Deep Learning 2025 Assignment 1

This is a **group assignment**, and its deadline is **Friday, Nov 28, 2025, 22:00**. You must submit your solution electronically via the Absalon home page.

Rules:

- For individual assignments the solution you hand in should be made and written by you. You are not allowed to share your solution with other students.
- The same is true on a group level for group assignments.
- We encourage learning by any means using all possible sources (peers, TAs, teachers, generative AIs, online media, books, etc.). We encourage you to use the exercise classes and the Absalon forum to get help. The exercise sessions exist to help you with the assignments, and you are welcome to ask any questions related to the teaching material and the assignments on the forum.
- You should follow normal scientific citation practice both for course material (textbook, Absalon, assignment text) and if you use methods or notation outside the course material.
- If you are in doubt about plagiarism or citation rules, please ask the teachers or TAs.

Please be very observant of these rules. We do not want any plagiarism cases, both for your and our sake.

Requirements for the Assignment hand-in:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your complete source code in the PDF file except if you are asked to do so. However, you can show important parts (i.e., a few lines) of your source code.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF.
- *Important: Do not zip the PDF file*, since zipped files cannot be opened in the speed grader. Zipped pdf submissions will not be graded.
- Your PDF report should be self-sufficient. That is, it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Your code should include a README text file describing how to compile and run your program, as well as a list of all relevant libraries needed for compiling or using your code.
- Handwritten solutions will not be accepted, please use the provided LaTeX template to write your report.
- Figures should be readable. They should have proper captions (same holds for tables) and labels, a legend when appropriate. In general, write proper sentences in the report.
- You are only asked to perform tasks that make sense from a machine learning point of view. If you do something that that does not make much sense, it is most likely wrong.

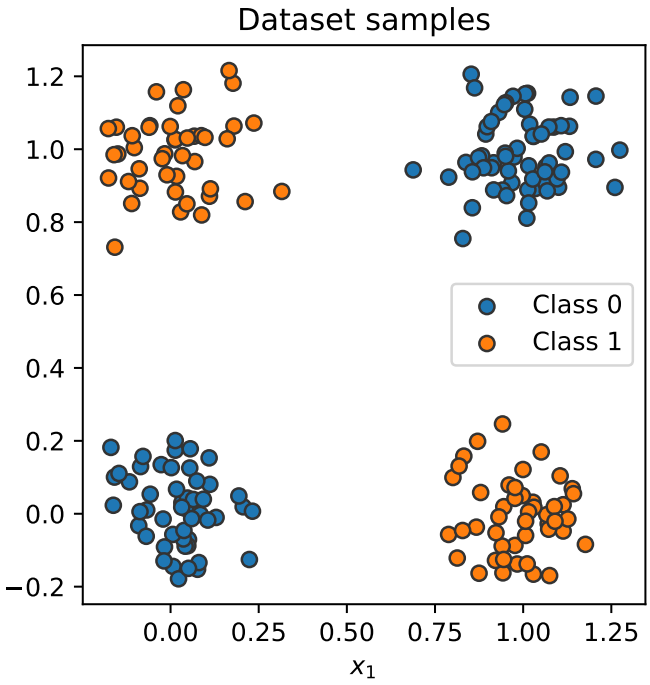

Figure 1: An example of the noisy-xor dataset with standard deviation $s = 0.1$.

1 Backpropagation pen and paper

In this part of the assignment we will work with the notebook `DL_Backpropagation_pen_and_paper.ipynb`. Your task is to:

1. Answer the exercises in the notebook. Only exercise c) and j) should be included in the PDF file.

2 AutoDiff framework

In this part of the assignment we will work with the notebook `DL_AutoDiff_Nanograd.ipynb`. Your task is to:

1. Answer the exercises in the notebook. Only exercise b) and l) should be included in the PDF file.

3 Deep versus wide networks

In this part of the assignment, you are going to work with feed-forward neural networks, and you are going to study the effect of widening and deepening networks on a non-linear classification task. You will work with the noisy-xor problem with data points defined as

$$(x_1, x_2, y) = (m_1 + s\eta_1, m_2 + s\eta_2, m_1 \text{ xor } m_2) \ ,$$

where $(m_1, m_2) = \{(0,0), (0,1), (1,0), (1,1)\}$ are the four cluster centers (chosen with equal probability), s is a user-defined standard deviation and η_1 and η_2 are normally distributed zero mean and unit variance random variables. An example of this dataset is shown in Figure 1.

We define the decision boundaries for binary classification as the curves in input space that the model predict to have 50-50 percent probability for each class: $p(y = 1|x_1, x_2) = p(y = 0|x_1, x_2) = 1 - p(y = 1|x_1, x_2) = 0.5$. A linear model $p(y = 1|x_1, x_2) = \sigma(w_1x_1 + w_2x_2 + w_0)$, where $\sigma(z) = 1/(1 + \exp -z)$ is the logistic function has single linear decision boundary. The equation that determines the position of the decision boundary is $w_1x_1 + w_2x_2 + w_0 = 0$ since $\sigma(0) = 0.5$.

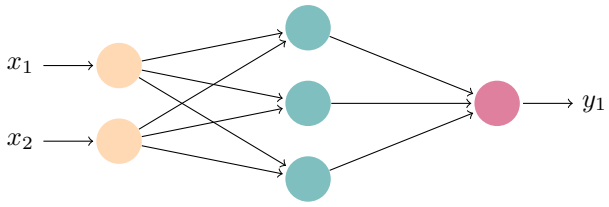

Figure 2: A simple feed-forward network shown as a graph. The middle three (green) units are hidden.

In the noisy-xor problem case we cannot draw a single line to separate the orange from the blue points. The problem is therefore a non-linear classification problem. Neural networks with hidden units can classify such data because the hidden units can create non-linear transformations of the input space, and the output unit can learn a linear classifier that uses these transformed data representations as input.

Consider the example of a simple feed-forward neural network shown in Figure 2. A PyTorch print of an implementation of this network is as follows:

```
Net(
  (model): Sequential(
    (0): Linear(in_features=2, out_features=3, bias=True)
    (1): Tanh()
    (2): Linear(in_features=3, out_features=1, bias=True)
    (3): Identity()
  )
)
```

where we see that the activation functions in the first and inner layers are tanh, and in the output layer, it is the identity function. The activation function of the output unit $\sigma()$ is built into the binary cross entropy loss function.

Your tasks are to:

1. Use an LLM to write a PyTorch program to be handed in the file `feedforwardAssignment.ipynb`. The program should contain a data generation class, a model class implementing for example the network listed above, a training loop (with mini-batch learning using the PyTorch DataLoader method and the Adam optimizer), visualization to make plots like Figure 1 for both the training and a validation set. When making programs with an LLM, it works well to give a short description of the overall task and then in the back and forth manner break the task into subtasks that each can be validated independently.

   (a) There are two versions of the binary cross entropy loss function in PyTorch. What is the difference, and which one should you use?

2. Update `feedforwardAssignment.ipynb` such that it tests a range of networks for how well they can classify the noisy-xor data source. As a minimum try repeatedly all combinations of $0 \ldots 3$ hidden layers with widths $1 \ldots 3$. For each combination of depth and width, calculate the mean and standard deviation of the resulting loss function on a new data set.

3. Give a brief interpretation of the effect of changing the depth and width on the quality of classification in the above experiment. Calculate how many parameters each of the networks tested above has. Identify the training parameters used, and describe how changing them, may influence your conclusions on the quality of the classification.

4. The file `XOR_NNsandbox.ipynb` available on Absalon visualizes the decision boundaries of the minimal XOR network with one hidden layer and two hidden units. Train your network with this architecture (still using tanh hidden activation functions), save weights during training and make a movie that show how the three decision boundaries evolve during training. Include this code in `feedforwardAssignment.ipynb`. Include a few snapshots from training in your report. It is encouraged to use an LLM and code from `XOR_NNsandbox.ipynb` to answer this question.

4 MLOps

The learning goal of this part of the assignment is to get hands-on experience with using an MLOps framework in practice.

Below, you are asked to use an MLOps platform such as Weights and Biases. Tutorial and example notebook using W&B for tasks similar to the below are available on Absalon (week 2 module). You are free to use other similar frameworks as long as your choice supports the requested tasks.

Your tasks are to:

1. Consider `feedforwardAssignment.ipynb` and choose a promising network configuration. Modify the notebook to log the training progress in the MLOps platform, and show the resulting plots in your report. Describe the modifications you made to use the MLOps platform and include code excerpts of the modifications in your report (should only be a few lines of code).
2. Implement a hyperparameter sweep using an MLOps platform, e.g., a W&B sweep. Sweep over the parameters you find most important (but keep the sweep of a size compatible with the compute resources you have - the size of the sweep will not affect the grading). Visualize the results using, e.g., a parallel coordinate plot in W&B.
3. Describe the experimental setup and your results in the report. You must describe what you did, present the results, discuss the results, and draw very careful preliminary conclusions.