CS4452 Assignment 2

Instructor: Yalda Mohsenzadeh

February 2026

Submission Instructions

Please submit a `.ipynb` file named `<your_name>.ipynb` with all cells executed and the relevant source code included.

Problem 1: Classification Problem

In this assignment, you will implement several core components of a neural network from scratch, including normalization layers, activation functions, loss functions, and regularization techniques.

You are not allowed to use PyTorch built-in layers such as `nn.Tanh`, `nn.Dropout`, `nn.BatchNorm`, or `nn.CrossEntropyLoss`. All required components must be implemented manually using tensor operations.

Download `assignment2.ipynb`, which contains code to generate the simulated data for classification and some helper functions for your convenience. You may modify the helper code if needed, but you must use the provided simulated dataset.

Q1 (10 points): Implement Instance Normalization

Let $z \in \mathbb{R}^{B \times D}$ denote the input to the normalization layer.

Forward Pass

For each sample b:

$$\mu_b = \frac{1}{D} \sum_{i=1}^{D} z_{b,i}$$

$$\sigma_b^2 = \frac{1}{D} \sum_{i=1}^{D} (z_{b,i} - \mu_b)^2$$

$$\hat{z}_{b,i} = \frac{z_{b,i} - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}}$$

$$a_{b,i} = \gamma_i \hat{z}_{b,i} + \beta_i$$

Backward Pass

Let

$$\delta^{(a)} = \frac{\partial L}{\partial a}$$

Parameter gradients:

$$\frac{\partial L}{\partial \beta_i} = \sum_{b=1}^{B} \delta_{b,i}^{(a)}$$

$$\frac{\partial L}{\partial \gamma_i} = \sum_{b=1}^{B} \delta_{b,i}^{(a)} \hat{z}_{b,i}$$

Define

$$\delta_{b,i}^{(\hat{z})} = \delta_{b,i}^{(a)} \gamma_i$$

Let

$$\text{invstd}_b = \frac{1}{\sqrt{\sigma_b^2 + \epsilon}}$$

Then

$$\delta_b^{(z)} = \frac{1}{D} \text{invstd}_b \left(D\delta_b^{(\hat{z})} - \sum_{i=1}^{D} \delta_{b,i}^{(\hat{z})} - \hat{z}_b \sum_{i=1}^{D} \delta_{b,i}^{(\hat{z})} \hat{z}_{b,i} \right)$$

Q2 (10 points): Implement Dropout

Let p be the dropout probability.

Forward Pass (Training)

Sample mask

$$m_{b,i} \sim \text{Bernoulli}(1-p)$$

$$a_{b,i} = \frac{m_{b,i} z_{b,i}}{1-p}$$

Forward Pass (Evaluation)

$$a = z$$

Backward Pass (Training)

$$\delta^{(z)} = \frac{m}{1-p} \odot \delta^{(a)}$$

Backward Pass (Evaluation)

$$\delta^{(z)} = \delta^{(a)}$$

Q3 (10 points): Implement Softmax Cross Entropy

Let logits be $z \in \mathbb{R}^{B \times C}$.

Forward Pass

Softmax:

$$\hat{y}_{b,c} = \frac{e^{z_{b,c}}}{\sum_{k=1}^{C} e^{z_{b,k}}}$$

Cross entropy loss:

$$L = -\frac{1}{B} \sum_{b=1}^{B} \log \hat{y}_{b,y_b}$$

Backward Pass

Let Y^{onehot} be the one-hot encoding of labels.

$$\frac{\partial L}{\partial z} = \frac{1}{B} \left(\hat{y} - Y^{onehot} \right)$$

Q4 (10 points): Implement Tanh

Forward Pass

$$a = \tanh(z)$$

Backward Pass

$$\delta^{(z)} = \delta^{(a)} \odot (1 - a^2)$$

Q5 (10 points): Build and Train a Classification Model

Using your implemented components to build

- generate data with generate_spiral(n_per_class=150, n_classes=2, noise=0.05, rotations=1.5, seed=42)
-

- A 2-hidden-layer MLP with the following structure per hidden layer: Linear → InstanceNorm → Tanh → Dropout
- Do not use weight decay
- lr = 0.1
- trained on the provided spiral dataset

Your training loop must include

- Forward pass
- loss computation
- backward pass
- parameter update
- zeroing gradients

Report:

- final training and validating accuracy

Q6 (25 points): InstanceNorm vs BatchNorm

Replace InstanceNorm with BatchNorm in your model. Train both models under identical settings. Describe what you observed and why?

Q7 (25 points): InstanceNorm With and Without Dropout

Using InstanceNorm:

- Train one model without Dropout
- train one model with Dropout(p=0.2)

Describe what you observed and why?