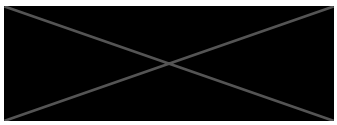
Labor Maschinelles Lernen: ROB43

MLFLOW UI and Tensorboard UI

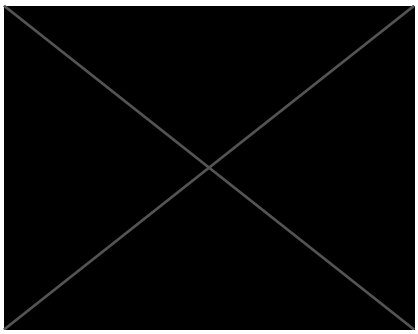
Assignment

eingereicht bei



von

Justin Stange-Heiduk



Datum: 19.02.2025

Abbildungsverzeichnis

1. Einleitung

Machine Learning hat sich in den letzten Jahren zu einer der zentralen Technologien in der Datenanalyse und Künstlichen Intelligenz entwickelt. Dabei werden Modelle trainiert, um Muster in Daten zu erkennen und Vorhersagen zu treffen. Doch mit der zunehmenden Komplexität von Machine-Learning-Experimenten wächst auch die Herausforderung, Modelle effizient zu verwalten, Trainingsergebnisse nachzuverfolgen und Optimierungen gezielt durchzuführen.

Hier kommen Tools wie MLflow und TensorBoard ins Spiel. Beide wurden entwickelt, um den Machine-Learning-Workflow zu unterstützen, unterscheiden sich jedoch teilweise in ihrer Funktionalität und ihrem Fokus. MLflow bietet vor allem Funktionen zur Organisation von Experimenten, Modellversionierung und Deployment, während TensorBoard auf die Visualisierung von Trainingsprozessen spezialisiert ist und tiefergehende Analysen der Modellarchitektur ermöglicht.

Ziel dieser Arbeit ist es, MLflow und TensorBoard hinsichtlich ihrer Funktionen, ihrer Benutzeroberfläche und ihres praktischen Einsatzes zu untersuchen. Dabei wird ein Beispielszenario vorgestellt, in dem beide Tools zur Analyse und Verwaltung von Machine-Learning-Experimenten verwendet werden. Anschließend werden die Benutzeroberflächen und ihre wichtigsten Komponenten detailliert betrachtet.

Die Arbeit beginnt mit einer Einführung in MLflow und TensorBoard, in der deren Architektur und zentrale Komponenten erläutert werden. Anschließend wird ein gemeinsames Beispielprojekt vorgestellt, das als Grundlage für die Analyse der beiden Tools dient. In diesem Projekt werden MLflow und TensorBoard parallel eingesetzt, um deren Funktionen praxisnah zu testen.

Darauf aufbauend folgt eine detaillierte Untersuchung der Benutzeroberflächen beider Tools, wobei die zentralen Anwendungsbereiche und Möglichkeiten zur Visualisierung und Verwaltung von Machine-Learning-Experimenten dargestellt werden. Im weiteren Verlauf werden MLflow und TensorBoard direkt miteinander verglichen, um ihre jeweiligen Einsatzgebiete klar abzugrenzen. Abschließend fasst die Arbeit die wichtigsten Erkenntnisse zusammen und gibt einen Ausblick auf mögliche Weiterentwicklungen und zukünftige Trends im Bereich der ML-Experimentverwaltung und Modellanalyse.

2. Grundlagen und Architektur

Ein effektives Experiment-Tracking und die Visualisierung von Trainingsprozessen sind zentrale Bestandteile moderner Machine-Learning-Workflows. Während MLflow primär für das Tracking von Experimenten und das Modellmanagement entwickelt wurde, konzentriert sich TensorBoard auf die detaillierte Analyse und Visualisierung von Trainingsprozessen. Um diese Werkzeuge effizient nutzen zu können, ist ein grundlegendes Verständnis ihrer Architektur und Systemkomponenten erforderlich.

Dieses Kapitel liefert eine detaillierte Einführung in die Systemarchitekturen von MLflow und TensorBoard. Zunächst wird erläutert, welche Ziele beide Tools verfolgen und welche zentralen Komponenten sie enthalten. Anschließend werden die technischen Unterschiede zwischen den beiden Systemen herausgearbeitet, um ein tiefergehendes Verständnis für ihre jeweilige Funktionsweise zu ermöglichen.

2.1 Einführung in Mlflow

Mit zunehmender Komplexität von Machine-Learning-Projekten steigt die Notwendigkeit, Experimente systematisch zu verwalten, Modelle reproduzierbar zu speichern und den gesamten Machine-Learning-Lifecycle nachverfolgbar zu gestalten.

MLflow wurde entwickelt, um diese Herausforderungen zu bewältigen. Als plattformunabhängiges Open-Source-Tool ermöglicht es MLflow, Experimente zu tracken, Modelle zu speichern, reproduzierbare Pipelines zu erstellen und Modelle für das Deployment bereitzustellen. Dabei ist MLflow nicht auf eine spezifische Machine-Learning-Bibliothek beschränkt, sondern kann mit TensorFlow, PyTorch, Scikit-learn und vielen weiteren Frameworks verwendet werden.

In den folgenden Abschnitten wird zunächst die Definition und Zielsetzung von MLflow erläutert, bevor die Architektur und die zentralen Komponenten detailliert beschrieben werden.

2.1.1 Definition und Zielsetzung

MLflow ist eine flexible und erweiterbare Open-Source-Plattform, die entwickelt wurde, um Workflows und Artefakte über den gesamten Lebenszyklus des maschinellen Lernens hinweg zu verwalten. Sie bietet integrierte Schnittstellen zu vielen gängigen ML-Bibliotheken, kann jedoch mit jeder Bibliothek, jedem Algorithmus oder Bereitstellungstool verwendet werden. Das Design von MLflow ist auf Erweiterbarkeit ausgelegt, sodass Benutzer Plugins schreiben können, um neue Workflows, Bibliotheken und Tools zu unterstützen. (MLflow, 2025a)

MLflow ist eine plattformunabhängige Open-Source-Plattform, die den gesamten Lebenszyklus von Machine-Learning-Modellen verwaltet. Sie wurde entwickelt, um zentrale Herausforderungen im ML-Workflow zu lösen, insbesondere im Bereich der Reproduzierbarkeit, Nachverfolgbarkeit, Modellverwaltung und Bereitstellung.

Um diesen Herausforderungen zu begegnen, verfolgt MLflow eine Reihe übergeordneter Ziele, die eine strukturierte, effiziente und flexible Entwicklung sowie den Betrieb von ML-Modellen ermöglichen (MLflow, 2025b):

- Nachverfolgbarkeit und Transparenz:
  Einheitliches Tracking von Experimenten, Parametern und Ergebnissen, um Modellentwicklungen nachvollziehbar zu machen.

- Reproduzierbarkeit von ML-Projekten:
  Sicherstellung der Konsistenz von Modellen über verschiedene Runs und Umgebungen hinweg, um verlässliche und wiederholbare Ergebnisse zu gewährleisten.

- Standardisierte Bereitstellung und Verwaltung von Modellen:
  Vereinfachung des Übergangs von Entwicklung zu Produktion durch ein strukturiertes Modellmanagement mit Versionierung und Deployment-Funktionen.

- Flexibilität und Kompatibilität:
  Unterstützung verschiedener Machine-Learning-Frameworks und Integration in bestehende ML-Workflows, um eine breite Anwendbarkeit sicherzustellen.

MLflow bietet somit eine ganzheitliche Lösung für die Verwaltung von Machine-Learning-Prozessen, die sowohl für Forschung als auch für den produktiven Einsatz geeignet ist.

2.1.2 Architektur und Komponenten

MLflow ist eine modulare Open-Source-Plattform, die Architektur von MLflow besteht aus fünf Hauptkomponenten (MLflow, 2025a), die nahtlos zusammenarbeiten, aber auch unabhängig voneinander eingesetzt werden können:

- MLflow Tracking (MLflow, 2025c):
  - Diese Komponente bietet eine API und eine Benutzeroberfläche zum Protokollieren von Parametern, Codeversionen, Metriken und Artefakten, die während des Machine-Learning-Prozesses entstehen.
  - Sie ermöglicht es, Experimente systematisch zu verfolgen und Ergebnisse zu vergleichen, unabhängig davon, ob sie in Standalone-Skripten, Notebooks oder anderen Umgebungen ausgeführt werden.
  - Die Ergebnisse können lokal oder auf einem zentralen Tracking-Server gespeichert werden, was die Zusammenarbeit zwischen verschiedenen Teams erleichtert.

- MLflow Projects (MLflow, 2025d):
  - MLflow Projects standardisiert die Paketierung von ML-Code, Workflows und Artefakten in einem reproduzierbaren Format.
  - Jedes Projekt kann als Verzeichnis mit Code oder als Git-Repository vorliegen und verwendet eine Projektbeschreibung, um Abhängigkeiten und Ausführungsanweisungen zu definieren.
  - Dies erleichtert die Wiederverwendbarkeit und gemeinsame Nutzung von ML-Projekten, insbesondere in kollaborativen Teams.

- MLflow Models (Mlflow, 2025e):
  - MLflow definiert ein standardisiertes Format zur Speicherung von Modellen, das die Nutzung in verschiedenen nachgelagerten Tools ermöglicht.

- Dies umfasst Echtzeit-Serving über eine REST-API oder Batch-Inferenz auf Apache Spark.
- Das Format unterstützt verschiedene "Flavors", die von unterschiedlichen ML-Frameworks interpretiert werden können, was die Flexibilität bei der Bereitstellung von Modellen erhöht.

- MLflow Model Registry (MLflow, 2025f):
  - Diese Komponente bietet ein zentrales Repository zur Verwaltung des gesamten Modelllebenszyklus.
  - Sie unterstützt Funktionen wie Modellversionierung, Statusübergänge (z. B. von "Staging" zu "Production") und Anmerkungen.
  -Dies erleichtert großen Organisationen die kooperative Verwaltung von Modellen und die Nachverfolgung ihrer Entwicklung.

- MLflow Recipes (MLflow, 2025g):
  - MLflow Recipes ist ein Framework zur Standardisierung von Machine-Learning-Workflows, das vordefinierte Templates für gängige ML-Aufgaben wie Regression und Klassifikation bereitstellt.
  - Es reduziert den Aufwand für Boilerplate-Code und bietet eine intelligente Ausführungs-Engine, die Ergebnisse zwischenspeichert und nur geänderte Schritte erneut ausführt.
  - Die strukturierte, git-integrierte Architektur erleichtert den Übergang von Entwicklung zur Produktion, indem sie sicherstellt, dass alle Modellcodes, Daten und Konfigurationen überprüfbar und bereitstellbar sind.

Durch die Integration dieser Komponenten bietet MLflow eine umfassende Plattform, die Effizienz, Konsistenz und Nachvollziehbarkeit im gesamten Machine-Learning-Lebenszyklus sicherstellt.

2.2 Einführung in TensorBoard

Mit der steigenden Komplexität von Machine-Learning-Modellen wächst auch die Anforderung an eine detaillierte Analyse und Visualisierung von Trainingsprozessen. Die Optimierung neuronaler Netzwerke erfordert eine tiefgehende Untersuchung von Metriken, Modellarchitekturen und Gewichtsverteilungen, um Überanpassung zu vermeiden und die Leistung systematisch zu verbessern.

TensorBoard wurde entwickelt, um diese Herausforderungen zu bewältigen. Als leistungsstarkes Open-Source-Tool für die visuelle Analyse von Machine-Learning-Experimenten ermöglicht es TensorBoard, Trainingsmetriken zu verfolgen, Modellarchitekturen zu visualisieren, Embeddings darzustellen und Performancedaten auszuwerten. Dabei ist TensorBoard besonders eng mit TensorFlow integriert, kann aber auch mit anderen Frameworks wie PyTorch oder Keras verwendet werden.

In den folgenden Abschnitten wird zunächst die Definition und Zielsetzung von TensorBoard erläutert, bevor die Architektur und die zentralen Komponenten detailliert beschrieben werden.

2.2.1 Definition und Zielsetzung

TensorBoard ist ein Open-Source-Visualisierungstool, das von den Entwicklern von TensorFlow bereitgestellt wird. Es dient der Überwachung und Analyse von Machine-Learning-Modellen während des Trainings und der Evaluierung. Ursprünglich für die Arbeit mit TensorFlow entwickelt, unterstützt TensorBoard mittlerweile auch andere Frameworks wie PyTorch. TensorBoard wurde entwickelt, um

ML-Experimente verständlicher und nachvollziehbarer zu machen, indem es eine zentrale Plattform für die Analyse von Modellen und deren Trainingsprozesse bietet. Durch die grafische Darstellung von Metriken, Modellstrukturen und Parameterentwicklungen hilft TensorBoard, Trainingsfortschritte zu überwachen, Probleme frühzeitig zu erkennen und Modelle gezielt zu optimieren. (Tensorflow, 2025a)

Um diese Anforderungen zu erfüllen, verfolgt TensorBoard die folgenden übergeordneten Ziele:

- Visualisierung von Trainingsmetriken (TensorFlow, 2025b):
  Ermöglicht das Verfolgen und Darstellen von Metriken wie Verlust (Loss) und Genauigkeit (Accuracy) über die Trainingszeit hinweg.

- Darstellung der Modellarchitektur (TensorFlow, 2025c):
  Bietet eine grafische Darstellung des Modells, einschließlich der einzelnen Schichten und Verbindungen, um die Struktur und den Datenfluss zu veranschaulichen.

- Analyse von Parameterverteilungen (TensorFlow, 2025e):
  Stellt Histogramme und Verteilungen von Gewichten, Biases und anderen Tensoren bereit, um deren Entwicklung während des Trainings zu beobachten.

- Projektion von Embeddings (TensorFlow, 2025e):
  Ermöglicht die Visualisierung hochdimensionaler Daten durch Projektion in niedrigdimensionale Räume, was insbesondere für die Analyse von Wort- oder Feature-Einbettungen nützlich ist.

- Integration von Medien (TensorFlow, 2025f):
  Unterstützt die Darstellung von Medien, die während des Trainingsprozesses verarbeitet oder generiert werden, um eine umfassende Analyse zu ermöglichen.

Durch diese Funktionen zielt TensorBoard darauf ab, den Trainingsprozess transparenter und verständlicher zu gestalten, sodass Entwickler und Forscher fundierte Entscheidungen zur Optimierung ihrer Modelle treffen können.

2.2.2 Architektur und Komponenten

TensorBoard ist ein modulares Open-Source-Visualisierungstool, das eine Vielzahl von Funktionen zur Analyse und Überwachung von Machine-Learning-Modellen bereitstellt. Die Architektur von TensorBoard besteht aus mehreren spezialisierten Dashboards, die jeweils bestimmte Aspekte des ML-Trainingsprozesses visualisieren. Diese Komponenten arbeiten nahtlos zusammen, können jedoch auch unabhängig voneinander genutzt werden:

- TensorBoard Scalars (TensorFlow, 2025b):
  - Diese Komponente ermöglicht das Tracking und die Visualisierung von Trainingsmetriken, darunter Loss, Accuracy und andere relevante Modellmetriken.
  - Durch die grafische Darstellung von Zeitreihen kann überprüft werden, ob ein Modell zu lange trainiert, Overfitting zeigt oder noch Potenzial zur Verbesserung hat.
  - Das Time Series Dashboard von TensorBoard stellt diese Metriken auf eine leicht verständliche

Weise dar und erlaubt den Vergleich verschiedener Runs.

- TensorBoard Graphs (TensorFlow, 2025c):
  - Dieses Dashboard bietet eine grafische Darstellung der Modellarchitektur, um den Aufbau des neuronalen Netzes zu analysieren.
  - Es zeigt sowohl konzeptionelle Graphen als auch op-level Graphen, wodurch Entwickler Einblick in die Struktur und Berechnungen des Modells erhalten.
  - Diese Ansicht kann helfen, Bottlenecks zu identifizieren oder Optimierungsmöglichkeiten zu erkennen, falls das Training langsamer als erwartet verläuft.

- TensorBoard HParams – Hyperparameter-Tuning (TensorFlow, 2025d):
  - ML-Modelle erfordern die Optimierung von Hyperparametern, z. B. Lernrate, Dropout-Wert oder Anzahl der Neuronen pro Schicht.
  - Das HParams-Dashboard in TensorBoard ermöglicht eine systematische Erfassung, Vergleich und Analyse dieser Hyperparameter und deren Einfluss auf die Modellperformance.
  - Dadurch wird der Experimentierprozess optimiert und die effizientesten Modellkonfigurationen schneller gefunden.

- TensorBoard Embedding Projector (TensorFlow, 2025e):
  - Hochdimensionale Daten, insbesondere Embeddings von Wörtern oder Feature-Daten, lassen sich mithilfe des Embedding Projectors visuell analysieren.
  - TensorBoard ermöglicht die Reduktion der Dimensionalität durch Algorithmen wie PCA oder t-SNE, um verborgene Strukturen in den Daten zu erkennen.
  -Dies ist besonders nützlich für die Untersuchung von Wort-Embeddings in NLP-Modellen oder Feature-Engineering in Tabulardaten.

- TensorBoard Image Summaries (TensorFlow, 2025f):
  - Modelle, die mit Bilddaten arbeiten, profitieren von der Möglichkeit, Bilder direkt in TensorBoard zu visualisieren.
  - Diese Komponente ermöglicht die Darstellung von Eingabedaten, generierten Bildern oder gewichteten Filterdarstellungen.
  - Dadurch lassen sich mögliche Fehlerquellen in den Daten oder Verzerrungen in neuronalen Netzwerken schneller identifizieren.

- TensorBoard Text Summaries (TensorFlow, 2025g)
  - Speziell für NLP-Modelle bietet TensorBoard die Möglichkeit, Textdaten während des Trainings zu analysieren.
  - Entwickler können damit generierten Text, verarbeitete Token oder diagnostische Logs innerhalb der Trainingspipeline darstellen und nachvollziehen.

- TensorFlow Profiler (TensorFlow, 2025h):
  - Da Machine-Learning-Modelle hohe Rechenkapazitäten benötigen, ist eine Performance-Analyse des Codes entscheidend.
  - Der TensorFlow Profiler hilft dabei, Bottlenecks im Rechenprozess zu identifizieren, die Speichernutzung zu optimieren und die Effizienz des Modells zu verbessern.

Durch die Kombination dieser spezialisierten Dashboards bietet TensorBoard eine umfassende Lösung zur Analyse und Optimierung von Machine-Learning-Modellen. Es ermöglicht nicht nur die Überwachung von Metriken, sondern auch die visuelle Inspektion von Architektur, Embeddings, Bildern, Texten und Performance-Daten, was die Modellentwicklung und Fehlersuche erheblich erleichtert.

3. Praxis: Implementierung eines Beispiel-Workflows für MLflow und TensorBoard

In den vorangegangenen Kapiteln wurden die Architektur und Kernkomponenten von MLflow und TensorBoard detailliert beschrieben. Während MLflow primär auf das Tracking von Experimenten und die Verwaltung von Modellen ausgerichtet ist, bietet TensorBoard eine umfassende Visualisierung von Trainingsprozessen und Modellarchitekturen.

Dieses Kapitel konzentriert sich auf die praktische Anwendung beider Tools. Ziel ist es, ein einheitliches Machine-Learning-Szenario zu definieren, das sowohl mit MLflow als auch mit TensorBoard umgesetzt wird. Dabei wird schrittweise gezeigt, wie Experimente protokolliert, Modelle gespeichert und Metriken visualisiert werden.
Zunächst wird in Abschnitt 3.1 das gemeinsame Beispielszenario definiert, das als Grundlage für die Implementierung dient. Anschließend erfolgt die praktische Umsetzung mit MLflow (Abschnitt 3.2) und die Integration von TensorBoard (Abschnitt 3.3). Die jeweiligen Abschnitte beinhalten Python-Code, Konfigurationsschritte und eine direkte Analyse der UI-Ergebnisse, um einen umfassenden Einblick in die Nutzung beider Tools zu ermöglichen.

3.1 Definition des gemeinsamen Beispielszenarios

Um MLflow und TensorBoard in einem praxisnahen Machine-Learning-Workflow zu demonstrieren, wird ein einheitliches Beispielszenario gewählt, das sich für beide Tools gleichermaßen eignet. Ziel ist es, ein Machine-Learning-Modell zu trainieren, das während des Trainings relevante Metriken und Modellinformationen protokolliert und diese sowohl in MLflow als auch in TensorBoard visualisiert.

Für eine optimale Vergleichbarkeit wird ein klassisches Deep-Learning-Modell auf Basis eines neuronalen Netzes trainiert. Dabei werden sowohl die Trainingsmetriken als auch die Modellarchitektur und Hyperparameter systematisch erfasst. Dieses Setup ermöglicht eine direkte Analyse und Gegenüberstellung beider Tools, da MLflow für das Experiment-Tracking und Modellmanagement genutzt wird, während TensorBoard die detaillierte Visualisierung der Trainingsdynamik und Modellstrukturen übernimmt.

3.1.1 Wahl eines geeigneten Machine-Learning-Projekts für beide Tools

Um die Funktionen von MLflow und TensorBoard praxisnah zu demonstrieren, wird ein Deep-Learning-Ansatz mit Transfer Learning gewählt. Das Ziel des Projekts ist die Klassifikation von Bildern zweier Kategorien – Apu Nahasapeemapetilon und Abraham Simpson aus The Simpsons. Dabei wird das vortrainierte Modell VGG16 als Feature-Extractor genutzt, während die oberen Schichten durch eigene trainierbare Layer ersetzt werden. Dieses Verfahren ermöglicht eine effiziente Anpassung des Modells an die neue Bildklassifikationsaufgabe.

Der gewählte Ansatz eignet sich ideal für die Evaluierung von MLflow und TensorBoard, da beide Tools ihre jeweiligen Stärken in unterschiedlichen Aspekten des Machine-Learning-Workflows ausspielen

können. MLflow wird genutzt, um Experimente zu tracken, Hyperparameter zu speichern und Modelle zu versionieren. TensorBoard dient zur detaillierten Analyse des Trainingsprozesses, einschließlich der Visualisierung von Metriken, Modellarchitektur und Optimierungsverläufen.

Während des Trainings werden die folgenden Schlüsselelemente erfasst und analysiert:

- Mit MLflow**:** Logging von Experimenten, Hyperparametern und Metriken, Speichern des trainierten Modells, Versionierung und Deployment. da der Fokus auf der Demonstration der Kernfunktionen von MLflow (Tracking, Models, Model Registry) liegt.
    - MLflow Projects wird primär genutzt, um Experimente reproduzierbar und portabel zu gestalten, insbesondere durch die Definition von Umgebungen mit Conda oder Docker. Da dieses Beispiel als Proof of Concept (PoC) dient und innerhalb einer kontrollierten Umgebung ausgeführt wird, ist die zusätzliche Komplexität von MLflow Projects nicht erforderlich.
    - MLflow Recipes hingegen ist für die Standardisierung und Automatisierung des Machine-Learning-Workflows gedacht, insbesondere für die strukturierte Verarbeitung von Daten, Feature Engineering, Training und Evaluation. Da dieses Beispiel darauf abzielt, die grundlegenden Konzepte von MLflow zu erklären und nicht eine vollständige Machine-Learning-Pipeline zu implementieren, wird auf MLflow Recipes verzichtet.

- Mit TensorBoard**:** Visualisierung von Loss & Accuracy, Modellarchitektur, Hyperparameter-Analyse, Performance-Optimierung und Bilddatendarstellung. Die **T**ensorBoard-Module Embedding Projector und Text Summaries werden in diesem Beispiel nicht verwendet, da sie primär für die Analyse von Wort-Embeddings und NLP-Modellen gedacht sind. Außerdem werden die Modulle Projects und Receipes nicht

3.2 Umsetzung mit MLflow: Logging und Visualisierung

Zur systematischen Erfassung und Verwaltung der Experimente wird zunächst ein MLflow-Experiment definiert und gestartet. Dies erfolgt durch mlflow.set_experiment("Simpsons_Classification"), wodurch sichergestellt wird, dass alle Trainingsläufe diesem Experiment zugeordnet werden. Innerhalb eines MLflow-Runs, der mit *mlflow.start_run()* initialisiert wird, werden alle relevanten Informationen zum jeweiligen Experiment protokolliert.

```
mlflow.set_experiment("Simpsons_Classification")
with mlflow.start_run() as run:
```

Während des Trainingsprozesses werden verschiedene Parameter geloggt, um spätere Vergleiche zwischen verschiedenen Runs zu ermöglichen. Dazu gehören die Anzahl der Convolutional-Filter in der zusätzlichen Schicht (filters), die Anzahl der Neuronen in der dichten Schicht (Dense_neurons), die Dropout-Rate (Dropout_rate), die Anzahl der Trainings-Epochen (epochs), die Batch-Größe (batch_size) sowie der verwendete Optimizer (optimizer). Diese Parameter bestimmen die Architektur des Modells und beeinflussen die Trainingsdynamik.

```
mlflow.log_param("batch_size", batch_size)
```

```
mlflow.log_param("filters", filters)
mlflow.log_param("Dense_neurons", Dense_neurons)
mlflow.log_param("Dropout_rate", Dropout_rate)
mlflow.log_param("epochs", epochs)
mlflow.log_param("optimizer", optimizer)
```

Zusätzlich zu den Parametern werden auch Metriken gespeichert, die eine quantitative Bewertung der Modellleistung ermöglichen. Während des Trainings werden pro Epoche sowohl der Trainings-Loss als auch die Trainings-Accuracy protokolliert. Parallel dazu werden die Validierungsmetriken (val_loss und val_accuracy) gespeichert, um die Generalisierungsfähigkeit des Modells auf unbekannte Daten zu bewerten. Die Metriken werden mit *mlflow.log_metric()* in MLflow gespeichert, wobei für jede Epoche die Werte mit dem entsprechenden Schritt (step=epoch) versehen werden, um eine zeitliche Analyse zu ermöglichen.

```
for epoch in range(epochs):
    mlflow.log_metric("train_loss", history.history["loss"][epoch], step=epoch)
    mlflow.log_metric("train_accuracy", history.history["accuracy"][epoch], step=epoch)
    mlflow.log_metric("val_loss", history.history["val_loss"][epoch], step=epoch)
    mlflow.log_metric("val_accuracy", history.history["val_accuracy"][epoch], step=epoch)
```

Nach Abschluss des Trainings wird das Modell gespeichert und in MLflow registriert. Dazu wird das Modell zunächst als TensorFlow-Modell mittels *custom_model.save()* lokal gesichert und anschließend mit *mlflow.tensorflow.log_model()* in MLflow gespeichert. Dadurch kann das Modell später einfach erneut geladen und weiterverwendet werden. Zusätzlich erfolgt eine Registrierung in der MLflow Model Registry, sodass verschiedene Modellversionen verwaltet und für das Deployment bereitgestellt werden können.

```
custom_model.save(model_path)

mlflow.tensorflow.log_model(custom_model, "model")

mlflow.register_model(model_uri, "Simpsons_Classifier")
```

Die Ergebnisse der Trainingsläufe können in der MLflow-Benutzeroberfläche analysiert werden. Die MLflow UI wird mit dem Befehl mlflow ui gestartet, wodurch eine webbasierte Oberfläche bereitgestellt wird. In dieser Oberfläche können verschiedene Runs verglichen werden, um den Einfluss unterschiedlicher Parameterkonfigurationen zu bewerten. Die gespeicherten Metriken werden dabei in Form von Diagrammen dargestellt, sodass die Entwicklung von Loss und Accuracy über die Epochen hinweg nachvollziehbar ist. Zudem ermöglicht die MLflow UI den direkten Zugriff auf die gespeicherten Modellartefakte, wodurch die verschiedenen Versionen des Modells verwaltet und gegebenenfalls für die Produktivsetzung vorbereitet werden können.

Durch die Integration von MLflow in den Trainingsprozess wird eine strukturierte und nachvollziehbare Verwaltung von Machine-Learning-Experimenten erreicht. Die Logging- und Versionierungsfunktionen erleichtern die Vergleichbarkeit verschiedener Trainingsläufe, wodurch eine effizientere Modelloptimierung möglich wird. Darüber hinaus erlaubt die Nutzung der MLflow UI eine detaillierte Analyse der gespeicherten Metriken und Modelle, wodurch die Entscheidungsfindung in Bezug auf Modellselektion und Deployment optimiert wird.

3.3 Umsetzung mit TensorBoard: Logging und Visualisierung

Während des Trainings werden relevante Daten über Trainingsfortschritt, Modellstruktur und Systemauslastung in TensorBoard gespeichert. Dies geschieht mithilfe von Callbacks, die automatisch Metriken und Modellinformationen während des Trainings loggen. Die Visualisierung dieser Daten ermöglicht eine bessere Analyse des Modellverhaltens und eine datenbasierte Optimierung der Hyperparameter.

Zunächst wird für das HParams-Dashboard von TensorBoard wird der verwendete Optimizer als Hyperparameter gespeichert, um verschiedene Runs vergleichen zu können. Danach werden die verschiedenen Runs definiert und mit einer Schleife ausgeführt.

```
P_OPTIMIZER = hp.HParam("optimizer", hp.Discrete(["adam", "rmsprop"]))

for i, optimizer_name in enumerate(HPARAMS["optimizer"].domain.values, start=1):
    run_name = f"run_{i}_{optimizer_name}"
```

Danach das Verzeichnis für das TensorBoard-Logging definiert. Dies geschieht durch die Angabe eines Speicherpfads, in dem die Trainingsdaten gespeichert werden. Der Pfad wird dynamisch erstellt, sodass verschiedene Trainingsläufe separat analysiert werden können.

```
log_dir = os.path.join("tensorboard_logs", f"{run_name}_{datetime.now().strftime('%Y%m%d-%H%M%S')}")
```

Während des Trainings werden die Loss- und Accuracy-Werte für das Training und die Validierung nach jeder Epoche gespeichert. Diese Metriken werden im Scalars-Dashboard von TensorBoard angezeigt.

```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
```

TensorBoard ermöglicht die Visualisierung der Modellarchitektur im Graphs-Dashboard. Dafür wird das Modell explizit in TensorBoard gespeichert.

```
tf.summary.create_file_writer(log_dir).set_as_default() tf.keras.utils.plot_model(custom_model, to_file=os.path.join(log_dir, "model.png"), show_shapes=True)
```

Zur Analyse der Rechenressourcen während des Trainings wird der TensorFlow Profiler verwendet. Dieser ermöglicht eine detaillierte Untersuchung der CPU- und GPU-Auslastung.

```
profiler_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, profile_batch=2)
```

Um zu überprüfen, ob die Eingabedaten korrekt geladen werden, können Beispielbilder in TensorBoard gespeichert werden. Diese werden im Images-Dashboard angezeigt.

```
    def log_images(dataset, writer, step):
        for images, labels in dataset.take(1):
            images = images / 255.0

            with writer.as_default():
```

```
        tf.summary.image("Training Images", images, step=step, max_outputs=5)

    image_writer = tf.summary.create_file_writer(log_dir)
```

Während des Modelltrainings werden die Callbacks für TensorBoard und den Profiler verwendet, um alle relevanten Daten automatisch zu loggen.

```
history = custom_model.fit(train_dataset, epochs=3,
                           validation_data=validation_dataset,
                           callbacks=[tensorboard_callback, profiler_callback])

log_images(train_dataset, image_writer, step=3)
```

Nach Abschluss des Trainings können die Ergebnisse in TensorBoard analysiert werden. Die Benutzeroberfläche wird mit folgendem Befehl gestartet:

```
tensorboard --logdir=tensorbord_logs
```

4. Analyse der UI-Funktionen anhand des Beispiels

In Kapitel 3 wurde die praktische Implementierung von MLflow und TensorBoard anhand eines gemeinsamen Machine-Learning-Workflows vorgestellt. Dabei wurden die zentralen Funktionen beider Tools zur Protokollierung von Parametern, Speicherung von Metriken und Verwaltung von Modellen genutzt.
Nachdem die Daten erfolgreich geloggt wurden, folgt nun in Kapitel 4 die detaillierte Analyse der Benutzeroberflächen (UIs) von MLflow und TensorBoard. Ziel ist es, anhand des implementierten Beispiels zu evaluieren, wie die protokollierten Informationen visualisiert und interpretiert werden können.
Dieses Kapitel untersucht die Struktur, Navigation und wichtigsten Funktionen beider UIs im Detail. Dazu werden zunächst die Aufbau und Navigation der Benutzeroberflächen beschrieben, gefolgt von einer genauen Analyse der verfügbaren Funktionen. Besonderer Fokus liegt dabei auf den Visualisierungsoptionen für Metriken, Modellregistrierung und Performance-Analysen, die für eine effektive Modelloptimierung essenziell sind.

4.1 MLflow UI: Aufbau und Nutzung

Die MLflow UI ist eine intuitive Möglichkeit, Metriken, Parametern und Modellen zu durchsuchen, zu vergleichen und zu verwalten. Die Oberfläche dient somit als zentrale Kontrollinstanz, um Modelle effizient zu analysieren, zu versionieren und auf deren Basis fundierte Entscheidungen zu treffen.

4.1.1 Überblick über die Benutzeroberfläche und Navigation

Die folgende Abbildung zeigt die Startansicht der MLflow UI im Experiments Tab, nachdem ein Experiment erfolgreich erstellt und ein Trainingslauf durchgeführt wurde.

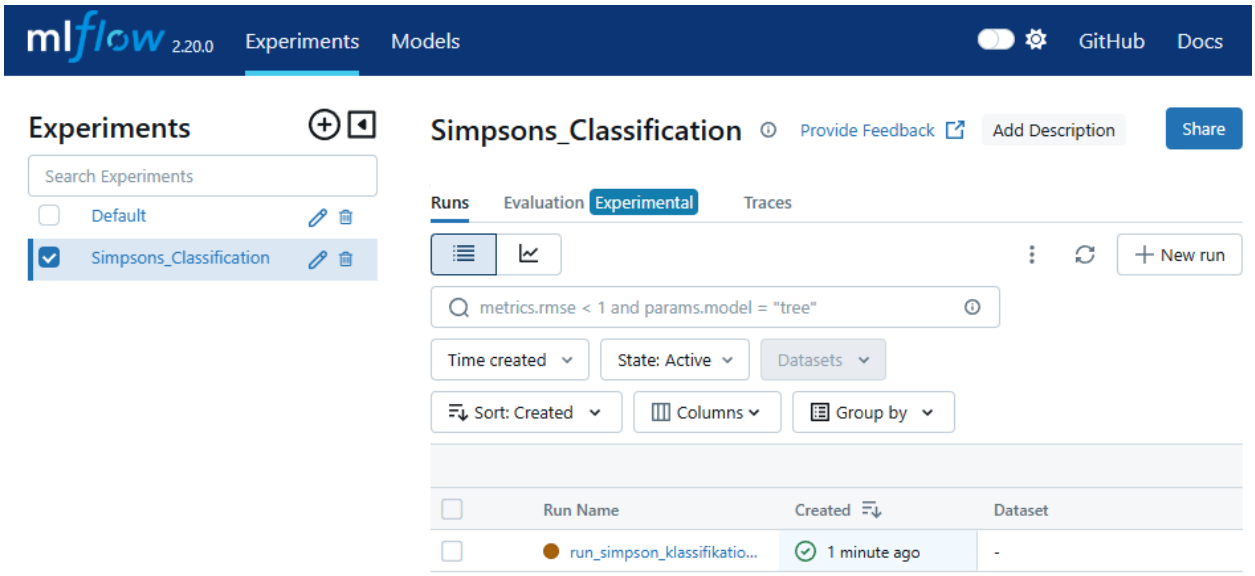


Abbildung 1: MLflow Experiments

Die MLflow UI ist in mehrere Hauptbereiche unterteilt:

- Navigation & Experimente-Verwaltung (linke Seite)
  - Zeigt eine Liste aller vorhandenen Experimente.
  - Hier können neue Experimente erstellt oder zwischen bestehenden gewechselt werden.
  - In diesem Beispiel ist das Experiment "Simpsons_Classification" aktiv.

- Experimentübersicht & Runs-Tabelle (zentraler Bereich)
  - Zeigt eine Tabelle aller Runs innerhalb des gewählten Experiments.
  - Jeder Run repräsentiert ein durchgeführtes Training mit bestimmten Parametern und Metriken.

- Filter- und Suchfunktionen (oberhalb der Runs-Tabelle)
  - Runs können gefiltert, sortiert und gruppiert werden.
  - Es können Bedingungen definiert werden, um nur bestimmte Runs anzuzeigen.
  - Der "New Run"-Button ermöglicht das manuelle Starten eines neuen Runs direkt aus der UI.

- Haupt-Tabs der MLflow UI Experiments
  - Oberhalb der Runs-Tabelle befinden sich drei Registerkarten, die eine gezielte Analyse der Runs ermöglichen

  - Runs-Tab:
    - Standardansicht, zeigt eine Tabelle mit allen ausgeführten Runs.
    - Möglichkeit zur Sortierung, Filterung und zum Vergleich von Runs.

  - Evaluation-Tab (experimentell):

- Falls aktiviert, bietet dieser Tab detaillierte Analysen, z. B. Metrik-Visualisierungen oder Modellvergleiche.
- In einigen MLflow-Versionen ist diese Funktion noch nicht vollständig implementiert.

- Traces-Tab:
  - Enthält detaillierte Logs für jeden Run.
  - Hilft bei der Fehlersuche und der Nachvollziehbarkeit von Trainingsabläufen.

4.1.2 Überblick der Runs

Klickt man auf einen Run in der Runs-Tabelle werden alle wichtige Information über den Run in der Übersicht aufgezeigt. Es werden die wichtigsten Details wie Ids, Status und Dauer des Runs angezeigt. Außerdem werden die geloggten Parameter angezeigt sowie die Metriken.
In den Tab *Model metrics* werden die Metriken über die Metriken über die jeweiligen Epochen geplottet wie in der folgenden Abbildung zu sehen.

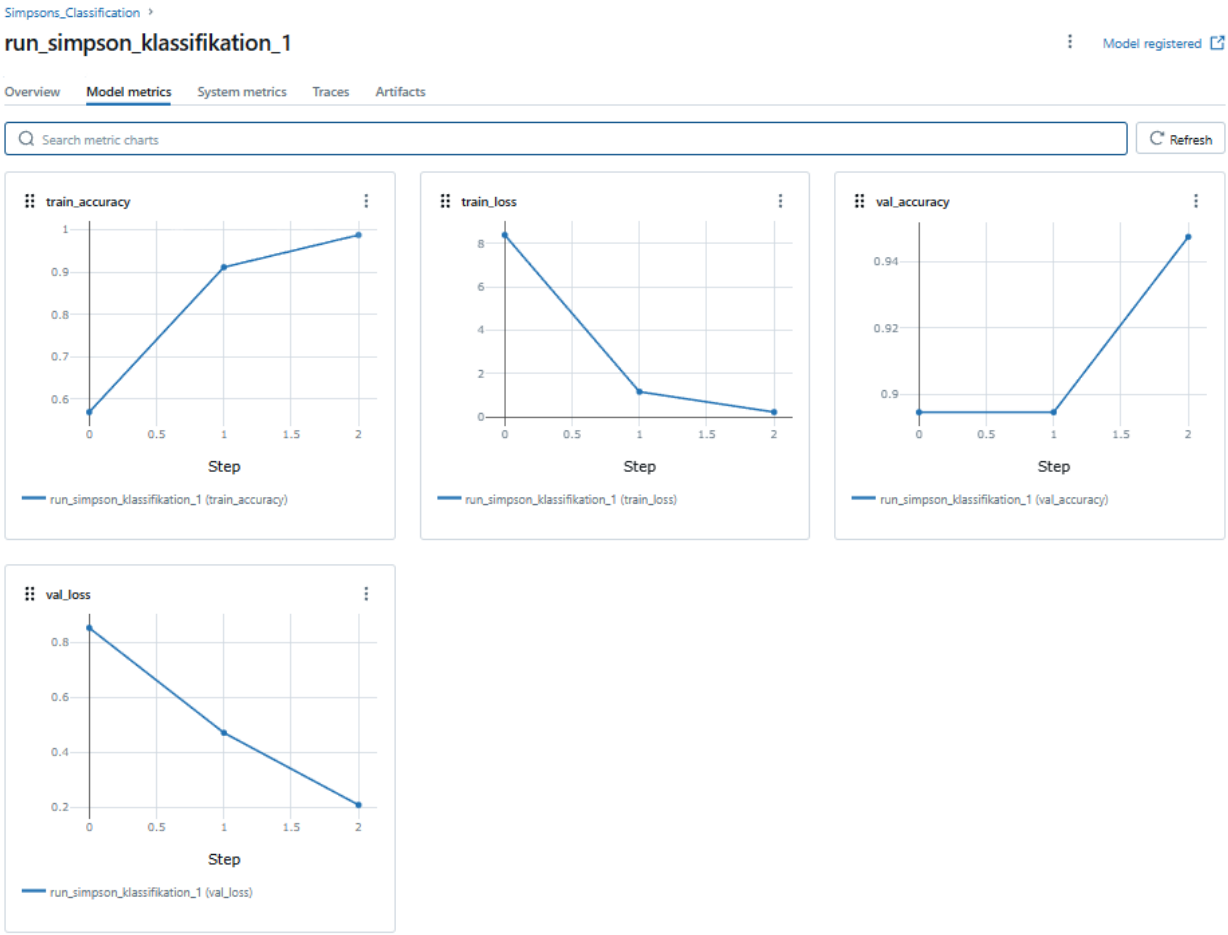


Abbildung 2: Mlflow Model metrics eines Runs

Zudem gibt es die Möglichkeitin MLflow System Metriken in den Tab *System metrics* zu speichern und zu analysieren. Dieser Tab ist für die Überwachung von Hardware-Ressourcen gedacht, die während des Experiments genutzt wurden.

In den Tab Artifacts ist für diesen Run, dass Model registiert worden. Dort kann die Modell- und Umgebungsdetails eingesehen werden.

Um Runs zu vergleichen wird ein weiterer Run gestartet mit zwei anderen Parametern: *Dense_neurons: 128* statt *256* und *Dropout_rate: 0.4* statt *0.5*.

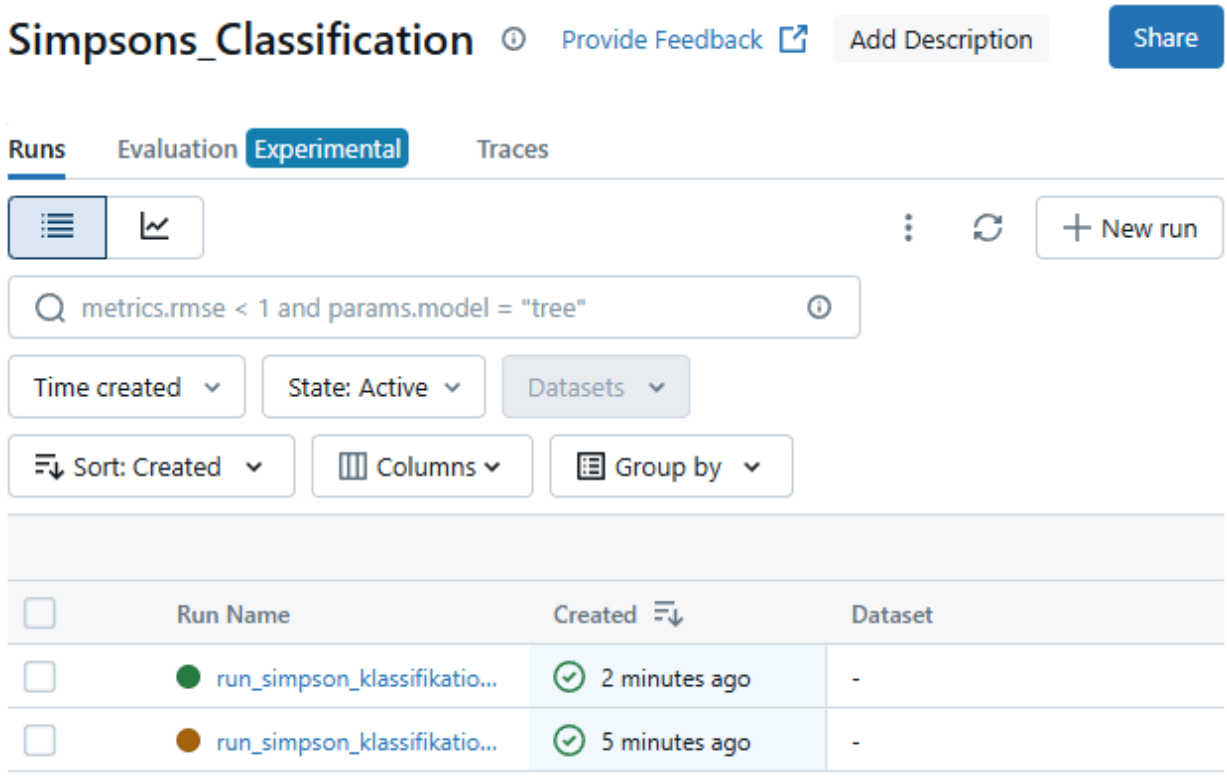


Abbildung 3: Mlflow Mehrere Runs verfügbar

In Abbildung sind die beiden Runs aufgeführt.

Um sie mit Mlflow UI zu vergleichen muss man in der Runs-Tabelle beide Runs auswählen und dann auf *Compare* drücken.
Dort gibt es nun vielzählige Möglichkeiten. Es können mehrere Plots mit verschiedenen Featuares und Metriken geplotet werden für den Vergleich der Runs. Es kann die Dauer des Trainigs, die unterschiedlichen Paramater, die jeweiligen Metriken und Artifakte betrachtet werden.

4.1.3 Modellregistrierung & Verwaltung von Modellen

Nachdem die wichtigsten Komponenten in den Tab *Experiments* aufgezeigt und erklärt wurden kommt nun der Tab *Models.*

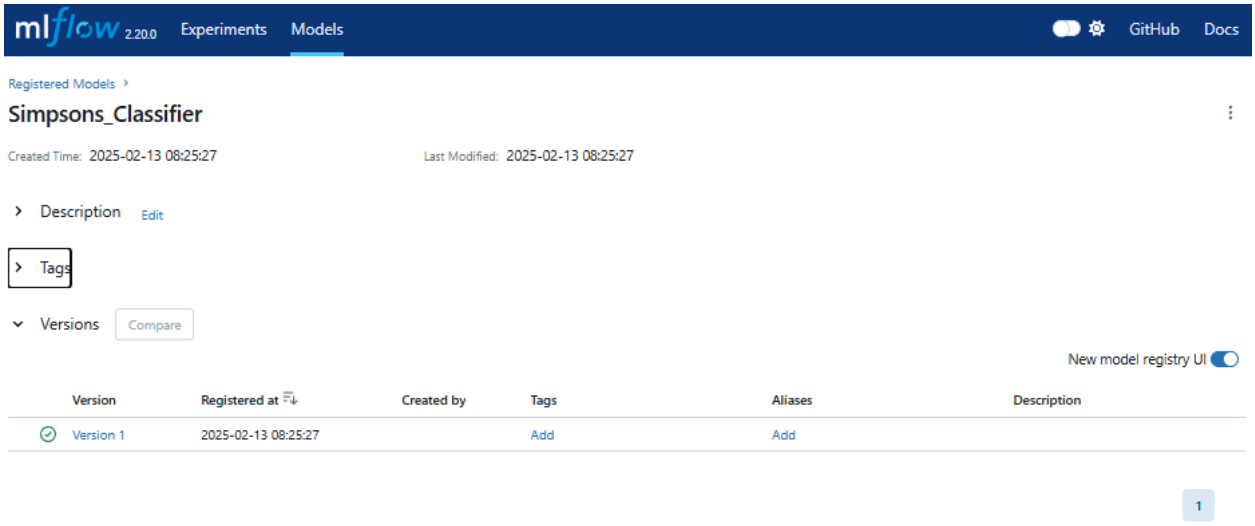


Abbildung 4: MLflow Models

Die Abbildung zeigt die MLflow Model Registry, in der das Modell "Simpsons_Classifier" erfolgreich registriert wurde. Die Model Registry dient zur Verwaltung, Versionierung und Bereitstellung von Machine-Learning-Modellen. In der oberen Leiste sind die Metadaten des Modells ersichtlich, darunter das Erstellungsdatum sowie die letzte Modifikation, die hier identisch sind, da bislang nur eine Version existiert.

Im Hauptbereich der Ansicht wird die erste Version des Modells aufgelistet. Jede Modellversion enthält Informationen wie das Registrierungsdatum, die Möglichkeit, Tags hinzuzufügen, sowie die Spalte *Aliases*, in der ein Modellstatus wie *Production* oder *Staging* festgelegt werden kann. Zudem bietet MLflow die Möglichkeit, eine Beschreibung zum Modell zu hinterlegen, um dessen Zweck oder spezielle Konfigurationen zu dokumentieren.

Ein zentrales Element der MLflow Model Registry ist die Versionierung. Sobald ein Modell mit identischem Namen erneut registriert wird, erstellt MLflow automatisch eine neue Version, wodurch sich Änderungen und Optimierungen im Trainingsprozess nachvollziehen lassen. Die Benutzeroberfläche erlaubt zudem den Vergleich zwischen Modellversionen, um Leistungsunterschiede zu analysieren.

4.2 TensorBoard UI: Aufbau und Nutzung

Die TensorBoard UI ist eine grafische Benutzeroberfläche, die speziell für die Analyse und Visualisierung von Machine-Learning-Trainingsprozessen entwickelt wurde. Sie ermöglicht es, Skalare und Metriken in Echtzeit zu verfolgen, Modellarchitekturen als Graphen zu visualisieren, Parameterverteilungen zu analysieren und Trainingsdaten in Form von Bildern darzustellen.

4.2.1 Überblick über die Benutzeroberfläche und Navigation

Zunächst wird die Startseite von Tensorboard nach dem erfolgreichen Logging der gewünschten Daten analysiert.

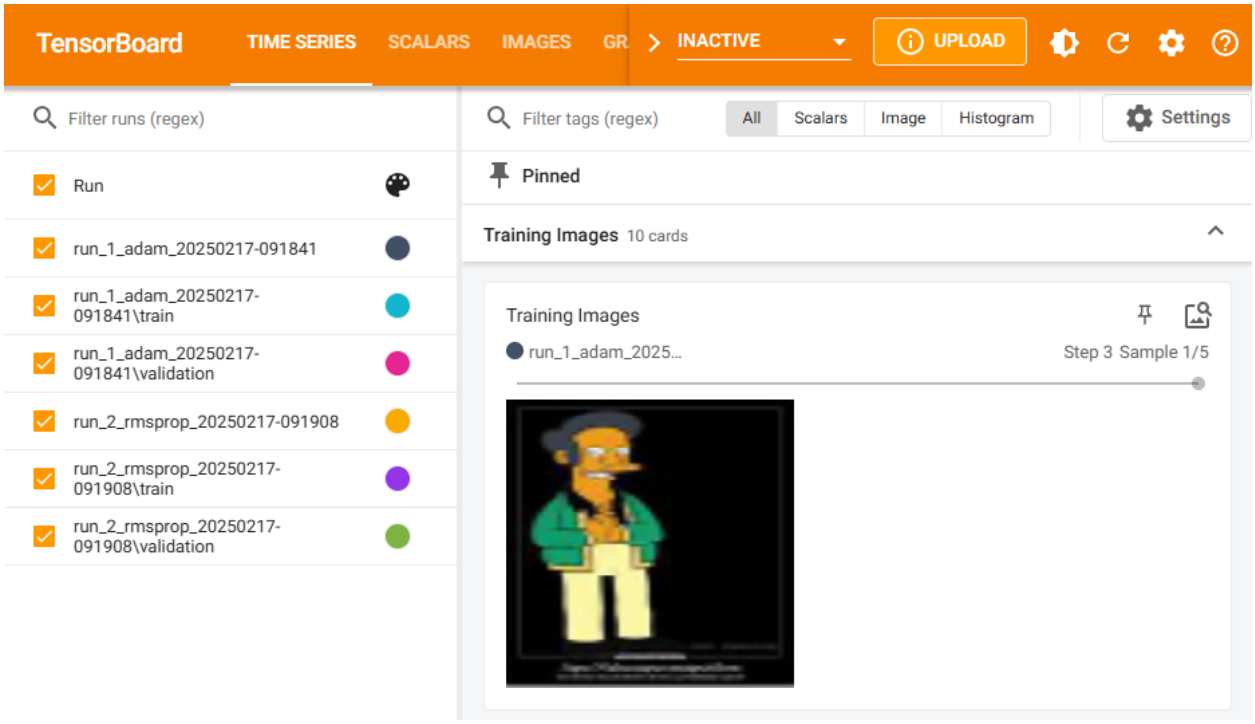


Abbildung 5: Tensorboard Startseite

Beim Start von TensorBoard wird eine mehrteilige Benutzeroberfläche geladen, die verschiedene Visualisierungs- und Analysefunktionen bietet. Die UI besteht aus mehreren Bereichen:

1. Navigationsleiste
    - TensorBoard bietet verschiedene Tabs zur Analyse von Trainingsprozessen, darunter:
        - Time Series: Zeitliche Darstellung von Metriken über mehrere Epochen hinweg.
        - Scalars: Zeigt die Entwicklung von Metriken wie Accuracy und Loss während des Trainings.
        - Images: Visualisiert Bilder, die während des Trainings geloggt wurden (z. B. Eingabebilder oder Augmentierungen).
        - Graphs: Stellt das neuronale Netz als Computational Graph dar.
        - Histograms und Distributions: Analysieren die Gewichtsverteilungen und Aktivierungsänderungen während des Trainings.
        - HParams: Ermöglicht die Visualisierung von Hyperparameter-Suchen, um verschiedene Modellkonfigurationen zu vergleichen.
    - Über die Schaltfläche Upload können externe Log-Dateien importiert werden.
    - Die Einstellungen und Aktualisierungsoptionen bieten Möglichkeiten zur Anpassung der Darstellung und zur Neuladung der Logs.
2. Dropdown-Menü für erweiterte Analysefunktionen

- Neben den Standard-Tabs bietet TensorBoard ein Dropdown-Menü, das Zugriff auf zusätzliche Module ermöglicht:
    - Inactive (Standardanzeige): Hierüber können spezielle Funktionen aktiviert oder deaktiviert werden.
    - Audio: Ermöglicht das Abspielen und Analysieren von Audio-Daten.
    - Debugger v2: Ein Debugging-Tool zur Fehleranalyse während des Modelltrainings.
    - Text: Darstellung und Analyse von Textdaten, z. B. für NLP-Modelle.
    - PR Curves (Precision-Recall Curves): Darstellung der Präzisions-Recall-Kurven für Klassifikationsmodelle.
    - Profile: Performance-Analyse des Modelltrainings, einschließlich Speicher- und Laufzeitanalyse.
    - Mesh: 3D-Visualisierung von Daten, z. B. für Punktwolken in Computer Vision.
    - Projecto“: Interaktive Darstellung von Einbettungen (Embeddings) zur Analyse von Feature-Transformationen.
    - What-If Tool: Ein interaktives Tool zur Modellinterpretierbarkeit und Fairness-Analyse.

3. Run-Filter und Auswahlbereich
    - Im linken Bereich werden alle Trainingsläufe („Runs“) mit farbcodierter Markierung aufgelistet.
    - Jeder Run hat mehrere Unterkategorien, darunter train und validation, die die jeweiligen Trainings- und Validierungsdaten anzeigen.
    - Ein Regex-Filter („Filter runs (regex)“) ermöglicht das gezielte Filtern und Vergleichen bestimmter Runs.
4. Hauptanzeigebereich
    - Hier werden die ausgewählten Metriken oder Visualisierungen dargestellt.
    - In der Abbildung ist der Bereich „Training Images“ geöffnet, in dem Bilder aus dem Trainingsprozess angezeigt werden.
    - Der Titel der Karte zeigt den zugehörigen Run und Step-Informationen an (Step 3 Sample 1/5).

4.2.2 Histogramme & Distributionsanalysen

Als nächstes werden die geloggten Daten näher betrachtet indem die eben gennanten Visualisierungs- und Analysefunktionen genauer betrachtet werden. Dabei bleiben wir zunächst bei der Navigationsleiste bei *Time Series* mit dem Filter Tag *Histrogram***.** Dort wird für jede Neuronale Schicht und ausgewählten

Trainingsrun das Bias und Kernel Histogram ausgegeben.

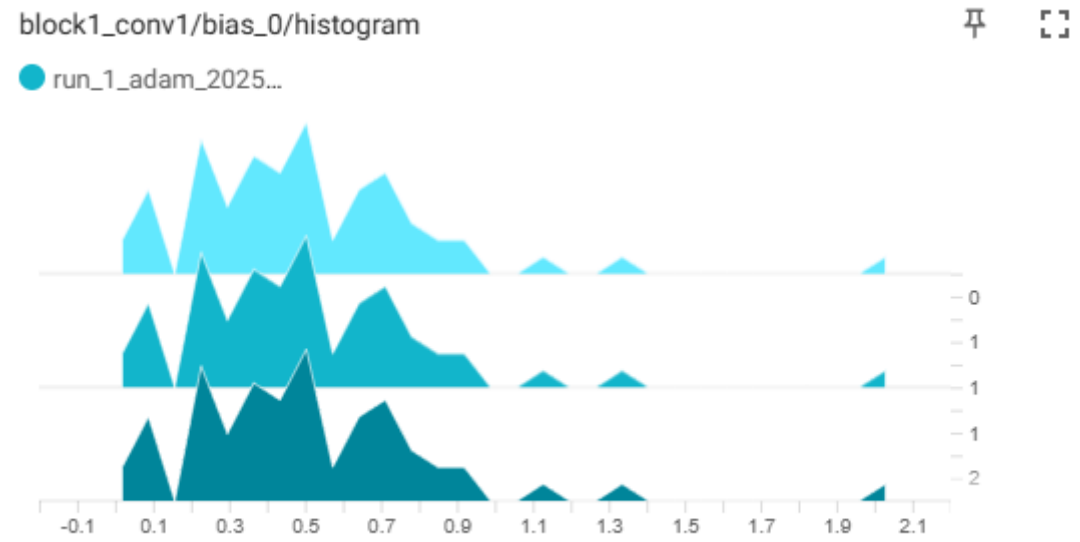


Abbildung 6: Tensorboard Bias Histogram

Die Abbildung zeigt ein Histogramm der Bias-Werte der ersten Convolutional Layer (block1_conv1/bias_0) in TensorBoard. Dieses Diagramm gehört zu den Histogramm-Visualisierungen, die TensorBoard bereitstellt, um die Verteilung von Modellparametern während des Trainings zu analysieren. Es dient dazu, Einblicke in die Gewichts- oder Bias-Änderungen über mehrere Trainingsschritte hinweg zu erhalten.

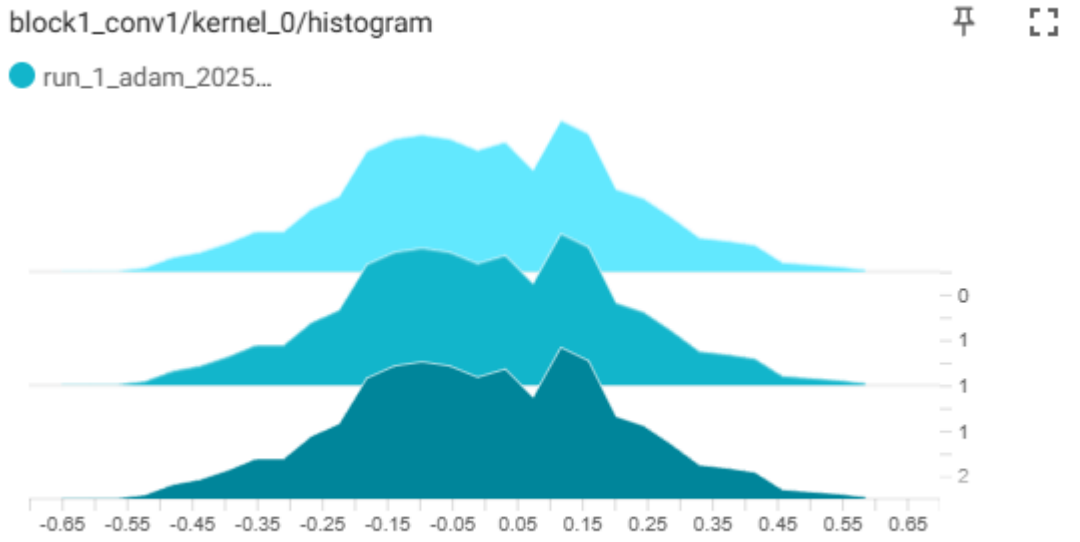


Abbildung 7: Tensorboard Kernel Histogram

Diese Abbildung aus TensorBoard zeigt ein Histogramm der Gewichte (Kernels) einer Convolutional Layer (block1_conv1/kernel_0) über mehrere Trainingsschritte hinweg. Diese Art von Histogramm ist

besonders hilfreich für die Analyse von Convolutional Neural Networks (CNNs) und gibt Einblicke in die interne Anpassung der Gewichte während des Trainingsprozesses.
Wechselt man in der Navigationsleiste von Time Series zu Distrubution werden ähnliche Grafiken angezeigt.

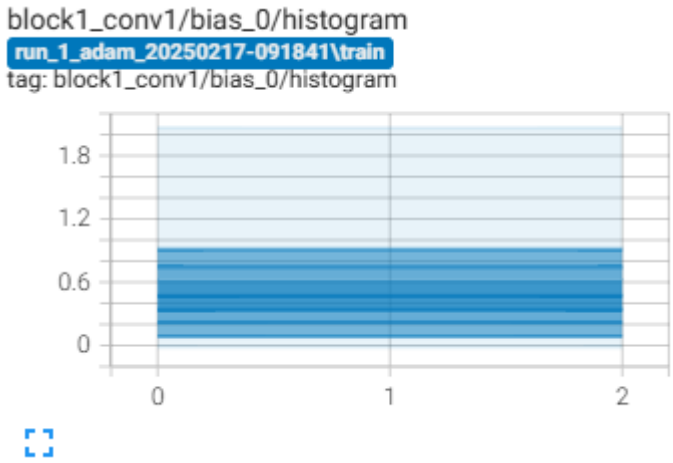


Abbildung 8: Tensorboard Distribution

Diese Abbildung stellt die Verteilung der Bias-Werte für die erste Convolutional Layer (block1_conv1/bias_0) während des Trainings dar. Die Visualisierung zeigt, wie sich die Gewichtsverteilungen über den Trainingsverlauf hinweg verändern. Im Gegensatz zu den Histogrammen, die verschiedene Epochen getrennt darstellen, zeigt die Distribution-Ansicht den Verlauf der Verteilung kontinuierlich.

4.2.3 Scalars

Wieder in der Navigationsleiste Time Series werden jetzt die Scalar Grafiken vorgestellt. Hier werden alle metrischen Logs die in Tensorboard erfasst wurden aufgezeigt.

- epoch_accuracy
    - Zeigt die Genauigkeit (Accuracy) des Modells nach jeder Epoche bzw. detaillierte, batchweise Änderungen während des Trainings bzw. Evulation.
    - Wird häufig verwendet, um den Fortschritt des Modells im Laufe der Zeit zu bewerten.
- epoch_loss
    - Repräsentiert den Loss-Wert nach nach jeder Epoche bzw. detaillierte, batchweise Änderungen während des Trainings bzw. Evulation.
    - Zeigt, wie gut das Modell die Fehler minimiert.
- evaluation_accuracy_vs_iterations
    - Stellt die Genauigkeit des Modells über verschiedene Iterationen der Evaluation hinweg dar.

- Dies ist besonders nützlich, um zu beobachten, ob das Modell auch bei späteren Evaluationsläufen eine stabile Leistung zeigt.

- evaluation_loss_vs_iterations
  - Zeigt den Verlustwert (Loss) über verschiedene Iterationen in der Evaluation.
  - Dies kann helfen zu beurteilen, ob das Modell weiterhin gut generalisiert oder ob es beispielsweise Overfitting entwickelt.

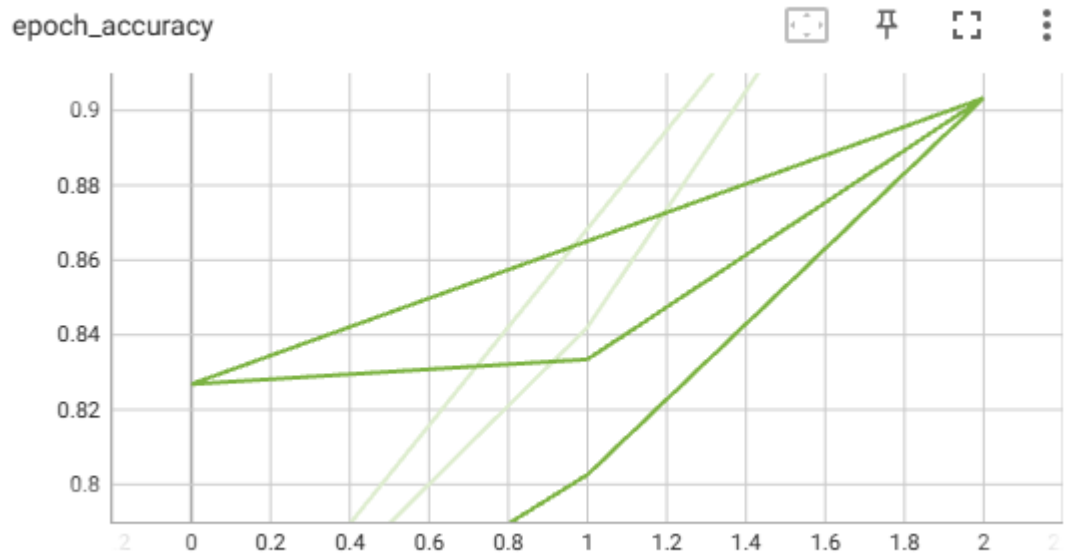


Abbildung 9: Tensorboard epoch_accuracy

Das Bild zeigt eine epoch_accuracy-Kurve in TensorBoard für den run_2_rmsprop_20250217-091908\validation. Auf der x-Achse sind die Trainings-Epochen dargestellt, während die y-Achse die Genauigkeit (Accuracy) des Modells über die Zeit hinweg zeigt. Mehrere Linien deuten darauf hin, dass verschiedene Runs oder Trainingsdurchläufe mit leicht unterschiedlichen Bedingungen oder Hyperparametern verglichen werden. Die Linien steigen im Verlauf der Epochen an, was darauf hinweist, dass die Modellgenauigkeit mit zunehmendem Training verbessert wurde.

Ähnliche Grafiken befinden sich in der Navigatorseite *Scalars.* Dort werden die Metriken nach jeder Epoche Trainings bzw. Evulation dargestellt. Aber dort können noch Einstellungen getroffen werden nach der Sortierung, wie man die Horizontalen Achse darstellen möchte sowie der Grad der Glättung.

4.2.4 HPARAMS

Als nächstes kommt die Grafiken für die Analyse der HPARAMS, indem in der Navigatorseite dort hin gelangen.

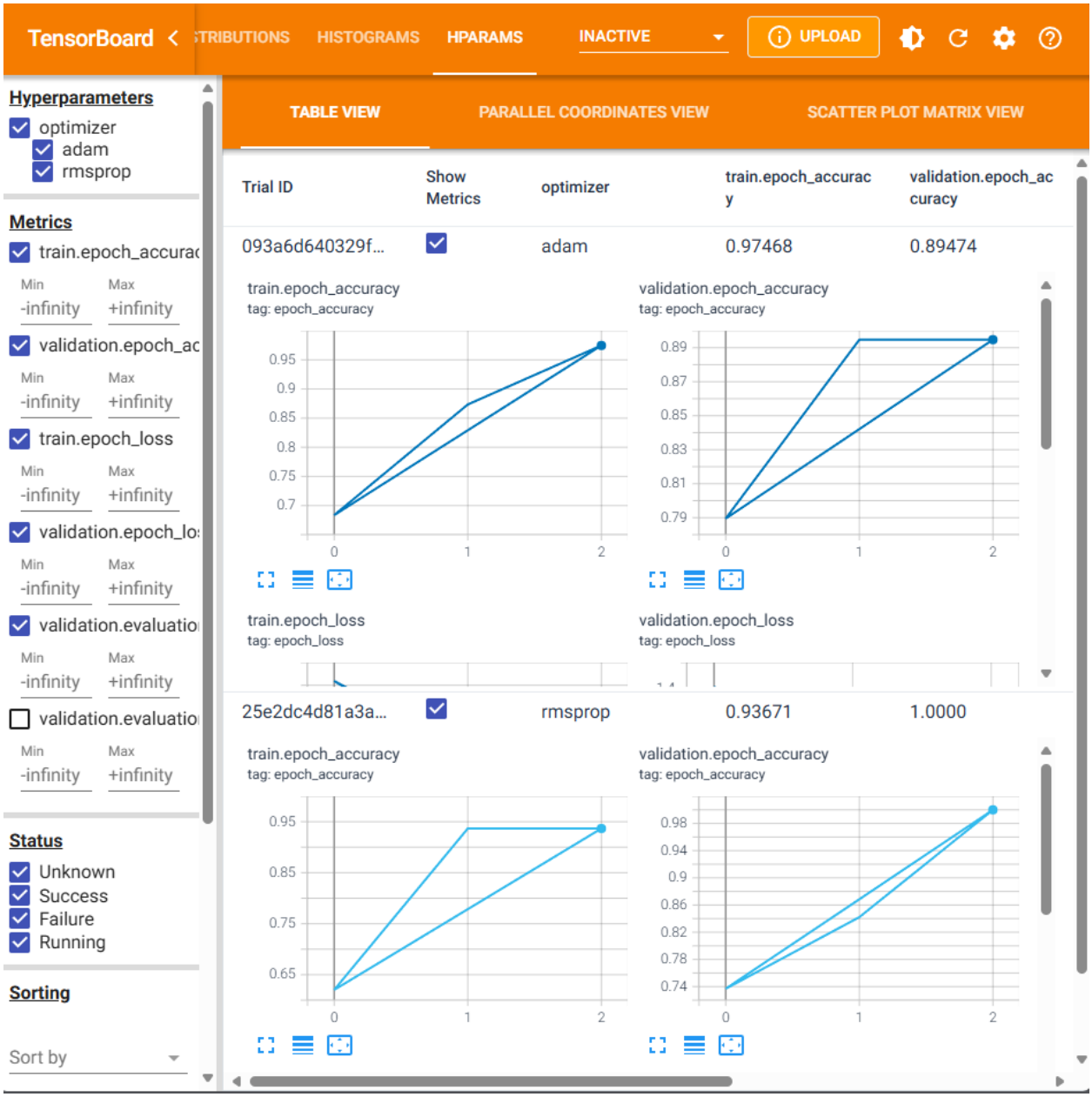


Abbildung 10: Tensorboard HPARAMS Table View

Das Bild zeigt die HPARAMS-Ansicht in TensorBoard, die zur Analyse von Hyperparametern und deren Einfluss auf das Training genutzt wird.

Elemente der HPARAMS-Ansicht

1. Linke Seitenleiste (Hyperparameters & Metrics)
    - Hyperparameters: Hier sind die getesteten Hyperparameter sichtbar. In diesem Fall wurden zwei verschiedene Optimizer („adam“ und „rmsprop“) für das Training ausprobiert.
    - Metrics: Zeigt die verfolgten Metriken, wie Trainings- und Validierungsgenauigkeit (epoch_accuracy) sowie Trainings- und Validierungsverlust (epoch_loss).

2. Hauptansicht – Tabellenansicht (Table View)
    - Hier werden verschiedene Trial IDs angezeigt, die unterschiedliche Trainingsläufe repräsentieren.
    - Jeder Lauf ist mit einem bestimmten Optimizer verknüpft, wodurch sich die Performance von „adam“ und „rmsprop“ direkt vergleichen lässt.
    - Spaltenübersicht:
        - train.epoch_accuracy: Die Trainingsgenauigkeit des Modells über die Epochen hinweg.
        - validation.epoch_accuracy: Die Genauigkeit des Modells auf den Validierungsdaten.
    - Diagramme:
        - Die linken Diagramme zeigen die Entwicklung der Trainingsgenauigkeit und des Trainingsverlusts pro Epoch.
        - Die rechten Diagramme zeigen die Entwicklung der Validierungsgenauigkeit und des Validierungsverlusts.
        - Durch den direkten Vergleich zwischen verschiedenen Optimierern kann festgestellt werden, welcher sich besser für das Problem eignet.
3. Filter- und Sortiermöglichkeiten
    - Status: Filtert nach Trainingsergebnissen (z. B. „Running“, „Success“, „Failure“).
    - Sorting: Läufe können nach bestimmten Metriken sortiert werden, um die besten Konfigurationen schnell zu identifizieren.

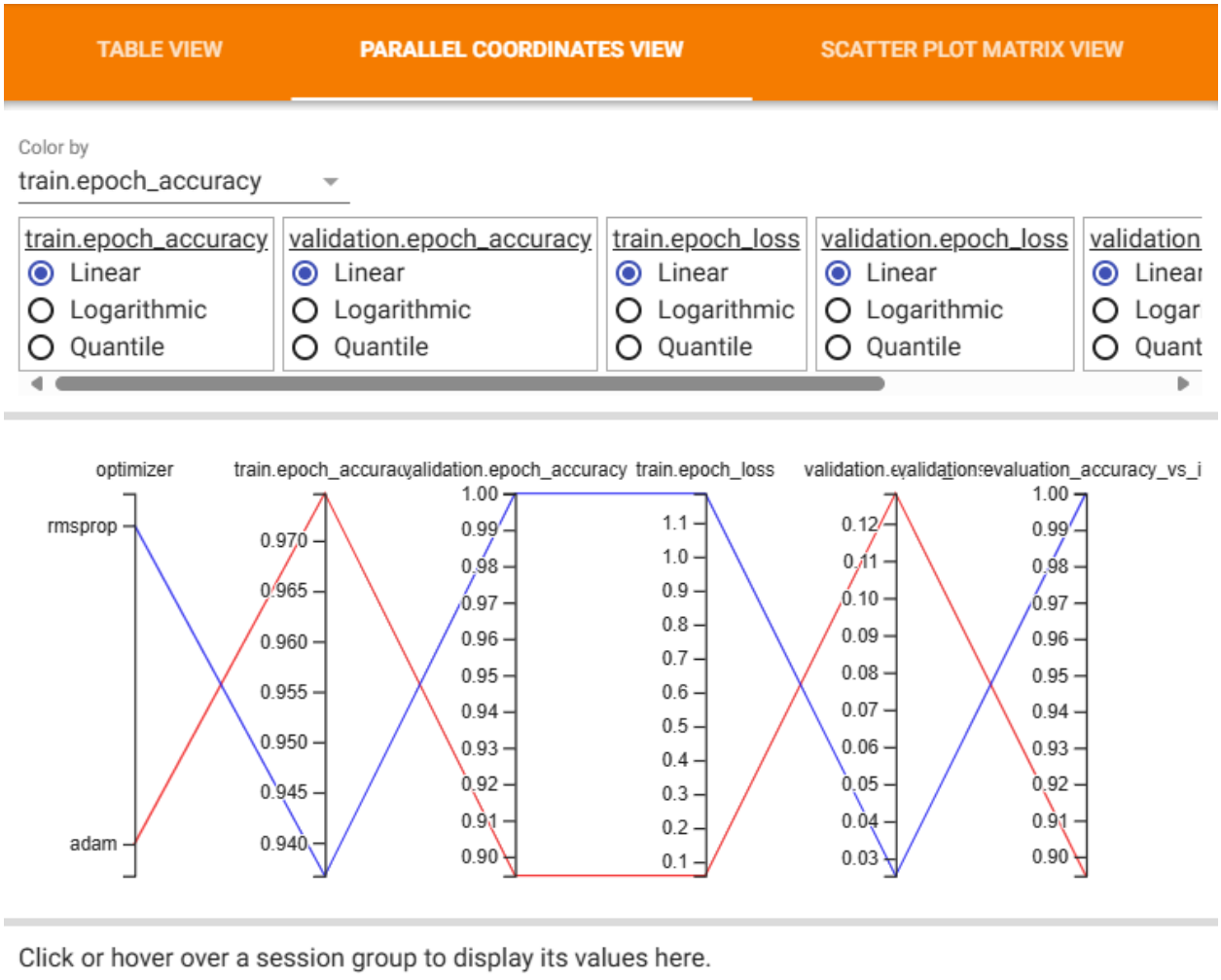


Abbildung 11: Tensorboard HPARAMS Parallel Coordinates View

Diese Abbildung zeigt die Parallel Coordinates View in TensorBoard, die zur Visualisierung der Hyperparameter-Optimierung dient. Diese Ansicht ermöglicht es, mehrere Parameter und Metriken auf einer gemeinsamen Skala darzustellen, um Zusammenhänge zwischen ihnen zu erkennen.

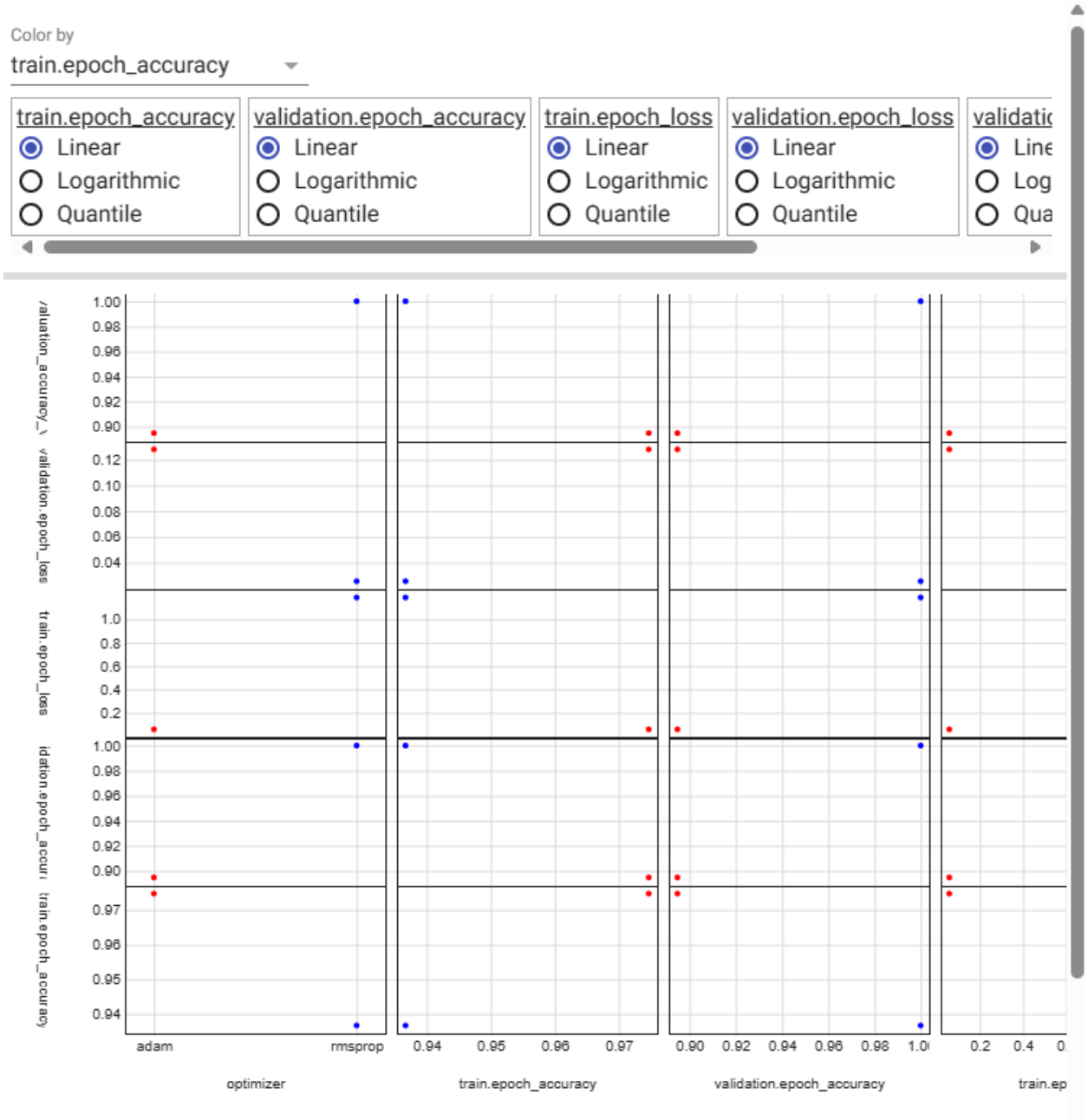


Abbildung 12: Tensorboard HPARAMS Scatter Plot Matrix View

Diese Abbildung zeigt die Scatter Plot Matrix View in TensorBoard, die zur Analyse von Hyperparametern und Modellmetriken verwendet wird. Diese Ansicht stellt die Beziehungen zwischen verschiedenen Metriken und Optimierungsparametern in Form einer Matrix aus Streudiagrammen dar.

4.2.5 Graphs

Zum Schluss bietet Tensorboard mit Graph in der Navigatorleiste die möglichkeit die Architektur eines neuronalen Netzes sowie dessen Berechnungsgraphen detailliert zu analysieren. Diese Funktion ermöglicht es, die Struktur des Modells, den Datenfluss sowie die verschiedenen Operationen zu inspizieren.

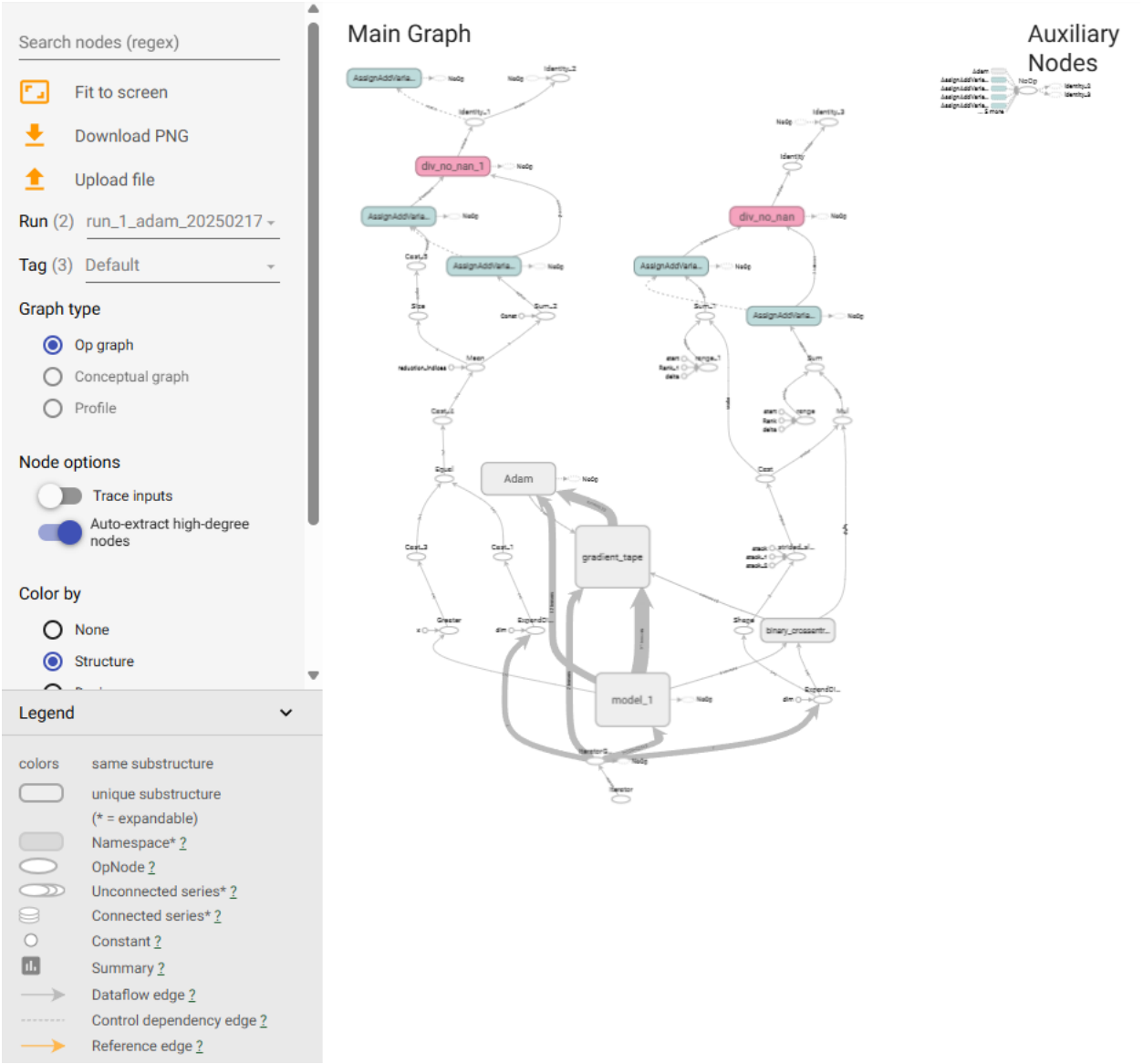


Abbildung 13: Tensorboard Graphs

Die gezeigte Abbildung stellt die Graph-Visualisierung eines neuronalen Netzes in TensorBoard dar. Es enthält mehrere Abschnitte:

1. Main Graph

- Der zentrale Bereich zeigt die Berechnungsgraphen des Modells.
- Zu sehen sind verschiedene Operationen (Op-Nodes), darunter mathematische Berechnungen, Parameter-Updates und Datenflüsse.
- Große, hervorgehobene Knoten wie "Adam" (der verwendete Optimizer) oder "model_1" (die Modellarchitektur) deuten auf wichtige Komponenten des Trainings hin.

2. Auxiliary Nodes („Hilfs-Knoten“)

- Rechts im Bild befinden sich zusätzliche Knoten, die oft Metadaten oder globale Variablen enthalten.

3. Farbgebung und Struktur

- In der Legende unten links wird die Bedeutung verschiedener Farben und Verbindungstypen erläutert:
    - Hellblaue und rosafarbene Knoten (z. B. „div_no_nan_1“ und „AssignAddVariableOp“) stehen für spezielle Operationen oder Kontrollmechanismen.
    - Graue Pfeile stellen Abhängigkeiten dar, während andere Verbindungen Datenflüsse und Berechnungen symbolisieren.
    - "gradient_tape" ist ein Mechanismus für automatische Differenzierung, der in TensorFlow verwendet wird, um Gradienten während des Backpropagations-Trainings zu berechnen.

4. Steuerung und Navigation

Auf der linken Seite der TensorBoard-Oberfläche gibt es verschiedene Steuerungsoptionen:

- „Graph Type“ ermöglicht es, zwischen Op-Graph, Conceptual Graph und Profiling-Ansicht umzuschalten.
- „Trace Inputs“ kann aktiviert werden, um detaillierte Input-Verbindungen anzuzeigen.
- „Color by“ kann zwischen „None“ und „Structure“ umgeschaltet werden, um unterschiedliche Darstellungen des Graphen zu ermöglichen.

5. Vergleich der MLflow UI und TensorBoard UI

Nachdem in den vorherigen Kapiteln die Funktionsweise und Nutzung der MLflow UI sowie der TensorBoard UI detailliert betrachtet wurden, erfolgt in diesem Kapitel eine systematische Gegenüberstellung beider Tools. Durch diese vergleichende Analyse soll herausgearbeitet werden, welche Stärken und Schwächen MLflow und TensorBoard besitzen und wie sich ihre Nutzung optimal an spezifische Anforderungen im Machine-Learning-Prozess anpassen lässt.

5.1 MLFlow und Tensorboard: Vergleich und Einsatzbereiche

MLflow und TensorBoard sind zwei der wichtigsten Tools im Machine-Learning-Workflow, doch sie haben unterschiedliche Schwerpunkte. Während MLflow eine ganzheitliche Plattform für das Experiment-Tracking, die Modellverwaltung und das Deployment bietet, konzentriert sich TensorBoard auf die tiefgehende Analyse von Modellarchitekturen und Trainingsprozessen. Beide Tools ermöglichen es, Trainingsverläufe systematisch zu dokumentieren, Modelle zu analysieren und die Entwicklung zu optimieren, unterscheiden sich jedoch in ihrer Funktionalität, Flexibilität und Spezialisierung.

MLflow deckt den gesamten ML-Lebenszyklus ab – von der Experimentverfolgung über die Modellversionierung bis hin zur Bereitstellung und Reproduzierbarkeit in produktiven Umgebungen. Mit MLflow Tracking lassen sich alle relevanten Parameter, Metriken und Artefakte eines Trainingslaufs systematisch speichern, sodass Experimente strukturiert analysiert und optimiert werden können.

Die MLflow Model Registry erlaubt es, verschiedene Modellversionen zu verwalten, zu testen und gezielt in eine Staging- oder Produktionsumgebung zu überführen. Zusätzlich ermöglichen MLflow Projects und MLflow Recipes die Standardisierung und Automatisierung von ML-Workflows, wodurch sich MLflow besonders für MLDevOps-Prozesse und den Einsatz in CI/CD-Pipelines oder Cloud-Deployments eignet.

Ein zentraler Vorteil von MLflow ist seine Flexibilität: Es unterstützt eine Vielzahl von Machine-Learning-Frameworks, darunter TensorFlow, PyTorch, Scikit-learn und XGBoost, und kann unabhängig vom gewählten ML-Stack eingesetzt werden. Dies macht es besonders attraktiv für Unternehmen und Forschungsteams, die unterschiedliche Machine-Learning-Modelle auf einer einheitlichen Plattform verwalten und vergleichen möchten.

Im Gegensatz zu MLflow, das sich auf die organisatorische Verwaltung von ML-Prozessen konzentriert, ist TensorBoard speziell für die detaillierte Analyse und Visualisierung von Modellarchitekturen und Trainingsmetriken konzipiert. Es bietet umfangreiche Funktionen zur Darstellung von Trainingsmetriken, Netzwerkarchitekturen, Gewichtsanpassungen und Hyperparametern, um tiefgehende Einblicke in die Trainingsprozesse neuronaler Netze zu ermöglichen.

Ein besonderer Vorteil von TensorBoard ist die Echtzeitüberwachung des Trainingsfortschritts. Durch Skalare wie Accuracy und Loss, Gewichtshistogramme sowie die Visualisierung des Netzwerkgraphen können Modellanpassungen und Performance-Verbesserungen gezielt gesteuert werden. Das HParams-Dashboard erlaubt zudem die direkte Analyse des Einflusses verschiedener Hyperparameter auf das Modelltraining, was insbesondere für Hyperparameter-Tuning essenziell ist.

TensorBoard entfaltet seine volle Leistungsfähigkeit innerhalb des TensorFlow-Ökosystems und ist insbesondere für Deep-Learning-Anwendungen optimiert. Zwar kann es mit anderen Frameworks genutzt werden, doch die Integration in TensorFlow und Keras ist am nahtlosesten.

Der größte Unterschied zwischen MLflow und TensorBoard liegt in ihrer strategischen Ausrichtung. MLflow ist ein universelles Tool für Machine Learning-Workflows, das über verschiedene Frameworks hinweg funktioniert und sich ideal für MLDevOps eignet. TensorBoard hingegen bietet eine tiefgehende Analyse des Trainingsprozesses, ist aber primär auf Deep Learning mit TensorFlow ausgerichtet.

MLflow eignet sich besonders für (neptune.ai, 2025):

- Teams, die verschiedene Machine-Learning-Frameworks nutzen und eine einheitliche Plattform für das Tracking, die Modellverwaltung und das Deployment benötigen.
- MLDevOps-Prozesse, die eine Integration mit CI/CD-Pipelines oder Cloud-Umgebungen erfordern.
- Projekte, die viele Experimente durchführen und Modellversionen langfristig verwalten möchten.

TensorBoard eignet sich besonders für (neptune.ai, 2025):

- Deep-Learning-Projekte mit TensorFlow, die eine detaillierte Analyse von Modellarchitekturen, Gewichtsverteilungen und Hyperparametern benötigen.
- Entwickler und Forscher, die neuronale Netze optimieren und deren Verhalten im Detail verstehen möchten.
- Projekte, bei denen eine Echtzeit-Visualisierung des Modelltrainings essenziell ist.

Obwohl sich MLflow und TensorBoard in ihrem Fokus unterscheiden, sind sie keineswegs konkurrierende Tools – im Gegenteil, sie ergänzen sich hervorragend. TensorBoard kann während des Modelltrainings genutzt werden, um detaillierte Einblicke in das Netzwerkverhalten zu gewinnen, während MLflow die systematische Verwaltung der Experimente, Modellversionierung und Bereitstellung übernimmt.
Durch eine Kombination beider Tools lassen sich sowohl tiefergehende Modellanalysen durchführen als auch Experimente langfristig verwalten und reproduzieren, was insbesondere in komplexen Machine-Learning-Projekten von Vorteil ist.

6. Zusammenfassung

In dieser Arbeit wurden MLflow und TensorBoard als zwei wichtige Tools für das Management und die Analyse von Machine-Learning-Projekten betrachtet. Beide Werkzeuge bieten umfassende Funktionen, unterscheiden sich jedoch deutlich in ihrer Ausrichtung und Anwendung.

MLflow dient vor allem der Nachverfolgung von Experimenten, der Modellverwaltung und der Bereitstellung von Machine-Learning-Modellen. Es bietet eine strukturierte Möglichkeit, verschiedene Modellversionen zu speichern und Experimente über mehrere Durchläufe hinweg zu vergleichen. TensorBoard hingegen ist besonders für die Visualisierung von Modellarchitekturen und Trainingsprozessen optimiert. Hier können unter anderem Trainingsmetriken, Gewichtsverteilungen und neuronale Netzwerke analysiert werden.

Die Untersuchung der Benutzeroberflächen zeigte, dass MLflow vor allem für die Verwaltung von Experimenten und Modellen geeignet ist, während TensorBoard detaillierte Einblicke in das Training von Modellen ermöglicht. Beide Tools ergänzen sich in vielen Bereichen und können gemeinsam genutzt werden, um eine umfassende Übersicht über den gesamten Machine-Learning-Workflow zu erhalten.

Je nach Anwendungsfall und Anforderungen eines Projekts kann eines der beiden Tools besser geeignet sein:

- MLflow ist ideal für Projekte, bei denen verschiedene Modellversionen strukturiert gespeichert, verglichen und verwaltet werden sollen. Besonders wenn Modelle in den produktiven Einsatz überführt werden, bietet MLflow eine klare Nachverfolgbarkeit und Organisation.

- TensorBoard hingegen eignet sich besonders für tiefgehende Analysen während des Modelltrainings. Es stellt detaillierte Visualisierungen bereit, die helfen, Modellveränderungen zu verstehen und die Trainingsprozesse zu optimieren.

Beide Tools haben somit ihre eigenen Stärken. Während MLflow insbesondere für strukturierte Machine-Learning-Workflows und Modellversionierung nützlich ist, bietet TensorBoard detaillierte Echtzeit-Analysen für das Training von Modellen.

Machine Learning entwickelt sich ständig weiter, und damit auch die unterstützenden Tools. Es ist zu erwarten, dass sowohl MLflow als auch TensorBoard in Zukunft noch mehr Funktionen bieten werden, um den Anforderungen immer komplexerer Modelle gerecht zu werden.

MLflow könnte stärker auf automatisierte Workflows und bessere Integration in Entwicklungssysteme ausgerichtet werden, um den gesamten Machine-Learning-Prozess weiter zu vereinfachen. TensorBoard hingegen könnte erweiterte Visualisierungen und neue Analysemöglichkeiten bieten, um noch detailliertere Einblicke in neuronale Netzwerke zu ermöglichen.

Ein interessanter Aspekt ist auch die Möglichkeit, beide Tools gemeinsam einzusetzen, um sowohl eine strukturierte Experimentverwaltung als auch eine detaillierte Modellanalyse zu ermöglichen. Durch diese Kombination ließen sich sowohl Trainingsprozesse optimieren als auch Modelle effizient verwalten.

Insgesamt bieten MLflow und TensorBoard wertvolle Unterstützung für Machine-Learning-Projekte, indem sie verschiedene Aspekte des Modelltrainings und -managements abdecken. Welche Lösung am besten geeignet ist, hängt letztendlich von den spezifischen Anforderungen des jeweiligen Projekts ab.

Literaturverzeichnis

MLflow. (2025a, Februar 10). *What is MLflow? —MLflow 2.7.0 documentation*. https://mlflow.org/docs/2.7.0/what-is-mlflow.html?utm_source=chatgpt.com

MLflow. (2025b, Februar 10). MLflow Overview. https://mlflow.org/docs/latest/introduction/index.html?utm_source=chatgpt.com

MLflow. (2025c, Februar 10). *MLflow Tracking*. https://mlflow.org/docs/latest/tracking.html

MLflow. (2025d, Februar 10). *MLflow Projects*. https://mlflow.org/docs/latest/projects.html

Mlflow. (2025e, Februar 10). *MLflow Models*. https://mlflow.org/docs/latest/models.htmlD

MLflow. (2025f, Februar 10). *MLflow Model Registry*. https://mlflow.org/docs/latest/model-registry.html

MLflow. (2025g, Februar 10). *MLflow Recipes*. https://mlflow.org/docs/latest/recipes.html

neptune.ai. (2025, Februar 17). *MLflow vs TensorBoard vs Neptune*. Neptune.Ai. https://neptune.ai/vs/tensorboard-mlflow

TensorFlow. (2025a, Februar 10). *TensorBoard | TensorFlow*. https://www.tensorflow.org/tensorboard

TensorFlow. (2025b, 02). *TensorBoard Scalars: Logging training metrics in Keras | TensorFlow*. https://www.tensorflow.org/tensorboard/scalars_and_keras

TensorFlow. (2025c, Februar 10). *Examining the TensorFlow Graph | TensorBoard*. TensorFlow. https://www.tensorflow.org/tensorboard/graphs

TensorFlow. (2025d, Februar 10). *Hyperparameter Tuning with the HParams Dashboard | TensorBoard | TensorFlow*. https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams

TensorFlow. (2025e, Februar 10). *Visualizing Data using the Embedding Projector in TensorBoard*. TensorFlow. https://www.tensorflow.org/tensorboard/tensorboard_projector_plugin

TensorFlow. (2025f, Februar 10). *Displaying image data in TensorBoard*. TensorFlow. https://www.tensorflow.org/tensorboard/image_summaries

TensorFlow. (2025g, Februar 10). *Displaying text data in TensorBoard | TensorFlow*. https://www.tensorflow.org/tensorboard/text_summaries

TensorFlow. (2025h, Februar 10): Profile model performance | TensorBoard. (2025, Februar 10). https://www.tensorflow.org/tensorboard/tensorboard_profiling_keras

Eidesstattliche Versicherung

Ich versichere, dass ich das beiliegende Assignment selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.