Hesam Pakdaman

hesampakdaman

Programming languages

Go | Python | Lisp | Rust

Experience

Software Engineer
Aug 2024—Feb 2025
Einride is a pioneer in autonomous and electric transportation, dedicated to delivering sustainable and efficient logistics solutions. I am part of the Data team, responsible for collecting telemetry from all vehicles and ensuring it is accessible to all engineering teams.

Einride

GCP | gRPC | PostgreSQL | Pub/Sub

Software Engineer
Mar 2023—June 2024
I helped developed the e-commerce platform Creator Studio, a venture within global fashion company H&M, for designing, ordering, and selling custom print-on-demand merchandise globally with an emphasis on minimal environmental impact.

H&M Group

k8s | Kafka | MongoDB | PostgreSQL

Machine Learning Engineer
Apr 2022—Feb 2023
Budbee is a Swedish last-mile delivery company focusing on efficient deliveries for online shopping across Europe. I was part of the ML team, providing predictions and data insights to support various departments within the company.

Budbee

LightGBM | Metaflow | MySQL | PyTorch

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 assist the team responsible for video segmentation.

Entecon

Matplotlib | NumPy | Pandas | PyTorch

Machine Learning Engineer
Jan 2018—Feb 2021
Hired at DING as a consultant for Convini, a Swedish company providing workplace food solutions through self-service stations. I built a deep learning system using cameras mounted on fridges to detect products customers selected.

DING

CUDA | FLIR | NumPy | PyTorch

Education

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

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 and a customize hash function tailored for the dataset. The project achieves high performance, processing the entire 13GB input file using all available CPU cores. On an Apple M1 Pro it processes the input file in ~ 2.75s.