MCT Program CSE473s: Computational Intelligence Fall 2025

Project Title: Build Your Own Neural Network Library & Advanced Applications

During this project you will develop a foundational neural network library from scratch using only Python and NumPy, validating it on the classic XOR problem. Second, you will use your library to build and train an unsupervised encoder-decoder (autoencoder) network for image reconstruction. Finally, you will use the trained encoder as a feature extractor, feeding its latent space representations into an SVM to perform supervised classification.

Project Objectives

- **Deepen Understanding**: Grasp the fundamental mathematics and algorithms (forward/backward propagation, optimization) that power machine learning.
- **Practical Implementation**: Code the core components of a neural network: layers, activations, losses, and optimizers.
- **Systematic Testing**: Validate your library's correctness.
- **Unsupervised Learning**: Implement an autoencoder for dimensionality reduction and reconstruction.
- **Feature Extraction**: Explore transfer learning by using the trained encoder's latent space to train a separate SVM classifier.
- **Comparative Analysis**: Critically compare your library's implementation and performance against an industry-standard framework (TensorFlow).

Core Library Requirements

You must create a modular Python library using **only NumPy** for numerical operations.

1. **Layer Abstraction**:
    1. A base Layer class with forward and backward methods.
    2. A Dense (fully connected) layer handling weights (W), biases (b), and their gradients (∂L/∂W, ∂L/∂b).
2. **Activation Functions**:
   Implement as Layer subclasses.
   Required:
    1. ReLU: f(x) = max (0, x)
    2. Sigmoid: $f(x) = \frac{1}{(1+e^{(-x)})}$
    3. Tanh: $f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
    4. Softmax.

3. **Loss Functions**:
   Functions to calculate the loss and its initial gradient.
   Required:
   Mean Squared Error (MSE): $L = \frac{1}{N} \sum (Y_{true} - Y_{pred})^2$
4. **Optimizer**:
   An SGD (Stochastic Gradient Descent) optimizer with a step method to update parameters using their gradients and a learning rate: $W_{new} = W_{old} - \eta(\partial L/\partial W)$.
5. **Network Model**:
   A Sequential or Network class to manage the list of layers and orchestrate the full train, forward, and backward passes.

Part 1: Library Implementation, Initial Test and Validation (XOR Problem)

Before tackling complex tasks, you must prove your library works by solving the XOR problem.

1. **Task**: Build a network that correctly classifies the 4 inputs of the XOR function.
2. **Architecture**: Use your library to build a simple MLP (e.g., 2-4-1 architecture with Tanh and Sigmoid activations).
3. **Training**: Train the network using your SGD optimizer and MSE loss.
4. **Goal**: Demonstrate that your network's predictions are correct for all 4 inputs after training.

Part 2: Autoencoder & Latent Space Classification

Task: Implement an autoencoder for MNIST image reconstruction and use the encoder for digit classification.

Autoencoder Implementation:

- **Encoder**: Dense + ReLU layers compressing 784 pixels → small latent space (32-64 dim)
- **Decoder**: Dense + ReLU/Sigmoid layers reconstructing 784-pixel images
- **Training**: Unsupervised learning using MSE loss (input = target output)

Classification Pipeline:

1. **Feature Extraction**: Use trained encoder to transform MNIST data into latent representations
2. **SVM Training**: Train SVM on latent features and labels
3. **Evaluation**: Report test accuracy, confusion matrix, and classification metrics

Testing and Evaluation

Your project will be graded on correctness, performance, and the quality of your analysis.

1. **Library Unit Testing (Gradient Checking)**:

   You *must* include a section in your notebook that proves your backpropagation is correct using **numerical gradient checking**.

   The formula is: $\partial L/\partial W \approx [L(W + \varepsilon) - L(W - \varepsilon)]/(2\varepsilon)$

   Your analytical gradient must be nearly identical to the numerical gradient.
2. **Model Performance (Your Library)**:
   1. **XOR**: Show the final predictions for the 4 XOR inputs.
   2. **Autoencoder**: Provide a loss curve and visualizations of original vs. reconstructed test images.
   3. **SVM Classifier**: Report the final classification accuracy on the test set.
3. **Baseline Comparison (TensorFlow/Keras)**:
   1. Implement the *exact same* network architectures (for both XOR and the autoencoder) in TensorFlow or Keras.
   2. Train these models and compare:
   3. Ease of Implementation
4. Training Time.
5. Final Reconstruction Loss.

Submission and Deliverables

You will submit your project via **GitHub**.

1. **Repository Setup**:
   1. Create a **public** GitHub repository.
   2. The submission is the URL to this repository.
   3. **Repository Structure**: (Red files structure is mandatory, green is optional)

```
.
├── .gitignore
├── README.md              <-- Main project documentation
├── requirements.txt       <-- (e.g., numpy, matplotlib, ….etc)
├── lib/                   <-- Your neural network library code
│   ├── __init__.py
│   ├── layers.py
│   ├── activations.py
│   ├── losses.py
│   ├── optimizer.py
│   └── network.py
├── notebooks/              <-- Your Jupyter Notebooks
│   └── project_demo.ipynb  <-- A notebook for all demos
└── report/
    └── project_report.pdf  <-- Your final PDF report
```

2. **Deliverables (Inside the Repo)**:
    1. **Source Code**: The well-commented Python library in the /lib folder.
    2. **Jupyter Notebook** (/notebooks/project_demo.ipynb): This single notebook must clearly demonstrate:
        - **Section 1**: Gradient Checking (proving your backprop is correct).
        - **Section 2**: The XOR problem (training and results) using **your library**.
        - **Section 3**: The Autoencoder (training, loss curve, and image visualizations) using **your library**.
        - **Section 4**: The Latent Space SVM Classification (feature extraction, SVM training, and evaluation metrics/confusion matrix).
        - **Section 5**: The TensorFlow/Keras implementations and comparisons.
2. **Report** (/report/project_report.pdf): A report explaining:
    1. Your library design and architecture choices.
    2. Results from the XOR test.
    3. Analysis of the autoencoder's reconstruction quality.
    4. A detailed analysis of your **SVM classification results**. Discuss the quality of the latent space features your encoder learned.
    5. A summary of your **TensorFlow comparison**.
    6. Challenges faced and lessons learned.

Warnings:

(1) Plagiarism is prohibited.
(2) Assignments with no reports will not be graded.

Students are encouraged to work in groups. Each group has at most five students.