Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science

WI4635 Linear Algebra and Optimization for Machine Learning 2025/2026
Project 2 – Neural Networks

Deadline: 23:59, January 12, 2026

Instructions and Assessment Criteria

- In this project, you will explore the power and some drawbacks of dense feedforward neural networks (multilayer perceptrons).
- Please hand in a Jupyter notebook with the Python or Julia code and brief explanations or discussions for each subquestion. For the code you should not use high-level machine learning libraries; instead implement your own functions that, for instance, use NumPy. Using a Python automatic differentiation library such as JAX is allowed and recommended to simplify gradient computations. Make sure the notebook runs end-to-end without us needing to modify anything.
- If you prefer to use an alternative notebook format or another open-source programming language, please discuss this with us. Submit the work via Brightspace.
- Please also prepare a short presentation (10 min) where you discuss the experiments you performed and which results you found particularly interesting. The presentations/interviews will be on January 14 and 15, 2026.
- You may use generative AI, but if you do, please make sure that you check (and discuss among each other) the AI-generated material. Make sure that you fully agree with and understand everything you hand in; this will also be tested in the interview (10 min) following the presentation. There will be an announcement on Brightspace to schedule the time slot for the presentation and interview.

1. Implement a dense feedforward neural network from scratch. Make your implementation flexible with respect to the input and output dimensions, the number of hidden layers and neurons per hidden layer, and the activation functions used. Use this implementation for the following two questions. Choose a suitable initialization for the network parameters.

2. Consider a standard benchmark dataset for classification: train a neural network to classify handwritten digits into the ten classes $0, 1, \ldots, 9$. As input for your model, use flattened vector representations of the *MNIST* images.[1]

   (a) For this multiclass classification task, train your neural network with cross-entropy loss and mini-batch gradient descent. Vary the neural network architecture (layers, neurons per layer, activation functions) and training hyperparameters (learning rate, batch size, epochs). Use grid search with k-fold cross-validation (e.g., $k = 5$) to select promising hyperparameters. Report the accuracy and learning curves for the best model.

   (b) Study how optimizer hyperparameters (batch size, learning rate) affect convergence speed and final performance, and discuss your observations.

   (c) Identify and visualize misclassified images for your best model, and provide possible explanations.

   Hint*: You can download MNIST in Python via fetch_ openml("mnist_ 784") from sklearn . datasets.*

3. Next, test the performance of your neural network implementation on a simple regression problem. Fit functions of the form

$$f_k(x) := \sin(k\pi x), \quad x \in [-1, 1],$$

   with $k \in \mathbb{N}$ using your neural network implementation. Use mean squared error (MSE) as the loss. Generate a sufficient number of training samples yourself, balancing approximation quality and computational cost.

   (a) Perform a grid search to find the best hyperparameters for fitting f_1, varying at least the parameters that you also tested in Question 2. Discuss your findings.

   (b) Try to fit f_k for different values of k. How do you need to change the hyperparameters when increasing k? Discuss your findings. How far can you increase k?

   (c) Finally, propose an approach to approximate f_k with higher values of k and test it. Examples could include different function spaces than neural networks, architectural tweaks, activation changes, initialization strategies, or training approaches. Motivate your choice and discuss your findings.

[1]MNIST: `http://yann.lecun.com/exdb/mnist/`