Conner Rose

linkedin.com/in/ConnerRose • github.com/ConnerRose • conner.n.rose@gmail.com • (517) 648-1359

Education

University of Michigan, *Ann Arbor, MI* **August 2022 – May 2026**

B.S.E. and M.S.E. in Computer Science, Completing Requirements for B.S. in Honors Mathematics *GPA: 3.79/4.0*

- **CS Coursework**: Programming and Data Structures, Data Science for Engineers, Data Structures and Algorithms, Discrete Mathematics, Machine Learning, Algorithm and Computation Theory, Computer Organization, Web Systems
- **Mathematics Coursework**: Calculus I-IV, Linear Algebra, Combinatorics and Graph Theory, Probability, Real Analysis, Graduate Probability Theory, Advanced Linear Algebra

Experience

Bloomberg L.P., *New York, NY* **June – August 2024**

Incoming Software Engineering Intern

Traders At Michigan, *Ann Arbor, MI* **September 2023 – Present**

Software Engineer – Project Lead

- Lead development of ETF trading game, utilizing **Django** and **React**, to be used in live competition of 100+ traders
- Design and deliver SWE curriculum to club members, supporting their preparation for software engineering interviews

Bloomberg L.P., *New York, NY* **May – August 2023**

CTO Office Intern – Compute Architecture and OSPO

- Designed automated access revocation system using **Python** and **LDAP**, deployed to **Docker**-containerized **Jenkins Pipeline**, ensuring appropriate removal of inactive accounts from Bloomberg's open-source GitHub repositories
- Developed GitHub crawler using **Python** to scan all projects contributed to by Bloomberg employees over 10 years, automating contribution cataloging and open-source license compliance verification, increasing audited projects by **3x**

Projects

Zinger's (ETF Trading Game) **March – April 2023**

Python, JavaScript, Django, React, Redis, Websockets

- Architected ETF Trading Game, utilizing **websocket**-based infrastructure, enabling real-time data and order matching
- Integrated **Redis** with **Django channels**, allowing for fast, single-server, many-client, bidirectional communication
- Implemented **fault-tolerant**, message-based order placing and trade execution system, supporting instantaneous updates while reducing server load and improving client-side performance, as compared to continuous polling

Historical Landmark Image Classifier **October – November 2023**

Python, PyTorch, Pandas, NumPy, Matplotlib, Computer Vision

- Designed and implemented **convolutional neural networks** for multiclass image classification of historical landmarks
- Researched **model architecture** and **data augmentation**, employing subsampling and noise generation to improve accuracy and mitigate overfitting while training model with 5 convolutional layers and **+2,000,000** learnable parameters
- Utilized **transfer learning**, leveraging multiclass model to initialize weights for training on binary classification target problem, reducing training time, preventing overfitting, and fine-tuning fully-connected layers to improve performance

Movie Review Prediction System **September – October 2023**

Python, Scikit-learn, Pandas, NumPy, Gensim, Matplotlib, Natural Language Processing

- Trained **support vector machines** capable of classifying positive and negative movie reviews achieving **92%** accuracy through **sentiment analysis** techniques, including learned **word association** and **negation handling**
- Leveraged **word embedding association test** to determine association of gendered language with positive/negative adjectives in reviews, identifying gender bias within dataset and resulting learned support vector machines

MST/TSP Solution Generator, *C++* **April 2023**

- Utilized **arbitrary insertion** heuristic approach to generate approximate solutions for the **traveling salesperson problem** with quadratic time complexity, allowing for computation for **+10,000-order** complete graphs in seconds
- Developed **branch and bound** algorithm to guarantee optimal solutions to the traveling salesperson problem and optimized via **solution tree pruning**, using MST-derived upper bound, reducing runtime by **90%**

Technical Skills

Languages: Python, C++, Rust, Java, JavaScript/TypeScript, HTML/CSS, SQL (SQLite), $\LaTeX$

Tools: Git, Docker, Jenkins, Django, Flask, React, Jupyter Notebook, MongoDB, Pandas, NumPy, Scikit-learn