Assignment 2

Assignment 2: From Linear Models to Deep Networks

Assignment Overview:

This assignment provides a comprehensive exploration of machine learning techniques for handwritten digit recognition, progressing from simple linear models to sophisticated deep neural networks. Students will implement and compare logistic regression, softmax regression, and custom neural networks using PyTorch.

Learning Objectives:

- **Foundation Models**: Implement logistic regression and softmax regression from scratch
- **Custom Neural Networks**: Design and build effective PyTorch models for digit recognition
- **Efficient Data Handling**: Master PyTorch data loaders for seamless dataset management
- **Training Optimization**: Develop specialized training loops with custom gradient descent
- **Architectural Analysis**: Explore various network architectures and regularization techniques
- **Performance Evaluation**: Conduct comprehensive model comparison and analysis
- **Research Skills**: Investigate advanced techniques like dropout and normalization

Problem Statement:

Develop a comprehensive handwritten digit recognition system that demonstrates the evolution from linear classifiers to deep learning. Using the MNIST dataset, you will implement multiple approaches and analyze their relative strengths and weaknesses.

Part A: Linear Classification Models (40 points)

A1. Data Preparation (10 points)

- Download the MNIST dataset from Kaggle
- Implement proper data preprocessing:
  - Normalize pixel values to [0,1] range
  - Flatten images for linear models (28×28 → 784 features)
  - Keep original shape for neural networks
- Split data stratified: 60% training, 20% validation, 20% test using `sklearn.model_selection.train_test_split`
- Create appropriate PyTorch DataLoaders

A2. Logistic Regression Implementation (15 points)

Binary Classification Task: Implement binary logistic regression to distinguish between two digits (e.g., 0 vs 1)

Requirements:

Prof. Dr. Marwan Torki

Eng. Youssef El-Ebiary
Eng. Youssef El-Kady
Eng. Ziad Fahmy
Eng. Sarah Mahmoud

- Implement logistic regression from scratch using PyTorch tensors
- Use sigmoid activation function
- Implement binary cross-entropy loss
- Use gradient descent optimization (learning rate: 0.01)
- Train for sufficient epochs until convergence

Deliverables:

- Training and validation loss curves
- Training and validation accuracy curves
- Final test accuracy and confusion matrix

A3. Softmax Regression Implementation (15 points)

Multi-class Classification: Extend to classify all 10 digits using softmax regression

Requirements:

- Implement softmax regression from scratch
- Use softmax activation and cross-entropy loss
- Compare with PyTorch's built-in implementations for verification
- Use same optimization settings as logistic regression

Deliverables:

- Training and validation metrics plots
- Test set evaluation with confusion matrix
- Per-class accuracy analysis

Part B: Neural Network Implementation (50 points)

B1. Custom Neural Network Architecture (20 points)

Requirements:

- Implement feedforward neural network using PyTorch
- Minimum architecture: Input → Hidden1 → Hidden2 → Output
- Use ReLU activation for hidden layers
- Flexible architecture allowing easy modification of layers/neurons
- Proper weight initialization (Xavier/He initialization)

B2. Training Infrastructure (15 points)

Custom Training Loop Requirements:

- Implement complete training loop from scratch
- Support for batch processing
- Proper gradient computation and backpropagation
- Training/validation split handling
- Progress tracking and logging

Training Configuration:

- Optimizer: Stochastic Gradient Descent (SGD)
- Learning rate: 0.01 (baseline)
- Loss function: Cross-entropy
- Batch size: 64 (baseline)

B3. Performance Visualization (15 points)

Required Plots:

- Training and validation loss over epochs
- Training and validation accuracy over epochs
- Learning curves with error bars
- Convergence analysis

Part C: Comprehensive Analysis (40 points)

C1. Hyperparameter Analysis (25 points)

Conduct systematic analysis by varying **one parameter at a time**:

Learning Rate Analysis:

- Test at least 4 values: [0.001, 0.01, 0.1, 1.0]
- Plot learning curves for each
- Analyze convergence speed and stability

Batch Size Analysis:

- Test at least 4 values: [16, 32, 64, 128]
- Compare training efficiency and final performance
- Analyze effect on gradient noise

Architecture Analysis:

- **Number of layers**: Test 2, 3, 4, 5 hidden layers
- **Neurons per layer**: Test [64, 128, 256, 512] neurons
- Create architecture comparison table

C2. Model Comparison (15 points)

Comparative Analysis:

- Compare logistic regression, softmax regression, and best neural network
- Analyze computational complexity and training time
- Discuss when to use each approach
- Create comprehensive performance summary table

Best Model Evaluation:

- Test optimal configuration on test set
- Report final accuracy and detailed confusion matrix
- Analyze misclassified examples
- Provide insights on model limitations

Part D: Advanced Techniques (Bonus - 20 points)

D1. Convolutional Neural Networks (10 points)

- Implement basic CNN architecture for MNIST
- Compare CNN vs fully connected network performance
- Analyze the benefit of spatial feature learning

D2. Regularization Techniques (10 points)

Dropout Analysis:

- Implement dropout layers with different rates [0.1, 0.3, 0.5, 0.7]
- Analyze effect on overfitting and generalization

Batch Normalization:

- Add batch normalization layers
- Compare training stability and convergence speed
- Analyze combined effect with dropout

Deliverables

Required Submissions

1. **Jupyter Notebook** (.ipynb file with comprehensive analysis in markdown cells)
2. **PDF Export** of the notebook
3. **Code Repository** (if using version control - recommended)

Alexandria University
Faculty of Engineering

Introduction to Machine Learning
Assigned: Wednesday, Oct 22, 2025
Due: Sunday, Nov 2, 2025

Grading Rubric

Component	Points	Criteria
Part A: Linear Models	40	Implementation correctness, analysis quality, visualizations
Part B: Neural Networks	50	Architecture design, training loop, performance evaluation
Part C: Analysis	40	Systematic experimentation, insights, model comparison
Part D: Bonus	20	Advanced techniques implementation and analysis
Total	130 + 20	(Bonus points can compensate for lost points)

Technical Tips

- Start with simple models and gradually increase complexity
- Use GPU acceleration for faster training (Google Colab recommended)
- Implement early stopping to prevent overfitting
- Save model checkpoints for long training runs

Dataset Access

- torchvision.datasets.MNIST

Academic Integrity Reminder: While collaboration within teams is encouraged, copying from other teams or external sources without attribution is strictly prohibited. All implementations should be original work by your team.

Final Notes

1. You should work in groups of maximum three, not finding a team is NOT an excuse.
2. You should deliver with a naming scheme id _assigment.zip.
3. Delivery will be ignored if you didn't follow the naming scheme provided in 2, any one of the team ids can be used.
4. Any form of academic dishonesty, including but not limited to using AI tools (such as ChatGPT or other code generation platforms), copying open-source code without proper attribution, or engaging in 'vibe coding' without genuine understanding, will be considered a serious violation and will be heavily penalized.

Prof. Dr. Marwan Torki

Eng. Youssef El-Ebiary
Eng. Ziad Fahmy

Eng. Youssef El-Kady
Eng. Sarah Mahmoud