Einführung in MLOps

04 MLOps Definition und Herausforderungen

Tobias Mérinat
teaching2025@fsck.ch

Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN

Department of Information Technology
Lucerne University of Applied Sciences and Arts
6343 Rotkreuz, Switzerland

14. und 15. Februar 2025

Prototypes are easy, Production is hard

Elon Musk

@elonmusk · **Follow**

Prototypes are easy, volume production is hard, positive cash flow is excruciating

9:40 PM · Jan 14, 2021

Warum AI Vorhaben fehlschlagen

The Root Causes of Failure for Artificial Intelligence Projects and How They Can Succeed

Avoiding the Anti-Patterns of AI

James Ryseff, Brandon De Bruhl, Sydne J. Newberry

1. stakeholders often misunderstand what problem needs to be solved using AI
2. many AI projects fail because the organization lacks the necessary data to adequately train an effective AI model
3. in some cases, AI projects fail because the organization focuses more on using the latest and greatest technology than on solving real problems for their intended users.
4. **organizations might not have adequate infrastructure to manage their data and deploy completed AI models, which increases the likelihood of project failure.**
5. in some cases, AI projects fail because the technology is applied to problems that are too difficult for AI to solve.

Technical Debt in ML

Hidden Technical Debt in Machine Learning Systems

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips
{dsculley,gholt,dgg,edavydov,toddphillips}@google.com
Google, Inc.

ML systems have a special capacity for incurring technical debt, because they have all of the maintenance problems of traditional code plus an additional set of ML-specific ssues. This debt may be difficult to detect because it exists at the system level rather than the code level. Traditional abstractions and boundaries may be subtly corrupted or invalidated by the fact that data influences ML system behavior. Typical methods for paying down code level technical debt are not sufficient to address ML-specific technical debt at the system level.

AI & Machine Learning

Key requirements for an MLOps foundation

September 1, 2020

Craig Wiley
Director of Product Management, Cloud AI and Industry Solutions

. . . They have all of the maintenance problems of traditional code plus an additional set of ML-specific issues: ML systems have unique hardware and software dependencies, require testing and validation of data as well as code, and as the world changes around us deployed ML models degrade over time. Moreover, ML systems underperform without throwing errors, making identifying and resolving issues especially challenging. . . .

Quote von hopsworks.ai (emphasis mine)

The existing mantra is that MLOps is about automating continuous integration (CI), continuous delivery (CD), and continuous training (CT) for ML systems. But that is too abstract for many developers. MLOps is really about ***continual development of ML-enabled products that evolve over time****. The available input* ***data (features) changes over time****, the target you are trying to predict changes over time. You need to make changes to the source code, and you want to ensure that any changes you make do not break your ML system or degrade its performance. And you want to accelerate the time required to make those changes and test before those changes are automatically deployed to production.*

MLOps und DevOps

- DevOps -> MLOps
- DevOps ist eine Grundlage für den erfolgreichen und effizienten Betrieb von Software
- MLOps >> DevOps
- MLOps begann als eine Reihe von Best Practices
- und hat sich in Richtung komplettes Lebenszyklusmanagement für Machine Learning entwickelt
- MLOps verbindet Data Science, Software-Engineering, Data Engineering und Operations
- MLOps umfasst Entwicklungskultur für die durchgängige Konzeption, Implementierung, Überwachung, Bereitstellung und Skalierbarkeit von Software mit einer Machine Learning Komponente.

MLOps und DevOps Prinzipien

- CI/CD-Automatisierung
- Workflow-Orchestrierung
- Reproduzierbarkeit
- Versionierung von Daten, Modellen und Code
- kontinuierliches Modell-Training
- kontinuierliche Evaluation von Metriken und Daten
- Nachverfolgung und Protokollierung von Metadaten
- Feedbackschleifen

Herausforderungen (Batch)

- Ausgangslage: Trainiertes Modell ist vorhanden
- Deployment als Batch Pipeline
  - Der einfache Teil
    - Daten lesen
    - Features berechnen
    - Genug Zeit
    - Predictions generieren und speichern
  - Aber was wenn
    - Die Feature-Berechnung mit sklearn und Pandas gemacht wurde, in Produktion aber auf dem Spark Cluster läuft?
    - Die berechneten Features gespeichert werden müssen, um sie wieder zu verwenden oder mit später anfallenden Labels zu joinen?
    - Keine Labels anfallen, und trotzdem Modellverschlechterung erkannt werden muss?

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN

Herausforderungen (Realtime)

- Deployment in einer App
  - Der einfache Teil
    - Einfache Feature-Berechnung mit Request-Daten
    - Modell laden, predicten, Resultat verwenden
  - Aber was wenn
    - Die Feature-Berechnung mit sklearn und Pandas gemacht wurde, die App aber in Typescript geschrieben ist?
    - Die Feature-Berechnung zu komplex für Realtime ist?
    - Das Modell auch Features benötigt, welche nicht im Request daherkommen?

Herausforderungen (Streaming)

- Wenn Realtime Feature-Berechnung nicht skaliert. . .
- . . . und Batch-Features nicht aktuell genug ist,
- benötigen wir Stream Processing
- joinen über die Zeit in verschiedenen Frequenzen aus verschiedenen Feature-Quellen
- Time-Travel für Retraining
  - Features können aus verschiedenen Quellen stammen
  - und in verschiedenen Frequenzen anfallen
  - Labels fallen später im Prozess an

Herausforderungen (Data Drift)

- Ein trainiertes Modell ist eine Momentaufnahme
- Ändern sich die Voraussetzungen, kann sich die Modellperformance verschlechtern
- Permanente Überwachung auf allen Stufen notwendig
  - Rohdaten
  - Features
  - Predictions (wenn möglich)
  - Metriken (wenn möglich)
  - Operative Metriken

High Level Anforderungen

- **Verlässlichkeit**: Auch unter ungünstigen Bedingungen (z.B. einem Hardware Crash oder korrupten Daten) produziert das System weiterhin korrekte Vorhersagen oder gibt zumindest zu erkennen, dass dies nicht mehr der Fall ist.
- **Skalierbarkeit**: Machine Learning Systeme können in verschiedenen Dimensionen wachsen: Gleiche Aufgabe aber komplexeres (und somit hoffentlich genaueres) Modell, mehr Traffic, mehr parallel laufende Modelle, komplexere Feature-Transformationen, strengere Anforderungen an Latenz und Durchsatz.
- **Wartbarkeit**: Dies beinhaltet die Fähigkeit, Probleme zu erkennen, zu analysieren und zu beheben; einfaches und möglichst automatisiertes Deployment, Governance und Reproduzierbarkeit sowie generell einfache und effiziente Zusammenarbeit aller involvierten Teams.
- **Anpassungsfähigkeit**: Machine Learning Systeme bestehen aus Code und Daten. Daten können sich je nach Anwendungsfall schnell ändern, weshalb Machine Learning Systeme effizient und einfach adaptierbar sein müssen. Dies setzt natürlich auch eine hohe Wartbarkeit voraus.

Herausforderungen von einfach bis schwer

- Level 1
  - Experiment Tracking und Model Registry
  - Model Deployment
  - Batch Prediction
- Level 2
  - Streaming Prediction
  - Drift Detection
  - Streaming Feature Engineering
- Level 3
  - Feature Store
  - Feature Platform
- Level 4
  - Continual Learning

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN

Proof of Concept to Production Gap

There is a thing called the **PoC to Production Gap** and it is real