Dataworkz

Eric Koepke

Data/Machine Learning Engineer

Born 08-11-1995
Living in Amsterdam

ME

Data Engineering	Apache Spark, Databricks, dbt, Pandas, Looker, AWS DMS, AWS Kinesis(Kafka)
Programming languages	Python, Java, C++, R, JavaScript, Lua, Bash, Powershell
Software Development	API development, FastAPI, Git/SCM, GitOps, Docker, CI/CD, Software Architecture, Agile, Scrum, DevOps
Machine Learning	PyTorch, TensorFlow, MLFlow, scikit-learn, NumPy, Hugging Face
Soft skills	Taking ownership of projects, Experimenting with new technologies, giving constructive feedback

I'm a curious guy with a passion for data and always looking for the next challenge to solve. I love **building data centric solutions** and honing my skills in **coding**, **software development** and **solution design**. I find it very important to keep software and architecture **as simple as possible** because it enables a product to be used and developed by a lot of people **over its lifetime**. For example, at October I **migrated** multiple **microservices** that were hosted on different sorts of endpoints to the same **serverless architecture**, that was easier to setup and maintain.

I started studying computer science because of my passion for programming and technical and mathematical problems. I then moved more towards **data science** and **machine learning** and now I am combining these different skills while focusing on **data engineering** and **MLOps**. I am comfortable using **Python**, **SQL** and other languages to **transform data** and **build data pipelines** with frameworks like **Spark** and **dbt.** In addition, I can build **complex machine learning** solutions like the continuous learning framework I built for Precibake. **Productionizing a model**, **building an API** around it and deploying it as a **containerized application** ultimately rounds off my skills and makes sure that my work is impactful. I'm at my best when I can combine my expertise from these different areas and help out others in a team.

Languages

Dutch (Fluent), English (Fluent), German (Mother tongue)

Education

Master’s in computer science
Technical University of Munich 2018 – 2020, Graduated among the best 25% of my class

Bachelor’s in computer science
Technical University of Munich 2018 – 2020

Working Experience

Data/Machine Learning Engineer | October | 11/2021 – 12/2023

October is a fintech startup that lends money to SMEs with the help of semi-automated data-driven software that is developed in house. As a machine learning engineer in a team of 5, I was involved in almost **all aspects of the data stack**: Functional and technical specification, data engineering, model development, model deployment, microservice setup, software development, monitoring, analytics, documentation and support.

One of the core products at October is a **risk model** that uses financial data from customers (SMEs) that apply for a loan from October. When I joined, the first version of that model had been in production for around 2 years without any major changes. The data had been collected and transformed ad hoc in the **Databricks** notebook environment, and the model was stored and committed to the repository hosting the model API. I was tasked with building an **architecture to monitor the performance of the model in production and enable regular semi-automated re-trainings**.

Model Monitoring Architecture

To be able to prioritize development better, the **monitoring** of the existing model had to be improved. So far, only the response from the model was saved in the database by the monolith backend application that was developed by a separate team. The feature data required to detect **data drift** was calculated inside the API, but not saved anywhere. To enable the team to properly monitor their own services, I started **logging** this data inside the API, output it to a stream in JSON format, and use **CloudWatch and Kinesis** to filter those logs and send them to an **S3 bucket** where they could easily be integrated into our **data lake**. Using our BI platform **Looker**, we could get a live **population stability dashboard** for the risk model to detect when performance degrades.

Data Pipeline Architecture

I was also responsible for developing a set of **pipelines** to prepare the data for model training using **PySpark in Databricks**. I needed to **join** together different sources of data, prevent introducing **biases**, and **validate** the correctness of data based on business rules. One challenge was to make sure that a number of transformations made in PySpark for **batch evaluation** would mirror the same transformations as they are applied in the API during **inference**, while allowing for the addition of new features later on. The solution we found was to define these transformations with **Pandas** in the API repository, use **Databricks Repos** to access them from the pipeline and the **Spark integration with Pandas** to execute them efficiently on the **Spark cluster**. Apart from the feature engineering pipelines for data science, I used **dbt** to organize the **data lake** more efficiently. By **involving and educating data analysts** about writing SQL in a repository with **automated tests** and the benefits of **CI/CD**, I enabled them to cooperate better and together we were able to build a growing ensemble of valuable views on the company data.

Model pipeline architecture

Finally, I established a model training pipeline that produces a new model version and compares its predictive performance against the last version. I used **MLFlow within Databricks** to log the model, metrics and parameters to create an organized **catalogue of versioned models.** The model was retrained a few times with relative ease after observing a data drift in Looker, while always **keeping business stakeholders involved and informed** to make sure that the reasons behind the data drift were understood. I changed the **deployment** from hosting the model on the API endpoint itself to using a **Databricks-managed** endpoint that would be called by the API handling the feature engineering and business context. This made the choice of model and its **software dependencies** independent of the rest of the codebase, which gave the team more flexibility going forward.

I did functional and technical specification, data engineering, model development, model deployment, microservice setup, software development, monitoring, analytics, documentation and support

I used Spark, AWS, Databricks, dbt, Pandas, MLFlow, FastAPI, Python, SQL, scikit-learn, Looker, Docker

Research Assistant | Robotics Chair at TUM | 10/2020 – 12/2020

After finishing my Master's thesis at the Robotics Chair at TUM, I was invited to continue my research as a temporary research assistant as **part of a research group**. For my thesis, I had researched the memory capabilities of **spiking neural networks** when used for simple **sequential tasks** like replaying an input signal with a delay. For this project, I was tasked with applying the learnings to a **reinforcement learning** problem because the group was focused on advancing robotic control through **neural networks**. Spiking neural networks can potentially be resource-lite enough to be used onboard a small robot.

I was given access to the university-hosted **LRZ compute cluster** and developed a small **scheduling** program that would use available slots on the cluster to deploy pre-defined experiments as **nvidia-docker containers**. This allowed me to do efficient **hyperparameter searches** and to pursue multiple research questions in parallel. I was able to deepen my understanding of **containerized applications** and learned how to access **GPU resources** within containers.

Additionally, I designed a **PyTorch reinforcement learning framework** that implements **Q-learning** for arbitrary environments and different actor policies and uses **multiprocessing** and clever transfers of data between CPU and GPU to fully exploit all available computational resources. **Optimizations** of the framework, again, allowed me to iterate on model architecture and parameters faster and more efficiently.

I did deploy GPU-enabled Docker containers on university cloud, develop training flow using multiprocessing, contribute as part of a research group

I used (Nvidia-) Docker, Python, PyTorch, TensorFlow

Data Innovation Lab | Precibake | 04/2019 – 08/2019

During my Master's program, I took part in the TUM Data Innovation Lab which brings together groups of **students from different academic disciplines** and sends them to an **industry partner** to work on a real-world data problem. Together with three other students with mathematics, statistics and biostatistics backgrounds, I got selected to solve a **continuous learning** problem at the startup Precibake (now Precitaste). For industrial bakeries, they fit ovens with webcams to allow monitoring and analysis of baking processes without manual configuration. As part of that product, a **classifier** detects what type of pastry is currently being baked in the oven. Because of evolving conditions, **data drift** had been observed, and the classifier was already automatically retrained regularly. With a high number of ovens in production that generate data continuously, the **data set was growing rapidly,** so training on the whole dataset was becoming unfeasible. The challenge: Automatically **retrain** the classifier on new data to adapt quickly while **keeping existing capabilities intact**.

In this context, I first encountered **Docker** because PreciBake was using it internally and it provided a simple way to give us access to compute resources and data. As the only one in the group with a computer science background, I quickly learned more about the system and **provided scripts and help for the team when running experiments on PreciBake's servers through Docker containers**.

As the project progressed, we implemented multiple state-of-the-art continuous learning techniques in **PyTorch**. Multiple of those restricted the change of weights in the model while re-training with new data to retain knowledge that was previously learned. As a consequence, I established a set of functions and classes that **implemented the core processes and provided abstractions** to add new techniques using **inheritance**. This **allowed everyone in the team to easily implement** the details of a different approach based on this framework and have **comparable results** in the end.

I did deploy Docker containers on a server, build extendable model training framework, take leadership on data infrastructure

I used Docker, Python, PyTorch, Git, scikit-learn

Hobbies

One of my favorite activities is tinkering with tech. This often involves software, but I also like to repair electronics or do some hands-on crafting. Apart from that I love languages. I like learning them, but I'm also interested in grammar, etymology and constructed languages. I like to play board games with friends and enjoy visiting museums. When I'm feeling more adventurous, I like to take my longboard for a ride, go bouldering or take a bakfiets to Ikea.