Einstieg in die KI-Inferenz

Red Hat

Beschleunigter Weg zur Effizienz

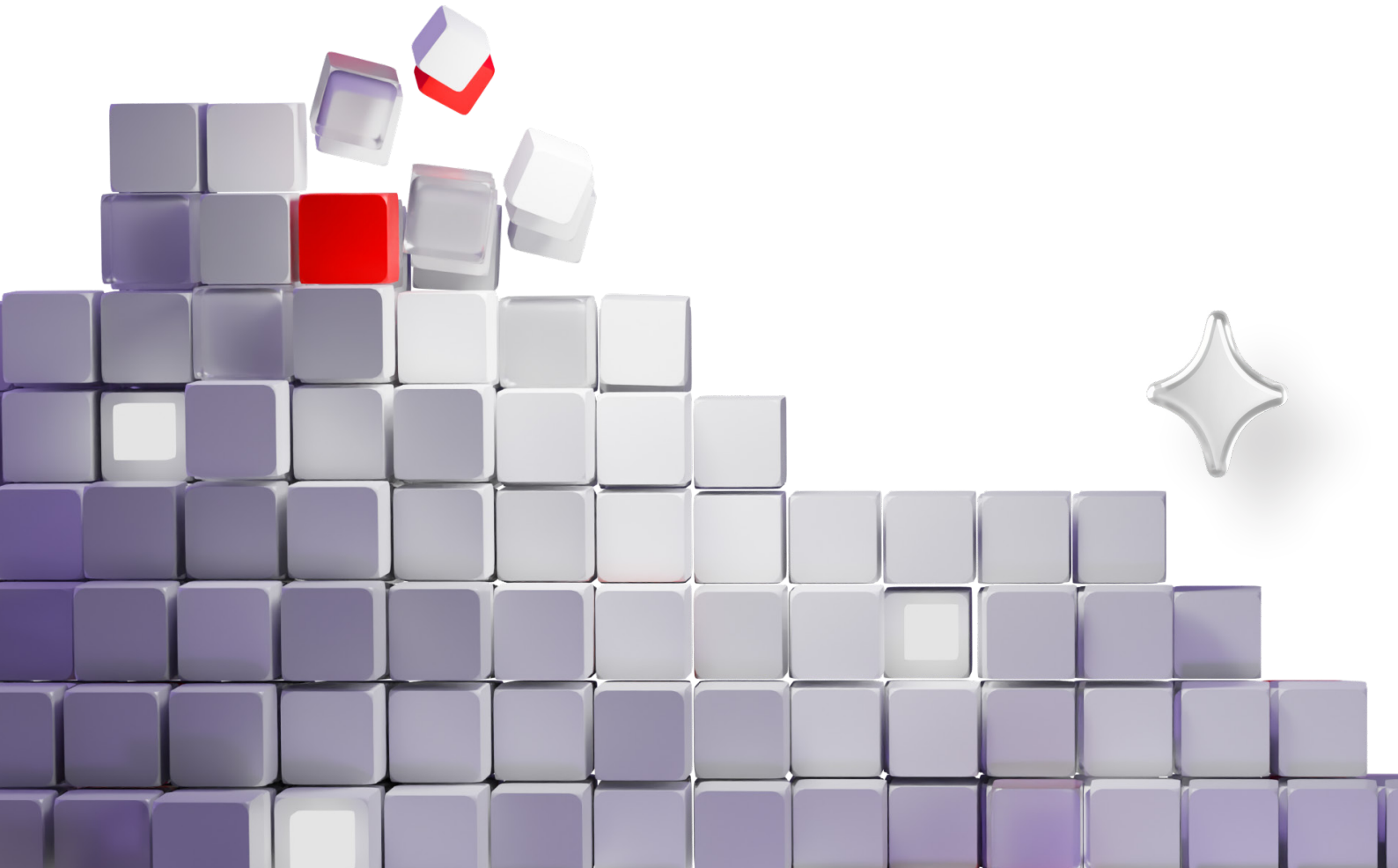
Inhaltsverzeichnis

Einleitung

Die Optimierung der KI-Modell-Inferenz ist eine der effektivsten Möglichkeiten, Infrastrukturkosten zu senken, Latenzzeiten zu verringern und den Durchsatz zu verbessern, insbesondere wenn Unternehmen große Modelle in der Produktion einsetzen.

Dieses E-Book bietet eine Einführung in die Grundlagen des Inferenz-Performance-Engineerings und der Modelloptimierung mit dem Schwerpunkt auf Quantisierung, Sparsity und anderen Techniken zur Reduzierung von Rechen- und Speicheranforderungen sowie auf Laufzeitsystemen wie Virtual Large Language Model (vLLM), die diese Vorteile für eine effiziente Inferenz bieten.

Außerdem werden die Inhalte des offenen Ansatzes von Red Hat, des validierten Modell-Repositorys und von Tools wie LLM Compressor und Red Hat® AI Inference Server erläutert. Unabhängig davon, ob Sie mit GPUs (Graphics Processing Units), Tensor Processing Units (TPUs) oder anderen Beschleunigern arbeiten, dieser Guide bietet praktische Einblicke in die Entwicklung intelligenterer und effizienterer KI-Inferenzsysteme.

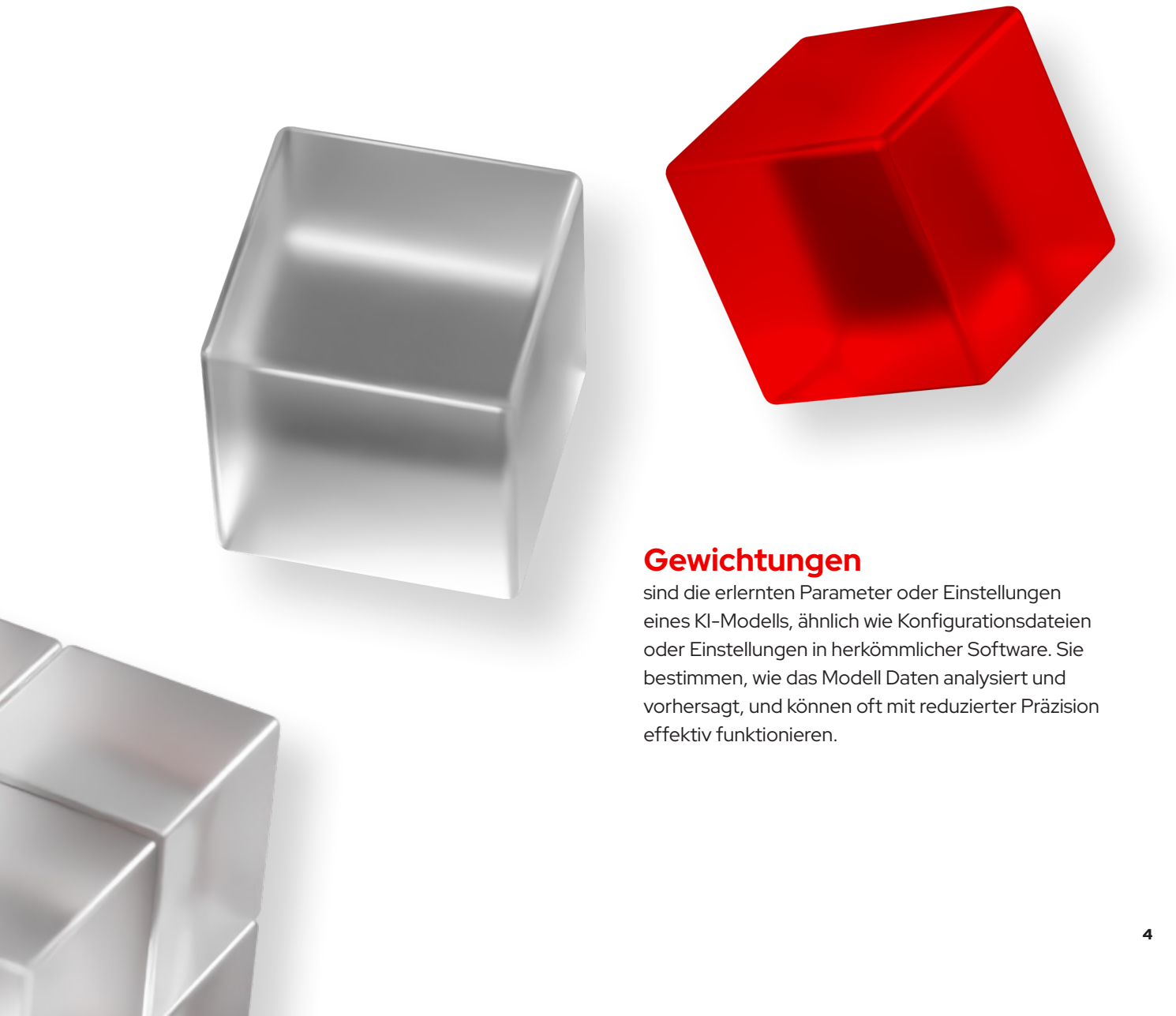


Die wichtigsten Begriffe auf einen Blick

Verstehen von Modellkomponenten

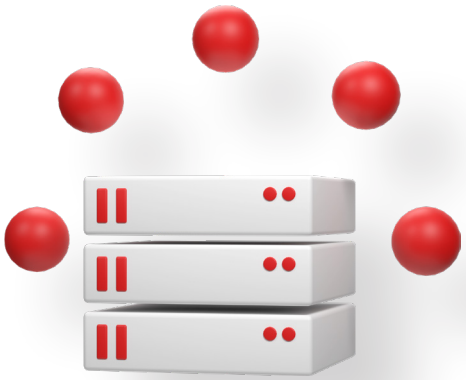
Aktivierungen

sind temporäre Daten, die während der Verarbeitung von Informationen (Eingabe-Token) durch ein Modell generiert werden, ähnlich wie bei einer Berechnung anfallende Zwischenergebnisse. Sie benötigen in der Regel eine hohe Präzision, um genaue Ergebnisse zu erhalten.



Gewichtungen

sind die erlernten Parameter oder Einstellungen eines KI-Modells, ähnlich wie Konfigurationsdateien oder Einstellungen in herkömmlicher Software. Sie bestimmen, wie das Modell Daten analysiert und vorhersagt, und können oft mit reduzierter Präzision effektiv funktionieren.



Quantisierung

Quantisierung reduziert die Größe und den Ressourcenbedarf von KI-Modellen, indem ihre Parameter (Gewichte) und Zwischendaten (Aktivierungen) in Formaten mit niedrigerer Präzision gespeichert werden, die weniger Bits pro Wert verwenden. Diese Technik ermöglicht eine effiziente Verwaltung von Ressourcen, ähnlich wie beim Komprimieren von Dateien auf einem Computer. Bei korrekter Ausführung **beeinträchtigt dies die Leistung des Modells nicht wesentlich**.

- **Die Gewichtungsquantisierung** reduziert die Storage-Größe der Modellparameter und **ermöglicht so eine effizientere Speichernutzung während der Inferenz**.[1]
- **Die Aktivierungsquantisierung** minimiert den Speicherbedarf von Zwischenausgaben (temporäre Daten) während der Inferenz und **macht die Ausführung schneller und effizienter**.[2]
- **Die KV-Cache-Quantisierung** verkleinert den Speicher-Footprint von zwischengespeicherten Schlüssel-Wert-Tensors und unterstützt Modelle **dabei, lange Prompts und gleichzeitige Anfragen effizienter zu verarbeiten**.[3]

Präzisionsebenen bei 16-Bit-, 8-Bit- und 4-Bit-Quantisierung:

- **16-Bit (FP16/BF16)** ist Standardpräzision und behält Genauigkeit bei, erfordert aber erheblichen Arbeitsspeicher, was die Anwendung für sehr große Modelle kostspielig macht.
- **8-Bit (FP8/INT8)** reduziert die Arbeitsspeichernutzung im Vergleich zu 16-Bit um etwa die Hälfte, was erhebliche Effizienzsteigerungen bei gleichzeitiger Beibehaltung der Modellgenauigkeit ermöglicht.
- **4-Bit (INT4)** reduziert die Modellgröße und die Arbeitsspeicheranforderungen erheblich und ermöglicht so ein Deployment auf weniger Ressourcen. Dies kann jedoch zu einer merklichen Verschlechterung der Genauigkeit führen, wenn dies nicht sorgfältig mit erweiterten Quantisierungsmethoden gemanagt wird.

1 Laboone, Maxime: „**Introduction to Weight Quantization**." *towards data science*, 7 Juli 2023.

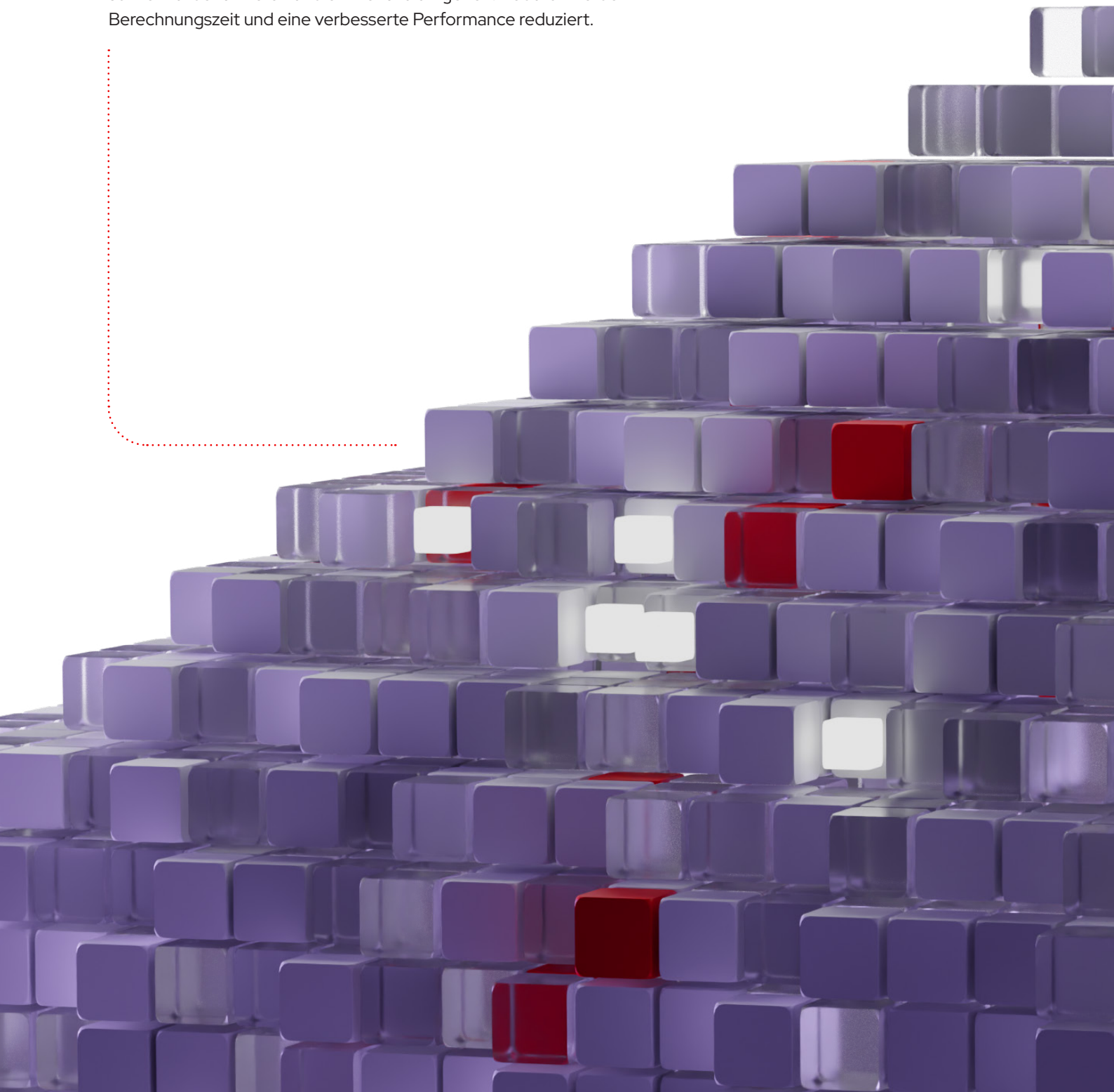
2 „**AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration**." GitHub, Zugriff am 8. Aug. 2025.

3 Turkganbay, Rohhan. „**Unlocking Longer Generation with Key-Value Cache Quantization**". *Hugging Face*, 16. Mai 2024.

Reduzierte Rechenlast mit Sparsity

Sparsity reduziert den Rechenaufwand, indem einige der Modellparameter absichtlich auf 0 gesetzt werden. So können Systeme unnötige Abläufe wie das Überspringen leerer Abschnitte in einem Formular umgehen. So werden Geschwindigkeit und Effizienz verbessert, ohne dass das Modell vollständig neu trainiert werden muss.

2:4 Sparsity ist **ein strukturierter Ansatz**, bei dem genau 2 von 4 Parametern auf 0 gesetzt werden. Dies ermöglicht es spezialisierter Hardware, Blöcke dieser inaktiven Parameter schnell zu identifizieren und effizient zu umgehen. Dadurch werden Berechnungszeit und eine verbesserte Performance reduziert.

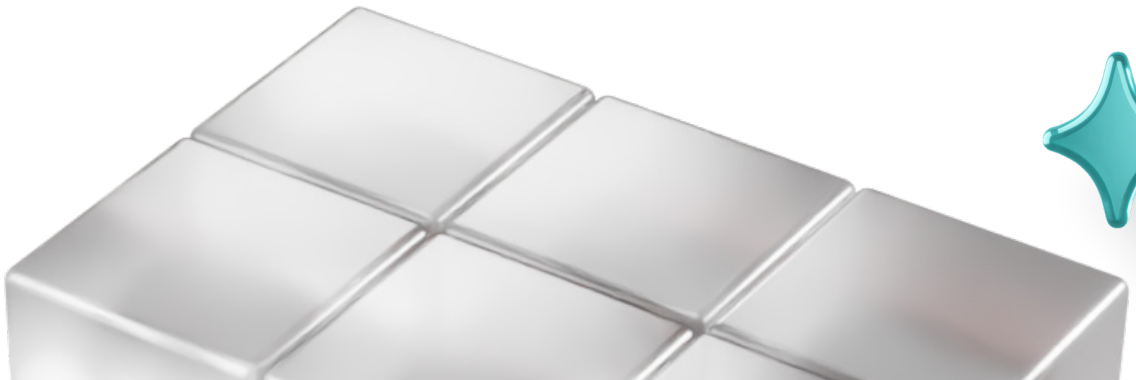


Die Evolution **großer Sprachmodelle**

Large Language Models (LLMs), die hauptsächlich auf Transformer-Architekturen aufbauen, haben sich von Forschungsexperimenten zu grundlegenden Tools für reale Anwendungen entwickelt. Ihr Umfang, der oft Dutzende oder Hunderte von Milliarden von Parametern umfasst, ermöglicht ein hohes Maß an Reasoning, Kreativität und Domain-Spezifikation. Dies geschieht durch einen Prozess namens Inferenz.

Inferenz ist das Verfahren, bei dem ein trainiertes Modell neue Eingabedaten verarbeitet und eine Ausgabe generiert, beispielsweise die Vorhersage des nächsten Wortes in einem Satz oder die Identifizierung eines Objekts in einem Bild. Im Gegensatz zum Training, bei dem aus großen Datensätzen gelernt wird, konzentriert sich die Inferenz darauf, das erlernte Wissen anzuwenden, um Echtzeitentscheidungen zu treffen. Daher muss die Inferenz schnell und effizient sein, insbesondere dann, wenn Modelle in Produktivumgebungen eingesetzt werden, um interaktive Anwendungen, Echtzeitanalysen oder umfangreiche Automatisierungen zu unterstützen.

Inferenzmodelle verarbeiten Eingabedaten wie Text, Bilder oder Audio als Token und durchlaufen sie mehrschichtige Transformer-Architekturen, um Vorhersagen zu generieren. Token sind die diskreten Einheiten, in die Eingabedaten unterteilt werden, bevor sie von einem Modell verarbeitet werden. In textbasierten Modellen können Token je nach verwendeter Tokenisierungsstrategie einzelne Zeichen, Teilwörter oder ganze Wörter darstellen.

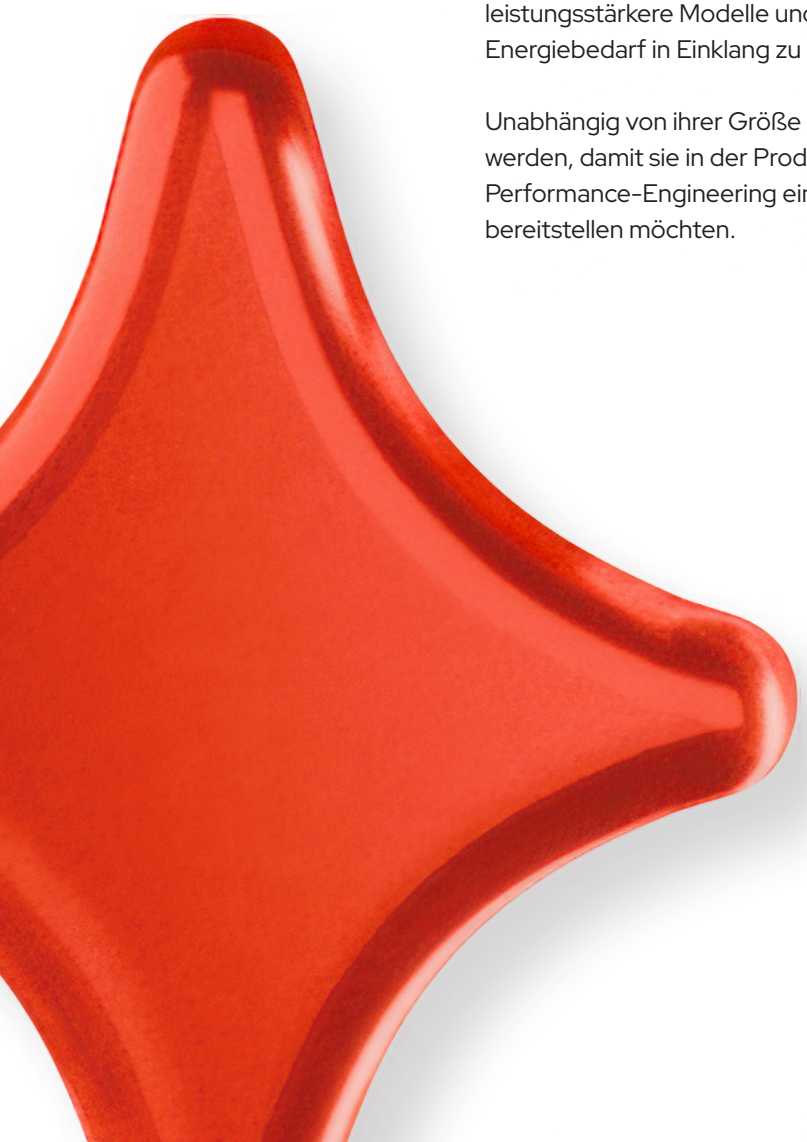


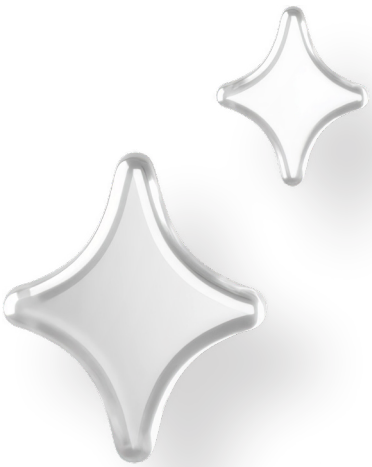
Diese Modelle übergeben Eingabe-Token durch tiefgreifende, mehrschichtige Transformer-Architekturen, die eine Abfolge mathematischer Operationen anwenden, um den Kontext zu analysieren, Beziehungen abzuwägen und wahrscheinliche Ausgaben zu bestimmen. Jede Ebene verfeinert das Verständnis des Modells der Eingabe und erzeugt letztendlich eine Vorhersage, jeweils mit einem Token. Diese schrittweise Token-Generierung ermöglicht hochgenaue und kontextgerechte Ausgaben, trägt aber auch zur Rechenintensivität von Inferenz-Workloads bei, insbesondere bei großen Modellen mit vielen Schichten.

Neben textbasierten LLMs stehen ähnliche Architekturen mittlerweile für eine Reihe von KI-Domains zur Verfügung, darunter Bildverarbeitungsmodelle und multimodale Systeme. Vision-Modelle wenden dieselben Prinzipien des tokenbasierten Transformer-Computings auf Bilder und Videos an. Anstatt Text in Token aufzuteilen, werden Pixeldaten in Einbettungen konvertiert. Diese Einbettungen erfassen räumliche Muster, Ränder, Konsistenzen und Beziehungen zwischen visuellen Elementen, sodass das Modell Aufgaben wie Bildklassifizierung, Objekterkennung, Segmentierung und visuelle Beantwortung von Fragen durchführen kann. Werden sie in der Produktion eingesetzt, können sie Use Cases wie automatisierte Inspektion, medizinische Bildgebung und Inhaltsmoderation unterstützen.

Mit der zunehmenden Einführung von KI in Unternehmen nehmen die Modellarchitekturen weiter an Größe und Komplexität zu. Neue Ansätze wie Mixture of Experts (MoE) zielen darauf ab, die Performance zu skalieren, indem nur Teile des Modells pro Inferenz aktiviert werden, wodurch die erforderliche Rechenleistung insgesamt reduziert wird. Diese Innovationen machen den Weg frei für noch leistungsstärkere Modelle und tragen dazu bei, Performance und Kosten- und Energiebedarf in Einklang zu bringen.

Unabhängig von ihrer Größe müssen Modelle effizient bereitgestellt und optimiert werden, damit sie in der Produktion verwendet werden können. Daher ist Inferenz-Performance-Engineering eine wichtige Priorität für Unternehmen, die Modelle bereitstellen möchten.



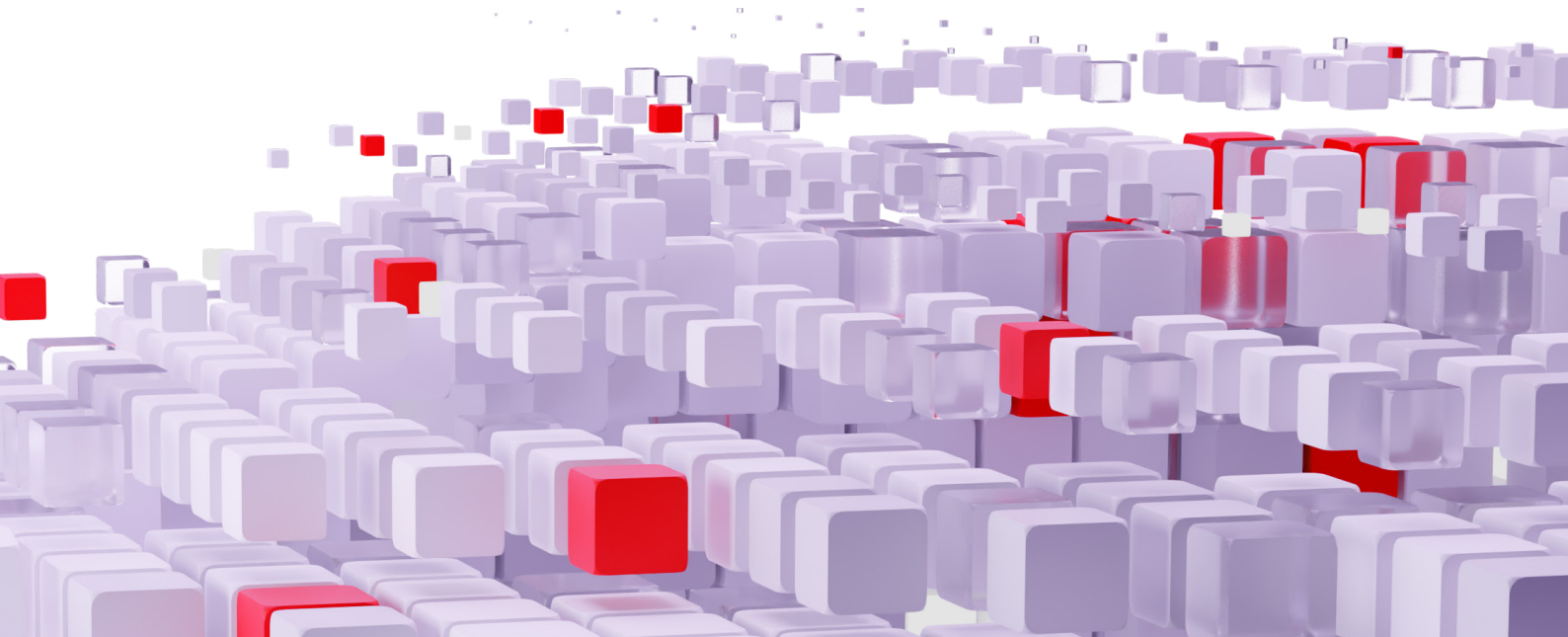


Herausforderungen beim **Bereitstellen von Inferenzen**

Die Bereitstellung von Inferenz für große Modelle ist mit mehreren Herausforderungen verbunden.

Modelle mit Milliarden von Parametern benötigen beträchtlichen GPU-Arbeitsspeicher, um ihre Gewichtungen und Zwischenzustände zu speichern, wie etwa Schlüssel-Wert-Caches (Key Value, KV). Wenn die Anzahl der gleichzeitigen Anforderungen oder die Länge der Eingaben zunimmt, werden Speicherbeschränkungen zu kritischen Engpässen, die den Durchsatz und die Reaktionsfähigkeit des Modells einschränken. Grundlegende Bereitstellungsmethoden sind oft mit ineffizienten Batch-Techniken verbunden, was zu nicht ausreichend ausgelasteten Hardwareressourcen und erhöhten Latenzzeiten führt.

Darüber hinaus können Implementierungen von Aufmerksamkeitsmechanismen in Transformer-Architekturen rechenintensiv sein, insbesondere bei langen Eingaben, wodurch sich die Antwortzeiten erheblich verlangsamen. Der Umgang mit diesen Herausforderungen erfordert ausgefeilte Runtime-Optimierungen wie effiziente Speicherverwaltung, fortschrittliche Batch-Strategien und optimierte Aufmerksamkeitsmechanismen wie Paged Attention. Zusammen tragen diese dazu bei, die Performance und Reaktionsfähigkeit in realen Anwendungen zu verbessern.



Ein Full-Stack-Ansatz für **die Inferenz-Performance**

Inferenzoptimierung bezieht sich auf den Prozess der Verbesserung der Effizienz eines KI-Modells nach dem Deployment in der Produktion. Der Einsatz von LLMs in der Produktion kann schnell teuer werden, insbesondere bei hohen Token-Volumes, langen Prompts und wachsenden Nutzungsanforderungen. **Für die Kostenoptimierung bei der Inferenz werden der Speicherbedarf reduziert, der Durchsatz erhöht und die Hardwareanforderungen minimiert, ohne dass dabei Genauigkeit oder Benutzererlebnis beeinträchtigt werden.**

Während das Modelltraining im Allgemeinen eine Aufgabe mit einer einzigen Instanz ist (mit Ausnahme von Modell-Neutrainings), erfolgt die Inferenz kontinuierlich, und es werden Echtzeitausgaben als Reaktion auf Eingaben der Nutzenden erzeugt. Für LLMs und Vision-Modelle kann die Inferenz schnell zum teuersten und ressourcenintensivsten Teil eines KI-Deployments werden, insbesondere wenn sie in hybriden oder globalen Infrastrukturen skaliert werden.

Für die effektive Bereitstellung von LLMs in großem Umfang ist eine umfassende Optimierungsstrategie für den gesamten Stack erforderlich, die sowohl das Modell selbst als auch die Bereitstellungslaufzeit berücksichtigt. Obwohl wir uns in erster Linie auf die Optimierung der Modellparameter durch **Quantisierung** und **Sparsity** konzentrieren, können zusätzliche Performance-Verbesserungen durch die Verfeinerung des Inferenzbereitstellungsprozesses durch Techniken wie **Chunked Prefill**,[4] **Prefix Caching**,[5] **spekulatives Dekodieren**,[6] und **disaggregiertes Vorfüllen und Decodieren** erzielt werden.[7]

4 „**Optimization and Tuning**." *vLLM*, 7. Aug. 2025.

5 „**What is Automatic Prefix Caching?**" *vLLM*, Zugriff am 8. Aug. 2025.

6 „**How Speculative Decoding Boosts vLLM Performance by up to 2.8x**." *vLLM*, 17. Okt. 2024.

7 Du, Kuntai: „**vLLM Office Hours – Disaggregated Prefill and KV Cache Storage in vLLM**" – 14. November 2024. *YouTube*, 18. Nov. 2024.

Überblick über Inferenzlaufzeiten und Modellformate

Da grundlegende Laufzeiten einen Engpass darstellen, erfordert die effiziente Bereitstellung großer Modelle die Wahl der richtigen Inferenzlaufzeit. Zu den beliebten Runtimes gehören:

- **vLLM**: Virtual Large Language Model ist eine Library von Open Source-Codes, die von der vLLM-Community verwaltet werden. Dies ermöglicht LLMs, Berechnungen effizienter und in großem Umfang durchzuführen. Konkret handelt es sich bei vLLM um einen Inferenzserver, der den Output von gen KI-Anwendungen mithilfe einer effizienteren Nutzung des GPU-Speichers beschleunigt. Es ist branchenweit weit verbreitet, da es einen überlegenen Durchsatz und eine Performance mit niedriger Latenz bietet. Unterstützt werden Innovationen wie Paged Attention, das die effiziente Verarbeitung von mehr Tokens im GPU-Speicher ermöglicht.

- **Triton**: Triton wird oft mit einer Standalone-Runtime verwechselt und fungiert eher als Frontend-API (Application Programming Interface) für verschiedene Backend-Engines, darunter TensorRT und vLLM. Triton mit TensorRT bietet zwar eine etwas höhere Performance auf NVIDIA-GPUs, geht aber mit einer höheren Komplexität der Einrichtung und eingeschränkter Modellunterstützung einher. Kunden geben häufig an, dass das Erreichen von Performance-Verbesserungen mit Triton deutlich mehr Aufwand erfordert als mit vLLM.

- **SGLang:** SGLang ist ein neuerer Eintrag, der von vLLM abgeleitet ist und für bestimmte Use Cases optimiert ist. Es verwendet viele der gleichen zugrunde liegenden Komponenten wie vLLM, unterstützt jedoch weniger Modellarchitekturen. Auch wenn es in bestimmten Anwendungsbereichen vLLM überlegen sein mag, machen seine begrenzte Flexibilität und der geringe Support durch die Community es für eine breite unternehmensweite Einführung weniger attraktiv.

Ein dualer Ansatz zur Modelleffizienz

1. Optimieren der Inferenzlaufzeit (vLLM)

Runtime-Einschränkungen

Wie im letzten Kapitel erwähnt, **kann die effiziente Bereitstellung von LLMs aufgrund der mit grundlegenden Inferenzmethoden verbundenen Einschränkungen eine Herausforderung darstellen**.

Zu diesen Laufzeiteinschränkungen gehören die ineffiziente GPU-Speichernutzung, die suboptimale Batch-Verarbeitung und die langsame Tokengenerierung. Laufzeiten speichern Zwischenberechnungsdaten wie KV-Caches in der Regel ineffizient, verbrauchen viel GPU-Arbeitsspeicher und schränken die Kapazität für gleichzeitige Anforderungen ein. Darüber hinaus können vereinfachte Batch-Strategien dazu führen, dass GPUs ungenutzt oder nicht ausreichend ausgelastet sind, was den Durchsatz erheblich reduziert. Darüber hinaus haben grundlegende Laufzeiten mit langsamen Aufmerksamkeitsmechanismen zu kämpfen, was bei langen Eingabesequenzen zu einer verlängerten Latenz führt.

Warum vLLM?

vLLM hilft bei der Lösung vieler Herausforderungen bei der Laufzeit, indem es fortschrittliche Techniken bereitstellt, die speziell für die Inferenz-Performance optimiert wurden:

- **Kontinuierliches Batching:** vLLM minimiert die GPU-Leerlaufzeit durch die gleichzeitige Verarbeitung von Tokens von mehreren eingehenden Anforderungen. Anstatt jeweils eine einzelne Anfrage zu bearbeiten, werden Tokens aus verschiedenen Sequenzen in Batches gruppiert, was die GPU-Auslastung und den Inferenzdurchsatz erheblich verbessert.

- **PagedAttention:** vLLM verwendet eine neuartige Speicherverwaltungsstrategie namens PagedAttention, die große KV-Caches effizient verwaltet. Diese Technik weist GPU-Speicherseiten dynamisch zu und verwaltet sie, wodurch die Anzahl der gleichzeitigen Anforderungen erheblich erhöht und deutlich längere Sequenzen ohne Speicherengpässe unterstützt werden.

Weitere Informationen finden Sie in diesem **technischen Blog über vLLM**.

Vorteile des vLLM-Deployments

Umfassende Integrationsfunktionen: vLLM kann Modelle direkt aus gängigen Repositories wie Hugging Face laden und dient als hochleistungsfähiges Backend in Frameworks wie Triton Inference Server. Die Kompatibilität mit einer Vielzahl von Hardwareplattformen, darunter NVIDIA-GPUs, AMD-GPUs und Google-TPUs, vereinfacht das Deployment im Unternehmensbereich weiter.

Standardisierung und Anbieterunabhängigkeit: Durch die Verwendung einer weit verbreiteten Runtime wie vLLM profitieren Unternehmen von den Vorteilen der Standardisierung, die eine zuverlässige Performance in verschiedenen Hardwareumgebungen unterstützen und die Bindung an proprietäre Lösungen vermeiden.

Weitere Informationen zu den Parallelverarbeitungstechniken von vLLM finden Sie in diesem **technischen Deep-Dive-Blog**.

2. Optimieren des KI-Modells

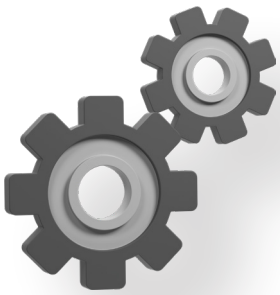
Die Bedeutung der Optimierung großer Sprachmodelle

Eine der größten Herausforderungen in der Produktion ist die Verwaltung der Speicher- und Recheneffizienz. Große Modelle erfordern häufig große Mengen an GPU-Arbeitsspeicher, um Parameter und Kontext im KV-Cache zu speichern, insbesondere bei langen Prompts oder mehreren gleichzeitigen Anforderungen. Wenn Modelle nicht optimiert sind, können sie ineffizient arbeiten, was zu höheren operativen Kosten führt. Die Latenz ist ein weiterer wichtiger Aspekt: Nutzende erwarten Echtzeit-Antworten, und Verzögerungen, die durch große Modellgrößen oder ineffiziente Ausführung verursacht werden, können sich negativ auf das Erlebnis und die Effizienz der nachgelagerten Workflows auswirken.

Warum ein Modell komprimieren?

Durch die Komprimierung eines Modells können Unternehmen einige der wichtigsten Herausforderungen bewältigen, mit denen Unternehmen beim Deployment von KI in großem Umfang konfrontiert sind: Kosteneffizienz und Performance-Optimierung.

Mit zunehmender Größe der Modelle auf Milliarden von Parametern wird ihre Bereitstellung in der Produktion ressourcenintensiv und erfordert viel Speicher und Rechenleistung. Modellkomprimierungstechniken, einschließlich Quantisierung und Sparsity, reduzieren leicht die Präzision und Anzahl der Parameter und verringern gleichzeitig den Speicher-Footprint und die Rechenanforderungen erheblich, ohne die Genauigkeit wesentlich zu beeinträchtigen. Durch Komprimieren von Modellen können Unternehmen KI-Workloads effizienter ausführen und dabei weniger GPUs oder andere Beschleuniger verwenden. Dadurch lassen sich die Betriebskosten drastisch reduzieren und auch schnellere Inferenzen ermöglichen. Dies ist besonders wichtig für Anwendungen, die Echtzeit-Antworten erfordern.



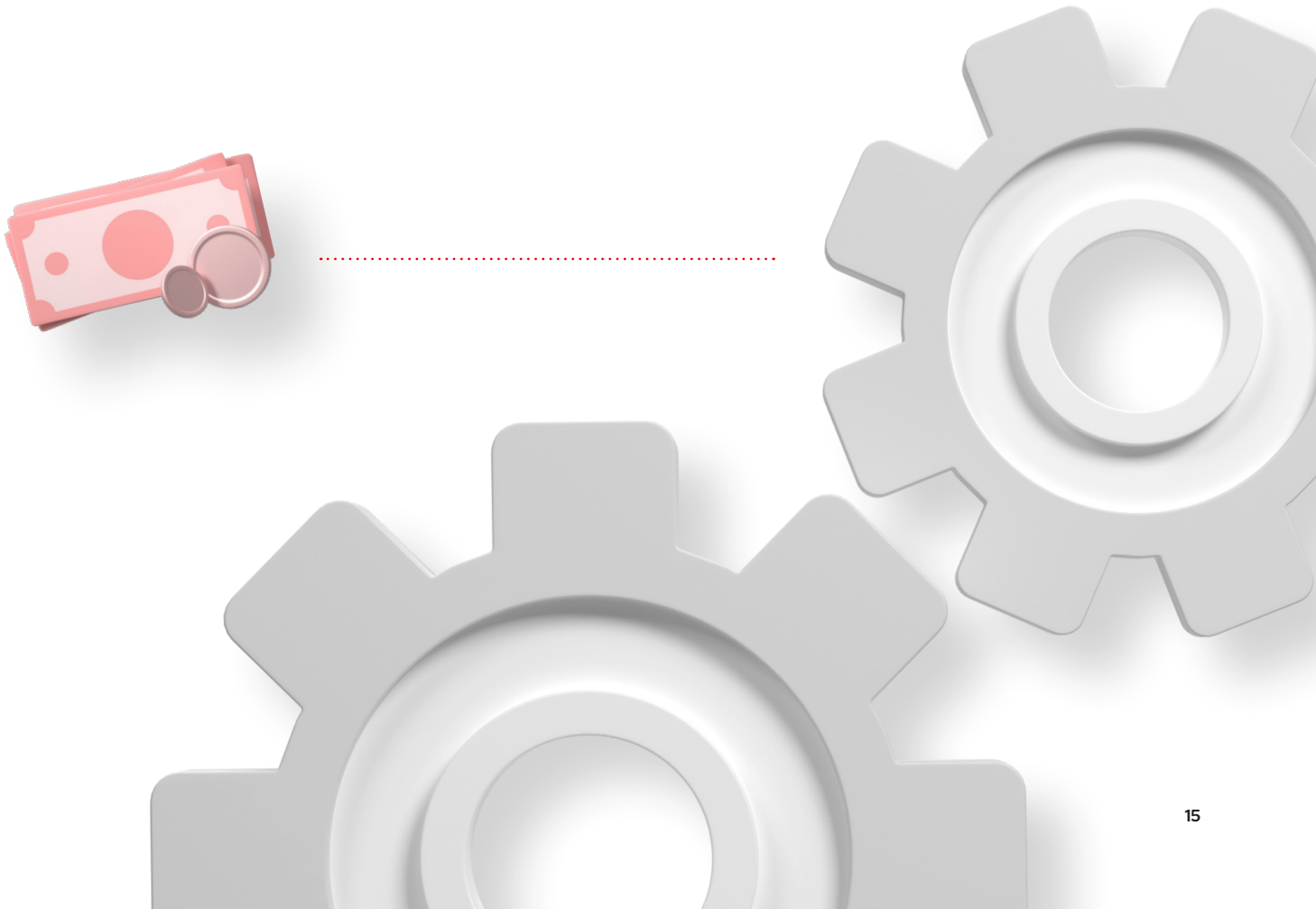
Wie kann mein Modell für die Inferenz kostenoptimiert werden?

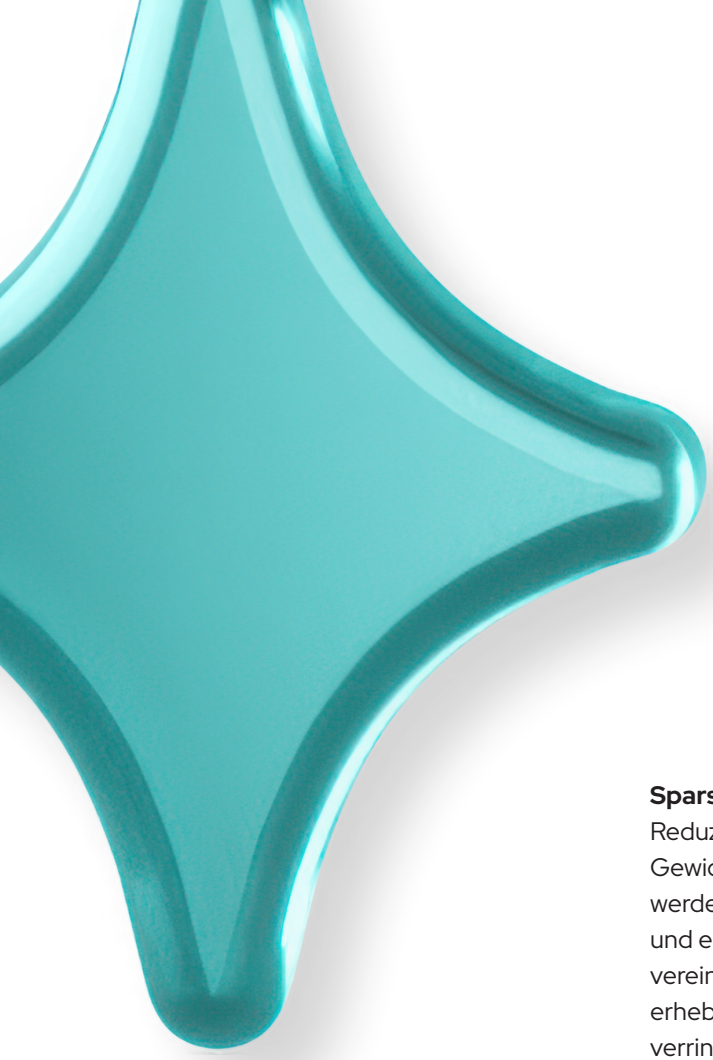
Eine der effektivsten Möglichkeiten, diese Kosten zu reduzieren, besteht darin, das Modell zu komprimieren. Komprimierungstechniken wie Quantisierung und Sparsity verkleinern die Modellgröße und reduzieren die Rechenanforderungen, sodass Inferenz-Workloads auf weniger oder kleineren GPUs ausgeführt werden können.

Durch Quantisierung wird ein Modell optimiert, indem die Präzision seiner numerischen Werte, insbesondere der Gewichtungen und Aktivierungen des Modells, reduziert wird. In der Regel arbeiten Modelle mit einer Präzision von 16 Bit (oder sogar 32 Bit) und verwenden Formate wie FP16 oder BF16.

Quantisierung komprimiert diese Werte auf Formate mit niedrigerer Präzision wie 8-Bit (INT8 oder FP8) oder sogar 4-Bit-Ganzzahlen (INT4). Durch diesen Prozess wird der für die Modellparameter benötigte Speicher erheblich reduziert, sodass Modelle wie ein Llama mit 70 Milliarden Parametern von etwa 140 GB auf nur 40 GB verkleinert werden können. Durch diese Reduzierungen wird nicht nur Arbeitsspeicher für zusätzliche Berechnungen frei, sondern der Durchsatz wird auch erhöht, insbesondere in Situationen mit begrenztem Speicher. Beispielsweise verarbeitet eine GPU mit 48 GB VRAM ein 40-GB-Modell schneller als ein 140-GB-Modell.

Eine aggressive Quantisierung kann jedoch die Genauigkeit aufgrund von Präzisionsverlust beeinträchtigen. Um dies zu verhindern, verwendet die fein abgestimmte Quantisierung Skalierungsfaktoren, die die Modellgenauigkeit beibehalten und erreicht häufig weniger als 1 % Degradation. Durch Quantisierung lassen sich durch optimierte Hardwarenutzung der Rechendurchsatz verdoppeln und so Latenzzeiten und Betriebskosten erheblich senken.





Sparsity optimiert ein Modell durch Einleiten einer strukturierten Reduzierung von Parametern, wobei ein großer Anteil der Gewichtungen des Modells auf 0 gesetzt wird. Bei dieser Technik werden redundante oder weniger kritische Gewichtungen identifiziert und eliminiert, wodurch die Berechnungen während der Inferenz vereinfacht werden. Sparsity kann die Komplexität des Modells erheblich reduzieren, wodurch Speichernutzung und Rechenlast verringert werden, was schnellere Inferenzen und geringere Betriebskosten ermöglicht.

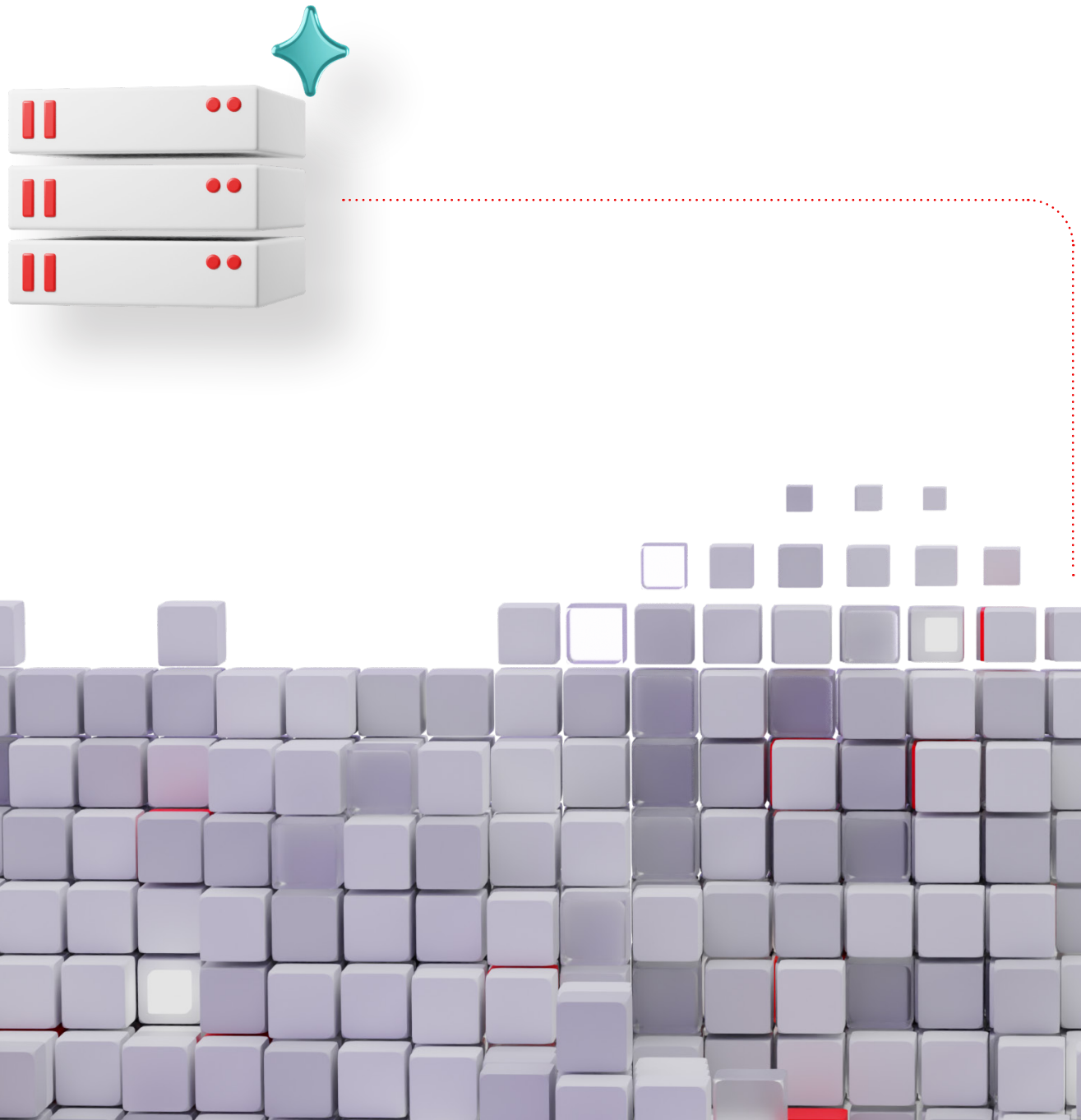
Um eine effektive Sparsity zu erreichen, muss das Modell jedoch neu trainiert werden – ein rechenintensiver Schritt, der erhebliche Vorabressourcen erfordert. Die Effizienz von Sparsity hängt von den Hardwarefunktionen ab, beispielsweise halbstrukturierter Sparsity, was von modernen Beschleunigern wie GPUs unterstützt wird, bei denen bestimmte Muster von Nullgewichtungen schnellere Berechnungen ermöglichen. Der Hauptvorteil ist die Fähigkeit, den Rechenaufwand bei richtiger Implementierung erheblich zu reduzieren.

Sparsity kann zwar bemerkenswerte Vorteile bringen, insbesondere in Kombination mit anderen Optimierungsmethoden wie Quantisierung, erfordert aber in der Regel einen aufwendigeren Optimierungsprozess. Daher wird sie für Szenarien mit umfangreichem Umfang oder speziellen Hardware-Setups empfohlen. Durch sorgfältiges Anwenden von Sparsity können Unternehmen die Inferenzeffizienz verbessern. Aufgrund der damit verbundenen Komplexität wird jedoch die Quantisierung häufiger als primäres Optimierungsverfahren empfohlen.

Durch die Einführung von Komprimierungs-Workflows und validierten Runtimes können Unternehmen ihre operativen Kosten besser verwalten, die Skalierbarkeit unterstützen und sich auf zukünftige Erhöhungen der KI-Nutzung vorbereiten, ohne die Infrastrukturressourcen zu überlasten.

Wird die Genauigkeit beeinträchtigt?

Modellkomprimierungstechniken wie Quantisierung und Sparsity reduzieren Speicher- und Rechenanforderungen, sind aber speziell darauf ausgelegt, ein akzeptables Maß an Genauigkeit beizubehalten. Beispielsweise ermöglicht die 8-Bit-Quantisierung in der Regel eine Genauigkeit nahezu der Baseline bei halbiertem Speicherbedarf. Selbst 4-Bit-Modelle können eine starke Performance beibehalten, wenn sie mit modernen Quantisierungstechniken wie Gewichtungsrunden und Optimierung optimiert werden. Strukturierte Sparsity-Patterns, wie etwa 2:4-Sparsity, ermöglichen es Hardwarebeschleunigern, redundante Abläufe zu überspringen, ohne die Ausgabequalität zu beeinträchtigen. In vielen Produktionsszenarien erzielen Teams erhebliche Ressourceneinsparungen bei minimaler oder gar keiner Verringerung der Modellperformance. Tests und Validierung sind weiterhin wichtig, aber für die meisten Anwendungen führt eine gut implementierte Komprimierung zu einer hocheffizienten Inferenz mit intakter Genauigkeit.

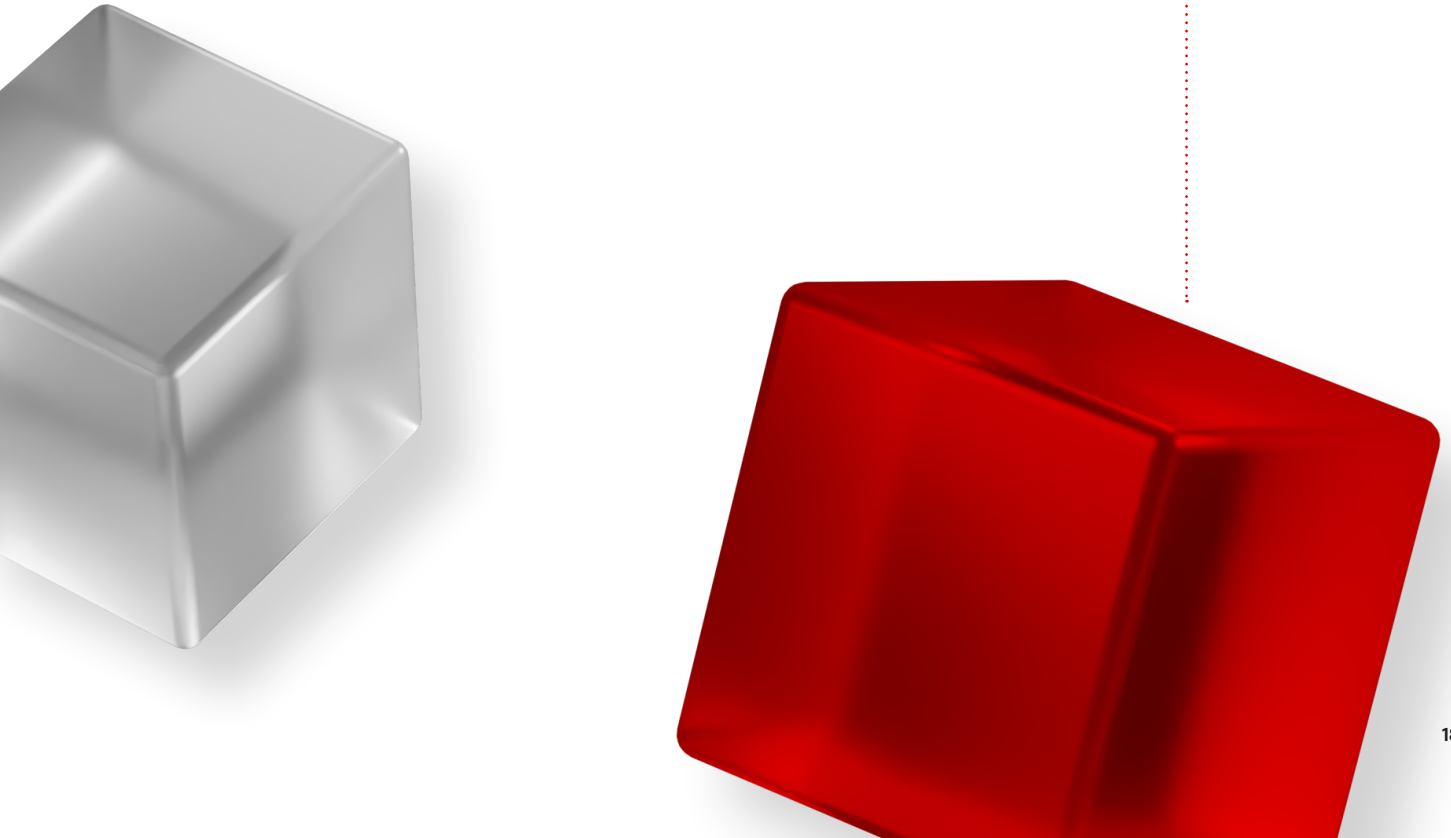


Red Hat AI

Was ist Red Hat AI?

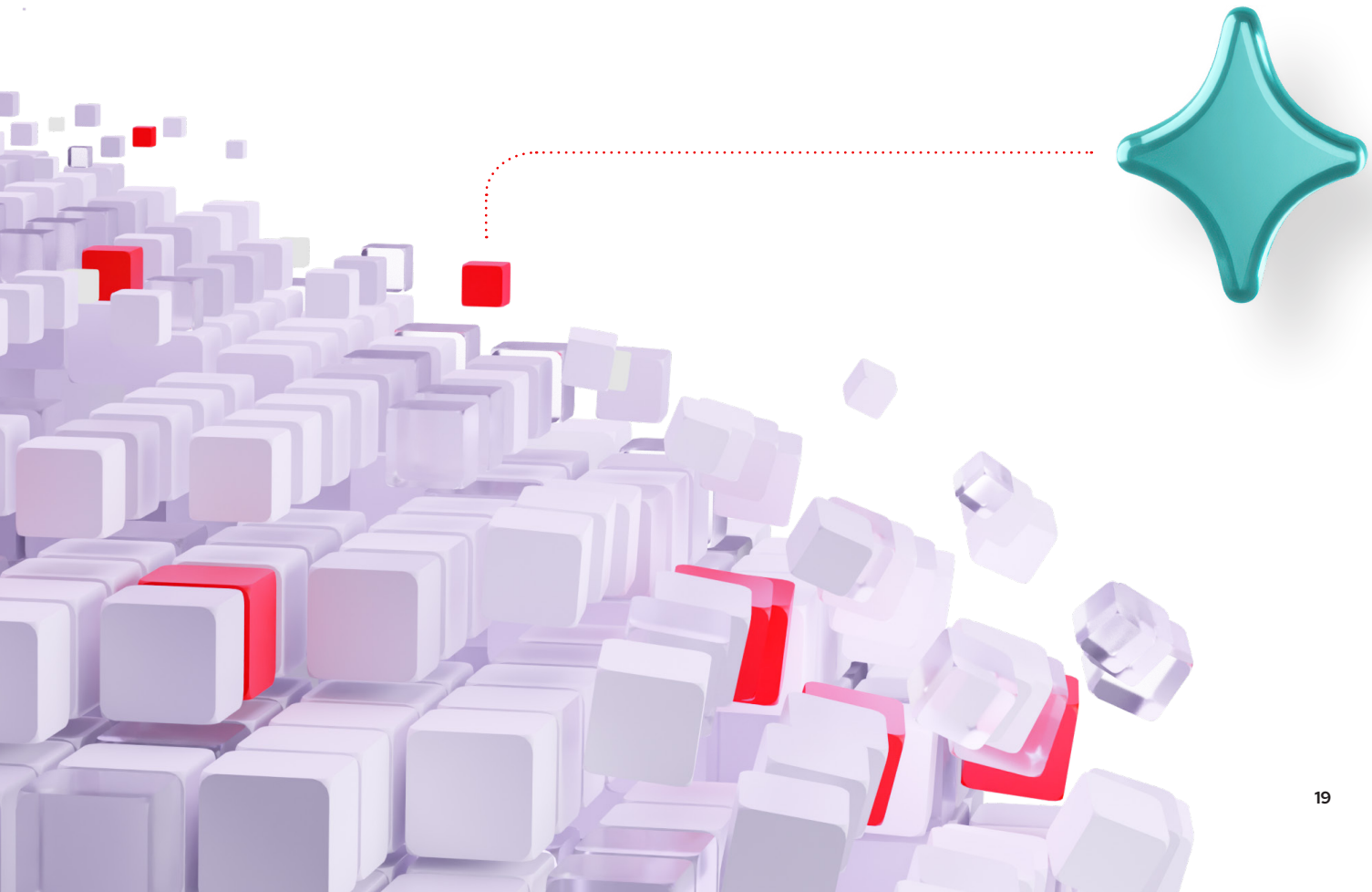
Red Hat AI ist eine Plattform, die KI-Innovationen beschleunigt und die operativen Kosten für die Entwicklung und Bereitstellung von KI-Lösungen in Hybrid Cloud-Umgebungen reduziert. Sie vereinfacht die Integration mit privaten Daten, hilft bei der Kostensenkung mit optimierten Modellen und effizienter Inferenz und beschleunigt die Bereitstellung agentischer KI-Workflows mit einer skalierbaren, flexiblen Plattform.

Red Hat AI ermöglicht Unternehmen, den Lifecycle sowohl von prädiktiven als auch von gen KI-Modellen in großem Umfang zu verwalten und zu überwachen – von Deployments mit einem einzelnen Server bis hin zu horizontal skalierten verteilten Plattformen. Die Plattform basiert auf Open Source-Technologien und Partnernetzwerken, die auf Performance, Stabilität und GPU-Unterstützung für verschiedene Infrastrukturen ausgerichtet sind.



Red Hat AI umfasst:

- **Optimierte und validierte Modelle:** Vorab evaluierte und leistungsgetestete Modelle, um den Aufwand für Tests und Fine Tuning zu reduzieren.
- **LLM Compressor:** Ein Toolkit, mit dem Nutzende Quantisierung und Komprimierung auf gängige Modelle anwenden und so den Ressourcenbedarf für die Inferenz reduzieren können, ohne auf Genauigkeit verzichten zu müssen.
- **Modellanpassung:** Tools zur Feinabstimmung oder Anpassung von Basismodellen an spezifische Unternehmensanforderungen.
- **Hochleistungsfähige Inferenzlaufzeit:** Eine optimierte vLLM-basierte Runtime, die moderne Batch- und Speicherverwaltungstechniken für eine effiziente, skalierbare und zuverlässige Modellbereitstellung nutzt.
- **LLMOps:** Praktiken und Tools, die das Bereitstellen, Monitoring und Verwalten von LLMs in Produktivumgebungen optimieren.
- **KI-Sicherheit und -Bewertungen:** Frameworks und Methoden zur Bewertung der Genauigkeit, Fairness und Robustheit von Modellen, um sicherzustellen, dass KI-Implementierungen verantwortungsvoll und zuverlässig sind.
- **Flexible und konsistente Skalierung:** Infrastrukturunterstützung, die beim Skalieren von KI in Hybrid Cloud-Umgebungen für Flexibilität und Konsistenz sorgt.
- **Beschleunigte agentische KI-Bereitstellung:** Funktionen für das schnelle Deployment fortschrittlicher, autonomer KI-Systeme, die Unternehmen an der Spitze der KI-Innovationen halten.



Optimieren von Modellen mit Red Hat AI

Red Hat AI unterstützt Unternehmen bei der Optimierung von KI-Modellen mit fortschrittlichen Techniken, die darauf ausgelegt sind, Effizienz, Genauigkeit und Kosteneffektivität in Einklang zu bringen.

Red Hat AI konzentriert sich auf 2 primäre Aspekte der Modelloptimierung: effiziente Runtime und komprimierte Modelle. Durch die Kombination dieser Ansätze bietet das KI-Portfolio von Red Hat eine schnelle Inferenzperformance und reduziert gleichzeitig die erforderlichen Rechenressourcen. Insbesondere verwendet der Red Hat AI Inference Server kontinuierliches Batching und speichereffiziente Methoden, um sicherzustellen, dass die Modelle mehr Token pro Sekunde verarbeiten und so einen höheren Durchsatz bei geringerer GPU-Nutzung erreichen.

Red Hat AI LLM Compressor bietet einen standardisierten Ansatz für die Anwendung der in diesem E-Book behandelten Komprimierungstechniken und soll eine Optimierung mit einer beibehaltenen Genauigkeit von 99 % ermöglichen. Damit können Nutzende optimierte Versionen gängiger Modelle generieren, die für Inferenzlaufzeiten wie vLLM abgestimmt sind. Dies erleichtert die Ausführung leistungsstarker, komprimierter Modelle auf einer Vielzahl von Hardware-Konfigurationen.

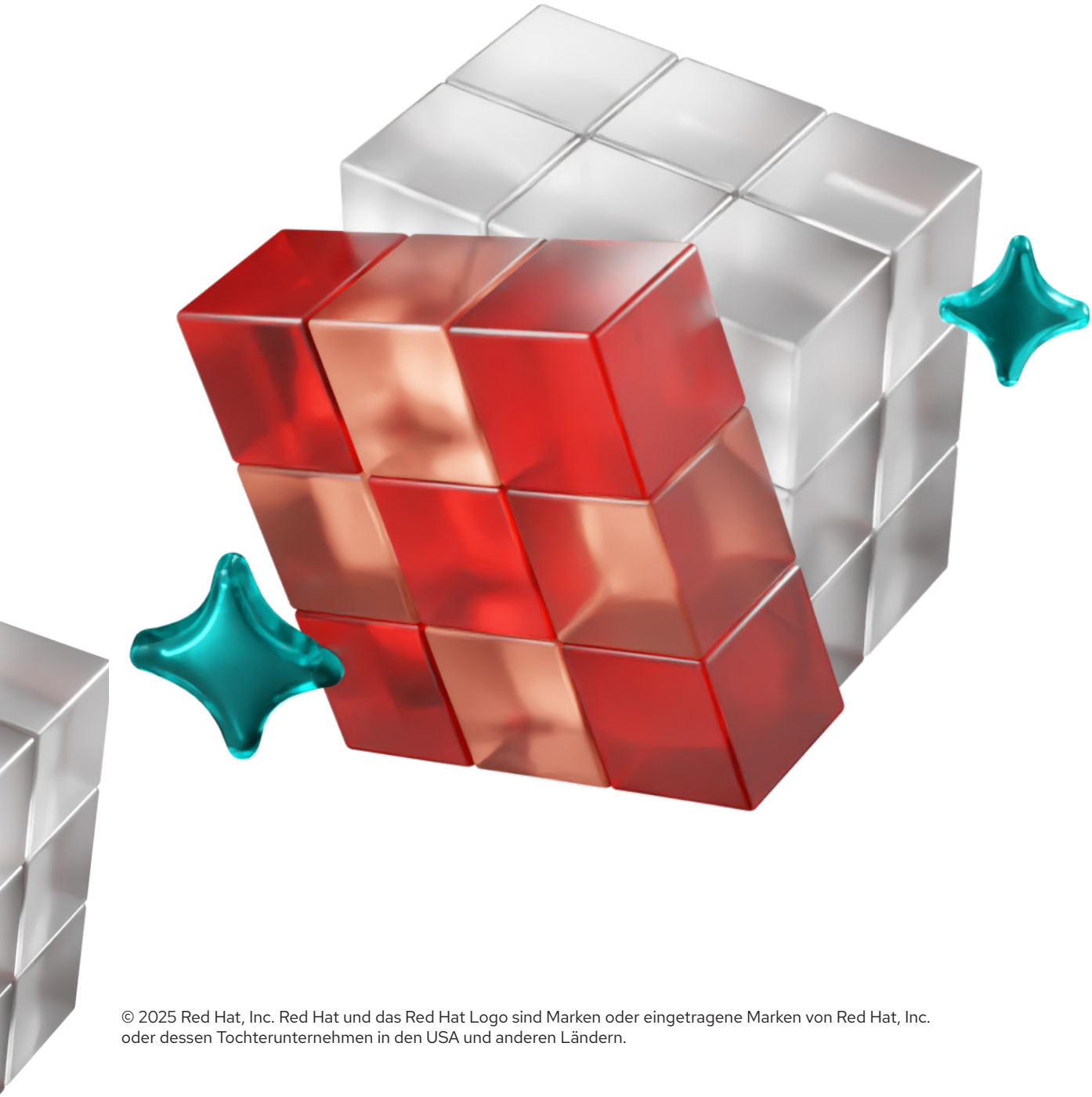
Red Hat AI bietet eine umfassende Validierung, damit Unternehmen optimierte Modelle sicher auswählen, bereitstellen und skalieren können. Angesichts der großen Auswahl an verfügbaren LLMs stehen Unternehmen oft vor der Herausforderung, die Modelle zu finden, die ihren Use Cases im Hinblick auf Genauigkeit, Performance und Kosteneffizienz am besten entsprechen. Um diese Herausforderungen zu bewältigen, verwendet Red Hat AI Open Source-Validierungstools (wie GuideLLM, Language Model Evaluation Harness und vLLM), um die Modell-Performance bei mehreren Evaluierungsaufgaben rigoros zu vergleichen. Diese Validierung unterstützt die Reproduzierbarkeit und eine fundierte Modellauswahl und reduziert Komplexität und Unsicherheit.

Red Hat AI bietet außerdem **Kapazitätsrichtwerte**, um Unternehmen bei der genauen Planung der KI-Infrastruktur und der Optimierung der Nutzung von Ressourcen zu unterstützen. So werden häufige Probleme wie nicht ausreichende Hardwareauslastung, hohe Computing-Kosten und Ineffizienzen zur Inferenzzeit behoben. Mit dieser Kombination aus validierten Modellen, optimierten Einstellungen für das Deployment und maßgeschneiderte Hardware-Empfehlungen können Unternehmen ihre Flexibilität erhöhen, Deployments beschleunigen, eine vorhersehbare Performance erreichen und gleichzeitig Kosten effektiv verwalten.

Mit Komprimierungstechniken und optimierten Laufzeiten ermöglicht Red Hat AI den Einsatz von LLMs in großem Umfang. So können Teams steigende Anforderungen erfüllen und gleichzeitig die Kontrolle über Kosten, Komplexität und Nutzung von Rechenressourcen behalten.

Nächste Schritte

Sind Sie bereit, die Kosten und Komplexität beim Bereitstellen von LLMs zu reduzieren? Erfahren Sie mehr über **Red Hat AI Inference Server** oder kontaktieren Sie den Red Hat Vertrieb.



© 2025 Red Hat, Inc. Red Hat und das Red Hat Logo sind Marken oder eingetragene Marken von Red Hat, Inc. oder dessen Tochterunternehmen in den USA und anderen Ländern.