Assignment 1
From Supervised to Semi-Supervised Learning
A Complete Machine Learning Pipeline

COSC 4P96: Topics in Computational Intelligence
Brock University

Due Date: 2026-02-09 at 11:59 PM

1 Introduction and Motivation

Building effective machine learning systems requires mastery of multiple interconnected components: data preprocessing, model architecture design, training optimization, regularization, and the ability to leverage unlabeled data. Each component influences overall system performance, and understanding their interactions is crucial for developing robust solutions.

This comprehensive assignment challenges you to build an end-to-end machine learning pipeline that progressively incorporates increasingly sophisticated techniques. You will begin with a supervised neural network baseline and systematically add components including proper data preprocessing, architecture selection, and semi-supervised learning methods. Throughout, you will analyze how each component contributes to the final performance.

The assignment simulates a realistic scenario where only a fraction of your data is labeled, requiring you to make intelligent use of both labeled and unlabeled examples.

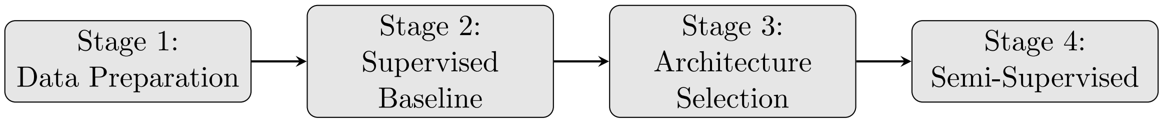
2 Learning Objectives

Upon completing this assignment, you will be able to:

1. Design and implement a complete machine learning pipeline from data preprocessing to model evaluation
2. Apply appropriate data preprocessing techniques including scaling, normalization, and handling of outliers
3. Implement neural network training with proper optimization, including learning rate scheduling and momentum
4. Apply architecture selection techniques such as regularization or pruning
5. Implement semi-supervised learning methods to leverage unlabeled data
6. Conduct systematic ablation studies to understand component contributions
7. Perform rigorous statistical analysis and present results professionally

3 Pipeline Overview

Your complete pipeline will consist of the following stages:


Stage 1: Data Preparation
Stage 2: Supervised Baseline
Stage 3: Architecture Selection
Stage 4: Semi-Supervised

Each stage builds upon the previous one, and you will analyze the contribution of each component to overall performance.

4 Assignment Tasks

4.1 Stage 1: Data Preparation and Preprocessing (15 marks)

a) **Dataset Selection:** Choose a classification dataset with at least 10,000 samples. Suggested options:

- Fashion-MNIST (images, 70,000 samples, 10 classes)
- CIFAR-10 (images, 60,000 samples, 10 classes)
- UCI Letter Recognition (tabular, 20,000 samples, 26 classes)
- Covertype (tabular, 581,012 samples, 7 classes)

b) **Data Splitting:** Organize your data as follows:

- **Test set:** 10% (held out for final evaluation only)
- **Validation set:** 10% (for hyperparameter tuning and early stopping)
- **Training pool:** 80% (used for training experiments)

c) **Semi-Supervised Setup:** From the training pool:

- Designate **20%** as labeled data (labels are available)
- Designate **80%** as unlabeled data (labels are hidden during training)

d) **Data Preprocessing:** Implement and compare at least TWO of the following:

- Min-max scaling to $[0, 1]$ or $[-1, 1]$
- Z-score normalization (zero mean, unit variance)
- Per-sample normalization (unit length)

e) **Data Augmentation:** For image datasets, implement at least TWO augmentation techniques:

- Random horizontal/vertical flips
- Random crops or translations
- Random noise injection
- Color jittering (if applicable)

For tabular datasets, implement noise injection with appropriate variance.

f) **Deliverable:** Document your preprocessing choices with justification. Include statistics about your dataset (class distribution, feature ranges before/after preprocessing).

4.2 Stage 2: Supervised Neural Network Baseline (20 marks)

a) **Network Architecture:** Design a feedforward neural network with:
   - At least 2 hidden layers
   - Appropriate activation functions (sigmoid, tanh, or ReLU)
   - Softmax output layer for classification
   - Bias units for all layers

b) **Training Implementation:** Implement the following:
   - Backpropagation with cross-entropy loss
   - Optimizer: Stochastic Gradient Descent (SGD)
   - Mini-batch stochastic gradient descent
   - Momentum term (experiment with $\alpha \in [0.5, 0.99]$)
   - Learning rate (experiment with $\eta \in [0.001, 0.1]$)

c) **Weight Initialization:** Implement and compare:
   - Random uniform in $[-1/\sqrt{\text{fanin}}, 1/\sqrt{\text{fanin}}]$
   - Random normal with small variance
   - Xavier/Glorot initialization (if using ReLU, use He initialization)

d) **Overfitting Detection:** Implement early stopping by monitoring validation error. Use the criterion:

$$\mathcal{E}_V > \bar{\mathcal{E}}_V + \sigma_{\mathcal{E}_V}$$

   where $\bar{\mathcal{E}}_V$ is the moving average and $\sigma_{\mathcal{E}_V}$ is the standard deviation of recent validation errors.

e) **Baseline Training:** Train your model using **only the 10% labeled data**. Record:
   - Training and validation loss/accuracy curves
   - Final test accuracy
   - Training time (epochs to convergence)

f) Run experiments at least **3 times** with different random seeds such as using $\{1, 123, 12345\}$. A guideline how to set seeds manually: https://docs.pytorch.org/docs/stable/notes/randomness.html.

4.3 Stage 3: Architecture Selection and Regularization (20 marks)

Implement **TWO** of the following techniques to improve your baseline model:

Option A: Weight Decay Regularization

i. Add L2 regularization to your loss function:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \frac{\lambda}{2}\sum_i w_i^2$$

ii. Experiment with $\lambda \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$

iii. Analyze the effect on weight magnitudes and generalization

Option B: Dropout Regularization

i. Implement dropout during training (randomly set hidden unit outputs to zero)

ii. Experiment with dropout rates $p \in \{0.1, 0.25, 0.5\}$

iii. Remember to scale activations appropriately during inference

Option C: Network Pruning

i. Train a larger network than necessary (e.g., 2x hidden units)

ii. Implement magnitude-based pruning: remove weights with smallest absolute values

iii. Experiment with pruning rates $\{10\%, 25\%, 50\%\}$ of weights

iv. Fine-tune after pruning and compare performance

Option D: Adaptive Learning Rate

i. Implement learning rate scheduling:

- Step decay: reduce η by factor every k epochs
- Exponential decay: $\eta(t) = \eta_0 \cdot \gamma^t$

ii. Alternatively, implement per-weight adaptive learning rates

iii. Compare convergence speed and final accuracy

Deliverable: For each technique, report improvement (or degradation) compared to baseline. Discuss why each technique helps or hurts performance on your dataset.

4.4 Stage 4: Semi-Supervised Learning (30 marks)

Now extend your model to leverage the 90% unlabeled data. Implement **ONE** of the following:

Method A: Label Propagation + Neural Network

i. Construct a similarity graph over all training data (labeled + unlabeled)

ii. Implement label propagation to generate pseudo-labels:

- Define edge weights using a similarity metric (e.g., Gaussian kernel on features or learned embeddings)
- Iteratively propagate labels: $Y \leftarrow TY$, then normalize and clamp
- Run until convergence

iii. Train your neural network on the combination of:

- Original labeled data (with true labels)
- Unlabeled data with propagated pseudo-labels (weighted by confidence)

iv. Experiment with different similarity metrics and confidence thresholds

Method B: Co-Training with Two Views

i. Create two "views" of your data:

- For images: split features (e.g., left/right halves, different augmentations)
- For tabular: split features into two disjoint subsets

ii. Train two separate neural networks, one on each view

iii. Implement the co-training algorithm:

- Each model predicts on unlabeled data
- If Model A is confident but Model B is not, add to B's training set (and vice versa)
- Retrain and iterate

iv. Final prediction: combine outputs of both models (e.g., average or voting)

Method C: Consistency Regularization with Data Augmentation

i. Modify your training objective to include a consistency term:

$$\mathcal{L} = \underbrace{-\sum_{i\in\text{labeled}} \log p_\theta(y_i|\mathbf{x}_i)}_{\text{supervised loss}} + \lambda \underbrace{\sum_{j\in\text{unlabeled}} \|p_\theta(y|\mathbf{x}_j) - p_\theta(y|\mathbf{x}'_j)\|^2}_{\text{consistency loss}}$$

where $\mathbf{x}'_j$ is an augmented version of $\mathbf{x}_j$

ii. Implement a warm-up schedule for λ: start at 0, gradually increase over training

iii. Use your data augmentation from Stage 1 to generate $\mathbf{x}'$

iv. **Important:** Treat $p_\theta(y|\mathbf{x}_j)$ as a fixed target (stop gradients)

Deliverable: Compare semi-supervised performance against the supervised baseline. Analyze:

- Improvement in test accuracy
- Training dynamics (how quickly does the model improve?)
- Failure modes (when does semi-supervised learning hurt?)

4.5 Stage 5: Comprehensive Analysis and Ablation Study (15 marks)

a) **Component Contribution Analysis:** Create a table showing the contribution of each pipeline component:

Configuration	**Test Accuracy**	**Improvement**
Baseline (10% labeled, no preprocessing)	$\mu \pm \sigma$	—
+ Preprocessing	$\mu \pm \sigma$	$+X\%$
+ Better Initialization	$\mu \pm \sigma$	$+X\%$
+ Regularization Technique 1	$\mu \pm \sigma$	$+X\%$
+ Regularization Technique 2	$\mu \pm \sigma$	$+X\%$
+ Semi-Supervised Learning	$\mu \pm \sigma$	$+X\%$
Full Pipeline	$\mu \pm \sigma$	**Total**

where μ is average result of runs and σ is standard deviation result of runs.

b) **Comparison with Fully Supervised:** Train your best model using 100% of labels. How close does your semi-supervised pipeline (with 10% labels) get to this upper bound?

c) **Varying Labeled Data:** Test your full pipeline with different amounts of labeled data: 5%, 10%, 25%, 50%. Plot test accuracy vs. percentage of labeled data.

d) **Statistical Significance:** Use appropriate statistical tests (e.g., paired t-test) to determine if improvements are statistically significant.

e) **Computational Cost Analysis:** Report training time for each stage. Discuss the trade-off between added complexity and performance improvement.

5 Deliverables

Submit a single compressed file (.zip) containing:

1. **Report** (PDF, maximum 15 pages excluding appendices):
   - Introduction and problem statement
   - Methodology for each stage
   - Results with tables and figures
   - Discussion and analysis on results
   - Conclusion and future work
   - References to papers, blogs, or books. Do NOT cite lecture notes.
2. **Source Code:**
   - Well-organized Python files or Jupyter notebooks
   - Clear separation between pipeline stages
   - Comprehensive comments explaining key algorithms
   - Provided `requirements.txt` to install required packages to used in the codes.
3. **README:** Instructions to reproduce your results
4. **Results Data:** CSV files with all experimental results

6 Grading Criteria

Component	**Marks**	**Percentage**
Stage 1: Data Preparation	15	15%
Stage 2: Supervised Baseline	20	20%
Stage 3: Architecture Selection (2 techniques)	20	20%
Stage 4: Semi-Supervised Learning	30	30%
Stage 5: Analysis and Report Quality	15	15%
Total	**100**	**100%**

Bonus Opportunities (up to 15 marks):

- Implementing multiple semi-supervised methods with comparison: +10 marks
- Exceptional visualization of results and pipeline: +5 marks
- Novel extensions or insights beyond requirements: +5 marks
- Code quality and reproducibility: +5 marks

7 Implementation Guidelines

- **Allowed:** Python, NumPy, Matplotlib/Seaborn for visualization
- **Allowed with restrictions:** PyTorch or TensorFlow may be used for automatic differentiation, backpropagation, and optimization methods such as SGD, or Adam, and layers such as Dense (linear), convolutional, or batch normalization, but you must implement the core algorithms yourself (semi-supervised methods, pruning, etc.)
- **Not allowed:** Pre-built semi-supervised learning libraries, pre-trained models, or high-level training loops that hide the implementation details
- Set random seeds for reproducibility
- Use version control (Git) and include your commit history and upload to GitHub.

8 Suggested Resources

- Course Lecture Slides: Lectures 1–4
- Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction.* MIT Press. Chapters 13, 19.
- Engelbrecht, A. P. (2007). *Computational Intelligence: An Introduction.* Wiley.
- Zhu, X., & Goldberg, A. B. (2009). "Introduction to Semi-Supervised Learning." Morgan & Claypool.

9 Academic Integrity

This is an **individual assignment**. You may discuss high-level concepts with classmates, but all code and writing must be your own work. Properly cite any external resources, papers, or code snippets that influenced your implementation. Violations of academic integrity will result in a grade of zero and may lead to further disciplinary action.

Questions about this assignment should be directed to the instructor or teaching assistant during office hours or via email. Start early—this is a comprehensive assignment that requires careful implementation and analysis.