**Title:** High Performance Neural Network from Scratch using C++ and OpenMP

**Author:** Anshika Gaur  
**Course:** COIS 4350 – High Performance Computing  
**Instructor:** Dr. Brian Srivastava   
**Date:** April 6th, 2025

**1. Abstract**

This project explores the development of a simple feedforward neural network entirely from scratch in C++ and evaluates its performance with and without OpenMP-based parallelization. The network is trained on the Wine Quality dataset and demonstrates a significant reduction in training time when multithreading is applied. This project showcases how a basic machine learning model can benefit from high performance computing principles, even at a small scale.

**2. Dataset & Preprocessing**

* **Dataset used:** Wine Quality (Red), from UCI Machine Learning Repository
* **Input features:** 11 numerical values per sample (e.g., acidity, pH, sulphates)
* **Target:** Wine quality (normalized between 0 and 1)
* CSV was read using a custom parser and included handling of malformed values.

**3. Neural Network Design**

* Architecture:
  + Input Layer: *11 nodes*
  + Hidden Layer: *16 nodes*
  + Output Layer: *1 node*
* Activation Function*: Sigmoid*
* Loss Function: *Mean Squared Error (MSE)*
* Training Algorithm: *Batch Gradient Descent*
* Epochs: *100*
* Language: *C++*

**4. Parallelization with OpenMP**

OpenMP was applied to the inner training loop using #pragma omp parallel for reduction.  
  
This allowed the program to take advantage of multiple CPU cores when updating loss values during each epoch.   
We measured training times before and after applying OpenMP. The results showed a clear performance boost.

**5. Results & Analysis**

**Performance Table:**

| **Version** | **Training Time (ms)** | **Notes** |
| --- | --- | --- |
| Without OpenMP | 395 ms | Single-threaded |
| With OpenMP | 147 ms | Multithreaded with omp |

**Speedup Calculation:**

Speedup = 395 / 147 ≈ 2.69x

This shows a substantial improvement in training efficiency with parallelism. While the neural network is simple and small, OpenMP still significantly cut down the training time, highlighting how even lightweight models can benefit from HPC techniques.

**6. Reflection & Learnings**

This project helped deepen my understanding of how neural networks function internally, especially when implemented from scratch without using libraries. More importantly, I learned how to parallelize tasks using OpenMP and how to measure performance gains accurately using chrono timers.

This project has been pushed to GitHub for future reference and included in my portfolio site. I plan to experiment further by increasing network depth, comparing with MPI, or deploying this as a Web Assembly demo online.

**7. Files & Submission**

* neural\_net.cpp : full implementation of the model
* wine.csv : dataset
* README.md : summary and usage
* word\_report.docx : this document

**GitHub Repo:** <https://github.com/gauranshika29/hpc-neural-network>

**Portfolio Site (Coming Soon!):** [YourPortfolioURLHere]

*End of Report*