Homework 4: Multilayer Perceptron, LSTM, and Autoencoders from Scratch on Financial Data (100 points)

Prof. Amine Aboussalah
Due Date: Friday 28th November 2025.
Submission will be done on Brightspace.

Datasets

- **Multilayer Perceptron**: S&P 500 Daily Prices (Predict next-day Up or Down movement)
- **LSTM**: USD-EUR Exchange Rates (Forecast next day's price)
- **Autoencoder**: Credit Card Transactions (Detect fraud through anomaly detection)

Overview

In this homework, you will build deep learning models **entirely from scratch** (only using `numpy` for matrix multiplications) and apply them to real-world **financial datasets**.

You will practice:

- Building an MLP for classification
- Implementing an LSTM for sequence prediction
- Using an Autoencoder for anomaly detection

Do not use TensorFlow, PyTorch, or Keras.

You may use external libraries only for loading datasets and basic plotting (e.g., pandas, matplotlib).

Task 1: Multilayer Perceptron for Stock Movement Prediction (30 points)

Objective

Implement a basic feedforward neural network to predict next-day stock price movement (Up/Down) using historical stock data and simple technical indicators.

Instructions

Implementation (15 points)

1. **Build an MLP (4 points):** 2 hidden layers with ReLU activations and Softmax output for binary classification (Up/Down).
2. **Forward and Backward Propagation (4 points):** Implement both manually, without using built-in frameworks (e.g. PyTorch) for training.
3. **Gradient Derivation (5 points):** Provide mathematical derivation of gradients used in backpropagation
4. **Loss Function (1points):** Use cross-entropy loss for classification
5. **Hyperparameter Support (1 points):** Enable tuning of learning rate, batch size, number of hidden units, and epochs

Application & Evaluation (15 points)

1. **Dataset (2 points)**
    - Use historical stock data (SP500) with engineered features:
        - Previous 5–10 days returns
        - Optional Technical indicators (e.g. simple moving average)
    - Label: Next-day movement: Up=1, Down=0
    - Use **first 5000 samples** for quick experimentation
2. **Data Split (1 point):** Train/Validation/Test = 60% / 20% / 20%
3. **Hyperparameter Experiments (3 points):** Train and test with different learning rates, hidden units, and epochs
4. **Evaluation Metrics (3 points):** Accuracy, precision, recall, F1-score, confusion matrix. Provide a detailed interpretation of the metrics.
5. **Training Visualization (3 points):** Plot training and validation loss over epochs
6. **Analysis (3 points):** Discuss overfitting vs. underfitting and justify architectural choices

Task 2: LSTM for Currency Exchange Rate Prediction (35 points)

Objective

Implement an **LSTM model** from scratch to **predict future USD-EUR exchange rates** based on historical data.

Instructions

Implementation (14 points)

- Build an **LSTM cell** that includes:
  - Forget gate
  - Input gate
  - Output gate
  - Cell state updates **(5 points)**
- Support **mini-batch training** and **sequence generation**. **(4 points)**
- Train using **Mean Squared Error (MSE)** loss (regression task). **(3 points)**
- Implement **gradient clipping**. **(2 points)**

Application and Evaluation (21 points)

- Download USD-EUR exchange rate daily data. (e.g., Investing.com) **(2 points)**
- Normalize the exchange rates between 0 and 1. **(2 point)**
- Create sequences of 30 past days to predict the next day's rate. **(3 points)**
- Split into 80% training / 20% testing. **(2 points)**
- Train your LSTM.
- Report and plot:
  - Training loss curve **(2 points)**
  - Predicted vs. True prices on the test set (scatter plot) **(3 points)**
- Evaluate using:
  - Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) **(3 points)**
- Comment on the performance of your model and compare it to a built-in implementation of LSTM either using PyTorch or any other relevant library. (4 points)

Task 3: Autoencoder for Fraud Detection (35 points)

Objective

Implement an autoencoder from scratch to detect fraudulent credit card transactions by learning compact feature representations. Indeed, when we train only on normal transactions, it becomes very good at reconstructing normal patterns, but bad at reconstructing unfamiliar, anomalous patterns like fraud.

Instructions

Implementation (20 points)

- Build a Feedforward Encoder (1–2 hidden layers with ReLU activations). **(5 points)**
- Build a Feedforward Decoder (symmetric to encoder). **(5 points)**
- Train using Mean Squared Error (MSE) loss (reconstruction error). **(5 points)**
- Support mini-batch training. **(3 points)**
- Allow the latent space dimension (compressed space) to be configurable. **(2 points)**

Application and Evaluation (15 points)

- Load the **Credit Card Fraud Detection Dataset**. **(2 points)**
- Normalize all features between 0 and 1. **(1 point)**
- Train the Autoencoder (on normal transactions only). **(3 points)**
- Test by measuring **reconstruction error** on both fraudulent and normal transactions. **(2 points)**
- Plot:
    - Histogram of reconstruction errors for fraud and non-fraud **(3 points)**
    - ROC curve based on reconstruction errors for fraud detection **(2 points)**
- Discuss:
    - How changing the latent dimension affects fraud detection performance? **(2 points)**

General Requirements (up to -15 points penalty)

- Submit as a **single Jupyter Notebook** named **LASTNAME_FIRSTNAME**.
- Code must be **original** and **well-commented**.
- Include explanations of:
    - How you designed your models
    - How you chose hyperparameters (e.g., learning rate, sequence length, hidden size)
    - Plots must be **clear and labeled**.