COPS Summer of Code 2025

Intelligence Guild

Club Of Programmers, IIT (BHU) Varanasi

Artificial Neural Networks

20 – 29 May 2025

Official IG Website: https://cops-iitbhu.github.io/IG-website/

All deadlines are strict. No extensions will be granted.

Introduction

COPS Summer of Code (CSOC) is a flagship initiative under the Club Of Programmers, IIT (BHU) Varanasi, with all verticals contributing through focused tracks. This document embarks the journey of deep learning and contains the contents of ANN.

Modules will be released from time to time. **Adhere strictly to deadlines**. Submissions will be evaluated on approach, technical correctness, and clarity. The most technically accurate solution may not necessarily be the one chosen; clarity of thought and a well-reasoned approach will be valued more.

Communities

All communication for the programme will be conducted strictly via Discord. Do not reach out through other channels. Resources and updates will be posted on Github, and all notifications will be made via Discord.

Final Report

A concise report may be submitted along with your final assignment. While **not mandatory**, it may strengthen your overall evaluation. Reports must be written in LaTeX and submitted in PDF format only. We are not interested in surface-level descriptions — focus strictly on your analysis, approach, and reasoning. The report itself constitutes the final assignment. No additional files are to be submitted. Refer to the Assignment section for details.

Contact Details

In case of any doubts, clarifications, or guidance, you can contact one of us. We request that you stick to Discord as the preferred mode of communication for all the questions that you have as it will also benefit others. However, you can reach out to us through other means in case we fail to respond on Discord.

- Yashashwi Singhania - 7905584242
- Vivek Kumar - 9873432572
- Tejbir Panghal - 9034705165
- Tanish Aggarwal - 9569884059
- Sakshi Kumar - 8073247266
- Manav Kumar Jalan - 9395002919

Resources

In this module we will learn the heart of machine learning: **Artificial Neural Network**. This module will be your launchpad into the fascinating world of deep learning, where simple units connect to solve complex problems.

Just a quick note before we jump into the resources, don't feel restricted to only these! If any topic feels unclear or if you're curious to explore further (which we highly recommend if you have the time), feel free to consult other resources. These are simply the ones we found helpful and worth sharing.

Theory:

- If you're looking to truly understand what's happening inside a neural network and not just code it, this is the perfect place to start. 3Blue1Brown Neural Networks video explains core concepts very well.
- This playlist by Sir Mitesh Khapra is great for understanding neural network's maths. It starts from basics and go all the way deep into neural networks.
  Consider watching Lec 2.1 to 4.9, they cover the fundamentals of neural nets.
- If you stil dont understand Backpropagation, then we would recommend watching this video.
- If needed: Derivation of the Gradient of the cross-entropy Loss
- How is the gradient calculated for the middle layer's weights?
- Vanishing and Exploding Gradients in Deep Neural Networks

Coding

Now that you've built a solid theoretical understanding of neural networks, it's time to bring that knowledge to life with code.
This section will guide you through implementing what you've learned.

- This playlist by Sentdex offers a practical introduction to neural network implementation, walking you through the coding process step by step.
- This book by sentdex also includes the process to code neural network.

Frameworks

Once you're comfortable with the fundamentals and have built a basic neural network from scratch, it's time to level up. This is where deep learning frameworks come in powerful tools like TensorFlow, PyTorch, and Keras that simplify model building, training, and deployment. They let you focus more on experimenting and less on reinventing the wheel, making your journey faster, more scalable, and production-ready.

Although we'll be focusing on PyTorch in this module

To get hands-on with PyTorch, here are two excellent playlists to choose from:

- PyTorch for Deep Learning: Videos 1-2
- Deep Learning with PyTorch: Zero to GANs: Videos 1-13

Here is a blog, to get you familiar with PyTorch.

Additional Resources:

- Vanishing Gradient Problem in ANN
- Early Stopping
- Dropout layers
- Feature Scaling
- Batch Normalization
- How to deal with Class Imbalance Chapter -4
- Different type of optimisers: lec 5.1 to 5.9(if you have spare time)

Assignment: Neural Network Implementation on Medical Appointment No-Show Dataset:

Objective

Implement and compare two neural network models for predicting patient no-shows using the same dataset: one built from scratch (without deep learning frameworks) and one using PyTorch.

Data Set: Medical Appointment No-Show Dataset

Project Tasks

Part 1: Neural Network from Scratch

- Build your neural network using only core Python and libraries such as NumPy or Pandas for numerical operations.
- Do not use any deep learning frameworks (e.g., PyTorch, TensorFlow).
- **Avoid** using large language models (LLMs) such as GPT for code generation, but if you do **clearly mention in the report** where you've used it or else you will be disqualified directly.

Part 2: PyTorch Implementation

- Re-implement the same neural network architecture using PyTorch.

Important Constraint

- Although the dataset is imbalanced, do not use oversampling, undersampling, or any data augmentation techniques to balance the classes.

Performance Metrics

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

Evaluation and Analysis

1. Convergence Time

- Measure and compare the training time and speed of convergence for both implementations.
- Reflect on the reasons why one might converge faster than the other.

2. Performance Metrics

Compare the accuracy, F1 score, and PR-AUC of both models on the validation set(or via cross-validation):

- Accuracy
- F1 score
- PR-AUC

3. Memory Usage

- Analyze and discuss the memory consumption differences between your from-scratch implementation and the PyTorch model.

4. Confusion Matrix and Inference

- For each implementation (from-scratch and PyTorch), draw the confusion matrix. Analyze and briefly discuss what insights it provides regarding your model's performance.

5. Analysis and Discussion

- Provide insights into why differences in convergence speed, performance, and memory usage exist.
- Consider factors like framework optimizations, hardware acceleration, numerical stability, and code efficiency.

Submission Guidelines

- Create a GitHub repository named `<roll_number>-CSOC-IG` (e.g., `23014019-CSOC-IG`)
- Repository organization:
  - A folder named Vanilla Neural Networks containing all source code implementations
  - The final report in PDF format, authored using LaTeX

  Everything must be in the github repo itself.
- Submit the repository link via the provided Google Form here
- **Note:** The report constitutes the primary assignment submission. No additional files are required
- **Deadlines are strict and will not be extended**

Final Remarks

Ensure that your submission reflects a clear understanding of the concepts and methodologies applied. Focus on the analytical aspects and the rationale behind your implementations. We look forward to your insightful contributions.

***Adios**, and keep learning!*