Hesam Pakdaman

hesampakdaman

Programming languages

Python Lisp Rust

Experience

H&M Group

Software Engineer
Mar 2023—
Developing an e-commerce platform named Creator Studio; for designing, ordering, and selling custom print-on-demand merchandise globally, with a focus on sustainability and minimal environmental impact.
Kafka Kubernetes MongoDB PostgreSQL Terraform

Budbee

Machine Learning Engineer
Apr 2022—Feb 2023
Budbee (part of Instabee Group) is a Swedish last-mile delivery company founded in 2016, focusing on efficient deliveries for online shopping across several European countries. I was part of the ML team, providing predictions and data insights to support various departments within the company.
LightGBM Metaflow MySQL PyTorch Snowflake

Entecon

Machine Learning Engineer
Feb 2021—Mar 2022
Entecon is a Swedish consultancy firm. I was contracted to work for Nielsen, a US-based company providing advanced video metadata solutions to leading media companies. My role was to assisst the team responsible for video segmentation, focusing on accurately categorizing commercial and program content.
Pandas PyTorch Matplotlib NumPy

DING

Machine Learning Engineer
Jan 2018—Feb 2021
At DING I was hired as a concluntant for Convini, a Swedish company providing workplace food solutions through self-service stations. I designed a deep learning system using cameras mounted on fridges to accurately detect which products customers selected.
CUDA CloneZilla FLIR NumPy PyTorch

Education

KTH Royal Institute of Technology
Civilingenjör i Teknisk fysik
- 2015—2018 MSc. Computer Science
- 2012—2015 BSc. Engineering Physics

Recent hobby projects

1 billion row challenge. This challenge involves processing one billion temperature measurements to compute the minimum, mean, and maximum temperatures per weather station. I implemented this in Rust, leveraging its std::sync::mpsc and std::thread standard libraries for efficient, parallel data handling and memory-mapped files for optimized I/O performance. By customizing the hash function for the dataset, the project achieves high performance, processing the entire 13GB input file using all available CPU cores. On a MacBook M1 Pro (2021), it processes the input file in ~2.75s, showcasing Rust's high-performance capabilities.

Integer factorization. In this project, I explored various number factorization algorithms. The aim was to learn more about Rust's advanced features such as generics, dynamic dispatch, and procedural macros while implementing design patterns like Strategy, Builder, and New Type. The project includes algorithms such as Miller-Rabin, Fermat's factorization method, Pollard's Rho, and trial division. A trait system was used to apply these strategies, and a test framework was developed using the Builder pattern to ensure robust and modular code.