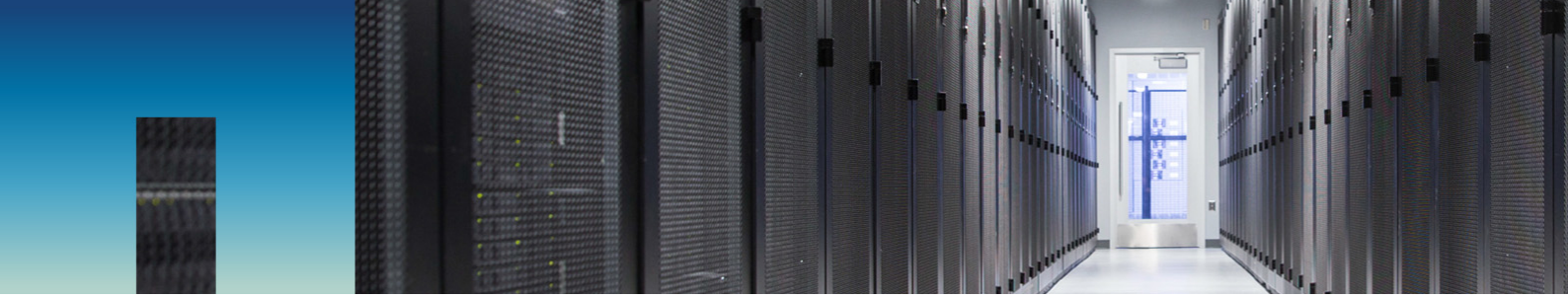


Whitepaper

Aufbau einer Daten-Pipeline für Deep Learning

Ein KI-Projekt von der Pilotphase bis zur Produktion führen

Santosh Rao, NetApp März 2019 | WP-7299

Zusammenfassung

Dieses Whitepaper beschreibt, was zu beachten ist, um ein Deep-Learning-Projekt von der ersten Konzeption bis zur Produktion zu führen. Es hilft Ihnen, die eigenen Geschäfts- und Datenanforderungen zu verstehen und eine mehrstufige Daten-Pipeline für Datenaufnahme, Datenvorbereitung, Training, Validierung und Nutzung eines KI-Modells zu entwerfen.

NetApp®

INHALT

TABELLENVERZEICHNIS

ABBILDUNGSVERZEICHNIS

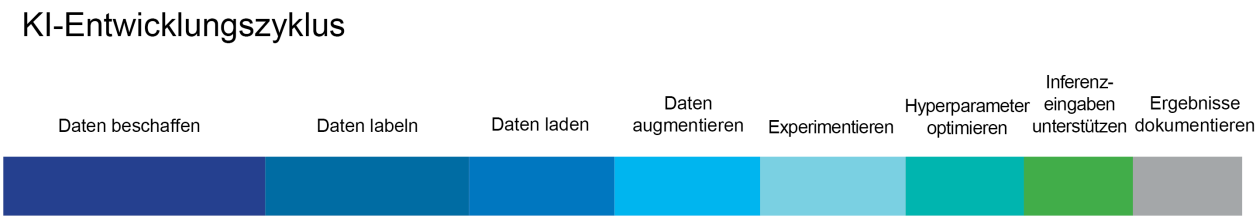
1 Zielpublikum

Dieses Whitepaper richtet sich vorrangig an Data Engineers, Infrastructure Engineers, Big Data Architects und Branchenberater, die die Möglichkeiten von Deep Learning (DL) ausloten oder sich damit befassen. Es bietet auch hilfreiche Informationen für Infrastrukturteams, um die Anforderungen von Data Scientists zu verstehen und zu erfüllen, wenn KI-Projekte von der Pilotphase in die Produktion übergehen.

2 Einführung

Der Erfolg eines KI-Projekts hängt von vielen Faktoren ab, angefangen bei der Auswahl des bestmöglichen Anwendungsfalls, über die Zusammenstellung eines Teams mit den richtigen Kompetenzen, bis hin zur Entscheidung für die bestmögliche Infrastruktur. Angesichts der Komplexität eines solchen Vorhabens wird die Bedeutung von Daten in diesem Prozess meist unterschätzt. Die in Abbildung 1 dargestellte Zeitleiste für ein typisches KI-Projekt verdeutlicht jedoch, dass ein Großteil der Zeit für datenbezogene Aufgaben wie das Sammeln, Labeln, Laden und Ergänzen von Daten aufgewendet wird.

Abbildung 1) Datenbezogene Aufgaben beanspruchen die meiste Zeit bei Deep-Learning-Projekten



Genau hier setzt das Konzept einer Daten-Pipeline an. Eine Daten-Pipeline ist die Software und unterstützende Hardware, die erforderlich ist, um effizient alle Daten zu sammeln, vorzubereiten und zu managen. Diese Daten werden benötigt, um einen KI-Algorithmus zu trainieren, zu validieren und in Betrieb zu nehmen.

In den frühen Phasen der KI-Planung und -Entwicklung ist die Bedeutung einer gut durchdachten Daten-Pipeline nicht immer sofort ersichtlich, aber ihre Bedeutung nimmt mit steigendem Datenvolumen und mit dem Übergang des trainierten Modells vom Prototypen zur Produktionsreife zu. Letztendlich hängt der gesamte Erfolg eines Projekts von der Effektivität der Pipeline ab. Wer sich früh genug mit der Realisierung der Datenanforderungen auseinandersetzt, kann sich später aufwendige Architekturänderungen ersparen.

Dieses Whitepaper hilft dabei, ein Verständnis für die Elemente einer effektiven Daten-Pipeline für KI-Projekte zu entwickeln:

- Was sind die gängigsten Optionen im Software-Stack für die einzelnen Stufen?
- Wann sollten die verschiedenen Software-Optionen angewendet werden?
- Wie arbeiten die Software und Hardware zusammen?

Dieses Whitepaper befasst sich hauptsächlich mit dem Aufbau einer Daten-Pipeline für Deep Learning, aber viele der hierin enthaltenen Informationen lassen sich auch auf andere Anwendungsfälle für Machine Learning und auf Big-Data-Analysen übertragen.

Herausforderungen für eine erfolgreiche KI-Implementierung

Bei KI-Projekten gibt es in den unterschiedlichsten Bereichen große Herausforderungen.

Data Engineering. Bei KI-Projekten muss in der Regel besonders viel Zeit für datenbezogene Aufgaben wie Labeling, Normalisierung, Tokenisierung usw aufgewendet werden.

Berechnung. Das Training eines Deep-Learning-Algorithmus erfordert eine hohe Anzahl von Kernen. Für das KI-Training mit großen Datensätzen braucht man Grafikprozessoren (GPUs) mit Tausenden von Kernen und speziell entwickelte Hardware-Interconnects.

Algorithmusentscheidungen. Das Training wird in der Regel parallel auf vielen GPUs gleichzeitig ausgeführt.

- Wenn das Modell in den GPU-Speicher passt, werden Kopien des Modells auf vielen GPUs genutzt und alle GPUs zeitgleich mit individuellen Untermengen der Daten versorgt, um die KI zu trainieren. Die GPUs müssen synchron oder asynchron miteinander kommunizieren. Hierbei gilt es, einen Kompromiss zwischen Konvergenz und niedrigeren Kommunikationskosten zu finden.
- Wenn das Modell nicht in den Speicher einer einzigen GPU passt, muss es aufgeteilt und über mehrere GPUs ausgeführt werden.

Infrastrukturherausforderungen. GPUs, CPUs, Netzwerk-Switches und schneller Speicher müssen sorgfältig zusammengestellt werden, um eine optimale Konfiguration zu erzielen und Engpässe zu vermeiden. Dies gilt sowohl für das Trainieren des Modells als auch für das Inferencing im Anschluss.

Betriebsherausforderungen. Diese Herausforderungen unterteilen sich in die Bereiche ScienceOps und DevOps:

- ScienceOps befasst sich mit Datenherkunft, Experiment-Management, Hyperparameter-Tuning und anderen Aufgaben, die für ein gut trainiertes Deep-Learning-Modell erforderlich sind.
- DevOps befasst sich z. B. mit der Übermittlung von KI-Trainings-Jobs (per Batch oder interaktiv), dem Job-Monitoring, der Zeitplanung usw. Nach der Fertigstellung eines Modells fallen weitere DevOps-Aufgaben an, um das Modell über Code zu integrieren. Außerdem müssen Implementierungen gemanagt werden, wenn das Modell mit der Zeit neu trainiert oder verbessert wird oder die Software, die die Modellausgaben verwendet, weiterentwickelt wird.

3 Was ist eine Daten-Pipeline?

Eine Daten-Pipeline dient dem Zweck, das Sammeln und das Management der für das Training des Modells erforderlichen Datensätze zu ermöglichen. Dazu müssen die Daten in eine Form gebracht werden, die das Modell aufnehmen und verstehen kann. Die zugrunde liegende Architektur einer Pipeline hängt von den jeweils verwendeten Quellen und Datentypen ab. Datentypen und Datenquellen werden in Abschnitt 5. Eigene Datenanforderungen verstehen, erläutert.

Eine Daten-Pipeline wird logisch in Stufen unterteilt, wie in Abbildung 2 dargestellt.

Abbildung 2) Stufen einer Daten-Pipeline für Deep Learning

In den jeweiligen Stufen eines typischen Deep-Learning-Anwendungsfalls passiert Folgendes:

- **Datenaufnahme**: Daten werden in der Regel im Edge-Bereich aufgenommen, z. B. von Autos, Point-of-Sale-Geräten oder Kameras an Montagebändern. Je nach Anwendungsfall müssen diese Daten in Echtzeit gestreamt werden oder sie werden im Edge-Bereich zwischengespeichert und in regelmäßigen Intervallen übertragen.
- **Datenvorbereitung**: Um Daten für das Training vorzubereiten, ist fast immer eine Vorverarbeitung erforderlich, in welcher Form auch immer. Die Vorverarbeitung findet häufig in einem Data Lake statt.
- **Training**: Für das Training werden möglicherweise Datensätze aus dem Data Lake in einen Trainings-Cluster mit teilweise sehr hohen I/O-Anforderungen kopiert.
- **Validierung**: Bevor ein Modell eingesetzt werden kann, muss es zunächst validiert werden, um sicherzustellen, dass die Ergebnisse brauchbar sind. Möglicherweise sind mehrere Zyklen von Datenvorbereitung, Training und Validierung erforderlich, um die Merkmale und Parameter zu bestimmen, die zu den besten Ergebnissen führen.
- **Implementierung**: Das fertige Modell wird zur Softwareentwicklung und anschließend in die Produktion verschoben. Je nach Anwendungsfall wird das Modell ggf. auch für Edge-Prozesse implementiert. Die Ergebnisse des Modells werden überwacht. Rückmeldungen fließen als neue Daten zusammen mit anderen neuen Daten zurück in den Data Lake, um den Prozess zu iterieren und das Modell regelmäßig neu zu trainieren.
- **Archivierung.** Die für die Trainingsläufe genutzten Daten werden ggfs. zeitlich unbegrenzt gespeichert. Viele KI-Teams archivieren solche Daten in einem Objektspeicher in einer Private oder Public Cloud.

Jede einzelne Stufe ist wichtig, jedoch sind die Übergänge zwischen den Stufen häufig genauso wichtig. Die Daten müssen effizient von der Aufnahme über die Vorbereitung zum Training und schließlich zur Validierung fließen. In den nachfolgenden Abschnitten dieses Whitepapers wird genauer auf die einzelnen Stufen und wichtigen Übergänge eingegangen.

Software 1.0 vs. Software 2.0

Bei der traditionellen Softwareentwicklung schreiben die Entwickler den Code auf Laptops oder kleinen Workstations – die fertige Software wird später mithilfe eines klar definierten Workflows auf High-End-Produktionsservern implementiert:

Entwicklung → Implementierung → Produktion

Bei KI-Entwicklung ist der „Entwicklungs-Server“ hingegen meist ein GPU-Supercomputer (Trainings-Cluster) und das trainierte Modell wird später auf einem kleineren System mit GPU-Unterstützung oder zusätzlicher Spezialhardware für das Inferencing eingesetzt. Die Implementierung eines Modells vom Trainings-System auf ein Inferenz-System nennt sich Serving. Der Workflow für die KI-Implementierung sieht demnach so aus:

Training → Serving → Inferencing

4 Eigene Geschäftsanforderungen verstehen

KI entwickelt sich zunehmend zu einer elementaren Technologie mit Vorteilen für alle Branchen. Sie führt zu tiefgreifenden Veränderungen in der Landwirtschaft, Fertigungsindustrie, Automobilindustrie, Pharmaindustrie oder Finanzdienstleistungen. In vielen Branchen sind Machine Learning und Deep Learning schon jetzt wesentliche Faktoren für die Wettbewerbsfähigkeit und Zukunftsfähigkeit von Unternehmen.

Am Anfang eines jeden KI-Projekts steht die Analyse der jeweiligen geschäftlichen Anforderungen, um den besten Anwendungsfall für das eigene Unternehmen zu bestimmen. Viele Teams beginnen mit einem für ihre Branche typischen Anwendungsfall, vor allem beim ersten KI-Projekt.

Es ist sinnvoll, zunächst „klein anzufangen“, mit einem Anwendungsfall mit guten Aussichten auf schnelle Ergebnisse zu beginnen und diesen dann auf andere Geschäftsbereiche auszuweiten. So hat Liberty Mutual Insurance mit einem digitalen Assistenten (oder Chatbot) für interne Mitarbeiter angefangen – ein Anwendungsfall mit geringen Risiken und ohne direkte Auswirkungen auf Kunden. Nach der erfolgreichen Umsetzung erprobte Liberty Mutual ein technologisch ähnliches Projekt im Kunden-Callcenter und brachte die Software schließlich über die Tochtergesellschaft Workgrid Software auf den Markt.

Abbildung 3 zeigt einige gängige Anwendungsfälle für verschiedene Branchen.

Abbildung 3) Gängige KI-Anwendungsfälle in verschiedenen Branchen

5 Eigene Datenanforderungen verstehen

> *Nur wenn ein Unternehmen über relevante Daten verfügt und die erforderlichen Schritte zum Vorbereiten und Wandeln der Daten für das Training von Modellen unternommen hat, können leistungsstarke Rechenressourcen und Frameworks für Machine Learning genutzt werden.*
>
> – Implementing AI: From Exploration to Execution

Nach der Auswahl eines KI-Anwendungsfalls müssen die entsprechenden Daten kritisch beurteilt werden. Wenn die erforderlichen Daten fehlen, müssen Wege gefunden werden, um diese zu beschaffen. Manchmal muss ein Anwendungsfall so lange aufgeschoben werden, bis die Prozesse und Infrastrukturen vorhanden sind, um die erforderlichen Daten zu sammeln.

Wenn Big Data und KI für ein Unternehmen neu sind, sollten alle Daten des Unternehmens betrachtet werden. Mit den richtigen Governance-Richtlinien sind alle Datensätze verfügbar, nutzbar und konsistent, wenn die Data Science Teams sie benötigen. Darüber hinaus müssen auch wichtige Vorgaben hinsichtlich des Datenschutzes und der Datensicherheit beachtet werden.

Wichtig zu wissen ist, dass nicht alle Datensätze gleich behandelt werden können; es gibt viele unterschiedliche Datentypen. So kann ein Anwendungsfall einen bestimmten Datentyp oder mehrere Typen erfordern. Die genutzten Datentypen bestimmen die Software-Tools, die auf den jeweiligen Stufen der Daten-Pipeline zum Einsatz kommen.

Tabelle 1 zeigt eine Vielzahl gängiger Datentypen mit Beispielen und Quellen.

Tabelle 1) Gängige Datentypen für Deep Learning

Datentyp	Beispiele	Quellen
Bilder	• Montagebandkameras • Medizinische Bildgebung • Raumbezogene Bilder • Geologische Darstellungen • Wärmebilder • Lidar-3D-Punktwolke	• NFS • Lustre • GPFS
Video	• Sicherheitskameras • Autonome Fahrzeuge • Drohnen • Kollaborative Roboter	
Audio	• Sprachnachrichten • Kundenservice-Anrufe • Akustische Daten	
Zeitreihendaten	• Internet of Things (IoT) • Wertpapierkurse • Wissenschaftliche Daten	• NoSQL-Datenbanken (Cassandra, AeroSpike)
Text	• Protokolldaten • Unstrukturierter Text • Dokumente	• Splunk, ELK usw. • NFS, Hadoop/HDFS • NoSQL-Datenbanken (MongoDB)
Graphen	• Social-Media-Daten • GPS-Navigation	• Graphen-Datenbanken

Jeder Datentyp wird ggfs. individuell gespeichert. Wie in Tabelle 1 zu sehen, müssen ggf. Daten aus einer Vielzahl von Quellen berücksichtigt werden, von Hadoop über NoSQL-Datenbanken bis hin zu Dateisystemen wie NFS oder Lustre. Manchmal werden bereits gesammelte Daten benötigt oder Streaming-Daten von Videokameras, Sensoren, Applikationen, E-Commerce-Transaktionen usw. (Streaming-Daten werden in Kapitel 6 behandelt: „Daten aufnehmen und vom Edge zum Core oder in die Cloud verschieben.“)

Ein zu trainierender Algorithmus kann ausschließlich Daten aus internen Quellen erfordern oder auch Daten aus externen Quellen wie Wetterdaten, demografische Daten oder Social-Media-Posts.

Für alle Quellen (intern und extern) müssen die Rechte an den Daten geklärt sein, sowie Compliance-Richtlinien und -Vorgaben eingehalten werden. Zudem muss sichergestellt werden, dass innerhalb des erforderlichen Zeitrahmens ein konsistenter Zugriff auf die Daten möglich ist.

Die drei großen Vs

KI lernt anhand der Merkmale und Attribute von Daten. Die drei großen Vs von Big Data – Volumen, Vielfalt und Verlässlichkeit – haben somit direkten Einfluss auf die Genauigkeit eines KI-Modells.

Volumen: Als Faustregel gilt: je größer das Volumen der Daten, desto höher die Leistung eines Deep-Learning-Systems.

Vielfalt: Vielfalt bedeutet eine Vielzahl verschiedener Merkmale und Attribute im Datensatz. Je größer die Vielfalt, desto genauer kann ein Deep-Learning-Modell generalisieren.

Verlässlichkeit: Der Lernprozess für maschinelles Sehen und andere Modelle erfordert gelabelte Daten. Das korrekte Labeling der Daten (Verlässlichkeit) ist entscheidend für genaue Modelle. Manuell gelabelte Trainingsdatensätze lassen sich nur mit hohem finanziellen und zeitlichen Aufwand erstellen. Deshalb werden neue Verfahren entwickelt, um Trainingsdatensätze programmgesteuert zu erzeugen.

5.1 Datenanforderungen von Anwendungsfällen in diversen Branchen

Im Gegensatz zu herkömmlicher Software sind für Deep-Learning-Systeme vor allem Daten und nicht der Code wichtig.

– Allegro.ai

Falls die vorhergehenden Diskussionen zu plastisch erscheinen, sollten folgende Anwendungsfällen ein besseres Verständnis vermitteln.

Einzelhandel: Bestandsoptimierung

Ein global agierendes Handelsunternehmen, wie zum Beispiel eine Supermarktkette, muss viele Faktoren berücksichtigen, um die Bestände an den jeweiligen Standorten zu optimieren. Der Einzelhandel ist extrem dynamisch: verderbliche Ware muss täglich frisch bestellt werden, Getränke und andere Waren hingegen nur ein- bis dreimal die Woche.

Zu den erforderlichen Daten gehören:

- **Lokalisation**: Welche Produkte sind am Standort des Geschäfts beliebt?
- **Produkttrends**: Wie sind die Verkaufsprognosen für die angebotenen Produkte?
- **Historische Trends**: Wie hoch war der Umsatz im letzten Monat? Letztes Jahr?

- **Werbeaktionen**: Welche Wirkung haben Coupons oder andere Werbeaktionen?
- **Demographie**: Wer lebt in der Nähe des Standortes?
- **Wetter**: Wie ist die Wettervorhersage für den Prognosezeitraum und wie wirkt sich das Wetter auf den Gesamtumsatz und den Umsatz für einzelne Produkte aus?

Gesundheitswesen: Intelligenter Inhalator

Allein in den Vereinigten Staaten leiden 25 Millionen Menschen an Asthma – jeder 13. ist betroffen. Ein paar Sensoren und eine Bluetooth-Verbindung in Asthma-Inhalatoren können eine Fülle nützlicher Daten bereitstellen. Patienten verwenden Inhalatoren, wenn Symptome auftreten. Dadurch ergibt sich die Gelegenheit, Nutzungs- und Standortdaten zu korrelieren. Die Kombination von Patientendaten mit Informationen wie dem Wetter, der Luftqualität und Pollenwerten kann Patienten helfen, mögliche Auslöser in Echtzeit zu vermeiden. Somit können sie Risiken minimieren und ihren allgemeinen Gesundheitszustand verbessern.

Maschinelles Sehen (Computer Vision)

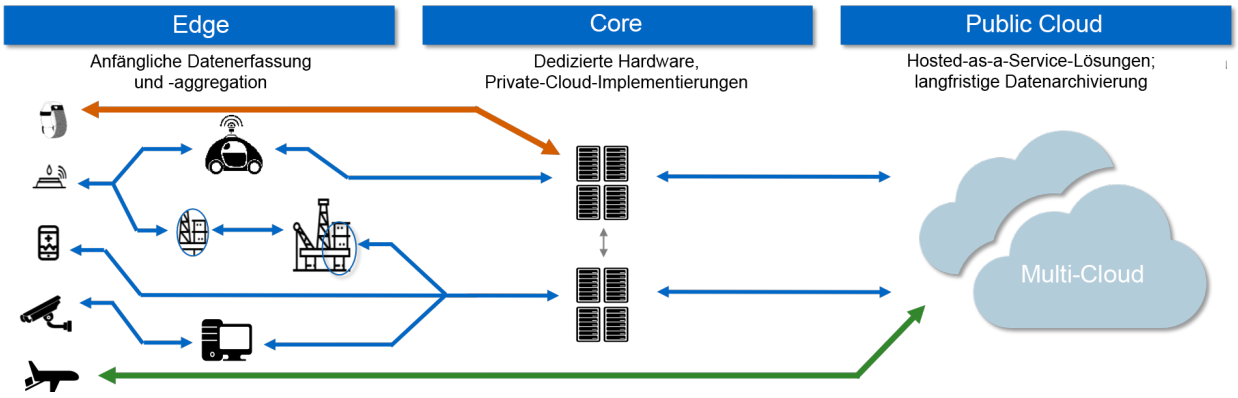
Bevor die irrtümliche Annahme entsteht, dass bei jedem Anwendungsfall für Deep Learning eine Vielzahl von Datensätzen und Datentypen korreliert werden müssen, werfen wir einen Blick auf das maschinelle Sehen. Hierfür gibt es Anwendungsfälle in fast jeder Branche und sie alle beruhen in der Regel auf Bildern oder Videomaterial. So nutzt die Fertigungsindustrie maschinelles Sehen, um Anomalien in Endprodukten zu erkennen. Im Gesundheitswesen dient maschinelles Sehen der Interpretation von CT-, MRT- oder anderen diagnostischen Bildern. Aber auch das Generieren von photoTANs fürs Online-Banking über ein Smartphone erfolgt über eine Anwendung des maschinellen Sehens, die schon längst so alltäglich geworden ist, dass man sie leicht übersieht.

Der NetApp Partner Allegro.ai ist auf die Verwendung von Deep Learning im Bereich des maschinellen Sehens spezialisiert und kennt die Datenanforderungen genau. Allegro.ai formuliert die zentrale Herausforderung für das maschinelle Sehen wie folgt: „Ein neuronales Netz für maschinelles Sehen ist in der Praxis nur dann erfolgreich, wenn es zuvor intensiv mit Datensätzen trainiert wurde, die der späteren Produktionsumgebung möglichst nahe kommen. Nicht allein die Menge der Daten bestimmt den Erfolg, sondern vielmehr der Grad, zu dem die Trainingsbilder und -videos dem entsprechen, was die Edge-Geräte aufnehmen."

6 Daten aufnehmen und vom Edge zum Core verschieben

Die Datenaufnahme ist ein wichtiger Faktor für jeden Deep-Learning-Anwendungsfall, insbesondere für solche, die auf Datenquellen am Edge-Bereich zurückgreifen. So steht die Datenaufnahme deutlich weniger im Mittelpunkt, wenn Daten aus einem Bildarchiv verarbeitet werden, als wenn IoT-Daten von Sensoren in Produktions- und Vertriebseinrichtungen gesammelt werden. Abbildung 4 veranschaulicht den Datenfluss zwischen Edge, Core und Cloud.

Abbildung 4) Beim Training fließen Daten häufig von Edge-Geräten zu Core-Rechenzentren oder in die Cloud



Bei der Datenaufnahme eignen sich bestimmte Datentypen für einen *Batch*-Ansatz, sodass sie zwischengespeichert und in regelmäßigen Intervallen weitergeleitet werden, während andere einen *Streaming*-Ansatz erfordern. In beiden Fällen ist das Ziel meist ein Data Lake, in dem die Daten nach Bedarf ausgewertet und umgewandelt werden können.

Der Typ der aufzunehmenden Daten bestimmt in der Regel die Art des Datenflusses. So erfordern Text- und Zahlendaten häufig die Datenverschiebung per Streaming, während Bild-, Video- und Audiodaten sich sowohl für Streaming als auch für die Datenverschiebung im Batch-Verfahren eignen.

Wenn ein Deep-Learning-Modell nach dem Training eingesetzt wird, kann die Art der Datenverschiebung vom Streaming auf den Batch-Ansatz umgestellt werden. So werden z. B. beim Training eines Modells für maschinelles Sehen alle erforderlichen Bilder gestreamt. Beim Einsatz des trainierten Modells im Edge-Bereich ist es jedoch möglich, die Daten lokal zu speichern und nur noch „interessante“ Bilder weiterzuleiten – z. B. jene, mit denen das Modell sich schwer tut.

6.1 Datenverschiebung per Stream

Bei der Datenverschiebung per Stream kommen häufig Open-Source-Tools von Apache zum Einsatz, wie Kafka, Flume oder NiFi:

- Kafka ist ein gängiger Message Broker, über den Datenverbraucher Streams von Datenproduzenten abonnieren können. Er zeichnet sich durch Zuverlässigkeit und Skalierbarkeit aus. Schwachpunkte sind der erforderliche Programmieraufwand und die Beschränkungen bei der Nachrichtengröße.
- Flume eignet sich hervorragend, um große Mengen von Streamingdaten nach Hadoop zu verschieben. Schwachpunkte sind potentielle Datenverluste in bestimmten Fehlerfällen und Beschränkungen bei der Nachrichtengröße.
- NiFi ermöglicht eine Echtzeitsteuerung für die Datenverschiebung zwischen beliebigen Quell- und Zielorten und eignet sich hervorragend für die Verschiebung von geschäftskritischen Daten mit Sicherheits- und Compliance-Anforderungen. Es gibt keine Beschränkungen hinsichtlich der Nachrichtengröße. Im Vergleich zu Kafka fehlt jedoch die Möglichkeit der Datenreplizierung.

Die Links in der Liste oben helfen bei der Entscheidung für das richtige Tool. Der Artikel Big Data Ingestion: Flume, Kafka, and NiFi bietet einen hilfreichen Vergleich der drei Tools.

Falls ein Anwendungsfall IoT-Daten von Industrieanlagen oder Sensoren erfordert, muss möglicherweise eines der in diesem Bereich üblichen Spezialprotokolle berücksichtigt werden. Das IoT reicht bis in die 80er und 90er Jahre zurück und hat eigene Protokolle und Standards hervorgebracht, um eine Vielzahl stromsparender Geräte mit wenig Speicher zu unterstützen. Zu den gängigen Protokollen gehören:

- **MQTT**: Message Queue Telemetry Transport
- **DDS**: Data Delivery Service
- **AMQP**: Advanced Message Queuing Protocol

Im Bereich der drahtlosen Datenübertragung ist Zigbee weit verbreitet, aber auch bekannte Protokolle wie Bluetooth, WiFi und Mobilfunknetze. Die neue 5G-Mobilfunktechnologie soll in Zukunft eine große Rolle für das IoT spielen.

6.2 Datenverschiebung im Batchverfahren

Wenn die Möglichkeit besteht, Daten im Edge-Bereich zu speichern und regelmäßig zu übertragen, kann hierfür eine Reihe von Linux-Befehlen genutzt werden, z. B. rsync, cp oder mv. Hierfür müssen jedoch Skripte geschrieben werden. Über REST-API-Aufrufe wie „PUT Object“ können Daten auch direkt in einen Objektspeicher wie Amazon S3 verschoben werden.

Manche Anbieter-Tools eignen sich hervorragend für solche Aufgaben. Sie umgehen die Notwendigkeit von Skripten und Programmen. Sie integrieren Verschlüsselung und Sicherheit sowie Daten-Effizienztechnologien wie Deduplizierung und Komprimierung, um die Netzwerkbandbreite und Storage-Kapazität effizient zu nutzen.

- **NetApp SnapMirror**: Ermöglicht die Datenaggregation und Datenverschiebung vom Edge- in den Core-Bereich mit Datenreduzierung und Verschlüsselung für mehr Effizienz und Sicherheit.
- **NetApp Cloud Sync**: Ermöglicht die nahtlose und sichere Datensynchronisierung vom Edge-Bereich in die Cloud.

7 Daten für das Training vorbereiten

Die Datenvorbereitungsstufe der KI-Daten-Pipeline umfasst eine Reihe von Funktionen, um einen Datensatz zu erstellen, mit der ein Deep-Learning-Modell trainiert und validiert werden kann. Die Datenvorbereitung ist jedoch nicht immer klar abgegrenzt. Sie kann während der Aufnahme und teilweise vor dem Training erfolgen, aber auch vollständig parallel zum Trainingsprozess.

Mögliche Schritte der Datenvorbereitung:

- **Daten auswerten**: Auf welcher Hypothese beruht das Modell und welche Merkmale der Daten sind hierfür entscheidend?
- **Datentypen und -formate bereinigen**: Konsistente Daten sorgen für einen reibungslosen Ablauf des Trainings. Sie sollten nur nicht deutlich konsistenter sein als die Livedaten, die das Modell später verarbeitet.
- **Trainingsdatensatz anpassen**: Das Merkmal, anhand dessen das Modell trainiert wird, sollte angemessen repräsentiert sein. So lässt sich zum Beispiel ein Modell zur Erkennung von Anomalien mit einem Datensatz voller Bilder von einwandfreien Teilen nicht effizient trainieren.
- **Datensätze labeln**:Überwachtes Lernen („supervised“) erfordert gelabelte Datensätze.
- **Datensätze in Trainings-, Validierungs- und Testdatensätze aufteilen**: Große Datenmengen sind erforderlich, um einen Trainingsdatensatz, einen separaten Validierungsdatensatz (anhand dessen das Modell nicht trainiert wurde) und einen Testdatensatz zur Beurteilung der Leistung des trainierten Modells zusammenzustellen.

Die Datenvorbereitung selbst ist gerade in den frühen Phasen der Modellentwicklung häufig ein iterativer Prozess des Ausprobierens, um herauszufinden, welcher Prozess die besten Ergebnisse liefert. Tabelle 2 zeigt gängige Datenvorbereitungsaktivitäten für verschiedene Datentypen.

Tabelle 2) Gängige Datenvorbereitungsschritte für verschiedene Datentypen

Datentyp	Gängige Datenvorbereitungsschritte
Bilder	• Größe und Auflösung aller Bilder vereinheitlichen • Sicherstellen, dass ausschließlich Schwarzweißbilder oder ausschließlich Farbbilder genutzt werden • Merkmale auf den Bildern labeln • Daten-Ungleichgewichte korrigieren (mittels Oversampling, Undersampling, Daten-Augmentation, Klassengewichtung)
Video	• Einzelbilder im JPG- oder BMP-Format extrahieren • Bildgröße nach Bedarf skalieren • Daten-Ungleichgewichte korrigieren (mittels Oversampling, Undersampling, Daten-Augmentation, Klassengewichtung)
Audio	• Abtastrate wählen • Signal aus dem Zeitbereich in den Frequenzbereich transformieren • Größe komprimieren
Zeitreihendaten	• Normalisieren (alle Werte zwischen 0 und 1) • Standardisieren (Werte neu skalieren, sodass der Mittelwert 0 und die Standardabweichung 1 beträgt)
Text	• Normalisieren (Groß-/Kleinschreibung und Zeichensetzung entfernen, Zahlen in Text umwandeln usw.) • Tokenisieren (Text in „Tokens“ segmentieren, die Wörter, Sätze oder Absätze repräsentieren) • Text bereinigen (Kopf- und Fußzeilen, HTML, Metadaten usw. entfernen)

7.1 Daten schneller labeln

Die Datenvorbereitung ist so zeitaufwendig, dass sich eine Reihe von Unternehmen darauf spezialisieren, diesen Prozess zu vereinfachen. Viele Deep-Learning-Anwendungsfälle setzen auf überwachtes Lernen und brauchen hierfür gelabelte Daten. Das Labeln der Daten stellt viele Deep-Learning-Projekte vor eine große Herausforderung.

Gelabelte Datensätze sind selten in der jeweils erforderlichen Spezifität verfügbar. Das Labeln von Daten ist somit meist ein arbeitsaufwendiger manueller Vorgang. Unternehmen für das Outsourcing von Geschäftsprozessen bieten Dienstleistungen für das manuelle Labeln von Daten an.

NetApp kooperiert mit den folgenden Unternehmen, die neue Wege beschreiten, um Daten vorzubereiten, automatisch zu labeln und den Prozess insgesamt zu beschleunigen:

- Figure Eight entwickelt eine Plattform, die Text-, Bild-, Audio- und Videodaten in individuelle Trainingsdaten wandelt, die mithilfe von Machine Learning gelabelt werden. Die Technologie von Figure Eight lernt aus gelabelten Beispielen der Benutzer und labelt dann den Rest der Daten.
- Hive bietet eine Data-Labeling-Lösung speziell für Anwendungsfälle des maschinellen Sehens.

8 Daten für das Training bereitstellen

Wenn der Datensatz vorbereitet ist, muss er der Trainingsplattform für das Modelltraining bereitgestellt werden. Große Deep-Learning-Modelle nutzen immer häufiger geclusterte Systeme mit vielen parallel arbeitenden GPUs als Trainingsplattform.

Die Daten können der Trainingsplattform auf drei Wegen bereitgestellt werden:

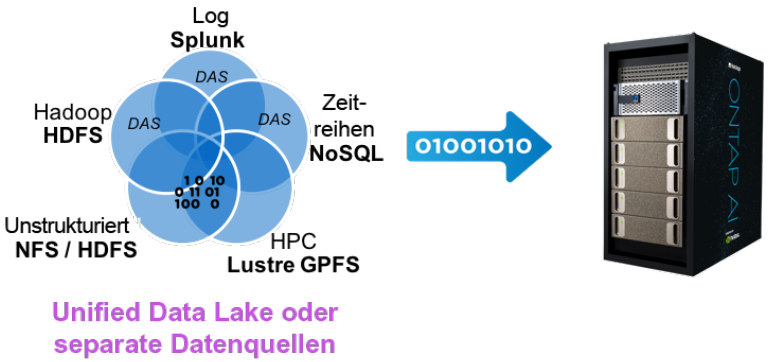
- Daten auf die Trainingsplattform kopieren
- Der Trainingsplattform Zugriff auf den Speicherort der Daten gewähren
- Tiering auf die Trainingsplattform

Die gewählte Methode hängt von mehreren Faktoren ab, einschließlich des Speicherorts der Daten, der Größe des Datensatzes und der Funktionen der Trainingsplattform.

Daten auf die Trainingsplattform kopieren

Abbildung 5 zeigt die am häufigsten genutzte Methode, um Daten auf der Trainingsplattform bereitzustellen: das Kopieren des Trainingsdatensatzes auf die Trainingsplattform. Diese Methode wird in 80 bis 90 % aller Fälle genutzt.

Abbildung 5) Daten aus einem Data Lake oder individuellen Datenquellen auf die Trainingsplattform kopieren



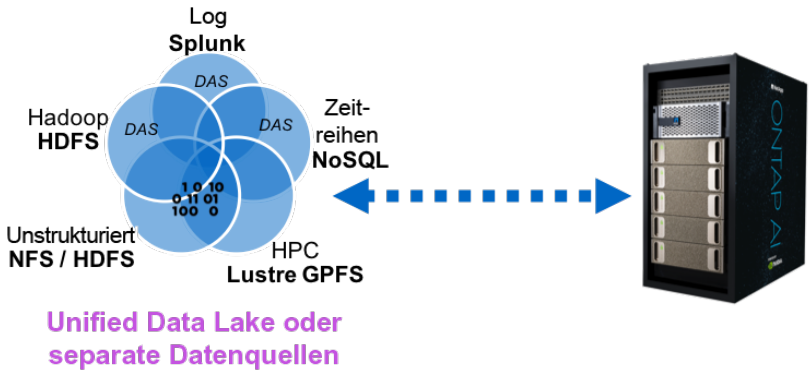
Der Datensatz wird in der Regel in einem Data Lake vorbereitet und dann auf die Trainingsplattform kopiert. Wenn kein zentrales Data Lake vorhanden ist, können die Daten nach Bedarf auch von unterschiedlichen Datenquellen auf die Trainingsplattform kopiert werden. Wenn die Daten im Vergleich zum Speicher und/oder der Storage-Kapazität der Trainingsplattform zu groß sind, können die Daten der einzelnen Datenquellen auch auf schnellem Storage in Trainingsplattform-Nähe zwischengespeichert werden. NetApp FlexCache ist eine hierfür geeignete Lösung.

Eine weitere Möglichkeit ist es, die Daten direkt bei der Aufnahme auf die Trainingsplattform zu kopieren. Streaming-Data-Broker wie Kafka vereinfachen diesen Ansatz. Ein Datenproduzent kann mehrere Datenverbraucher bedienen, sodass die aufgenommenen Daten zur selben Zeit an ein Data Lake und an die Trainingsplattform übertragen werden können.

8.1 Die Trainingsplattform greift auf den Speicherort der Daten zu

Eine selten genutzte Alternative für das Training ist der direkte Zugriff auf den Speicherort der Daten, wie in Abbildung 6. Auch hier erfolgt der Zugriff meist auf ein Data Lake oder direkt auf andere Datenquellen. Diese Option bietet sich an, wenn der Datensatz auf hochperformantem Storage gespeichert ist und die Größe des Datensatzes die Kapazität der Trainingsplattform übersteigt. Ein zusätzlicher schneller Cache ist eine Alternative für Datenquellen, die zu wenig Leistung bieten.

Abbildung 6) Trainingsplattform greift auf Speicherort der Daten zu



In der Theorie kann so eine einzige Kopie eines Datensatzes für Big-Data-Analysen, Machine Learning und Deep Learning genutzt werden. In der Praxis müssen die Daten für Deep Learning jedoch meist speziell vorbereitet und transformiert werden, sodass eine eigene Kopie erforderlich ist. Das Klonen der Daten ist eine Alternative zu einer vollständigen physischen Kopie, da nur etwaige Änderungen zusätzlichen Speicherplatz belegen. Mit NetApp FlexClone können beliebig viele Klone erstellt werden. Die Kapazität wird hierbei nur inkrementell und ohne Beeinträchtigung der Leistung verbraucht.

8.2 Tiering auf die Trainingsplattform

Tiering-Lösungen sind eine weitere Möglichkeit, um Datensätze für das Training bereitzustellen. Dies bietet sich an, wenn Daten aus „kaltem“ Storage abgerufen werden. In solch einem Fall müssten Sie viel Zeit und Arbeit investieren, um die Daten dorthin zu verschieben, wo sie benötigt werden. Tiering-Lösungen verschieben Daten aus dem „kalten“ Storage transparent, sobald ein Zugriff erfolgt. Die manuelle oder skriptbasierte Datenverschiebung entfällt. Wenn ein Modell reift und wiederholt neu trainiert wird, bietet Tiering einen guten Weg, um den Zugriff auf ältere Versionen der Trainingsdatensätze zu gewährleisten.

NetApp FabricPool ermöglicht zum Beispiel Tiering von Daten zwischen On-Premises- und Cloud-basiertem Objektspeicher. Je nach Trainingsplattform kann das Tiering direkt auf die Plattform selbst oder auf Speicher in Plattform-Nähe erfolgen.

9 Ein Deep-Learning-Modell trainieren

Sobald der Datensatz vorbereitet ist, ist es an der Zeit, das Deep-Learning-Modell zu trainieren. Das Training ist fast immer eine Frage des Ausprobierens, um gute Ergebnisse zu erzielen (und den Rechenaufwand zu minimieren).

Erfolgreiches Training von Deep-Learning-Modellen beruht im Wesentlichen auf drei Aspekten:

- Die bestmögliche Modellarchitektur ermitteln
- Die Berechnung skalieren
- Sehr große Trainingsdatensätze verwenden

Das Training kann eine Vielzahl von Experimenten umfassen. Data Scientists möchten womöglich die neuesten und innovativsten Modellarchitekturen ausprobieren. Infrastructure Engineers möchten vielleicht neue Parallelisierungsmethoden testen – oder bessere GPUs einbinden, um die Berechnung zu skalieren. Data Engineers bereiten neue und größere Datensätze vor.

Dieses Kapitel befasst sich mit einigen Grundkonzepten der verschiedenen Typen von neuronalen Netzen sowie der Deep-Learning-Frameworks, die möglicherweise zum Einsatz kommen. Es stellt auch einige Softwareplattformen vor, die große Teile des Deep-Learning-Prozesses integrieren und vereinfachen.

Rechen- und I/O-Anforderungen von Deep Learning

Die Datensätze für das Training von Deep-Learning-Modellen werden zunehmend größer. Mit der Größe der Datensätze steigen auch die Rechenanforderungen für das Training. Zum Glück lassen sich die Rechenanforderungen eines Deep-Learning-Modells empirisch vorhersehen. Ein Modell kann auf kleine Datensätze angewendet werden, um die passende Modellarchitektur zu bestimmen, und dann mit größeren Datensätzen verbessert werden, um die gewünschte Genauigkeit zu erzielen.

Je größer und umfangreicher die Modelle sind, desto eher werden die Rechenressourcen, insbesondere der verfügbare GPU-Speicher, zum Engpass für das Training. Dieses Problem lässt sich durch ein Upgrade auf die neuesten GPU-Architekturen (mit mehr Speicher) oder die Migration zu GPU-„Supercomputern“ lösen, die eine Vielzahl miteinander verbundener GPUs zu einer großen GPU zusammenfassen.

Parallelisierungstechniken wie Daten-, Modell- oder Pipeline-Parallelisierung können die Trainingsdauer verkürzen. Hierfür muss jedoch die Modellarchitektur überarbeitet werden, um diese Ansätze zu unterstützen.

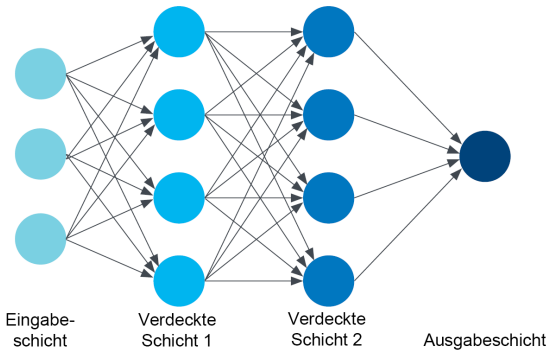
Unzureichende Rechenleistung kann dazu führen, dass Data Scientists von Modellen mit zu hohen Rechenanforderungen absehen. Dies führt zu einem Kompromiss zwischen Rechenleistung und statistischer Genauigkeit. Anwendungsfälle wie die medizinische Bildgebung erfordern eine sehr hohe Genauigkeit, aber in anderen Fällen – z. B. bei Empfehlungsdiensten („andere Kunden kauften auch“) – können schnellere Antwortzeiten der höheren Genauigkeit vorgezogen werden.

Die Ansätze zur Lösung der Probleme mit der Größe von Datensätzen und der Trainingsdauer werden kontinuierlich weiterentwickelt.

9.1 Typen von neuronalen Netzen

Für den Einstieg in die Welt von Deep Learning ist es hilfreich, die unterschiedlichen Klassen von künstlichen neuronalen Netzen zu kennen und wofür sie sich eignen. Es ist nicht zwingend erforderlich, die Architektur aller Typen von neuronalen Netzen zu verstehen, um sie zu benutzen. Es ist jedoch hilfreich, sich mit ein paar Grundkonzepten vertraut zu machen. Ein neuronales Netz besteht aus einer Eingabe- und einer Ausgabeschicht mit dazwischenliegenden verdeckten Schichten.

Abbildung 8) Vereinfachte Darstellung eines mehrschichtigen neuronalen Netzes



Die Knoten der verdeckten Schichten eines neuronalen Netzes führen meist einfache mathematische Berechnungen durch, sogenannte *Aktivierungsfunktionen*. Die Architektur und der Typ eines neuronalen Netzes werden bestimmt von den Verbindungen zwischen den Knoten, den mathematischen Funktionen sowie der Breite und Tiefe des Netzwerks.

Tabelle 3 zeigt einige der gängigsten Netzwerktypen. Diese Auflistung erhebt keinen Anspruch auf Vollständigkeit und es werden ständig neue Ansätze entwickelt. Zudem gibt es auch Hybridformen, die Elemente verschiedener Netzwerktypen kombinieren.

Tabelle 3) Gängige neuronale Netze und ihre Anwendungsfälle

Typ des neuronalen Netzes	Gängige Anwendungsfälle
Convolutional Neural Network (CNN) Klassifizieren, gruppieren und identifizieren	• Foto- und Videoerkennung • Empfehlungen • NLP
Rekurrentes neuronales Netz (RNN) Long Short-Term Memory (LSTM) Muster in Datensequenzen erkennen; kann Ursache und Wirkung in Verbindung bringen	• NLP • Video • Text • Sprache • Zeitreihen
Generative Adversarial Network (GAN) Eignet sich für kreative Text- und Bildverarbeitung	• Bilderzeugung anhand von Beschreibungen • Bild-zu-Bild-Wandlung • Video-Lernen und -Synthese
Deep reinforcement learning (DRL) Kann lernen, wie man ein Ziel erreicht (z. B. ein Spiel spielen)	• Netzwerk-Scheduler in Mobilfunknetzen • Recovery-Controller für mobile Roboter • Management großer Flotten • Medizinische Bildgebung

Schematische Darstellungen und detailliertere Beschreibungen dieser und weiterer Netzwerktypen finden sich in „The mostly complete chart of Neural Networks, explained“

Aus praktischer Sicht erfordern das Design und Training eines neuronalen Netzwerks viele Überlegungen. Dazu gehören die Auswahl der geeigneten Konnektivität (Netzwerktyp), Optimierer, Fehlerfunktionen und Aktivierungsfunktionen sowie die Optimierung der Hyperparameter.

9.2 Gängige Deep-Learning-Frameworks

Für die meisten Teams, die ein Deep-Learning-Projekt beginnen, sind die Spezifikationen des neuronalen Netzes wichtiger als das verwendete Framework. Die Entscheidung für ein Framework beruht in der Regel auf Erfahrungswerten – sofern zumindest ein Teammitglied bereits über Erfahrungen mit einem der Frameworks verfügt –, dem jeweiligen Anwendungsfall und der unterstützen Programmiersprache.

Einige gängige Deep-Learning-Frameworks sind:

- TensorFlow
- Keras
- PyTorch
- Caffe und Caffe2
- Microsoft Cognitive Toolkit (CNTK)
- MXNet
- DeepLearning4j
- Chainer
- Neural Network Libraries
- PaddlePaddle

9.3 Deep-Learning-Softwareplattformen

Viele Softwareunternehmen bieten Plattformen, die den Prozess der Entwicklung von KI-Modellen vereinfachen, um Daten schneller gewinnbringend zu nutzen. NetApp arbeitet mit einigen dieser Unternehmen zusammen. Dazu zählen:

Deep Learning

- **Allegro.ai.** Allegro.ai entwickelt eine Deep-Learning-Plattform, die für maschinelles Sehen optimiert ist. Der Schwerpunkt liegt dabei auf Datenvorbereitung (inkl. Data-Labeling), Training und Implementierung. Allegro.ai unterstützt diverse KI-Frameworks, einschließlich TensorFlow, PyTorch, Keras und Caffe.
- **Element AI.** Element AI will anderen Unternehmen zum Erfolg mit KI verhelfen. Zu den Zielbranchen gehören Finanzdienstleistungen, der Einzelhandel, Cybersicherheit, das Transportwesen und die Logistikbranche.

Maschinelles Lernen

- **H2O.ai.** Die Verbreitung von Tools für maschinelles Lernen und Deep Learning macht es selbst Profis schwer, sich auf diesem Gebiet zurechtzufinden. H2O.ai will den Zugang zu KI demokratisieren. Die H2O-Plattform ist eine Open-Source-Plattform für maschinelles Lernen. Sie vereinfacht die Nutzung gängiger Algorithmen für Data Science und Machine Learning.

9.4 Modelle validieren und bewerten

Für Data Scientists ist der entscheidende Teil des Trainingsprozesses die Frage, ob das Modell genug aus den Daten gelernt hat.

- Die **Modellvalidierung** beantwortet die Frage, ob das Modell vernünftig skaliert. Stürzt es ab? Verfügt es über ausreichende Rechen-, Netzwerk-, Speicher- und Sicherheitsressourcen? Wenn die Modellvalidierung scheitert, muss ein Operations Engineer die Rechenanforderungen des Modells überarbeiten.

- Die **Modellbewertung** bestimmt die statistische Leistung eines Modells. Wie genau arbeitet das Modell mit Livedaten? Wie präzise ist das Modell für eine Untermenge der Daten? Wenn die Modellbewertung scheitert, muss der Data Scientist den Entwurf überarbeiten und sich mit der Datenherkunft und Modellarchitektur befassen.

Sie können einen einzigen dedizierten Cluster für die Modellvalidierung/-bewertung und das Serving bzw. die Bereitstellung des Modells oder zwei separate Cluster nutzen.

10 Modell-Serving und -Bereitstellung

Auf dieser Stufe der Pipeline geht das Deep-Learning-Modell von der Entwicklung in die Produktion über. Ab diesem Zeitpunkt geht es mehr um Lifecycle Management und weniger um Data Science.

Wenn das erste Training abgeschlossen und das Modell zur Zufriedenheit des Teams validiert und bewertet wurde, muss es operationalisiert werden. Dieser Prozess umfasst häufig:

- Optimierung des Modells für Inferencing-Leistung (z. B. über NVIDIA TensorRT).
- Serving. Bestimmte Anwendungsfällen erfordern die Bereitstellung des Modells auf Inferencing-Systemen im Edge-Bereich. So befindet sich bei autonomen Fahrzeugen Inferecing-Hardware an Bord.
- Regelmäßiges Neutrainieren mit aktuellen Daten, um sicherzustellen, dass das Modell aktuell bleibt.
- Integration von Modellausgabe und benutzerdefinierte Software.
- Sorgfältiges Management der Modellversionen sowie von Trainings-, Validierungs- und Testdatensätzen.

Das Ausmaß der Aktivitäten und Anforderungen ändert sich oft in dieser Phase. Zu den neuen Anforderungen gehören:

- **Mandantenfähige Umgebung**: Eventuell befinden sich mehrere Deep-Learning-Modelle in der Software-Pipeline. Eine Zugriffssteuerung ist erforderlich, um zu verhindern, dass jeder auf alles Zugriff erhält.
- **Versionierung von Datensätzen**: Eine sorgfältige Versionskontrolle aller Trainings-, Validierungs- und Testdatensätze ist unumgänglich.
- **Modellmanagement:**
  - Jedes Mal, wenn das Modell neu trainiert wird, wird eine neue Version erstellt.
  - Eventuell sind unterschiedliche Versionen desselben Modells erforderlich, die für unterschiedliche Inferencing-Hardware optimiert sind.
- **Automatisierung**: Diese Prozesse sollten weitestgehend automatisiert ablaufen, um Engpässe und Fehler zu vermeiden.

10.1 Plattformoptionen

Wie bei den anderen Teilen der Pipeline stellen diverse KI-Anbieter auch hier Plattformen zur Verfügung, mit denen sich diese Aspekte der Deep-Learning-Pipeline vereinfachen lassen.

- Allegro.ai. NetApp arbeitet gemeinsam mit Allegro.ai an einer einheitlichen Plattform, die sich um alle Aspekte des Datensatzlebenszyklus kümmert. Dadurch entfallen sämtliche Überlegungen bezüglich der Tools und Infrastruktur.
- Datatron bietet eine Plattform mit Fokus auf Versionierung und Bereitstellung, Monitoring und Management.
- Algorithmia automatisiert DevOps für maschinelles Lernen. Modelle lassen sich über einen serverlosen Ansatz als skalierbare Microservices bereitstellen.
- Skymind. Der Skymind Intelligence Layer (SKIL) hilft Enterprise-IT-Teams, Modelle für maschinelles Lernen in großen Umgebungen zu managen, bereitzustellen und neu zu trainieren. SKIL kann Modelle aus vielen verschiedenen Frameworks importieren.
- Das NeoPulse-Framework von Dimensional Mechanics unterstützt die Entwicklung, Bereitstellung und Distribution benutzerdefinierter KI-Modelle on-Premises oder in der Cloud.

Versionsverlauf

Version	Datum	Dokumentversionshistorie
Version 0.1	März 2019	Alphaversion.

Überprüfen Sie mithilfe des Interoperability Matrix-Tools (IMT) auf der NetApp Support-Website, ob die in diesem Dokument angegebenen Produktversionen und Funktionen in Ihrer IT-Umgebung unterstützt werden. NetApp IMT definiert die Produktkomponenten und -versionen, die für von NetApp unterstützte Konfigurationen verwendet werden können. Die jeweiligen Ergebnisse sind von der kundenspezifischen Installation bzw. den technischen Daten abhängig.