Neural Networks Lab Manual – Lab 1/4

From the Single Unit to MNIST — A Process-Over-Product Sequence

---

How This Manual Works

This lab sequence is built on a single premise: **the learning is not in the typing.** You are permitted — encouraged — to use AI to write, debug, and refactor your code, your training loops, your sweep harnesses, and your plotting. What you cannot outsource, and what this manual grades, is the reasoning around the code: predicting what will happen before you run it, assessing why it happened, diagnosing what went wrong, and reconciling the moments when the machine surprised you.

Consequently, **there are no points for code anywhere in this manual.** A perfect, elegant, fully-working implementation earns zero points on its own. The points live entirely in four recurring artifacts, described below. This is deliberate. Working code you cannot explain is exposed the moment you are asked to predict, derive, or diagnose — and those are the only things assessed here.

The AI use policy

- **Permitted:** using AI to write and debug code, build the experiment harness, generate plots, and explain concepts to you. You may implement your code in Java or Python.
- **Not permitted for credit:** having AI write your predictions, derivations, diagnoses, or reconciliations. These must be your own reasoning, committed before you see results (for predictions) or written in your own words (for everything else). A prediction ledger that reads as if written after the run earns nothing — being wrong in advance is worth full marks; being retroactively right is worth none.
- **Specifically forbidden in Lab 1 and Lab 2:** surrogate gradients / straight-through estimators for the step function. If your AI assistant offers one, decline it. The step function's dead gradient is the entire lesson; a surrogate hides it.

The four recurring artifacts

Every lab is assessed through some combination of these. They are defined once here and referenced throughout.

1. **Prediction ledger.** Before running an experiment, you write down what you expect to happen and why. Predictions are graded on the quality of reasoning, never on correctness. A well-reasoned wrong prediction that you then reconcile is the ideal outcome.
2. **Mechanism explanation.** A short, written account of why a result occurred.
3. **Signature library.** A cumulative, annotated collection of loss-curve shapes (and decision-boundary shapes) keyed to their causes, built across Labs 1–4 and used in the capstone. Each entry: a sketch or saved plot, the cause, and the one-line diagnostic tell.
4. **Reconciliation.** Whenever a result diverges from your prediction, a short note: what you expected, what happened, and what your mental model was missing. This is usually where the real learning is, and it is weighted accordingly.

Shared infrastructure (build once, ungraded)

Before Lab 1, stand up a small harness you will reuse throughout. AI may write all of it.

- **Logging** that captures per-epoch train and loss you can plot.

- **A 2D toy dataset** for Lab 3 — two-moons or two-spirals — chosen because you can see the decision boundary. Brittleness is far more visceral as a warped boundary than as a number.
- **One-variable-at-a-time hygiene:** hold dataset and seeds fixed, change one hyperparameter at a time, until the coupling study in Lab 3 where changing two is the point.

Grading overview

Lab	Title	Points
1	The Single Unit — Perceptron Rule vs Gradient Descent	15
2	The XOR Network — Representability vs Learnability	20
3	Activations and the Anatomy of Hyperparameters	25
4	MNIST — Discipline at Scale	20
5	Capstone: The Diagnosis Clinic	20
	Total	100

Code: **0 points.** Everything above is prediction, derivation, diagnosis, and reconciliation.

Lab 1 — The Single Unit: Perceptron Rule vs Gradient Descent (15 pts)

Idea

A single unit can only separate a dataset in a linear fashion. AND is linearly separable; XOR is not. You will train a perceptron two different ways — the perceptron learning rule (for a step activation) and gradient descent (for a sigmoid activation) — and watch both succeed on AND and fail on XOR in visibly different ways.

Background

- **Step activation → perceptron learning rule.** The step function has zero derivative everywhere it is defined, so it cannot be trained by gradient descent. The perceptron rule sidesteps the derivative: w ← w + lr · (target − output) · input. The perceptron convergence theorem guarantees this terminates when the data is linearly separable.
- **Sigmoid activation → gradient descent.** A single sigmoid unit trained by gradient descent is exactly logistic regression. It is differentiable, so ordinary backprop applies.

Reminder — what a unit actually computes (the dot product)

This is covered in lecture, but it underpins every lab in this manual, so it is repeated here. A unit computes w·x + b, and **w·x is a dot product — a similarity score between the input x and the unit's weight vector w.** The dot product is largest when x points in the same direction as w, near zero when they are orthogonal, and negative when they oppose. So you can read a unit's weight vector as a **stored template**, and the unit as asking: how much does this input resemble my template? The bias b shifts how strong that resemblance must be before the unit fires.

This is the precise, operational sense in which a neural network is a **pattern matcher**: it is a composition of template-match scores, passed through non-linearities and combined. Keep this in view — in Lab 2 you will read the templates a trained network learned, in Lab 3 you will see what happens when an input resembles no stored template, and in Lab 4 you will look at templates directly as images.

Tasks

Please see the slides and worksheets from day 4 for the algorithms forward run as well as the training run. Please implement them.

For all tasks below, when running the code, please experiment with the following hyper parameters so as to optimize the training: values of initial weights, learning rate, number of epochs. See what it takes to minimize the number of epochs, and report your observations below.

A. AND with the step unit (perceptron rule). Predict, then implement, then train. Report epochs to convergence.

B. AND with the sigmoid unit (gradient descent). Predict, then implement, then train. Please do not use bias nodes.

Deliverables and points

Item	Artifact	Pts
Predictions for A–B, committed before running	Prediction ledger	3
Values of hyperparameters for *step* unit AND experiments. Must have at least 5 reasonable experiments.	Results table	4

Item	Artifact	Pts
Values of hyperparameters for *sigmoid* AND experiments. Must have at least 5 reasonable experiments.	Results table	4
Mechanism: Explain why the network learns or why it does not learn.	Mechanism explanation	4

No points for the implementation.

Appendix A — Prediction Ledger Template

> *Experiment: _____ Date/commit: _____*
> *What I will change: _____ (all else held fixed: _____)*
> *What I predict will happen: _____*
> *Why (mechanism I expect): _____*
> *What would surprise me: _____*

Committed before the run. Graded on reasoning, never on correctness.

Appendix B — Mechanism Explanation Rubric

A full-credit mechanism explanation:

- names the specific behavior observed,
- connects it to the relevant equation (update rule, gradient, activation derivative),
- would let a peer predict the same behavior in a new setting.

Appendix C — Reconciliation Template

> *I predicted: _____*
> *What actually happened: _____*
> *What my mental model was missing: _____*
> *Updated rule of thumb: _____*