, number of neurons in layers) and learning parameters (learning constant value and/or change during training).

2. In the second phase you'll have to experiment with various elements of training (and sometimes also classification) procedure. Possible modifications include, but are not limited to:

   a. Normalization of the data sets (this can be implemented as standard mean removal and scaling of the features by reciprocal of the standard deviation or via the PCA transformation).

   b. Including inertia (aka momentum) during the training.

   c. Updating weights only after processing several samples (a small batch of say 8 or 16, or 32 samples).

   d. Implementing dropout in layer(s) of network.

You should select just two modifications from the above list and check their influence on the training process and final classification result. To make results comparable you should pay attention to start from the same point (i.e. you should use exactly the same initial weights).

Included in the package are several .m files containing implementation (sometimes partial) of the steps required to train and test the network:

actf.m	-	neuron activation function
actdf.m	-	derivative of neuron activation function
crann.m	-	creation of output and hidden layers weight matrices (with random initialization)
anncls.m	-	neural network classifier
ann_training.m	-	neural network training function (reads data set, creates neural network and contain training loop currently for fixed number of epochs)

backprop.m	-	backpropagation function intended to perform one epoch of training (comments suggest stochastic training)
confMx.m	-	computes confusion matrix
compErrors.m	-	computes recognition, error and rejection coefficients
readSets.m	-	reads fashion mnist data set
readmnist.m	-	reads image and label data in mnist format

Red colour marks files which have to be completed properly by you.

Your report should include:

1. Short description of the learning method, esp. modifications of the reference solution.
2. Data concerning epoch count and training times.
3. Confusion matrices for reference and modified solution (you should select more interesting one of the two you'll have at the end) and analysis of differences between them.
4. As usual you should send not only the report but also the code you used in this assignment. This time you should pay attention to repeatability of the results: you should store random number generator parameters and send the vector of these parameters together with the report and code. I should be able to train exactly the same network(s) as you.

   To get the random number generator state use commands:

   ```
   rand();
   rstate = rand("state");
   ```

   and save it with:

   ```
   save rnd_state.txt rstate
   ```

You should upload your report and code in one zip archive in LeOn in task FashioNet mini-project (in the Lab & Projects section).