Alan Yuan

alan.yuan.jly@gmail.com | linkedin.com/in/jalnyn | github.com/jalnyn

Work Experience

Amazon May 2022 – Aug 2022
Software Developer - Intern *Vancouver, British Columbia*

- Engineered a microservice in **Java** to send notifications to **100mil+** customer of cashback on select products
- Ensured modularity by designing a plugin system for processing the customer orders for microservice
- Utilize **AWS** webservices such as **Lambda**, **SQS** and **SNS** to ensure scalability of the notification system

Intel May 2021 – May 2022
Software Engineer - Intern *Toronto, Ontario*

- Developed flagship product using **C++**, **Python** and **Bash** for speedup by re-routing the compilation
- Developed support software to generate 4000+ of completely random test-cases for edge-case testing
- Optimized support tool's Ram templates to reduce false positives and failing cases by around **70%**
- Implemented various new features and upgrades such as re-scripting tools to utilized new control system, dynamic database size notifier, hierarchy re-router for missing entities, automatic parameter setting aggregator and more
- Migrated all 600+ failing regression test cases to the new compilation flow leading to decreaesed build failures

Centivizer April 2020 - Sept 2020
Software Developer - Part-time *Toronto, Ontario*

- Designed and wrote backend application using **Node.JS** and **SimplePeer** to connect users via video call
- Establish communication between client and backend for video feed using **socket.io**
- Integrated video feature with user database through **RESTful API** using the **Axios** Library
- Decreased server load by re-working notification system to use a socket based approach

Education

University of Toronto 3.82 cGPA
BSc Computer Science Specialist, Major in Mathematics *Sep. 2018 – May 2023*

Relevant Coursework: Data Structures and Algorithms (A+), Operating Systems (A+), Parallel Programming (A+), Neural Networks and Deep Learning (A+), Intro to AI (A+), Introduction to Machine Learning (A+), Algorithm Design, Analysis & Complexity (A)

Projects

PAIR lab assistant | Private repo (paper under review) Sep 2021 – present

- Built on top of **Nvidia**'s Isaacsim to create a robot reinforcment learning **benchmark**
- Implementing a task-flow and enviroment randomizer for **causal reward** based research
- Creating physics scenes and testing **reinforcment learning** algorithms to be used as a benchmark
- Utilize **pytorch** and **PPO** implementations such as **rslrl**, **rlgames** and **rllib**
- Setup and trained a variaty of robots including **frankas and mobile manipulators**

Deep QLearning snake | Private repo June 2021 – July 2021

- Utilized **pytorch** to write a Deep Q-Learning algorithm
- Played the snake game with DQL agent reaching a high score of **40** after **5** minuites of training

CFR Minimization (Kuhn Poker, Tic-Tac-Toe and Coup) | Link: GitHub May 2021 – July 2021

- Developing a general framework to find nash equilibrium using CFR, CFR+ and MCCFR
- Implemented each of the algorithms to play tic-tac-toe and Kuhn poker

Tenant-Landlord Matching App | Links: server-side, client-side Aug 2020 – June 2021

- Fullstack development of an mobile application to match landlords and tenants
- Constructed front-end using **React Native** and common packages such as **React Navigation** and **axios**
- Features: Authentication, Images upload utilizing **multer**, Tinder-like swiping, instant messaging with **Socket.io**
- Utilized **Node.js**, **GraphQL**, and database **Postgres** to construct backend

CaNetDa: Deep learning for GeoGuesser in Canada | Link: GitHub Jan 2021 – April 2021

- Utilized a deep learning approach utilizing multiple deep learning techniques to have an AI play GeoGuesser.
- Utilized a **pytorch** implementation of **ResNet**, **EfficientNet** and **Vision Transformer** to predict the location
- With our approach, a accuracy of **60%** was consistently achieved out of 13 options

Tron UDP multiplayer | Link: GitHub Sep 2019 – Dec 2019

- Created a four player game for local networks using the **UDP** network protocol and C++
- **Forked** timer from the server to ensure the game runs on time
- Utilize **epoll** for both client and server to monitor the socket as well as the timer (server) and stdin (client)

BF-interpreter | Link: GitHub Mar 2018 – Nov 2018

- Built **interpreter** that runs BF in C
- Reads user input in **real-time** as BF shell and reads BF files
- Runs all example BF programs found on wikipedia

Technical Skills

Languages: Python, C/C++, JavaScript, Java, C#, R
Tools: Git, React Native, Node.js, MongoDB, SQL (Postgres), PyTorch, Numpy, Pandas, GDB, GraphQL