AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING

Computational Intelligence CSE473s
Fall 2025 / Mechatronics Senior-2

Final Project Report
– Team 18 –

Name	ID
Amr Khaled Taha	2101427
Mostafa Samy Mohammed	2100349
Mohamed Tarek Abdelwahab	2100287
Omar Mohamed Fathy	2100503

Submitted to:
Prof. Dr. Hossam El din Hassan
Eng. Abdallah Awdallah
Eng. Dina Zakaria

Table of Contents

Introduction

This project implements a modular neural network library from scratch using only Python and NumPy. The objective is to demonstrate the "**black box**" of deep learning by manually implementing the core algorithms of forward propagation, backward propagation (back-propagation), and gradient descent optimization.

Part 1 focuses on library implementation and validation. The core training pipeline includes:

- Layers:
  - Abstract layer class.
  - Fully Connected (Dense) layer.
- Nonlinear Activation functions (ReLU, Sigmoid, Tanh, Softmax).
- Loss computation using **Mean Squared Error** (MSELoss).
- Gradient-based weight updates using **Stochastic Gradient Descent** (SGDOptimizer).
- A Sequential model class (Network) to combine all components.
- Experimental validation using the **XOR problem**.

Part 2 focuses on Autoencoder & Latent Space Classification. It involves two main stages:

- Autoencoder Implementation: Which aims to build a network to reconstruct MNIST images dataset using autoencoder (encoder->decoder).
  - Encoder: Implement Dense and ReLU layers to compress 784 input pixels into a small latent space (32-64 dimensions).
  - Decoder: Implement Dense and ReLU/Sigmoid layers to reconstruct the 784-pixel images from the latent space.
  - Train the model using unsupervised learning with Mean Squared Error (MSE) loss, where the input is the target output.
- Classification pipeline: Which aims to use the pre-trained encoder's latent space–transformed from MNIST images data to latent space representation [Feature vector])–to train SVM on these latent features and labels.

System Architecture & Design

Overall Workflow

[Input –> Sequential.forward –> MSELoss –> MSELoss.backward –> Sequential.backward –> SGDOptimizer(step)]

Modular Design Strategy

Component	Responsibility
Sequential	Manages layers, full training loop (fit)
Dense	Trainable linear transformation (stores W, b, dW, db)
Activation Functions	Apply nonlinearity, no trainable parameters
MSELoss	Computes loss and their gradient w.r.t model output
SGDOptimizer	Updates parameters using gradients

Implemented Components

Dense Layer

- Trainable weights: W and b
- Uses scaled random initialization: np.random.randn() * scale
- Stores gradients: dW, db

Forward: [$Z = XW + b$]

Backward:

- [$dW = X^T * dZ$]
- [$db = \sum dZ$]
- [$dX = dZ * W^T$]

Activation Layers

Activation	Derivative
ReLU	$(1_{x>0})$
Sigmoid	$(y[1 - y])$
Tanh	$(1 - y^2)$
Softmax	Uses cross-entropy loss

- Non-trainable (only modify the gradient).

MSELoss class

Used for XOR regression-style learning.

- [$L = \frac{1}{2N} \sum (y - \hat{y})^2$]
- [$\frac{\partial L}{\partial \hat{y}} = \frac{1}{N} (\hat{y} - y)$]

SGDOptimizer class

Updates trainable parameters:

- $[W_{n+1} = W_n - \eta \cdot dW_n]$
- $[b_{n+1} = b_n - \eta \cdot db_n]$

Where (η) = learning rate.

Sequential class

Responsibilities:

- Store ordered layers
- forward() → inference
- backward() → propagate gradients
- fit() → complete training loop
    1. Forward pass
    2. Computing loss
    3. Backward pass
    4. Optimizer weight update

Result: **Simple high-level API** for training models.

Experiment 1: XOR Problem

Dataset

Input (x1, x2)	Target (y_true)
[-1, -1]	-1
[-1, 1]	1
[1, -1]	1
[1, 1]	-1

Model Architecture

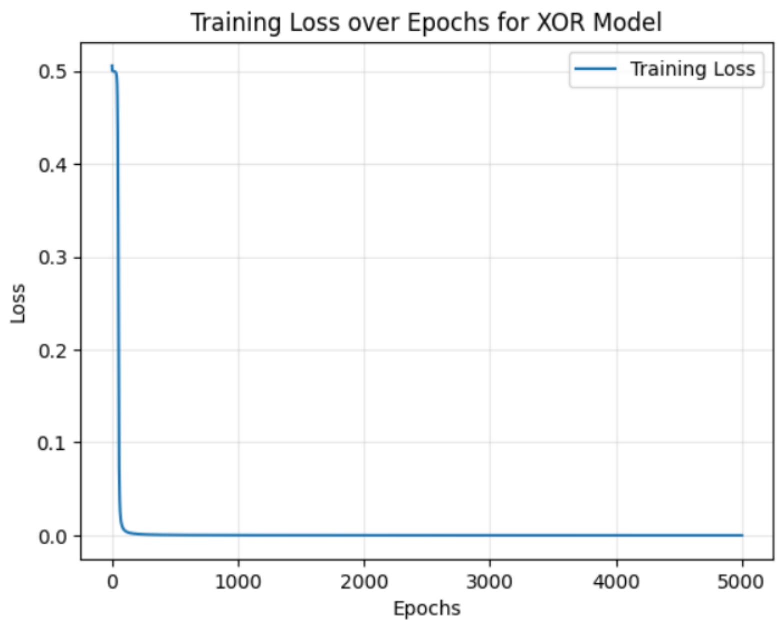
Layer	Scaling factor (Parameters initialization)	Details
Dense layer (2 –> 4)	0.1	Hidden layer
Tanh	–	Nonlinear activation
Dense layer (4 –> 1)	0.1	Output layer
Sigmoid	–	Output activation

Training configuration

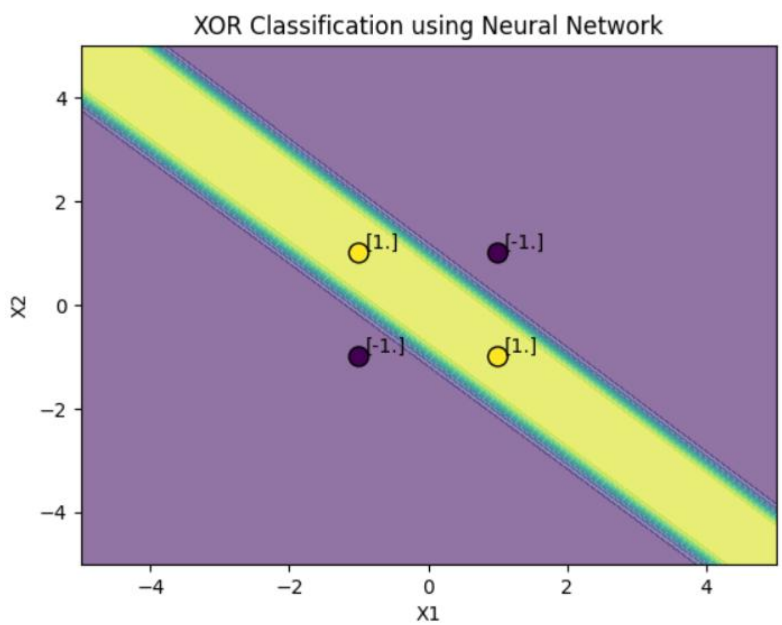
- Loss function: [**MSELoss]**.
- Optimizer: [**SGDOptimizer]** with learning rate (η) = 1.0, and epochs = 5,000.

Results & analysis

Training loss curve



Boundary layers plotting



Final Predictions

Input	Target	Predicted	Rounded
[-1, -1]	-1	-0.99080804	-1
[-1, 1]	1	0.99231939	1
[1, -1]	1	0.99275965	1
[1, 1]	-1	-0.99118204	-1

The model successfully learned XOR mapping, achieving correct classification.

Observations & Notes

- Increasing:
  - Hidden layer size
  - Learning rate Can improve convergence
- The pipeline correctly computes gradients and updates weights

This confirms our framework works end-to-end.

Experiment 2: Autoencoder (MNIST images Reconstruction)

Model architecture

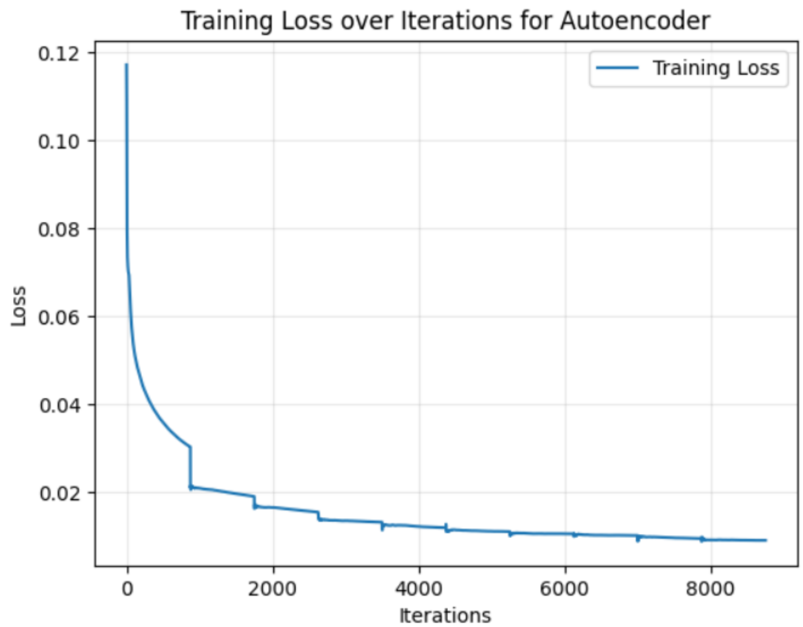
Layer	Scaling factor (Parameters initialization)	Details
Encoder		
Dense (784 –> 128)	0.1	Hidden layer
ReLU	–	Nonlinear activation
Dense (128 –> 64)	0.1	Hidden layer
ReLU	–	Nonlinear activation
Dense (64 –> 32)	0.1	Hidden layer
ReLU	–	Latent space feature vector (encoder output)
Decoder		
Dense (32 –> 64)	0.1	Hidden layer
ReLU	–	Nonlinear activation
Dense (64 –> 128)	0.1	Hidden layer
ReLU	–	Nonlinear activation
Dense (128 –> 784)	0.1	Hidden layer
Sigmoid	–	Reconstructed images [0, 1] (decoder output)

Training configuration

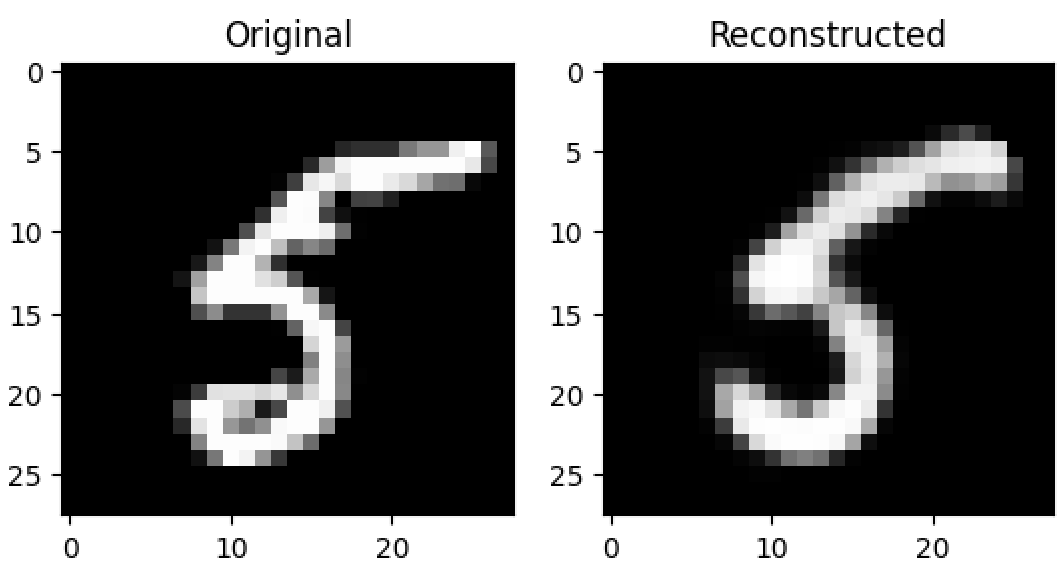
- MNIST images dataset preprocessing: Normalized to (0-1) digits.
- Loss function: [**MSELoss]**.
- Optimizer: [**SGDOptimizer]** with learning rate (η) = 0.1, and epochs = 20.

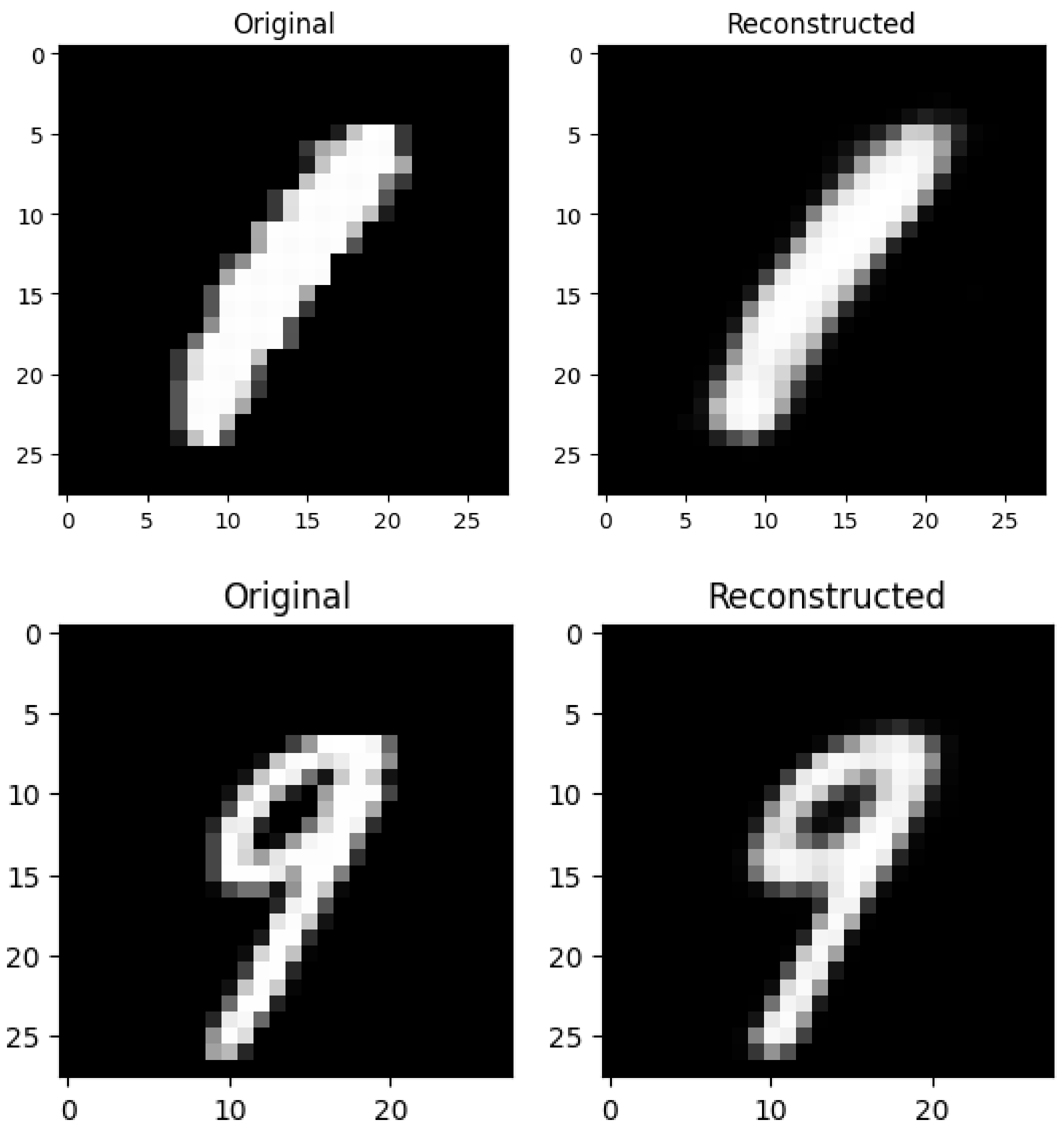
Results & analysis

Training loss curve



Reconstructed test images



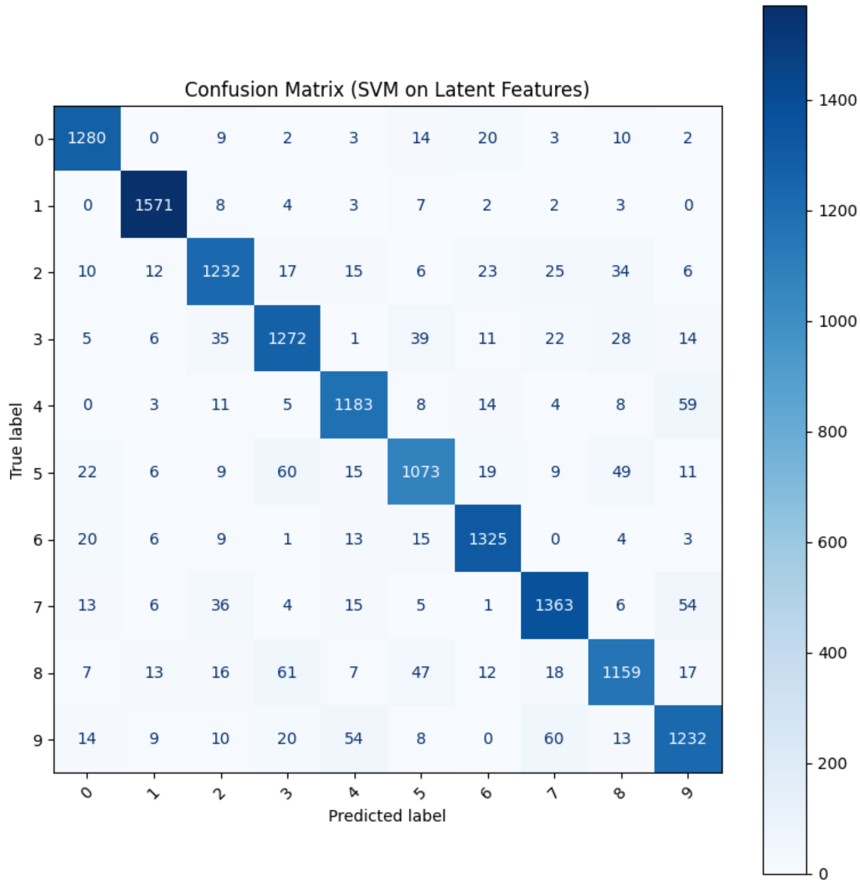


Experiment 3: SVM classifier [Built by sklearn]

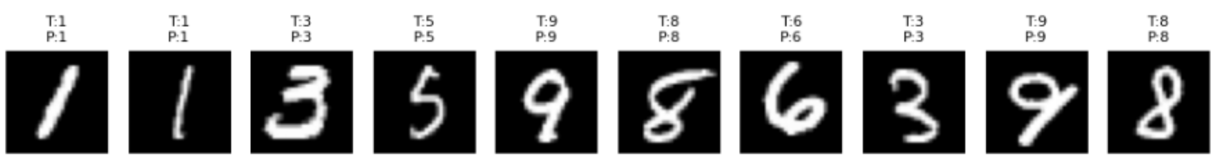
Classification report

```
Accuracy: 0.9064285714285715
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      1343
           1       0.96      0.98      0.97      1600
           2       0.90      0.89      0.89      1380
           3       0.88      0.89      0.88      1433
           4       0.90      0.91      0.91      1295
           5       0.88      0.84      0.86      1273
           6       0.93      0.95      0.94      1396
           7       0.91      0.91      0.91      1503
           8       0.88      0.85      0.87      1357
           9       0.88      0.87      0.87      1420

    accuracy                           0.91     14000
   macro avg       0.91      0.90      0.90     14000
weighted avg       0.91      0.91      0.91     14000
```

Confusion matrix



Results



Comparison with network built by TensorFlow/Keras

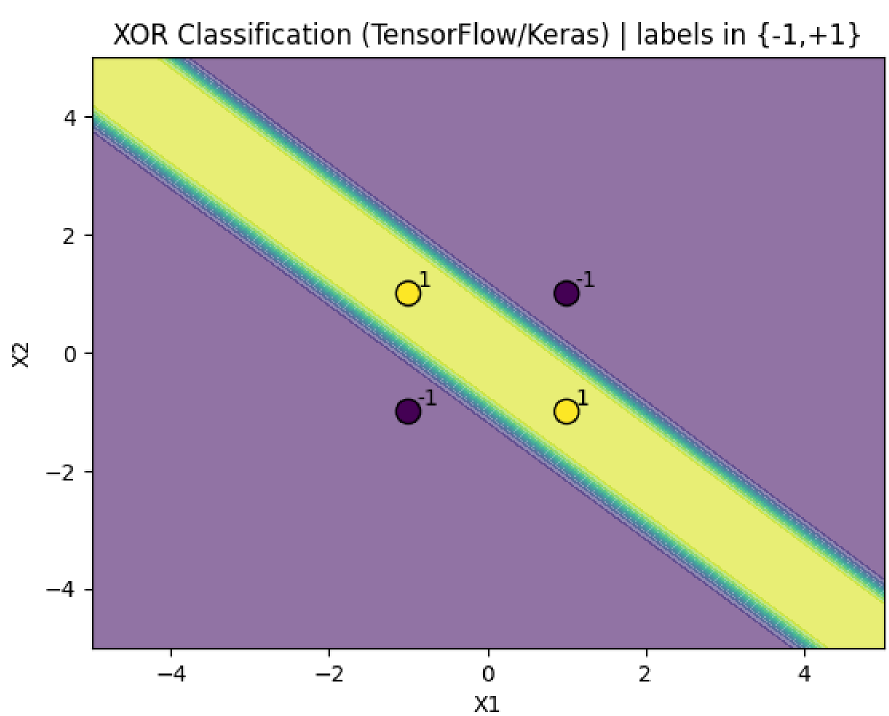
XOR problem

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 4)	12
activation_2 (Activation)	(None, 4)	0
dense_3 (Dense)	(None, 1)	5
activation_3 (Activation)	(None, 1)	0

Total params: 17 (68.00 B)
Trainable params: 17 (68.00 B)
Non-trainable params: 0 (0.00 B)

```
Train time (s): 170.08040391199938
Final loss: 3.4214885090477765e-05
Raw preds:
 [[-0.991]
 [ 0.992]
 [ 0.993]
 [-0.991]]
True:
 [-1  1  1 -1]
Pred sign:
 [-1  1  1 -1]
```



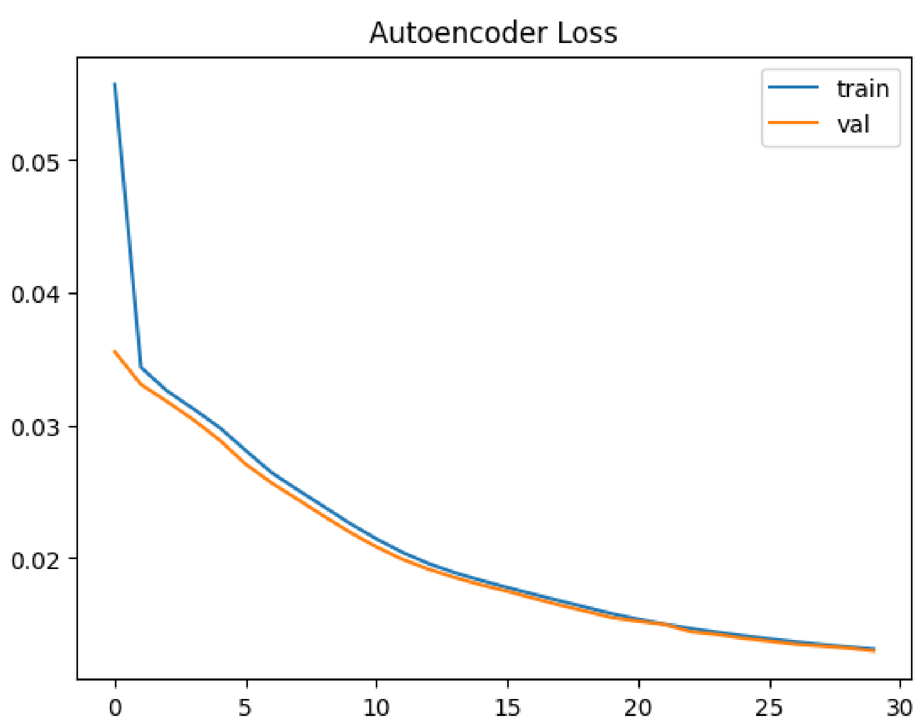
Autoencoder & classifier

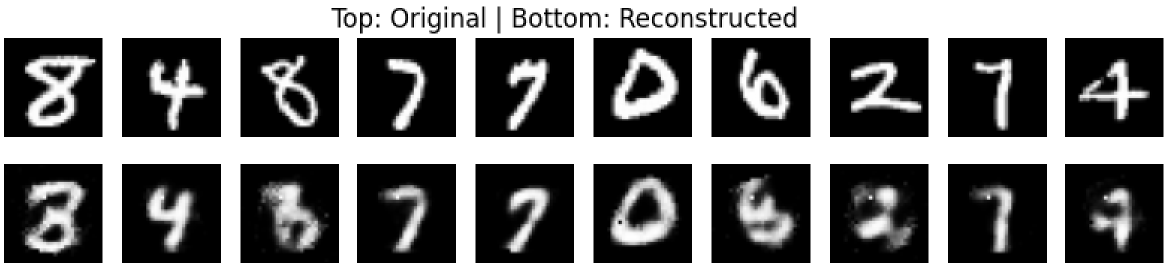
Model: "autoencoder_tf"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 784)	0
dense_4 (Dense)	(None, 128)	100,480
re_lu (ReLU)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8,256
re_lu_1 (ReLU)	(None, 64)	0
dense_6 (Dense)	(None, 32)	2,080
re_lu_2 (ReLU)	(None, 32)	0
dense_7 (Dense)	(None, 64)	2,112
re_lu_3 (ReLU)	(None, 64)	0
dense_8 (Dense)	(None, 128)	8,320
re_lu_4 (ReLU)	(None, 128)	0
dense_9 (Dense)	(None, 784)	101,136
activation_4 (Activation)	(None, 784)	0

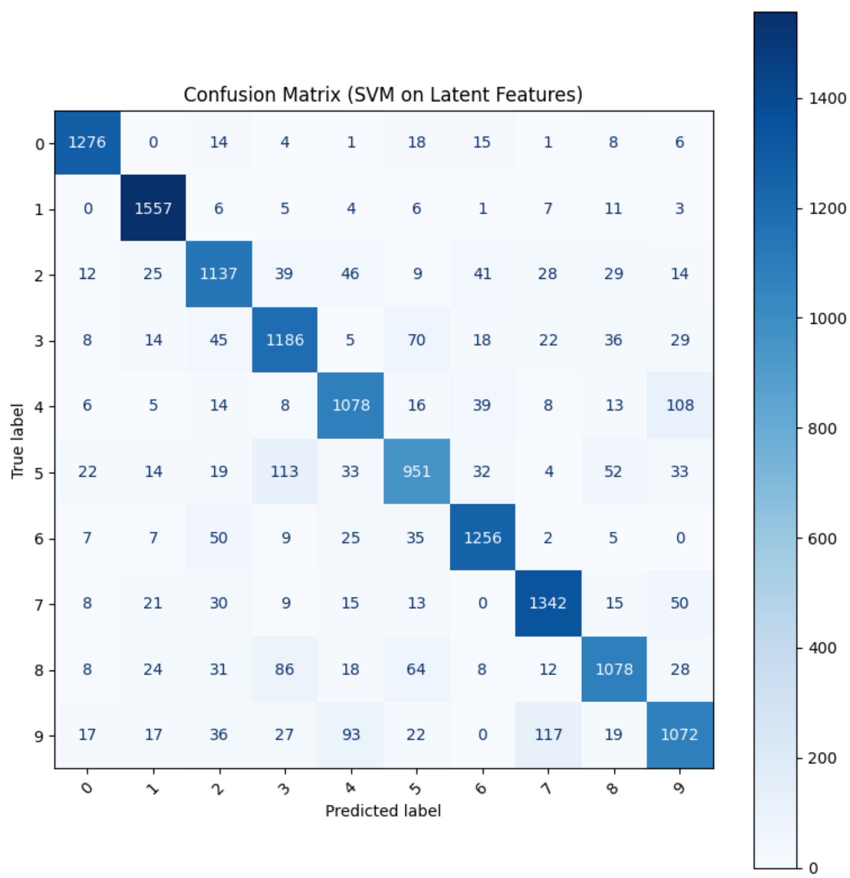
Total params: 222,384 (868.69 KB)
Trainable params: 222,384 (868.69 KB)
Non-trainable params: 0 (0.00 B)

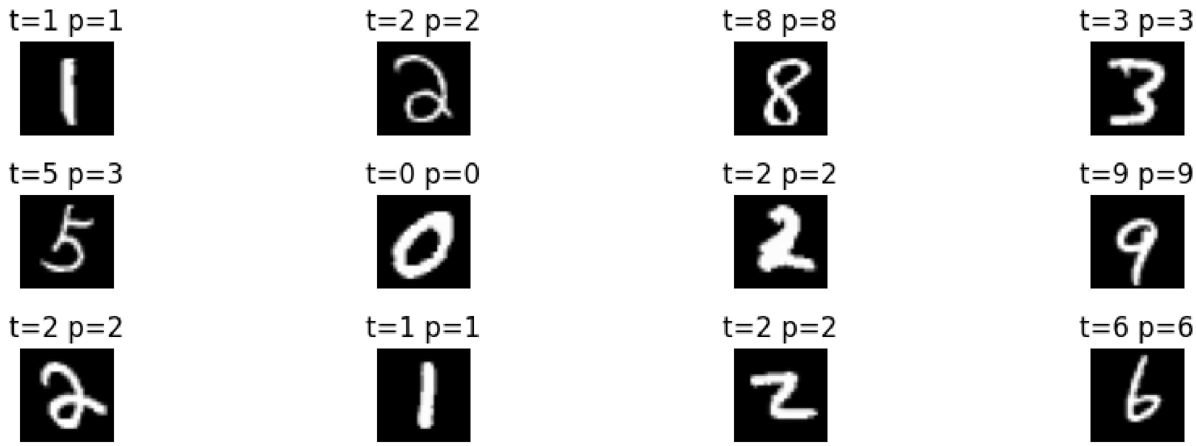
```
AE train time (s): 50.99489460499899
AE test reconstruction loss: 0.013000349514186382
```






```
SVM accuracy: 0.8523571428571428
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      1343
           1       0.92      0.97      0.95      1600
           2       0.82      0.82      0.82      1380
           3       0.80      0.83      0.81      1433
           4       0.82      0.83      0.83      1295
           5       0.79      0.75      0.77      1273
           6       0.89      0.90      0.90      1396
           7       0.87      0.89      0.88      1503
           8       0.85      0.79      0.82      1357
           9       0.80      0.75      0.78      1420

    accuracy                           0.85     14000
   macro avg       0.85      0.85      0.85     14000
weighted avg       0.85      0.85      0.85     14000
```





Conclusion

We have successfully built a fully functional neural network library from scratch. The library was rigorously tested via Gradient Checking, validated on the XOR problem, and capable of training deep Autoencoder architectures. The project demonstrated the power of non-linear feature extraction and provided deep insights into the mathematics of backpropagation.

Appendix

GitHub Repository

https://github.com/CSE473s-Course-Project-Fall25/neural-network-lib.git