Zara Syed

Algorithms — Optimizations — Firmware
+1 647 284 5350 zara.syed@uwaterloo.ca linkedin.com/in/zarasyeduw zarasyed.com

Skills

Languages: Python, C/C++, Matlab, HTML/CSS, SQL
Libraries: Tensorflow, Keras, PyTorch, Scikit-Learn, SciPy, Pandas, Numpy, Matplotlib
Tools: Simulink, vFlash, CANalyzer, Azure, Docker, Git, Regular Expressions, Jenkins, PTC

Experience

Magna Powertrain **September 2024 - Present**
Base Software Engineering Intern *Troy, MI*

- Revolutionized requirements traceability and achieved 100% audit readiness by automating requirements linking of 4000 functions across 10 million lines of C code using Python, Clang, LLVM, RegEx, and Excel.
- Automated performance evaluation consolidation for customer updates, with 99% task completion time reduction, parsing 130+ HTML Unit Test reports using Python, RegEx, Jenkins, and Excel saving 8-10 hours each release.
- Enhanced vehicle software reliability by developing CAN traffic analysis tool, using Python, RegEx, and CANalyzer to detect anomalies in millions of lines of diagnostic data in seconds.
- Conducted in-vehicle tests to evaluate CPU load across maneuvers, software versions, and vehicle types (PHEV and ICE) using vFlash and CANalyzer.

Magna Powertrain **Jan 2024 – Apr 2024**
Control Algorithms and Software Engineering Intern *St. Valentine, Austria*

- Developing patent-eligible deep learning solution for motor control systems, demonstrating graduate-level research rigor as an undergraduate.
- Eliminated time-intensive PID controller tuning process by replacing controller with reinforcement learning agent.
- Implemented real-time Python-MATLAB-Simulink synchronization interface for over 100+ hours of model training.
- Designed and implemented reinforcement learning algorithm and engineered mathematical reward function for motor control optimization within custom Gymnasium environment.

Magna Mechatronics, Mirrors, & Lighting **May 2023 – Sept 2023**
Machine Learning DevOps and Software Engineering Co-op *Newmarket, ON*

- Developed and deployed machine learning web app to advise engineers' automotive material choices by predicting stress-strain curves, using Python, Tensorflow, Flask, SQL, Docker, Azure DevOps, Azure App Services and with CI/CD.
- Engaged in cross-functional and international collaboration, including colleagues in Italy, China, and India.
- Trained machine learning model to estimate friction coefficient in automotive part materials with Python, Tensorflow, Keras.

ON Semiconductor **Sep 2022 – Dec 2022**
Digital Signals Processing Algorithm Developer *Waterloo, ON*

- Developed 32-bit fixed-point firmware functions for LPDSP32 using C, including signal windowing.
- Reduced memory usage by 75% and cycle count by 45% by leveraging conditional compilation and cyclical addressing in signal windowing function.
- Profiled cycle counts of 15+ functions using ChessDE and reported to customer facing documentation.

XSENSOR Technology Corporation **Jan 2022 – Apr 2022**
Machine Learning Intern *Calgary, ON*

- Developed Human pose estimation (HPE) pipeline which processed 2 million+ sensor inputs using Tensorflow, Keras, Pandas, Numpy, and Multiprocessing.
- Developed 85% accurate Anthropometric meta data extraction functionality for HPE pipeline.
- Built digital filter tuner used to tune FIR parameters to 87% accuracy for biosignal extraction.
- Prepared dataset report and augmentation and expansion strategy for CEO with 500k+ data points.

Projects

FashionMNIST Classication | *Python, PyTorch, Jupyter Notebook* | GitHub **Nov 2024 - present**

- Implementing GPU accelerated training of Convolutional Neural Network (CNN).

Real Time Operating System | *C, STM32* | GitHub **Sep 2023 – Dec 2023**

- Developed kernel and functionality for thread creation, thread scheduling, and multithreading.

Bluetooth Robotic Claw Arm | *Arduino Uno, Arduino mini* **Apr – June 2023**

- Robotic claw arm mimics real time human action using accelerometers, gyroscopes, flex sensors, DC & servo motors.

Autonomous Vehicle Simulation | *Python, Tensorflow* **Jan 2019 – Mar 2019**

- Built CNN to train self-driving car using end-to-end learning and computer vision on Udacity's self-driving car simulator.

Education

University of Waterloo **Sep. 2021 – April 2026**
Candidate for BASc, Honors Mechatronics Engineering *Waterloo, ON*

- **Relevant Courses:** Embedded Systems, Microprocessors, Computer Architecture, Real Time Operating Systems, Data Structures and Algorithms, Circuits, Power Electronics, Statistics

Zara Syed

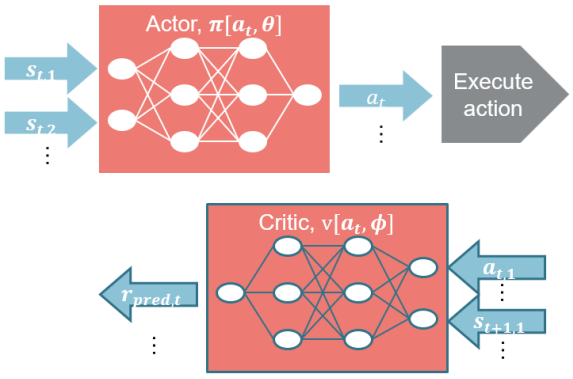
Algorithms • Optimizations • Firmware
Mechatronics Engineering Student @ UWaterloo

+1 647 284 5350
zarasyed.com
zara.syed@uwaterloo.ca
linkedin.com/in/zarasyeduw

Reinforcement Learning: Redefining Motor Control (Article)

Overview

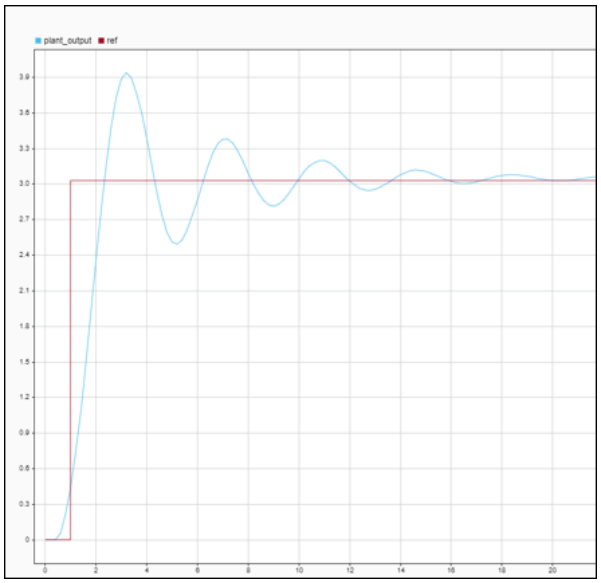
I developed an advanced deep learning solution for motor control systems, simplifying the complexity of traditional model-based control algorithms by replacing them with a single, adaptive reinforcement learning (RL) agent. Using an Actor-Critic Deep Deterministic Policy Gradient (DDPG) method, I trained the RL agent to replicate the functionality of a PID controller. This solution eliminates the need for time-consuming PID tuning while delivering precise, adaptive control across a wide range of operating conditions.



Actor-Critic Method

Learning Environment

The learning environment is built in Simulink, including a motor, a reference value, and a feedback loop. The motor is modeled by a plant with an arbitrary transfer function. The DDPG algorithm in Python processes real-time simulation data to train the agent. To ensure seamless interaction between Python and Simulink, I implemented a synchronization mechanism that allowed the simulation to run step-by-step, sending data to Python at each step, ensuring a smooth workflow.



Training Results: Plant Output Approaches Steady-State

Model Training

The learning framework involves 2 neural nets, based in Python: an actor, and a critic. The actor outputs what actions, or control value, it thinks the RL agent should perform in the environment. In Simulink, this action is executed. The updated environment state data is sent back to Python, where the critic attempts to predict the reward, as it does not have access to the reward function. The actual reward function output is compared to the critic's prediction, and the resultant error is used to advise both networks' updated parameters via backpropagation.

Results and Takeaways

The RL agent was able to reach the reference value. I was really surprised to see that the trained RL agent replicated the exact behavior of a PID controller, without ever having any reference to a PID controller. I also internalized that the reward function design is a strong contributor to the agent's learning!

Zara Syed

Algorithms • Optimizations • Firmware
Mechatronics Engineering Student @ UWaterloo

+1 647 284 5350
zarasyed.com
zara.syed@uwaterloo.ca
linkedin.com/in/zarasyeduw

No Libraries, No Shortcuts: Engineering Backpropagation from Scratch (Github)

Overview

To deeply understand the ins-and-outs of the backpropagation algorithm, I took on the challenge of building and training a neural network completely from scratch, without the use of any machine learning libraries. To demonstrate the application of the algorithm, the model classifies species of iris flowers. An in-depth code and algorithm walk-through is available at my Github.

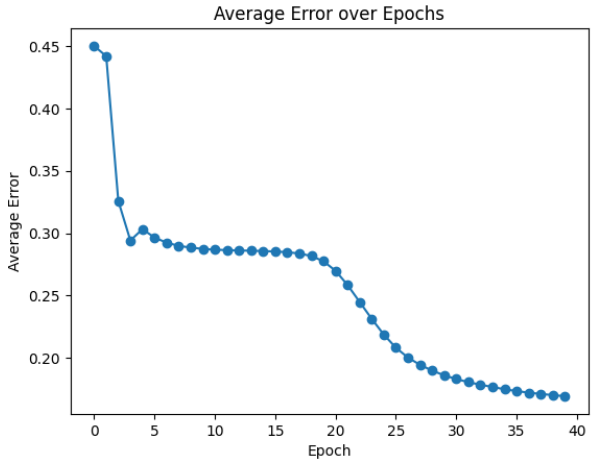
Image From UC Irvine Machine Learning Repository

Model

The network had three layers: an input layer, a hidden layer powered by the Leaky ReLU activation function, and an output layer using the Softmax function for probabilistic predictions.

Algorithm

After much literature review to fully understand the backpropagration algorithm, I developed my own implementation, applying calculus concepts including gradients, partial derivatives, and chain rule. The resulting functionality updates the parameters of the model in a ”backwards” fashion, hence the name: *backpropagation*. In short, for each prediction of the model, the error is computed with respect to each parameter. Then, a proportional update is made to each parameter. For more details, find my report on my Github repo.



Training Results: Error Over Time

Results

The maximum accuracy achieved was an impressive 97%, demonstrating a clear drop in error over time, proving the work of the backpropagation algorithm. This experience reinforced my passion for diving deep into complex algorithms and turning theory into results.