Mingyong Ma

3869 Miramar St, San Diego, US

858-539-6919 m7ma@ucsd.edu My Website github.com/mamioma

Education

University of California, San Diego **Sep. 2022 – June 2023**
Master of Science in Computer Science *San Diego, California*

Skills

Language: C++/C, Python, Go, Java, Spark, SQL, Javascript, HTML, CSS
Tools: fastAPI, Azure, REST API, Redis, Kafka, Spark, Pytorch, Git, LangChain, Postgres DB, Docker, Jmeter

Experience

Adobe (GenAI platform Team) **June 2023 – Sep 2022**
Software Engineer Intern *San Jose, California*

- Integrated **LLM** model **fine-tuning** and **inference** to Adobe AI platform **Firefall** using **LangChain** and **fastAPI**.
- Created a service that user can submit a **fine-tuning** task, and this request will be forwarded to **MMS** (Model Management System), which will download our code from remote artifactory **JFrog**, integrate into MMS and build a **Docker** container. After fine-tuning task is finished, it will save the fine-tuned model in **Azure Blob Storage**.
- Improved the fine-tuning API call from **blocking** to **asynchronous** utilizing **Message Queue**. This modification has reduced **overhead** and boosted **latency** by 90%.
- **Reduced the network I/O** from **13GB** to **32MB** per **inference** call. By utilizing **PEFT**, the based model is consistent for every fine-tuning job, thus is stored in **Redis** of the **Docker** container, with only the **Lora layer** being stored in **Azure Blob Storage**. Therefore, only the Lora layer (32MB) instead of the entire model (13GB) is downloaded into Docker container.
- Used **Jmeter** for **load-testing**, able to generate **1600 TPS** (token per second) with **multi-threading**.
- Innovatively proposed how to fine-tune **LLaMa2-7b** on a **CPU**, which offers alternative choice to save cost. No need to run GPU entirely a day.
- Implemented using **REST API** that able to **CRUD** a task, and save it in **postgres DB** with **almebic** version control.

Amazon (Camera & Perception Team) **June 2022 – August 2022**
Software engineer Intern *Shenzhen*

- Developed an **image processing** algorithm that combines **deep learning** techniques with the **Unsharp** algorithm, achieving 20% superior results compared to the camera algorithm used in tablets.
- Utilized **Canny Operator** for **edge enhancement** and **Unsharp** for mid-frequency enhancement. And introduced **ESR-GAN** to restore general real-world images by synthesising pairs with a more practical **degradation** process.
- Achieved automatic **object detection** on portraits utilizing **YOLOv5** and Implemented more refined **Super-Resolution** for every portraits.
- Conducted an evaluation of our proposed algorithm using **Imatest** software in the Amazon lab, observing an increase in **MTF-50**, which showcases an improvement in image sharpness.

Lenovo (Digital Transformation Team) **Nov 2021 – Feb 2022**
Data Analytic Intern *Beijing*

- Developed Spark SQL Catalyst Expressions (i.e., SQL functions) using Java/Scala to optimize the performance of DataFrame Transformation. Employed **Spark DataFrame** and **MapReduce** for data extraction from IDC and GFK.
- Conducted **time series forecasting** to predict future sales of Lenovo's notebook products and tablets, utilizing Lenovo's historical sales data as well as data from other companies such as IDC and GFK.
- Increased the forecasting accuracy of the model by 4.2% by implementing machine learning algorithms such as **Prophet** and deep learning models like **LSTM** or **GRU**.

Projects

Backend Intelligent ChatBot | *Java, Mybatis, Redis, Kafka* **January 2023**

- Developed a WeChat Work group assistant robot using **Java**, which possesses group management functions and intelligent Q&A features among others.
- Implemented communication between Android clients and the server using **WebSocket** and **Java Session API**; wrote code under the **Spring Boot** and **MyBatis** frameworks to interact with **MySQL**, maintaining the online legal aid service for the Longhua District People's Court in Shenzhen.
- Utilized middleware like **Redis** and **Kafka** to add multi-threading functionality for the publishing and subscribing of event and message storage, capable of handling over 1MB of message data per instance.