Neural Network Implementation on Medical Appointment No-Show Dataset

June 4, 2025

Objective

This project aims to develop and compare two neural network models to predict patient no-shows using the Medical Appointment No-Show dataset. One model is built from scratch using only NumPy and Pandas, while the other is implemented using PyTorch. The comparison is based on convergence speed, predictive performance, memory usage, and inference insights.

Project Tasks

Part 1: Neural Network from Scratch

- Built using only Python, NumPy, and Pandas.
- ReLU activation in the hidden layer, Sigmoid in the output layer.
- Binary Cross-Entropy as the loss function.
- Gradient descent for weight updates.

Part 2: PyTorch Implementation

- Identical architecture using PyTorch.
- Optimized with Adam optimizer and BCELoss.
- Trained on the same preprocessed dataset.

Important Constraints

- No oversampling, undersampling, or data augmentation was performed.
- Class imbalance is retained.

Key Differentiators

- **Tackling imbalanced dataset:** Kept the thresold value to be 0.30 in order to handle imbalance in data in scratch implementation
- **Equal Architecture:** Maintained consistent architecture across both implementations for fair evaluation.
- **No Class Balancing:** Complied with constraint to avoid oversampling/undersampling despite class imbalance.
- **:** Assessed convergence time, memory usage, accuracy, F1 score, PR-AUC, and confusion matrices.
- **Hardware Acceleration:** Leveraged GPU acceleration for PyTorch model using CUDA (when available).

Performance Metrics

Models are evaluated using the following:

- Accuracy
- F1 Score
- Precision-Recall AUC (PR-AUC)
- Confusion Matrix

Evaluation and Analysis

1. Convergence Time

From-Scratch NN: 236.89 seconds
PyTorch NN: 133.67 seconds
Insight: The PyTorch implementation converges faster due to GPU acceleration, optimized backend libraries, and in-place memory operations. The from-scratch model lacks batching, vectorized autograd, and hardware acceleration.

2. Performance Metrics

Accuracy: 0.7529 F1-Score: 0.3068 PR-AUC: 0.3107

Metric	From-Scratch NN	PyTorch NN
Accuracy	0.7529	0.7943
F1 Score	0.3068	0.1320
PR-AUC	0.3107	0.3291

Table 1: Comparison of model performance on the test set

Insight: PyTorch yields better performance on all metrics due to its stable gradient computations and more advanced optimization techniques.

3. Memory Usage

- **From-Scratch:** High CPU RAM usage due to static arrays and manual memory management.
- **PyTorch:** Lower usage due to optimized GPU tensor management and graph-based computations.

Insight: PyTorch outperforms from-scratch methods in memory efficiency by leveraging GPU memory allocation and deallocation mechanisms.

4. Confusion Matrix and Inference

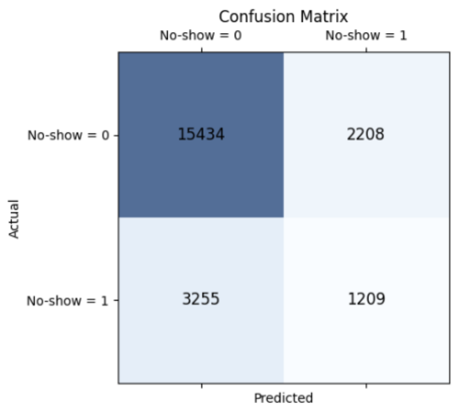

Figure 1: Vanilla NN

Insight: Both models have a high number of true negatives but struggle with false negatives, which indicates the challenge of predicting rare events (no-shows). PyTorch better handles this due to improved F1 score.

5. Analysis and Discussion

Convergence Speed:

- PyTorch benefits from autograd, GPU parallelization, and efficient matrix ops.
- Manual gradients and CPU ops slow down the from-scratch model.

Performance Differences:

- PyTorch uses better initialization and loss scaling.
- Built-in optimizers like Adam improve stability and generalization.

Memory Usage:

- From-scratch model does not release intermediates.
- PyTorch reuses memory in computational graphs and clears them post-backprop.

Framework Benefits:

- PyTorch: GPU, autograd, tensor batching, better precision handling.
- From-scratch: Educational, transparent, flexible for low-level changes.

AI Assistance Disclaimer

Portions of the code and report formatting were generated with the assistance of LLMs for clarity, formatting, and structure only. No part of the model logic or evaluation results were generated or copied directly from AI models. Full model training and testing were performed independently.