Fraunhofer

Timo Leitritz | Jonas Heinle

KI-Beschleunigung durch spezialisierte Hardware und Software

Hrsg.: Thomas Bauernhansl, Marco Huber, Werner Kraus

Im Rahmen des

KI Fortschrittszentrum
LERNENDE SYSTEME UND KOGNITIVE ROBOTIK

Vorwort

Künstliche Intelligenz (KI) ist eine der zentralen Technologien für die Zukunft. Ihre Einführung und der Einsatz fordern Unternehmen im besonderen Maß heraus.

Mit dem KI-Fortschrittszentrum des Fraunhofer IAO und Fraunhofer IPA unterstützen wir Unternehmen dabei, das Potenzial von KI zu erkennen und dieses wirtschaftlich nutzbar zu machen. An der Schnittstelle zwischen anwendungsorientierter Wirtschaft und exzellenter Forschung des Cyber-Valley-Konsortiums entwickeln wir innovative KI-Anwendungen für die Praxis und treiben damit die Kommerzialisierung von KI voran. Erklärtes Ziel ist dabei, menschzentrierte KI-Lösungen zu entwickeln. Denn nur wenn Menschen mit einer neuen Technologie intuitiv interagieren und vertrauensvoll zusammenarbeiten, kann ihr Potenzial optimal ausgeschöpft werden.

Die Studienreihe »Lernende Systeme« des KI-Fortschrittszentrums gibt Einblick in die Potenziale und die praktischen Einsatzmöglichkeiten von KI. Dabei werden übergreifende Themen wie Zuverlässigkeit, Erklärbarkeit (xAI), cloudbasierte Plattformen, Technologien und Einführungsstrategien diskutiert. Zudem werden einzelne Anwendungsbereiche in der Wissensarbeit, Bauwirtschaft, Produktion und dem Kundenservice im Detail beleuchtet.

Die vorliegende Studie untersucht den Einsatz von KI-Hardware und -Software im Kontext von Computer Vision anhand von Recherchen und konkreten Benchmarks auf mehreren KI-Plattformen. Sie gibt einen Überblick über die am Markt verfügbare Beschleuniger-Hardware und ordnet diese in Einsatzszenarien von IoT bis Datacenter ein. Zudem werden ausführliche Benchmarks mit Fokus auf Performance und Effizienz durchgeführt, um mehrere KI-Hardware-Software-Plattformen miteinander zu vergleichen und deren Vor- und Nachteile zu diskutieren.

Diese Ergebnisse liefern wichtige Erkenntnisse und Kriterien für die Evaluation von KI-Systemen. Darauf basierend können fundiertere Entscheidungen über die Wahl von KI-Hardware und Softwareoptimierungen für die Beschleunigung von KI-Systemen getroffen werden, um effizientere und kostengünstigere Systeme zu entwickeln.

Wir wünschen Ihnen eine spannende Lektüre und freuen uns, wenn wir in Zukunft auch Sie mit unserer Expertise auf Ihrem Weg zur menschzentrierten KI unterstützen dürfen.

Thomas Bauernhansl, Marco Huber, Werner Kraus
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA

Inhalt

Management Summary

Künstliche Intelligenz (KI) wird vermehrt eingesetzt, und gleichzeitig steigt der benötigte Rechenaufwand mit der Weiterentwicklung von KI stetig. Dadurch wachsen die Anforderungen an die Rechenhardware und -software. Unternehmen investieren daher verstärkt in spezialisierte KI-Beschleuniger.

Die Leistungsfähigkeit von KI-Systemen beruht auf dem abgestimmten Zusammenspiel von KI-Modell, Hardware und Software-Stack. Das Fraunhofer IPA untersucht deshalb verschiedene Beschleuniger hinsichtlich ihrer KI-Fähigkeiten. Die Ergebnisse zeigen: Diskrete GPUs liefern den höchsten Durchsatz, allerdings mit hoher Leistungsaufnahme. Edge-Beschleuniger wie Nvidias Jetson-Reihe, Hailos Hailo-8 und SiMas MLSoC bieten ein gutes Verhältnis aus Energieeffizienz und Integrationsfähigkeit. Zudem wird deutlich, dass besonders leistungsfähige KI-Hardware wie GPUs bei kleineren Modellen bzw. Batch-Größen nicht voll ausgelastet sind. CPUs ohne zusätzliche KI-Beschleuniger sind für komplexe KI-Modelle ungeeignet, bleiben aber wichtig für die Vor- und Nachverarbeitung und können unter Umständen besonders kleine Modelle ausreichend schnell ausführen.

Des Weiteren ist eine Softwareoptimierung durch reduzierte Präzision, optimierte Laufzeitumgebungen und Co. entscheidend. Für die industrielle Nutzung müssen außerdem Faktoren wie Lieferbarkeit, Echtzeitfähigkeit, Sicherheit und Erklärbarkeit berücksichtigt werden. Die Studie beinhaltet außerdem eine umfangreiche Marktübersicht verfügbarer KI-Hardware und ordnet diese den relevanten Einsatzgebieten zu.

1. Einleitung

Die rasanten Fortschritte rund um Künstliche Intelligenz (KI) prägen zunehmend industrielle Prozesse und Anwendungen. Insbesondere Methoden aus der Computer Vision (dt. maschinelles Sehen) eröffnen Unternehmen in der Automatisierung, Qualitätskontrolle, Robotik und weiteren Einsatzgebieten immense Potenziale, um effizienter und wettbewerbsfähiger zu werden. Gleichzeitig wachsen die Komplexität und der Ressourcenbedarf moderner KI-Modelle kontinuierlich an, was insbesondere bei Echtzeitanwendungen hohe Anforderungen an die eingesetzte Hardware stellt.

KI entwickelt sich im deutschen Mittelstand von einer Option zu einer zentralen Voraussetzung für Wettbewerbsfähigkeit. Eine KPMG Befragung [1] von 653 Führungskräften zeigt, dass 69 Prozent der Unternehmen bereits eine KI Strategie besitzen und 82 Prozent ihre Investitionen in die Technologie erhöhen wollen. Dabei erhoffen sie sich vor allem schnellere Datenanalysen, Automatisierung, neue Produkte und Umsatzzuwächse.

Vor diesem Hintergrund steigt das Interesse der Industrie an spezialisierter KI-Hardware bzw. KI-Beschleunigern signifikant. Diese speziell entwickelten Hardware-Komponenten versprechen deutliche Performance-Vorteile gegenüber rein CPU-basierten Lösungen und sollen gleichzeitig den Energieverbrauch reduzieren. Neben etablierten Beschleunigern wie GPUs rücken effizientere, spezialisierte KI-Chips zunehmend in den Fokus der Unternehmen. Gartner [2] prognostizierte für 2024 und 2025 einen globalen Umsatzanstieg bei KI-Beschleunigern um 33 beziehungsweise 29 Prozent, was maßgeblich auf die steigende Nachfrage aus den Bereichen generative KI und Computer Vision zurückzuführen ist.

Diese Studie verfolgt das Ziel, eine systematische Untersuchung der Leistungsfähigkeit unterschiedlicher KI-Beschleuniger in Bezug auf Software und Hardware durchzuführen, wobei der Schwerpunkt auf KI-Modellen und Beschleunigern aus der Computer Vision liegt. Es werden Modelle für Klassifikation, Objekterkennung und Segmentierung betrachtet, die unter anderem Transformer-basierte Architekturen nutzen. Dies ist die gleiche den großen Sprachmodellen (engl. Large Language Models) zugrundeliegende Architektur.

Die Inhalte und Erkenntnisse dieser Studie sollen Unternehmen fundierte Entscheidungsgrundlagen liefern, um KI-Hardware und zugehörige Software besser einzuordnen und optimal in bestehende oder zukünftige Anwendungen zu integrieren.

Die Studie gliedert sich wie folgt: Zunächst werden in Kapitel 2 Grundlagen zu KI-Hardware und -Software behandelt. Kapitel 3 stellt eine Marktübersicht aktueller KI-Beschleuniger vor. Darauf folgt in Kapitel 4 ein detaillierter Vergleich ausgewählter Hardware- und Softwarelösungen anhand konkreter Metriken wie Latenz, Durchsatz und Effizienz. Abschließend werden in Kapitel 5 Praxisbeispiele erörtert.

2. Grundlagen

Zum besseren Verständnis der folgenden Kapitel werden in diesem Grundlagen-Kapitel sowohl eine allgemeine Einführung zu KI gegeben als auch Details zu existierenden KI-Architekturen und Problemstellungen. Außerdem werden die verschiedenen Architekturen, Einsatzgebiete und Formate von KI-Hardware behandelt.

2.1. KI, Machine Learning und Deep Learning

KI beschreibt die Fähigkeit von Computern und Maschinen, Aufgaben zu lösen, die normalerweise menschliche Intelligenz erfordern. Dabei umfasst KI eine Vielzahl von Methoden und Techniken, darunter das Maschinelle Lernen (engl. Machine Learning, ML). Maschinelles Lernen bezeichnet Verfahren, bei denen Systeme aus Daten lernen, Muster erkennen und daraus Entscheidungen treffen oder Prognosen ableiten, ohne explizit für jede Aufgabe programmiert zu werden. Dieser Lernprozess wird als Training bezeichnet und wird auf ein zuvor entwickeltes Modell bzw. eine Modellarchitektur angewendet. Das Ausführen des Modells für eine Vorhersage wird als Inferenz bezeichnet. Beispiele für ML-Ansätze sind Support-Vector-Machines (SVM), Entscheidungsbäume oder künstliche neuronale Netze. Deep Learning (DL) ist ein Teilbereich von Machine Learning und konzentriert sich auf besonders tiefe neuronale Netze (vgl. Abbildung 1). Neuronale Netze besitzen lernbare Parameter, auch Gewichte genannt, die während des Trainingsprozesses eingestellt werden. Einen detaillierten Überblick dazu liefert das frei verfügbare Buch »Deep Learning« [3].

Abbildung 1: Künstliche Intelligenz vs. Maschinelles Lernen vs. Künstliche Neuronale Netze vs. Deep Learning. Quelle: Fraunhofer IPA.

Aktuelle Large Language Models (LLM) aus dem DL-Bereich wie DeepSeek-R1 [4] besitzen, je nach Anwendungsfall, bis zu 671 Milliarden lernbare Parameter. Aus dieser Menge an Parametern resultieren gigantische Mengen an Rechenoperationen, die für den Einsatz der KI ausgeführt werden müssen. Allerdings können auch wesentlich kleinere, effizientere Modelle für spezifische Anwendungsfälle, wie bspw. zur visuellen Qualitätsprüfung von Bauteilen (siehe auch https://www.ipa.fraunhofer.de/de/aktuelle-forschung/bild--und-signalverarbeitung.html), genutzt werden und ausreichend sein.

Für die Ausführung von KI-Modellen kann unterschiedliche Rechner-Hardware (siehe Abschnitt 2.3.1) eingesetzt werden, von der maßgeblich die Performance und Effizienz der KI-Systeme abhängt.

2.2. KI-Architekturen und Problemstellungen

KI-Modelle lassen sich in Architekturen und Blöcke unterteilen, die sich unterschiedlich auf Rechenhardware auswirken. Dies wird im Folgenden näher beleuchtet und zudem ein Überblick zu KI-Problemstellungen und zugehörigen State-of-the-Art-Modellen (SOTA) gegeben.

2.2.1. Grundbausteine der modernen KI

Moderne KI beinhaltet diverse Basis-Blöcke, auch Operatoren oder Schichten (engl. layers) genannt, die sich in den unterschiedlichsten Architekturen wiederfinden und miteinander kombiniert werden. Ein Auszug der wichtigsten Komponenten wird im Folgenden erläutert.

Aktivierungsfunktionen

Aktivierungsfunktionen sind elementare nichtlineare Operatoren, die den Ausgabewert einer Schicht transformieren. Ohne sie wäre ein Netzwerk – selbst bei vielen Schichten – äquivalent zu einer linearen Abbildung. Sigmoid und Tangens hyperbolicus (tanh) waren die ersten Standardfunktionen, wurden aber weitgehend durch ReLU [5] und Varianten (z. B. Leaky ReLU, GELU) ersetzt, da diese stabilere Gradienten und ein schnelleres Training ermöglichen. Sie sind in praktisch allen Architekturen unverzichtbar.

Fully Connected Layer
Eine Fully Connected Layer (FC-Schicht) besteht aus vernetzten künstlichen Neuronen und berechnet für jede Ausgabeeinheit eine gewichtete Summe aller Eingaben, ergänzt um einen Bias-Term. Auf die Ausgabe wird anschließend eine der oben genannten nichtlinearen Aktivierungsfunktionen angewendet [6]. Formal entspricht der lineare Anteil einer Matrix-Vektor-Multiplikation bzw. bei Verarbeitung mehrerer Eingaben einer Matrix-Matrix-Multiplikation. Durch die Stapelung mehrerer solcher Schichten in Kombination mit nichtlinearen Aktivierungsfunktionen können hochdimensionale nichtlineare Funktionen modelliert werden. FC-Schichten bilden den elementarsten Operator in neuronalen Netzen, da sie sowohl in Modellen mit ausschließlich FC-Schichten als auch als Teil komplexerer Modelle wie Transformern eingesetzt werden. FC-Schichten werden auch als Linear Layer, Dense Layer oder GEMM (General Matrix Multiplication) bezeichnet.

Convolutional-Schicht
Die Convolutional-Schicht (Faltungsschicht) berechnet lokale gewichtete Summen durch Verschiebung kleiner Filter über die Eingabe, was als Faltung bezeichnet wird [7]. Dabei werden Gewichte über den gesamten Eingabebereich geteilt, wodurch die Anzahl der Parameter reduziert wird und translations-invariante Merkmale entstehen. Faltungsschichten werden in 1D (Sequenzen), 2D (Bilder) und 3D (Volumendaten) eingesetzt und können ebenso als Matrix-Vektor-Multiplikation dargestellt werden. Sie sind die zentrale Operation in Convolutional Neural Networks (CNN, dt. Faltungsnetze), können aber auch in Kombination mit FC-Schichten und anderen Operatoren genutzt werden. Nach Faltungsschichten werden ebenso wie bei FC-Schichten Aktivierungsfunktionen angehängt.

(Self-)Attention
Der Attention-Operator [8] berechnet eine gewichtete Kombination von Eingaberepräsentationen, wobei die Gewichte dynamisch durch eine Kompatibilitätsfunktion (meist Skalarprodukte) bestimmt werden. Self-Attention bezeichnet den Fall, dass eine Sequenz mit sich selbst in Beziehung gesetzt wird. Diese Operation erlaubt es, Abhängigkeiten zwischen weit entfernten Elementen der Sequenz direkt zu modellieren und bildet die Grundlage moderner Transformer-Architekturen, die unter anderem in LLMs oder modernen Netzen der Computer Vision eingesetzt werden. Auch hier wird Attention häufig mit FC-Schichten kombiniert, etwa in den Feed-Forward-Blöcken der Transformer.

2.2.2. Anforderungen an Rechenhardware

KI-Modelle in Form von neuronalen Netzen lassen sich als gerichtete Graphen verschachtelter Funktionen wie folgt mathematisch auffassen:

$$y = f_L \circ f_{L-1} \circ \ldots \circ f_1(x)$$

wobei jede Schicht f_i auf Operatoren wie bspw. FC-Schichten, Faltungsschichten, Attention oder Aktivierungsfunktionen beruht.

Die dominante primitive Rechenarbeit in all diesen Operatoren ist die Multiply-Accumulate-Operation (MAC). Formal lässt sie sich als folgende Gleichung

$$y = Wx + b$$

darstellen, wobei $W \in R^{m \times n}$ die Gewichtsmatrix, $x \in R^n$ der Eingabevektor und $b \in R^m$ der Bias-Vektor ist. Elementweise ergibt sich die Berechnung jeder Komponente y_i von $y \in R^m$ zu

$$y_i = \sum_{j=1}^{n} W_{ij} \cdot x_j + b_i, i = 1, \ldots, m$$

Damit wird sichtbar, dass jeder Ausgabewert durch eine Sequenz von Multiplikationen und Additionen berechnet wird – der MAC-Operation.

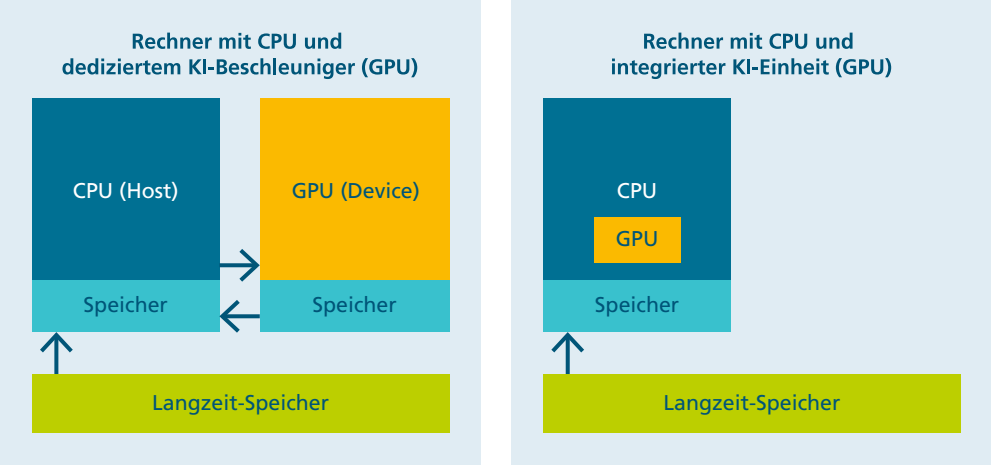


Abbildung 2: Grafik zur Verdeutlichung der Kopiervorgänge in unterschiedlichen Systemarchitekturen. Die grauen Pfeile repräsentieren Kopiervorgänge über PCI-Express, welche die KI-Ausführung bremsen können und langsamer sind als prozessorinterne Speicherzugriffe. Quelle: Fraunhofer IPA.

Somit sind für die Ausführung bzw. Berechnung von KI viele Multiplikations- und Additionsoperationen notwendig, die je nach Problemstellung auf Millionen von Eingabedatenpunkten und über zahlreiche Schichten mit Millionen bzw. Milliarden von Gewichten berechnet werden müssen. Bei diesem Prozess müssen zudem häufig Daten aus dem Arbeitsspeicher des Rechners geladen und zwischengespeichert werden, weshalb die Geschwindigkeit bzw. Bandbreite der Speicheranbindung für KI-Anwendungen besonders relevant ist. Beim Einsatz von Beschleunigern mit separatem Speicher müssen zusätzlich Host-Device- und Device-Host-Kopiervorgänge miteinbezogen werden (siehe Abbildung 2). Der Speicherbedarf und die Rechengeschwindigkeit der KI-Anwendung hängen maßgeblich von der genutzten Rechenpräzision ab, bspw. FP32, TF32, BF16, FP16, FP8, FP4, INT8 etc. (siehe Abschnitt 2.4).

Der KI-Beschleuniger AMD Instinct MI300X erreicht beispielsweise bis zu 1,3 PFLOPS (Peta Floating Point Operations per Second) in BF16/FP16 sowie bis zu 2,6 PFLOPS in FP8 und bietet 192 GB HBM3 mit bis zu 5,3 TB/s Bandbreite.

Die Rechenanforderungen während des Trainings sind im Vergleich zur Inferenz signifikant höher, da Ersteres zusätzlich zum Vorwärtspass den Backpropagation-Algorithmus [6], Gradientenakkumulation und Optimizer-Kernel wie z.B. Adam [9] inkludiert. Daraus ergeben sich deutlich höhere Speicheranforderungen, etwa durch das Vorhalten von Aktivierungen, eine in der Regel doppelte Anzahl an Multiply-Accumulate-Operationen (MACs) sowie häufig die Notwendigkeit höherer Genauigkeit (FP32) in kritischen Rechenschritten. Inferenz hingegen entspricht ausschließlich der Ausführung des Vorwärtspasses. Hier genügt in vielen Fällen eine geringere numerische Präzision (bspw. FP8, INT8).

Die in neuronalen Netzen eingesetzten Operatoren unterscheiden sich in ihren Rechen- und Speicheranforderungen. Hierbei verursachen große FC-Schichten eine besonders hohe Menge an Speicherzugriffen und MAC-Operationen. Typische Faltungsschichten benötigen durch das Schiebefensterprinzip und die daraus folgende Datenwiederverwendung weniger Bandbreite und sind dadurch effizienter [10]. Aufmerksamkeitsmechanismen wie Scaled Dot-Product Self-Attention verursachen quadratische Abhängigkeiten in der Sequenzlänge und erfordern entsprechend große Matrizen [8]. Neuere Arbeiten zeigen, dass hier nicht die Rechenleistung,

sondern häufig Speicherbandbreite und Speicherzugriffsmuster den dominanten Engpass darstellen, woraus Optimierungen wie FlashAttention hervorgegangen sind [11]. Elementweise Operationen wie ReLU-Aktivierungsfunktionen sind dagegen rechenarm.

2.2.3. Problemstellungen und SOTA-KI-Modelle

Neuronale Netze werden heute auf zahlreichen Datenmodalitäten eingesetzt, darunter Bilder, Videos, Audio, tabellarische Daten, Zeitreihen und Text. Unabhängig von der Modalität folgt der Aufbau moderner Architekturen meist einem modularen Prinzip, nach dem sich ein Modell grob in Backbone, Neck und Head gliedern lässt, angelehnt an den Aufbau des menschlichen Oberkörpers. Der Backbone dient dabei als universeller Merkmalsextraktor und ist häufig austauschbar, während der Head die aufgabenspezifische Transformation vornimmt, beispielsweise Klassifikation, Regression oder Sequenzgenerierung. In vielen Fällen können hybride Architekturen mehrere Heads umfassen, sodass ein Modell gleichzeitig unterschiedliche Aufgaben löst, etwa die Kombination von Klassifikation (bspw.: Welche Klasse hat das Objekt?) und Regression (bspw.: Welches Volumen hat das Objekt?). Durch dieses Prinzip ist es möglich, aufwendiges Pretraining auf großen Datensätzen nur einmalig für den Backbone auszuführen und diesen dann für verschiedenste Anwendungsfälle wiederzuverwenden. In diesem Fall kann der Backbone auch eingefroren werden, um die Anzahl an zu trainierenden Gewichten zu reduzieren und das Training zu beschleunigen oder Overfitting zu verhindern.

Abbildung 3: Vergleich zwischen Klassifikation, Objekterkennung und semantischer Segmentierung. Quelle: Fraunhofer IPA/Foto: Rainer Bez.

Eine typische Problemstellung ist die Klassifikation von Bildern. Hier geht es darum, einem Eingabebild eine feste Klasse zuzuordnen. Klassische CNNs wie VGGNet [12] oder ResNet [13] haben Maßstäbe gesetzt, indem sie zeigten, dass Architekturen mit tiefen Faltungsschichten eine robuste Merkmalsextraktion und Klassifizierung ermöglichen. In jüngerer Zeit wurden diese Ansätze durch Vision Transformer ergänzt, die auf Self-Attention-Mechanismen beruhen und einen alternativen Weg der Merkmalsextraktion darstellen [14].
Beispiel-Modelle: ResNet-50, EfficientNet, Vision Transformer (ViT).

Die Objekterkennung (eng. Object Detection) erweitert die Bildklassifikation um die Fähigkeit, mehrere Objekte innerhalb eines Bildes nicht nur zu klassifizieren, sondern auch räumlich zu lokalisieren. Frühe Durchbrüche wurden durch zweistufige Ansätze wie Faster R-CNN erzielt, die Region-Proposals erzeugen und diese anschließend klassifizieren [15]. Demgegenüber stehen einstufige Modelle wie YOLO, die Objekterkennung in einem einzigen Schritt realisieren

und damit Echtzeitanwendungen ermöglichen [16]. Neuere Arbeiten wie DETR haben zudem gezeigt, dass Transformer-basierte Architekturen auch in der Objektdetektion konkurrenzfähig sind [17].
Beispiel-Modelle: Faster R-CNN, YOLO-Familie, DETR, DINOv2.

Bei der semantischen Segmentierung von Bildern wird jedem Pixel eine semantische Klasse zugeordnet. Damit können nicht nur Objekte erkannt, sondern auch ihre genauen Umrisse bestimmt werden. U-Net hat in diesem Feld insbesondere für medizinische Bildverarbeitung eine prägende Rolle gespielt [18]. 2023 wurde das Segment Anything Model (SAM) vorgestellt, das als universelles Segmentierungsmodell auch ohne spezialisierte Anpassung auf neue Datensätze anwendbar ist [19].
Beispiel-Modelle: U-Net, SegFormer, SAMv2.

Generative KI in Form von großen Sprachmodellen (LLMs) repräsentiert eine weitere zentrale Problemstellung. Das Ziel ist hierbei die Modellierung und Generierung natürlicher Sprache, meist durch die iterative Vorhersage des nächsten Worts bzw. Tokens in einem Satz. Einen grundlegenden Paradigmenwechsel brachte die Einführung des Transformer-Modells [8], das Self-Attention als Kernbaustein etablierte. Darauf aufbauend wurden Modelle wie LLaMa [20] entwickelt, die auf Milliarden von Parametern skalieren.
Beispiel-Modelle: Llama 4, DeepSeek-R1, OpenAI GPT-5, Google Gemini.

Im späteren Performancevergleich in Abschnitt 4 werden Bildklassifikations-, Objekterkennungs- und Segmentierungsmodelle untersucht. LLMs werden nicht explizit als Benchmark-Modelle eingesetzt, es werden aber deren grundlegende Operatoren (Attention, FC-Schicht) durch das Benchmarken von Transformer-basierten Ansätzen wie ViT [14], DETR [17] und SegFormer [21] getestet.

2.3. KI-Hardware

Die Performance eines KI Systems hängt wesentlich von der darunterliegenden Hardware ab, die sich hinsichtlich Rechenleistung, Energieeffizienz und Flexibilität stark unterscheidet. Im Folgenden wird auf die verschiedenen Architekturen, Einsatzgebiete und Formate von KI-Hardware eingegangen.

2.3.1. Architekturen

KI kann auf diversen Hardwaretopologien ausgeführt werden. Notwendig ist dabei eine Recheneinheit, die die Berechnungen durchführt. Ein vollständiges KI-Hardwaresystem besteht mindestens aus einer CPU (Central Processing Unit) und typischerweise zusätzlichen KI-Beschleunigern. Diese Beschleuniger können direkt im CPU-Chip integriert sein (System on Chip, SoC) oder separat über standardisierte Schnittstellen wie bspw. PCIe angebunden werden (siehe Abbildung 2). KI-Beschleuniger können in drei Architekturen eingeteilt werden: GPU (Graphics Processing Unit), FPGA (Field Programmable Gate Array) und ASIC (Application Specific Integrated Circuit), wie in Abbildung 4 dargestellt.

Abbildung 4: Übersicht zu Rechenarchitekturen für KI-Ausführung. Von links nach rechts: Intel Core i7 8700K, Nvidia RTX A2000, Xilinx Zynq UltraScale+ Evaluationsboard, Hailo-8 m.2-Modul mit Kühlkörper. Quellen: Alexandru Bogdan/Unsplash, Fraunhofer IPA/Foto: Timo Leitritz.

CPU
Die CPU ist die klassische Allzweck-Recheneinheit in Computersystemen. Sie besteht aus einer geringen Anzahl an leistungsstarken Kernen, typischerweise 4 bis 16 Kerne, die universell programmierbar sind und komplexe Steuerlogik sowie serielle Aufgaben effizient ausführen können. Über SIMD-Erweiterungen wie Intel AVX oder Arm NEON sind auch Vektoroperationen beschleunigt möglich, was einfache Parallelisierung erlaubt. Für große neuronale Netze ist die Parallelität jedoch nicht ausreichend bzw. ineffizient, sodass CPUs vor allem für kleinere Modelle, Kontrollaufgaben und Vor- oder Nachverarbeitungsschritte genutzt werden.

GPU
Die GPU wurde ursprünglich für Grafikberechnungen entwickelt und besitzt daher eine Architektur mit tausenden vergleichsweise einfachen Rechenkernen. Sie ist auf massiv parallele Verarbeitung ausgelegt, insbesondere für Vektor- und Matrixoperationen. Damit eignet sie sich ideal für das Training großer neuronaler Netze, bei denen viele ähnliche Operationen gleichzeitig ausgeführt werden müssen. GPUs sind heute der Standard in Forschung und Praxis für Deep Learning, da ihre Architektur den hohen Rechen- und Speicheranforderungen dieser Modelle besonders gut entspricht. Die Leistungsaufnahme ist allerdings hoch.

FPGA
Ein FPGA (Field Programmable Gate Array) ist ein integrierter Schaltkreis, dessen Logikblöcke und Verbindungen nach der Fertigung frei rekonfigurierbar sind. Dadurch lassen sich Datenpfade exakt an eine gewünschte Berechnung anpassen, beispielsweise für spezialisierte neuronale Netze. Die Architektur verteilt Speicher und Rechenlogik eng miteinander, was niedrige Latenzen und eine effiziente Datenverarbeitung ermöglicht. Häufig wird dabei mit reduzierter Zahlendarstellung wie INT8 gearbeitet, um Speicherbedarf und Rechenaufwand zu reduzieren. FPGAs eignen sich deshalb gut für eingebettete Systeme oder Anwendungen mit strengen Energie- und Latenzanforderungen.

ASIC
Ein ASIC (Application Specific Integrated Circuit) ist ein spezieller Chip, der ausschließlich für einen bestimmten Anwendungsfall gefertigt wird. Im Bereich der KI bedeutet das, dass sämtliche Logik auf die Ausführung neuronaler Netze optimiert ist. Dadurch erreicht ein ASIC eine

besonders hohe Rechendichte und Energieeffizienz. Beispiele sind Googles Tensor Processing Units (TPUs) oder Hailos Hailo-8. ASICs bieten eine hohe Performance bei geringer Leistungsaufnahme, haben aber teilweise Einschränkungen bei der Kompatibilität mit bestimmten Schichten. Weitere typische Bezeichnungen für KI-ASICs sind: NPU (Neural Processing Unit), IPU (Intelligence Processing Unit), DLA (Deep Learning Accelerator).

Kombinationen
In modernen Systemen werden diese Architekturen oft kombiniert, um unterschiedliche Aufgaben optimal abzudecken. Ein Beispiel sind SoCs (System-on-Chip), die CPU-Kerne für allgemeine Aufgaben, GPU-Kerne für KI- bzw. generische parallele Berechnungen und spezialisierte Tensor-Cores als ASIC-Einheiten für reine Deep-Learning-Operationen integrieren. Dadurch lässt sich eine breite Palette an Rechenanforderungen effizient in einem System abbilden.

2.3.2. Einsatzgebiete

KI-Beschleuniger können je nach ihren Spezifikationen und Anwendungsfällen in verschiedene Einsatzgebiete eingeteilt werden. Es werden zu jedem Einsatzgebiet zwei typische Bezeichnungen genannt:

- **Datacenter/Server:** In Datacentern werden hunderte bis tausende extrem leistungsfähige KI-Beschleuniger in Server-Clustern verschaltet, um massive Modelle zu trainieren und zu betreiben. KI-Beschleuniger in diesem Einsatzgebiet stellen sehr viel Rechenleistung, Speicher und Bandbreite bereit. Die hohe Leistungsaufnahme muss durch aufwendige Kühlsysteme gehandhabt werden.

- **Workstation/Professional:** Für die Entwicklung werden typischerweise Workstations zum Testen und Trainieren von KI-Modellen genutzt. Eine Workstation setzt dafür meist ein bis vier professionelle GPUs oder vergleichbare KI-Beschleuniger ein.

- **Consumer/Client:** Im Consumer-Bereich (alternativ: Client-Bereich) existiert eine Vielzahl an unterschiedlichen Geräten mit integrierten und dedizierten KI-Beschleunigern. So integrieren nahezu alle modernen Smartphone-, Tablet und Laptop-SoCs KI-ASICs, meist NPU genannt. Außerdem existieren hier bereits zahlreiche KI-Applikationen, wie das Editieren und Generieren von Bildern auf dem jeweiligen Gerät. Auch Consumer-Desktop-Rechner beinhalten meist KI-Beschleuniger in Form einer integrierten oder dedizierten GPU.

- **Edge/Embedded:** Edge-Beschleuniger (alternativ: Embedded-Bereich) sind für Anwendungen wie Robotik oder autonomes Fahren konzipiert, bei denen kurze Latenzen, hohe Datenraten und energieeffiziente Inferenz entscheidend sind. Sie kombinieren spezialisierte Hardware und angepasste Software, erreichen hohe Rechenperformance bei nur wenigen Watt Leistungsaufnahme und können gleichzeitig mehrere Modelle ausführen. Durch Techniken wie Quantisierung ermöglichen sie trotz begrenzter Leistung hohe Frame-Raten und effiziente Verarbeitung direkt vor Ort bzw. »on the edge«.

- **TinyML/IoT:** Im TinyML-Bereich (alternativ: IoT-Bereich, Internet of Things) wird die Inferenz auf Prozessoren mit sehr niedriger Leistungsaufnahme ausgeführt. Hardwareseitig dominieren hier Mikrocontroller-basierte SoCs mit zusätzlichen DSP-/Vektor-Einheiten oder KI-ASICs, die im Milli-Watt-Bereich arbeiten und mit wenig KB- bis MB-Speicher auskommen.
  Im TinyML-Bereich lässt sich außerdem ein interessanter Trend beobachten: Es werden zusätzlich zu den herkömmlichen ARM- oder Xtensa-Kernen auch RISC-V-Kerne verbaut. Diese RISC-V-Kerne können aufgrund der Offenheit des Standards stark für KI-Anwendungen angepasst werden und sind bereits für ihre extrem ressourcenschonende Programmausführung bekannt.

2.3.3. Formate

KI-Beschleuniger liegen in diversen Formfaktoren vor, die sich in Übertragungsgeschwindigkeit, Kompaktheit und Kompatibilität stark unterscheiden:

- **PCIe:** PCIe-Steckkarten (Peripheral Component Interconnect Express) werden in Workstations, Data-Center und manchen Edge-Beschleunigern eingesetzt. PCIe-Anbindungen bieten sehr hohe Transferraten und eine direkte Anbindung an den Host über PCIe-Lanes (x1/x4/x8/x16). Die Bandbreite hängt von Generation und Lane Anzahl ab: Gen3 x16 ≈ 16 GB/s, Gen4 x16 ≈ 32 GB/s, Gen5 x16 ≈ 63 GB/s pro Richtung; x8 entsprechend halbiert.

- **M.2-Module:** M.2 ist ein kompakter Formfaktor, mit dem auch manche Embedded Systeme ausgestattet sind. Die Datenanbindung erfolgt meist über PCIe; je nach Keying stehen unterschiedlich viele Lanes zur Verfügung (M Key bis x4, B Key bis x2, E-/A,(A+E)-Key x1 bis x2). Daraus ergeben sich typische Nutzbandbreiten von ca. Gen3 x4 ≈ 4 GB/s und Gen4 x4 ≈ 8 GB/s pro Richtung (x2 entsprechend halbiert). M.2 ist bei modernen NVME-SSDs und WIFI-Karten sehr verbreitet, wird aber auch für KI-Beschleuniger eingesetzt.

- **USB:** KI-Beschleuniger mit USB-Anbindung eignen sich für kostengünstige Inferenz mit Plug-and-Play-Charakter, werden aber im Vergleich zu PCIe durch die begrenzte Übertragungsrate von ca. 400-450 MB/s bei USB 3.2 Gen 1 limitiert. Neuere USB-Generationen erreichen zwar wesentlich höhere Übertragungsraten, es werden aber kaum noch KI-Beschleuniger in Form eines USB-Dongles angeboten.

- **Single-Board-Computer:** Single-Board-Computer (SBC) integrieren alle Komponenten eines herkömmlichen Desktop-Rechners (CPU, GPU, RAM, HDMI, USB…) in einem sehr kompakten Format, wenig größer als eine Kreditkarte. Ein sehr bekanntes Beispiel für ein SBC ist der Raspberry Pi. Für KI spezialisierte SBCs integrieren zusätzlich KI-ASICs oder vergleichsweise starke GPUs. Ein Beispiel hierfür ist die Jetson-Serie von Nvidia.

- **SXM/OAM:** SXM/OAM Module für Rechenzentren sitzen direkt auf Trägerboards und erlauben hohe Leistungsaufnahmen sowie schnelle Direktverbindungen zwischen Beschleunigern. Beispiel: Der Nvidia DGX H100 Server nutzt acht H100-GPUs im SXM5-Format, die über NVLink/NVSwitch verbunden sind und pro GPU bis zu 900 GB/s Peer-to-Peer-Bandbreite bereitstellen. Insgesamt besitzt das System eine TDP von max. 10,2 kW. Ergänzend standardisiert das Open Compute Project (OCP) mit dem Open Accelerator Module (OAM) offene mechanische, elektrische und thermische Schnittstellen: Ein Universal Baseboad (UBB) kann bis zu acht OAM-Module aufnehmen und ermöglicht hohe Leistungsbudgets pro Modul und direkte Hochgeschwindigkeitsverbindungen.

2.4. KI-Software

KI Modelle benötigen zusätzlich zu passender Hardware einen umfangreichen Software-Stack, um effizient ausgeführt zu werden. Generische Hardware wie CPUs und GPUs ermöglichen den Einsatz unterschiedlicher Software-Backends, während spezialisierte ASICs wie Hailo-8 nur mit spezialisierter, teilweise proprietärer Software funktionieren.

Die wichtigsten Frameworks für KI-Entwicklung und -Ausführung werden hier aufgelistet:

KI-Entwicklungs-Frameworks:

- Zweck: Erstellen, Trainieren und Testen von Modellen.
- Beispiele: PyTorch, TensorFlow, Jax, Keras, PaddlePaddle, MXNet uvm.

Optimierungs- und Inferenz Frameworks:

- Zweck: Optimieren und Ausführen von Modellen auf spezialisierter Hardware.
- Beispiele: Nvidia TensorRT, AMDs ROCm, ONNX Runtime + Microsoft Olive, Intel OpenVino, HailoRT, TFLite, ARM NN uvm.

Compute-Backends:

- Zweck: Standardisierte und optimierte Low-Level-Berechnungs-Bibliotheken für CPU und GPU.
- Beispiele: BLAS, Eigen, OpenCL, CUDA, ROCm.

Diese Inferenz-Frameworks in Kombination mit diversen Compute-Backends erlauben es unter anderem, Modelle, die in den KI-Entwicklungs-Frameworks erstellt und trainiert wurden, auf verschiedenen CPU, GPU, FPGA oder ASIC-Plattformen auszuführen. Dabei spielen hardware-orientierte Optimierungen wie Präzisionsreduktion und Layer-Fusion eine große Rolle für die Beschleunigung der Modelle, die im Folgenden näher beschrieben werden.

Präzisionsreduktion (Quantisierung) wandelt 32 Bit Fließkommawerte (=FP32) der Modell-parameter auf niedrigere Präzision um, etwa BF16 (16 bit), INT8 (8 bit) oder INT4 (4 bit). Das verringert Speicherbedarf und Rechenaufwand deutlich: Bei INT8 schrumpft die Modellgröße typischerweise um den Faktor 4 in Bezug auf die Original-Größe bei FP32, bei BF16 um den Faktor 2; Datenübertragungen laufen entsprechend schneller. BF16 ist ein 16 Bit Gleitkomma-format (1 bit Vorzeichen, 8 bit Exponent, 7 bit Mantisse) mit nahezu FP32 Dynamikumfang und wird häufig für Mixed Precision Training und Inferenz genutzt. Viele aktuelle Beschleuniger-Chips erlauben bzw. benötigen native INT8 Berechnungen. Man unterscheidet zwischen Post Training Quantization (PTQ) und Quantization Aware-Training (QAT), wobei QAT die Quantisie-rung bereits während des Trainings simuliert und so höhere Genauigkeit ermöglicht. [22]

Pruning/Sparsity entfernt überflüssige Verbindungen, einzelne Gewichte oder ganze Filter Kanäle aus einem neuronalen Netz. Zunächst wird das Modell vollständig trainiert, dann wird die Bedeutung einzelner Parameter (z. B. durch Magnitudenbewertung) bestimmt und die unwichtigen Gewichte entfernt, gefolgt von einer Feinabstimmung zur Wiederherstellung der Genauigkeit. Man unterscheidet unstrukturiertes Pruning (zerstreute Null Gewichte) und struk-turiertes Pruning (Entfernung ganzer Neuronen, Filter oder Schichten). Iteratives Pruning mit wiederholtem Entfernen und Finetuning führt meist zu besseren Ergebnissen als Einzelschnitte. Um die Vorteile von Pruning bzw. Sparsity tatsächlich nutzen zu können, benötigen Hardware Backends spezielle Unterstützung dafür. [22]

Graph-Optimierungen wie Operator-/Layer-Fusion fassen aufeinanderfolgende Operationen wie Convolution, BatchNorm und Activation in einem einzelnen Kernel zusammen. Dadurch sinkt die Anzahl an Speicherzugriffen und Kernel-Wechseln. Diese Technik bringt meist hohe Performancegewinne bei gleichbleibender Genauigkeit. Des Weiteren kann durch Entfernen redundanter Operationen, die Auswertung von Konstanten bereits vor der Laufzeit (Constant Folding) und Umordnung des Datenflusses für die Rechen-Kernel die Ausführung zusätzlich beschleunigt werden. [23] [24] [25]

Kernel- und Hardware-Tuning: Durch systematische Variation von Parametern wie Tile-Größen, Schleifenanordnung, Parallelisierung, Unrolling und Vektorisierung werden unterschiedliche Ausführungsstrategien direkt auf der Zielhardware getestet. Die schnellste Konfiguration wird automatisch gewählt, wodurch vorhandene Hardwarefunktionen – etwa CUDA-Kernel auf GPUs oder AVX512-Instruktionen auf CPUs – optimal genutzt werden. [24]

2.5. Industrielle Anforderungen an KI-Hardware und -Software

Für den industriellen Einsatz von KI-Hardware-Beschleunigern sind Anforderungen relevant, die weit über reine Rechenleistung hinausgehen.

In Industrieszenarien ist eine nahtlose Anbindung an etablierte Feldbus- und Industrial Ethernet Protokolle essenziell: PROFINET, EtherCAT, Ethernet/IP, Modbus TCP, OPC UA, PROFIBUS, CAN und viele weitere sind in industriellen Anlagen und Systemen verbreitet. Diese Kommunikationsstandards fordern entsprechende physische Schnittstellen und Software-Support der jeweiligen KI-Systeme.

Kompakte Bauformen erweisen sich als wesentlicher Vorteil für die Integration in bestehende industrielle Systeme, in denen Platzbeschränkungen häufig eine kritische Rolle spielen. Beispielsweise haben bestehende Industrie-PCs teilweise nur Platz für Einzelslot-PCIe-Erweiterungskarten. Ebenso ist der Platz in Schaltschränken limitiert. Eine geringe Wärmeabgabe mit einer Leistungsaufnahme $\ll$ 100 Watt kann ebenso entscheidend sein, da dadurch unter anderem auch passive Kühlung ohne Lüfter ermöglicht wird, was die Robustheit und Langlebigkeit weiter erhöht.

Des Weiteren wird in Bereichen der Automatisierungstechnik Echtzeitfähigkeit gefordert. Das KI-System aus Software und Hardware muss deterministische Verarbeitungszeiten gewährleisten, die mit den strengen Timing-Anforderungen von speicherprogrammierbaren Steuerungen und anderen Echtzeitsystemen kompatibel sind. Diese harten Echtzeitanforderungen sind nur umzusetzen, indem ein Echtzeit-Betriebssystem auf dem Host-System eingesetzt wird und der KI-Beschleuniger passend integriert und priorisiert ist.

Zertifizierungen für Schutz vor Vibration, Spritzwasser, Staubschutz und Co. stellen weitere wesentliche Kriterien in raueren Industrieumgebungen dar. Ebenso ist fehlerkorrigierender Speicher (ECC = Error Correcting Code) wichtig für eine robuste Ausführung der Programme.

Die langfristige Verfügbarkeit der Hardware und deren Ersatzteile ist essenziell für den Einsatz in Maschinen und Anlagen. Die Verfügbarkeit sollte sich dabei mindestens auf einen Zeitraum von zehn Jahren belaufen, da industrielle Anwender auf planbare Ersatzteil- und Wartungszyklen angewiesen sind.

Für den industriellen Einsatz ist es nicht ausreichend, dass ein KI-System lediglich korrekte Ergebnisse liefert. In vielen Anwendungen, etwa bei sicherheitskritischen Prozessen oder der Qualitätskontrolle, muss nachvollziehbar sein, wie ein Ergebnis zustande gekommen ist. Dies erfordert Modelle und Toolchains, die Methoden der erklärbaren KI (Explainable AI, XAI) unterstützen und sich in bestehende Engineering- und Monitoring-Prozesse integrieren lassen. Transparenz in Bezug auf Modellarchitektur, Trainingsdaten und Entscheidungslogik erleichtert Audits, Zertifizierungen und die Fehleranalyse im laufenden Betrieb. Nähere Informationen zu XAI können aus der Studie »Erklärbare KI in der Praxis«[1] des KI-Fortschrittszentrums entnommen werden.

Industrieanlagen sind zunehmend über Netzwerke und das Industrial Internet of Things verbunden, wodurch KI-Systeme potenziell neue Angriffsflächen darstellen. Hardware und Software müssen daher nach anerkannten Sicherheitsstandards entwickelt und getestet sein (z. B. IEC 62443[2]). Dazu gehören sichere Boot-Mechanismen, verschlüsselte Kommunikation, Authentifizierung und Zugriffskontrolle. Auch Updates und Patches müssen über langfristig gepflegte und abgesicherte Kanäle bereitgestellt werden, um Manipulationen und Betriebsunterbrechungen zu verhindern.

KI-Hardware und -Software-Lieferanten reagieren auf diese Anforderungen mit entsprechenden Industrievarianten ihrer Produkte bzw. setzen industrielle Produkte über Hardware-Integratoren um, die die Chips und Module in robuste Carrier-Boards einbauen.

[1] https://www.ki-fortschrittszentrum.de/veroeffentlichungen/erklaerbare-ki-in-der-praxis/
[2] https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

3. Marktübersicht KI-Hardware

Der Markt für KI-Beschleuniger ist sehr divers und volatil. So haben Startups wie Flex Logix und Perceive 2020/2021 KI-Beschleuniger angekündigt und teilweise in begrenzter Stückzahl herausgebracht, wurden aber kurze Zeit später von Analog Devices bzw. Amazon aufgekauft und haben seitdem keine Beschleuniger veröffentlicht. Weitere Startups sind nach initialem Marketing und Investments nicht aus der Prototypenphase herausgekommen. Dennoch haben sich einige Hersteller, allen voran Nvidia als Quasi-Monopolisten, etabliert. Die folgenden Tabellen geben einen Überblick zu verfügbarer Hardware mit KI-Fokus, gruppiert nach Einsatzgebiet. Die Liste hat keinen Anspruch auf Vollständigkeit, deckt aber die wichtigsten Hersteller bzw. Beschleuniger ab. Es werden hauptsächlich die neuesten Generationen der jeweiligen Hardware aufgelistet und davon stellvertretende Varianten. Zur besseren Einordnung werden die jeweiligen Prozessortypen und verfügbaren Formate angegeben.

Tabelle 1: KI-Hardware mit Einsatzgebiet Consumer. Preisbereich zwischen 200 und 2500 Euro. Leistungsaufnahme zwischen 50 und 600 Watt.

Hersteller	Bezeichnung	Format	Prozessortypen
AMD	Ryzen AI 300 Series	SBC, Chip	CPU, GPU, ASIC
AMD	Radeon RX 9070 XT	PCIe	GPU, ASIC
Intel	Core Ultra 200 Series	SBC, Chip	CPU, GPU, ASIC
Intel	ARC B580	PCIe	CPU, GPU, ASIC
Nvidia	GeForce RTX 5090	PCIe	GPU, ASIC

Tabelle 1 führt aktuelle CPUs und GPUs aus dem Consumer-Bereich auf. Die GPUs besitzen typischerweise PCIe x16 Anschlüsse und nehmen 70 bis 600 Watt an Leistung auf. Moderne GPUs integrieren zusätzlich zur klassischen Grafik-Einheit auch dedizierte KI-Rechenkerne, die eine Form von ASICs darstellen. Insbesondere Nvidia's GeForce Reihe aus dem Gaming-Bereich wird von Forschungsinstituten und Privatnutzern aufgrund des vergleichsweise guten Preis-Leistungs-Verhältnisses für die KI-Entwicklung umfangreich genutzt. Die Nutzung dieser GPUs für kommerzielle Zwecke in der Industrie ist von Nvidia untersagt (siehe Nvidia's EULA[1]). Des Weiteren integrieren mittlerweile nicht nur Laptop-CPUs (bspw. AMD Ryzen AI 300), sondern auch Desktop-CPUs (Intel Core Ultra) KI-Beschleuniger in einem Bereich von 13-50 TOPS. Diese CPUs könnten mittelfristig in Industrie-PCs angeboten werden. Aktuell gibt es bereits Angebote für industrielle Mini-PCs von Aaeon[2] und OnLogic[3] mit Intel und AMD CPUs mit integrierten KI-Beschleunigern.

[1] https://www.nvidia.com/content/DriverDownloads/licence.php?lang=us&type=GeForce
[2] https://www.aaeon.com/en/product/detail/up-system-intel-core-ultra-meteor-lake-up-xtreme-i14-edge
[3] https://www.onlogic.com/store/ml100g-56/

Tabelle 2: KI-Hardware mit Einsatzgebiet Workstation. Preisbereich zwischen 500 und 10 000 Euro. Leistungsaufnahme zwischen 200 und 600 Watt.

Hersteller	Bezeichnung	Format	Prozessortypen
AMD	AMD Radeon AI PRO R9700	PCIe	GPU, ASIC
Intel	Arc Pro B60	PCIe	GPU, ASIC
Nvidia	RTX PRO 4000 Blackwell SFF Edition	PCIe	GPU, ASIC
Nvidia	RTX PRO 6000 Blackwell Workstation-Edition	PCIe	GPU, ASIC
Tenstorrent	Blackhole p150a	PCIe	ASIC

Die GPUs und KI-Beschleuniger aus Tabelle 2 lassen sich dem Workstation-Bereich zuschreiben und sind alle im PCIe-Kartenformat. Die Architektur und Compute-Fähigkeiten der GPUs (Nvidia RTX PRO 6000 Blackwell, Intel ARC Pro B60) sind identisch mit denen der Consumer-Modelle, allerdings meist mit signifikant höherem GPU-Speicher (bspw. 32 GB gegen 96 GB für RTX 5090 gegen RTX PRO 6000 Blackwell). Besonders hervorzuheben ist die RTX PRO 4000 Blackwell SFF Edition GPU, da sie sehr kompakt ist und unter 75 Watt Leistungsaufnahme besitzt, was bedeutet, dass sie rein durch den PCIe-Anschluss mit Strom versorgt werden kann und keine zusätzlichen Kabel benötigt. Dies kann unter Umständen eine Integration in größere Industrie-PCs mit ausreichend Kühlung und PCIe-Erweiterungen ermöglichen. Als reiner KI-Beschleuniger bzw. ASIC ohne Grafikfunktionalität sticht der Tenstorrent Blackhole p150a aus der Liste heraus. Dieser Beschleuniger wurde von Grund auf vom Startup Tenstorrent für KI entwickelt.

Tabelle 3: KI-Hardware mit Einsatzgebiet Edge. Preisbereich zwischen 50 und 3500 Euro. Leistungsaufnahme zwischen 1 und 100 Watt.

Hersteller	Bezeichnung	Format	Prozessortypen
AMD	AMD Versal™ AI Core Series	Chip	CPU, GPU, FPGA
AMD	Kria K26	SBC, SoM	CPU, GPU, FPGA
Axelera	Metis AIPU	m.2, PCIe, Chip	ASIC
Basler	microEnable 5 marathon deepVCL (Frame Grabber)	PCIe	FPGA
Blaize	Blaize P1600	m.2, PCIe, SBC, SoM	CPU, ASIC
Google	Coral TPU	USB, m.2, PCIe, SBC, SoM, Chip	ASIC
Hailo	Hailo-10H	m.2, PCIe, SBC, Chip	ASIC
Hailo	Hailo-8	m.2, PCIe, SBC, Chip	ASIC
IDS	NXT Malibu	Kamera	CPU, ASIC
Intel	Mustang-F100-A10	PCIe	FPGA
kinara	ARA-2	USB, m.2, PCIe	ASIC
Luxonis	OAK 4 SoM (including Robotics Vision Core 4)	SoM, Kamera	CPU, GPU, ASIC
Mythic	M1076	m.2, Chip	ASIC
Nvidia	Jetson AGX Thor	SBC, SoM	CPU, GPU, ASIC
Nvidia	Jetson AGX Orin Industrial	SBC, SoM	CPU, GPU, ASIC
Nvidia	Jetson AGX Orin 32GB/64GB	SBC, SoM	CPU, GPU, ASIC
Nvidia	Jetson Nano Super	SBC, SoM	CPU, GPU, ASIC
Nvidia	Jetson Orin NX 8GB/16GB	SBC, SoM	CPU, GPU, ASIC
NXP	i.MX 9	SBC, SoM, Chip	CPU, GPU, ASIC
Qualcomm	QCS6490	SBC, Chip	CPU, GPU, ASIC
Rockchip	RK3588S2	SBC, Chip	CPU, GPU, ASIC
SiMa Technologies	MLSoC Gen 2 Modalix	SoM, Chip	CPU, ASIC
SiMa Technologies	MLSoC Gen 1	PCIe, SBC, Chip	CPU, ASIC

Im Edge-Bereich sind hauptsächlich ASICs oder Kombinationen mit ASICs verfügbar, da hier Energieeffizienz eine wesentlich größere Rolle spielt. Herauszustellen sind hier die Lösungen von Nvidia in Form der Jetson Serie, von Hailo in Form der Hailo-8/Hailo-10 Serie und von SiMa Technologies in Form der MLSoC Serie. Diese Plattformen bieten ein umfangreiches Software-Ökosystem, welches das Beschleunigen unterschiedlichster Modelle ermöglicht. Die Jetson-Module bzw. Jetson-SBCs sind vollständige Rechner mit ARM-CPU, CUDA-fähiger GPU und zusätzlichen ASIC KI-Rechenkernen (Tensor-Cores). Die Kompatibilität mit CUDA ermöglich ein sehr einfaches Deployment mit GPU-Beschleunigung von den meisten Torch-/TensorFlow-Anwendungen, was die Jetson-Hardware besonders attraktiv macht, auch wenn reine ASICs wie Hailo-8 eine bessere Effizienz erzielen. Dabei fällt das Quantisieren der Modelle in INT8 weg, das für Hailo-8 und Co. notwendig ist. Manche Jetson-Module gibt es zudem in Ausführungen, die explizit für die Industrie entwickelt wurden, bspw. in Form des Jetson AGX Orin Industrial[1]. Als Betriebssystem wird auf Jetson-Geräten eine angepasste Version von Ubuntu genutzt, die neueste Iteration ist Jetpack 7, welche nun auch Echtzeit-Support verspricht. Sowohl Nvidias Jetson-Reihe als auch Hailos Beschleuniger werden nicht nur als Development Kits angeboten, sondern auch von Drittanbietern und Integratoren in entsprechende Carrier-Boards mit variierenden Spezifikationen, Kühlung und Schnittstellen integriert. So bieten bspw. Hersteller wie Seeed, Aaeon, Advantech und Co. passiv gekühlte, robuste Jetson-Systeme. Für den Hailo-8 gibt es bspw. von Berghof Automation eine Raspberry Pi-basierte SPS »DC-Pi-Prime« mit Hailo-Chip[2], während Aaeon auf eine Intel-X86-Lösung mit integriertem Hailo-Chip setzt in Form des »UP SQUARED PRO 710H EDGE«[3].

SiMa Technologies bietet einen Beschleuniger an, der eine ARM-CPU mit einem performanten ASIC, einem Vision-Co-Prozessor und Video-Encoder/-Decodern kombiniert, ähnlich einem Nvidia Jetson. In der zweiten Generation wird der MLSoC zusätzlich im System-on-Module-Format angeboten, ebenfalls angelehnt an Nvidias Jetson Module. Im Gegensatz zum MLSoC Gen 1 von SiMa unterstützt die Gen 2 die effiziente Ausführung von LLMs durch signifikant größeren Speicher und zusätzliche Genauigkeitsformate in Form von BF16. Ähnlich verhält es sich beim Hailo-10H, der im Gegensatz zum ursprünglichen Hailo-8 auch für kompakte LLMs, dank des wesentlich größeren Speichers und Unterstützung von INT4 Precision von 8 GB, geeignet ist.

Herauszustellen sind außerdem Lösungen, die ASICs direkt in Kameras integrieren, wie die IDS NXT Malibu oder die OAK-Serie von Luxonis. Diese Integration ermöglicht es, Kameradaten direkt auf der Kamera zu verarbeiten und das Endergebnis auszugeben. Zudem kann dadurch Bauraum eingespart werden und eine geringere Leistungsaufnahme erzielt werden. Das Deployment von Modellen kann auf dieser Art von Hardware aber aufwendiger und fehleranfälliger sein.

Hardwarelösungen von NXP und Rockchip setzen auf ARM CPUs und deren integrierte KI ASICs. FPGAs wie AMDs Kria Plattform oder Intels Mustang Beschleuniger bieten eine eingeschränkte Unterstützung und sind aktuell nicht zu empfehlen.

[1] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/
[2] https://www.berghof-automation.com/de/loesungen/automatisierungstechnik/displaysteuerungen#tabpanel-5219
[3] https://www.aaeon.com/en/product/detail/up-systems-up-squared-pro-710h-edge

Tabelle 4: KI-Hardware mit Einsatzgebiet TinyML. Preise < 20 Euro. Leistungsaufnahme < 1 Watt.

Hersteller	Bezeichnung	Format	Prozessortypen
Ambient Scientific	GPX-10	Chip	CPU, ASIC
brainchip	Akida 2	Chip	CPU, ASIC
Espressif	ESP32-S3	Chip	CPU
Renesas Electronics	RZ/V2L	Chip	CPU, GPU, ASIC
Silicon Labs	EFR32BG24 Series 2	Chip	CPU, ASIC
STMicroelectronics	STM32N6	Chip	CPU, ASIC
Syntiant	NDP200	Chip	CPU, ASIC

Im TinyML-Bereich sind niedrige Leistungsaufnahmen von <2 Watt typisch, daher ist die KI-Performance der Chips bzw. Mikrocontroller aus Tabelle 4 signifikant niedriger als die der vorherigen Kategorien. Dennoch sind diese Mikrocontroller fähig, kleine Modelle mit ausreichender Geschwindigkeit auszuführen. Als Beispiel sei der ESP32-S3 genannt, auf dem dank Unterstützung für Vektor-Recheninstruktionen sehr kleine Objektdetektor-Modelle basierend auf YOLOv11 mit ca. zwei bis zehn FPS in Kombination mit einer Kamera lauffähig sind[1]. Im Gegensatz zu reinen Mikrocontrollern wie dem ESP32-S3 bieten Ambient Scientific und STMicroelectronics bspw. zusätzliche integrierte ASIC-Beschleuniger in ihren Chips an, die eine wesentlich effizientere Ausführung von KI ermöglichen.

Tabelle 5: KI-Hardware mit Einsatzgebiet Server. Preisbereich 1000 Euro (einzelne Beschleuniger) bis 1 Mio. Euro (vollständige Server-Racks) und mehr. Leistungsaufnahme zwischen 150 und 1000 Watt für einzelne Beschleuniger und >30 kW für Server-Racks.

Hersteller	Bezeichnung	Format	Prozessortypen
Achronix	Speedster7t AC7t6000	PCIe, SoM, Chip	FPGA
AMD	MI350X / MI355X	Server-Rack	GPU
Cerebras	CS-3	Server-Rack	ASIC
Graphcore	Bow-2000	Server-Rack	ASIC
Intel	Gaudi 3	Server-Rack	ASIC
Nvidia	RTX PRO 6000 Blackwell Server-Edition	PCIe	GPU, ASIC
Nvidia	B200	PCIe, Server-Rack	GPU, ASIC
Nvidia	GB200 NVL72	Server-Rack	CPU, GPU, ASIC
Qualcomm	Cloud AI 100 Ultra	PCIe	ASIC
Tenstorrent	Blackhole p150b	PCIe	ASIC

Die Kategorie Server fokussiert sich auf hohe Speichermengen, Bandbreite und Rechenleistung. State-of-the-Art Server GPUs von Nvidia besitzen eine Leistungsaufnahme von 600 Watt und mehr, wie bspw. die RTX PRO 6000 Blackwell Server-Edition, die zu Preisen ab ca. 8000 Euro verfügbar ist. Hardware aus Tabelle 5 ist für besonders große Modelle bzw. LLMs wie DeepSeek-R1 [4] und Llama 4[2] geeignet. Oft werden dabei mehrere Beschleuniger zusammengeschaltet, wie in Nvidias DGX-Lösungen oder AMDs MI-Serie. Ein solcher Server kann je nach Modell und Hardware-Ausstattung eine Vielzahl von Inferenz-Anfragen gleichzeitig beantworten und je nach Software-Plattform (bspw. Triton Inference Server[3]) Anfragen intelligent bündeln und abarbeiten. Abhängig von der Anforderung kann es auch vorteilhaft sein, einen

[1] https://github.com/espressif/esp-detection
[2] https://www.llama.com/models/llama-4/
[3] https://github.com/triton-inference-server/server

zentralen Server auf dem Shopfloor oder im Serverraum einer industriellen Anlage zu platzieren und über das Netzwerk Inferenz-Anfragen an diesen zu stellen. Eine kombinierte Architektur aus KI-Edge-Devices und KI-Servern ist ebenso denkbar, um Teile des KI-Systems auf beiden Gerätetypen auszuführen. Hierbei sind aber die zusätzliche Latenz durch die Netzwerkkommunikation und mögliche Probleme der Zuverlässigkeit der Verbindung zu beachten.

Zusammenfassend betrachtet gibt es mittlerweile für nahezu jegliche Bauraumgröße und Leistungsaufnahmeklasse Lösungen mit KI-Beschleunigern. Dabei ist zu beachten, dass KI-Leistungsangaben wie TOPS nicht immer eine vergleichbare Aussage über die Performance der Hardware liefern. TOPS werden dabei oft in Bezug auf bestimmte Präzisionen und Sparsity benannt und sind ebenso oft eher »theoretisch« erreichbar. Es empfiehlt sich, die Zielanwendung auf diverser Hardware zu evaluieren oder konkrete, öffentliche Benchmarkergebnisse der entsprechenden Modelle zu vergleichen. Des Weiteren integrieren viele Prozessor-Hersteller KI-ASICs aufgrund der wachsenden Zahl an KI-Applikationen standardmäßig in ihre Produkte, wodurch zukünftige Industrie- und Consumer-PCs automatisch effizienter mit KI umgehen können.

4. Benchmarks

In diesem Kapitel erfolgt ein Vergleich von drei KI-Beschleunigern und reiner CPU-Ausführung von typischen Computer-Vision-Modellen hinsichtlich der in Abschnitt 4.1.1 genannten Performance-Metriken. Dabei wird insbesondere auf die Effizienz der KI-Hardware eingegangen.

4.1. Methoden

Für das Benchmarking der KI-Hardware wurde eine spezielle Software-Plattform entwickelt und entsprechende Benchmarking-Hardware aufgebaut. Zudem wurden passende Metriken und Modelle selektiert, um einen möglichst detaillierten und repräsentativen Vergleich zu ermöglichen.

4.1.1. Metriken und Parameter

Zur Evaluation der KI-Hardware wurden folgende Metriken herangezogen:

- **Latenz:** Vergangene Zeit (in ms) von Anfrage der Inferenz bis zur Rückmeldung des Ergebnisses.
- **Durchsatz:** Anzahl erfolgreich verarbeiteter Einheiten pro Zeiteinheit, bspw. Iterationen pro Sekunde oder Bilder pro Sekunde (1/s).
- **Leistungsaufnahme:** Momentanleistung in Watt (W).
- **Energieeffizienz:** Durchsatz pro Leistungsaufnahme (1/s/W).
- **X-Auslastung:** Anteil aktiver Rechen-/I-/O-Zeit der Ressource X in Prozent.

Zusätzlich sind diese Parameter bzw. Kennzahlen relevant:

- **Batch-Größe (Batch Size, BS):** Anzahl der gestapelten Samples pro Inferenz. Bei Bildern bedeutet eine Batch-Größe von vier bspw., dass bei jeder Inferenz vier Bilder auf einmal als Eingabe dem Modell gegeben werden. Dies führt bei ausreichender Rechenkapazität des Beschleunigers zu einer besseren Auslastung und höherem Durchsatz bei geringem Overhead.

- **FLOPs (Floating Point Operations):** Diese Kennzahl beschreibt den Rechenaufwand pro Ausführung eines KI-Modells bei FP32-Genauigkeit. Da der Wert u.a. abhängig von der Eingabegröße ist (Angabe z. B. @224² = Zweidimensionales Bild mit Breite und Höhe von 224 Pixeln), werden beide Werte oft gemeinsam genannt.

- **TOPS (Tera Operations per Second):** TOPS beschreibt die Anzahl der Berechnungsoperationen pro Sekunde, die ein KI-System theoretisch ausführen kann, bei einer bestimmten Genauigkeit (bspw. INT8). In Kombination mit den FLOPs eines Modells kann so ein theoretischer Durchsatz auf gegebener Hardware ermittelt werden.

- **Anzahl der (Modell-)Parameter:** Anzahl der Parameter eines KI-Modells. Die Anzahl der Parameter bestimmt maßgeblich den Speicherbedarf (Anzahl Parameter × Präzision) in Bytes. Um eine bestimmte Batch-Größe eines Modells auszuführen, muss der Speicher des KI-Beschleunigers ausreichend groß sein.

4.1.2. Benchmark-Modelle

Tabelle 6 stellt eine Übersicht und einen direkten Vergleich der in diesem Benchmark verwendeten KI-Modelle dar. Mit dem Zeichen „≈" wird angedeutet, dass aufgrund von u. a. Klassenanzahl, Head/Decoder, Auflösung und Implementierung leichte Abweichungen vom Wert für das jeweilige Modell auftreten können.

Hinweis: Da der Hailo-8-KI-Beschleuniger speziell konvertiere Modelle benötigt, wurden die passenden Modelle aus dem Hailo Model Zoo[1] genutzt.

Tabelle 6: Überblick der verwendeten KI-Modelle.

Modell	Architektur	Aufgabe	Parameter (ca.)	FLOPs (ca.)	Quelle/Implementierung
ResNet-50	CNN, Residual (Bottleneck)	Klassifikation/Backbone	≈25.6M	≈4.1 GFLOPs @224²	[13] / https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html
SegFormer-B5	Hierarch. Transformer (MiT) + MLP-Decoder	Semantiksegmentierung	≈85M	≈170–220 GFLOPs @640²	[21] / https://github.com/NVlabs/SegFormer
YOLOv9c-MIT	One-Stage-Detektor (aggregierter CNN-Backbone/Neck, decoupled heads)	Objekterkennung	≈25,3M	≈102,1 FLOPs @640²	[26] / https://github.com/MultimediaTechLab/YOLO
DETR-R50	CNN-Backbone + Transformer Encoder-Decoder (Set-Prediction)	Objekterkennung	≈41M	≈86 GFLOPs @800²	[17] / https://github.com/facebookresearch/detr
ViT-B/16-224	Vision Transformer	Klassifikation/Backbone	≈86M	≈17.6 GFLOPs @224²	[14] / https://docs.pytorch.org/vision/main/models/generated/torchvision.models.vit_b_16.html

ResNet-50 ist ein tiefes CNN für die Bildklassifikation und -erkennung, eingeführt in [13] als Teil der »Residual Networks«. Es gilt als robuster Standard-Backbone in Forschung und Praxis, weil es eine gute Balance aus Genauigkeit, Rechenaufwand und Einfachheit bietet. Eine zentrale Idee sind sogenannte Residual-Verbindungen (»Skip Connections«): Ein Block lernt nicht direkt eine Abbildung H(x), sondern eine Residualfunktion F(x), die zur unveränderten Eingabe x addiert wird (y = F(x) + x).

[1] https://github.com/hailo-ai/hailo_model_zoo

Dadurch fließen Gradienten leichter durch viele Schichten, was das Vanishing-Gradient-Problem [27] mindert und das Training sehr tiefer Netze stabilisiert.

SegFormer-B5 ist die größte Variante der SegFormer-Familie von [21] für die semantische Segmentierung. Das Kernprinzip: Ein hierarchischer Transformer-Backbone (MiT, »Mix Transformer«) kombiniert mit einem leichtgewichtigen, rein MLP-basierten Decoder (»All-MLP«). So werden starke Multi-Scale-Repräsentationen mit geringem Decoder-Overhead vereint.

YOLOv9c MIT ist eine kompakte, echtzeitfähige Variante des YOLOv9 Objektdetektors, ausgerichtet auf ein ausgewogenes Verhältnis zwischen Genauigkeit, Geschwindigkeit und Ressourcenverbrauch. Kern der Architektur ist der in YOLOv9 eingeführte GELAN Backbone (Generalized Efficient Layer Aggregation Network), der effektive Merkmalsaggregation bei moderatem Rechenaufwand ermöglicht. Ergänzt wird dies durch den in YOLOv9 etablierten Trainingsansatz mit Programmable Gradient Information (PGI): Dabei wird der Gradientenfluss so gesteuert, dass Klassifikation und Lokalisation besser ausbalanciert werden, was die Konvergenz stabilisiert.

DETR-R50 (DEtection TRansformer) ist ein End to End Objektdetektor, der Objekterkennung direkt auf einem Transformer-Encoder/Decoder-Modell formuliert und dadurch klassische Bausteine wie Anker und Non Maximum Suppression (NMS) überflüssig macht. Das Modell kombiniert ein ResNet 50 Backbone (R50) zur Feature Extraktion mit einem Transformer Encoder Decoder. Der Encoder modelliert globale Abhängigkeiten über alle Bildpositionen, während der Decoder mit einer festen Menge an gelernten Objekt Queries auf die Encoder Repräsentation zugreift. Jede Query steht für eine potenzielle Instanz und erzeugt genau eine Vorhersage: eine Klassenverteilung und eine normalisierte Box zur Objektdetektion.

ViT-Base/16-224 ist das Vision-Transformer-Modell, das das Transformer-Prinzip aus der Sprachverarbeitung auf Bilder überträgt. Die Bezeichnung »Base/16-224« steht für eine Basis-Modellgröße, eine Patch-Kantenlänge von 16 Pixeln sowie eine Eingabeauflösung von 224×224 Pixeln. Ein Eingabebild wird dazu in ein 14×14-Gitter aus insgesamt 196 Patches zerlegt; jedes Patch wird linear zu einem 768-dimensionalen Token projiziert, ein lernbares Klassen-Token wird vorangestellt, und gelernte Position-Embeddings werden addiert. Anschließend werden die Tokens von 12 Transformer-Encoder-Blöcke mit LayerNorm, Multi-Head-Self-Attention und einem MLP mit etwa 3072 verborgenen Dimensionen sowie Residualverbindungen und GELU-Aktivierung verarbeitet.

4.1.3. Hardware und Aufbau

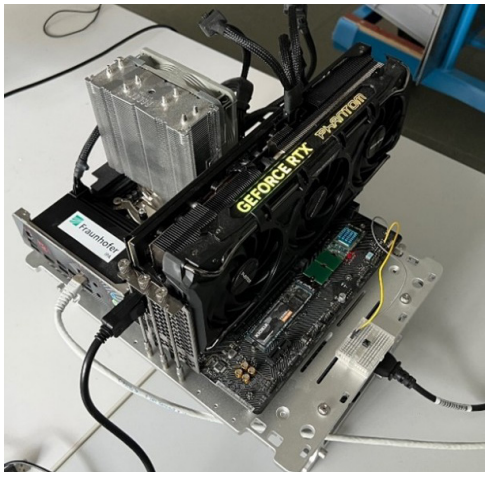


Abbildung 5: Benchmark-Rechner mit offenem Gehäuse und Intel Core i7-13700 CPU, Hailo-8 m.2 und Nvidia RTX 4090 sowie 64 GB DDR4-4800 Arbeitsspeicher. Quelle: Fraunhofer IPA/Foto: Timo Leitritz.

In dieser Studie wurde eine Desktop-Workstation (siehe Abbildung 5) mit Ubuntu 22.04.5 LTS und Linux-Kernel 6.8.0-78-generic auf x86-64 für Benchmarks eingesetzt. Die Hardware basiert auf einer ASRock Z790 PG Lightning Plattform und verfügt über einen Intel Core i7 13700 CPU. Eine Nvidia RTX 4090 24 GB ist per PCIe 4.0 x16 und ein Hailo-8-Modul per m.2-Slot angeschlossen. CUDA ist auf dieser Bench-Station in der Version 12.6 und cuDNN in Version 9.5.1 installiert. Die Softwareumgebung umfasst Python 3.10.18 sowie PyTorch 2.7.1+cu126.

Abbildung 6: Nvidia Jetson Orin NX 16 GB Industrial (Seeed Studio reComputer Industrial J4012), passiv gekühlt. Quelle: Fraunhofer IPA/Foto: Timo Leitritz.

In dieser Studie wurde außerdem ein Nvidia Jetson Orin NX 16 GB (siehe Abbildung 6) getestet. Das System ist passiv gekühlt und eignet sich damit für geräuschsensitive und staubige Umgebungen. Als Software-Stack wird JetPack 6.1 (L4T 36.4.0) verwendet, das den Nvidia Treiberstack inklusive CUDA, cuDNN, TensorRT beinhaltet. Der Leistungsmodus ist auf MAXN gesetzt, wodurch die verfügbaren Recheneinheiten mit maximalen Taktraten arbeiten können, um die Inferenzleistung zu maximieren. In der verwendeten virtuellen Python-Umgebung (.venv) läuft Python 3.10.12. Darin ist PyTorch 2.5.0a0+872d972e41.nv24.08 installiert, ein Nvidia angepasster Build (nv24.08), der von Source gebaut wurde. PyTorch verwendet hier CUDA 12.6 und cuDNN 9.3.0.

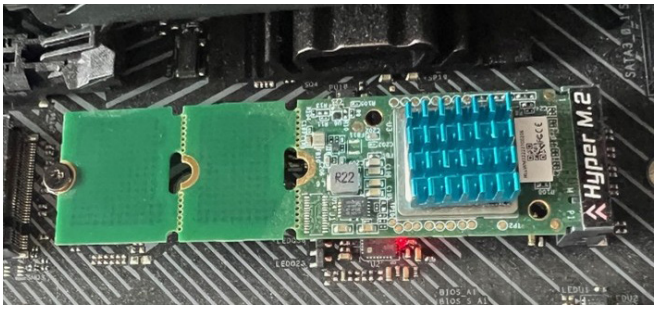


Abbildung 7: Hailo-8 m.2 Beschleunigerkarte mit passivem Kühlkörper. Quelle: Fraunhofer IPA/ Foto: Timo Leitritz.

Weiterhin wurde eine Hailo-8 M.2-Beschleunigerkarte mit passivem Kühlkörper eingesetzt (siehe Abbildung 7). Der Edge-AI-Beschleuniger ist auf energieeffiziente Inferenz optimiert und liefert eine theoretische Rechenleistung im zweistelligen TOPS-Bereich (max. 26 TOPS) bei einer Leistungsaufnahme von wenigen Watt. Die Anbindung erfolgt über den M.2-Steckplatz als PCIe-Gerät. Der passive Kühlkörper ermöglicht lüfterlosen Betrieb, setzt jedoch ausreichenden Gehäuse-Luftstrom bzw. eine geeignete Wärmeabfuhr voraus. Das Software-Ökosystem umfasst die HailoRT-Laufzeit, den Hailo Dataflow Compiler und Toolchains zur Modellkonvertierung (z. B. ONNX, Exporte aus TensorFlow/PyTorch).

4.1.4. Benchmarking-Software

Die Messungen werden mit einem eigens entwickelten Python-Skript durchgeführt. Für die präzise Zeitstempelung kommt die Python-Funktionalität perf_counter[1] zum Einsatz. Monitoring für Auslastung und Leistungsaufnahme läuft in einem separaten Prozess. Auf der Nvidia-Jetson-Plattform wird die Leistungsaufnahme über Tegrastats[2] erfasst. Auf der Bench-Station in Abbildung 5 erfolgte die Leistungsmessung mittels eines externen Strommessgeräts, das automatisiert ausgelesen wird.

Damit die Performancemessungen repräsentativ sind und das initiale Laden des Modells bei der ersten Inferenz umgangen wird, wird eine Aufwärmphase (engl.: Warmup) mit mehreren Inferenz-Schritten durchgeführt.

Nahezu alle Benchmarks wurden mit 20 Warmup-Iterationen und 200 Benchmark-Iterationen ausgeführt; eine Iteration ist dabei eine einzelne Ausführung des Modells, unabhängig von der Batch-Größe. Bei Benchmarks auf dem Intel i7 CPU für die Modelle SegFormer und YOLOv9c wurden fünf Warmup-Iterationen und 20 Benchmark-Iterationen durchgeführt. Hintergrund ist die extrem hohe Ausführungszeit für diese Modelle auf dieser Hardware.

[1] https://docs.python.org/3/library/time.html#time.perf_counter
[2] https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html

4.2. Ergebnisse

Die Ergebnisse der Experimente zur Performance, Effizienz und CPU-Bottlenecks werden in diesem Kapitel dargestellt. Zudem wird gezeigt, welche Performancesprünge durch spezialisierte Frameworks wie TensorRT und Präzisionsreduktion möglich sind. Es wird außerdem verglichen, wie sich dieselbe Hardware zwischen Linux und Windows verhält.

4.2.1. Latenz und Durchsatz

In diesem Abschnitt werden Latenz und Durchsatz der verschiedenen Modelle (siehe Abschnitt 4.1.2) und Hardware (siehe Abschnitt 4.1.3) untersucht. Batch-Größen wurden, falls für den individuellen Test angebracht, zwischen 1 und 32 variiert.

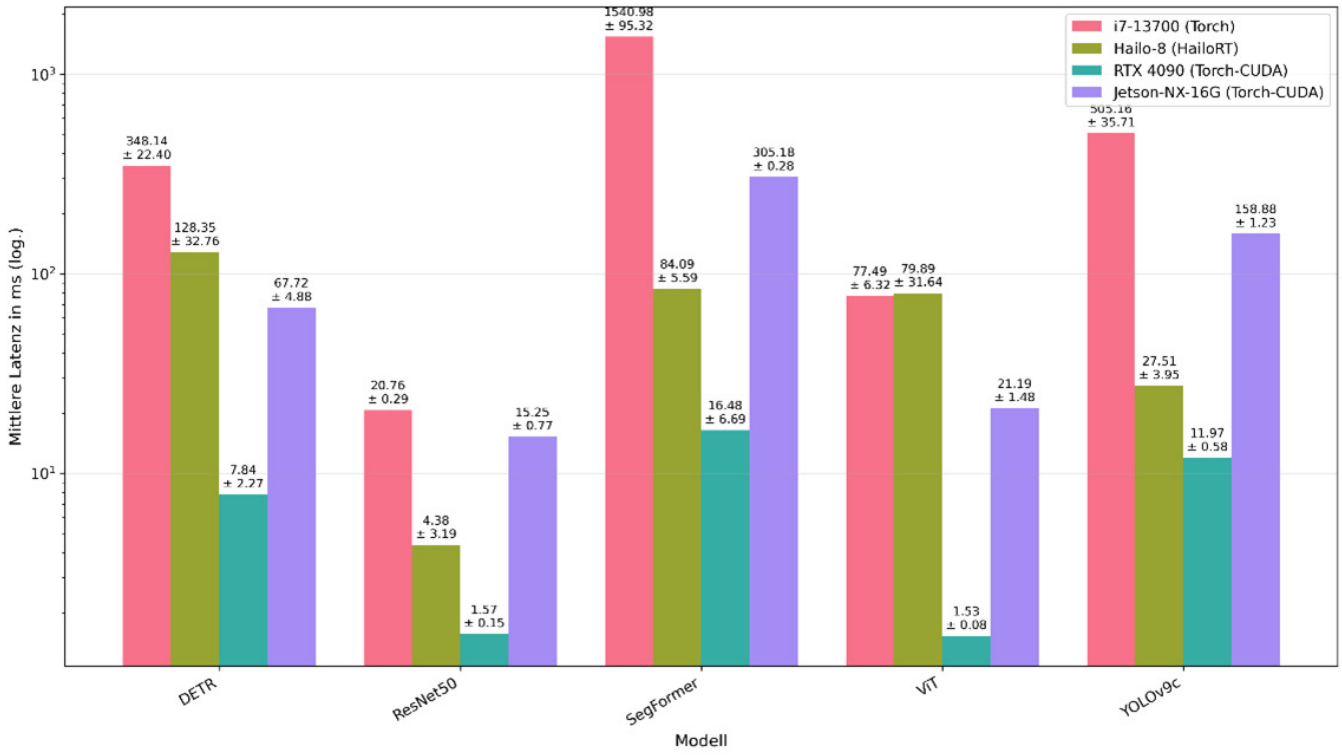


Abbildung 8: Mittlere Latenz und Standardabweichung mit Batch-Größe 1 auf allen Modellen und HW- bzw. SW-Plattformen. Gemessen wird ausschließlich die mittlere Latenz der Modellinferenz ohne Vor- und Nachbearbeitung. Niedrigere Werte sind besser.

Abbildung 8 zeigt eine Übersicht der mittleren Latenzen. Es ist zu erkennen, dass die RTX 4090 für alle Modelle mit großem Abstand am besten abschneidet, während die Ausführung auf der CPU ohne Beschleuniger nahezu durchgehend die höchste und damit schlechteste Latenz aufweist. Die Unterschiede zwischen den verschiedenen Modellen entsprechen grob den Rechenaufwänden (FLOPs) aus Tabelle 6, d. h. SegFormer ist bspw. signifikant aufwendiger zu berechnen als ResNet-50. Auffallend ist, dass die CPU beim ViT-Modell etwa gleich schnell ist wie der Hailo-8 Chip. Des Weiteren fällt auf, dass die Streuung beim Hailo-8 Chip für ViT, DETR und YOLOv9c im Vergleich zu den anderen Modellen und Hardware relativ hoch ist.

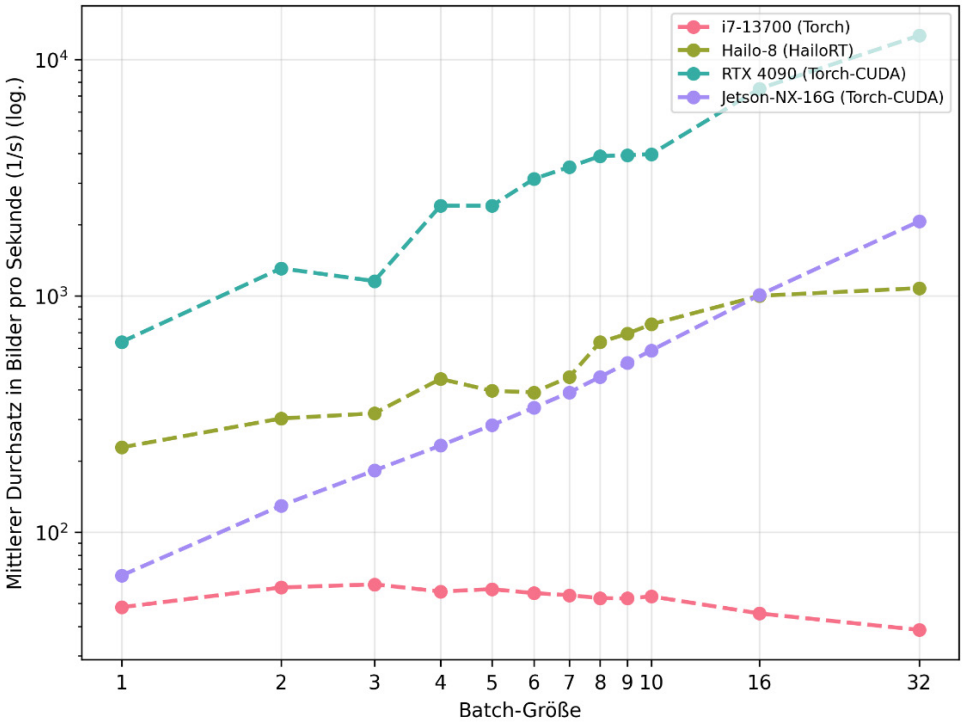


Abbildung 9: Mittlerer Durchsatz über diverse Batch-Größen für ResNet-50-Modell, ohne Vor- und Nachbearbeitung. Höhere Werte sind besser. Hinweis: Die horizontale Achse ist logarithmisch zur Basis 2 aufgetragen.

Abbildung 9 visualisiert, wie der Nvidia Jetson Orin NX 16 GB Industrial und der Benchmark-Rechner mit der RTX 4090 über die Batch-Größen 1 bis einschließlich 32 stetig mehr Durchsatz erzielen und damit bis einschließlich Batch-Größe 32 nicht voll ausgelastet werden. Ähnliches gilt für den Hailo-8, wenn auch mit weniger Steigung. Dies führt dazu, dass der Hailo-Chip vom Jetson bei Batch-Größe 16 eingeholt und ab dieser Größe übertroffen wird. Der i7-CPU skaliert nur minimal und fällt ab Batch-Größe 3 ab. Die Nvidia RTX 4090 erzielt erwartungsgemäß über alle Batch-Größen den mit Abstand höchsten Durchsatz.

Die Unterschiede in der Skalierung deuten auf unterschiedliche Kapazitäten der Beschleuniger hin. Der i7-CPU ist schon bei geringer Batch-Größe gesättigt und fällt danach aufgrund von Overhead ab (zu viele Berechnungen in der Pipeline, Stau). Außerdem wird klar, dass keine der Hardware bei Batch-Größe 1 voll ausgelastet ist. Dies führt auch zu einem recht moderaten Abstand der 4090 zu dem zweitbesten Beschleuniger (Hailo-8) bei Batch-Größe 1, trotz des enormen Unterschieds in der Nennleistung (ca. 5 Watt gegenüber bis zu 450 Watt). Der Unterschied vergrößert sich aber exponentiell bei steigender Batch-Größe, da die RTX 4090 besser ausgelastet wird.

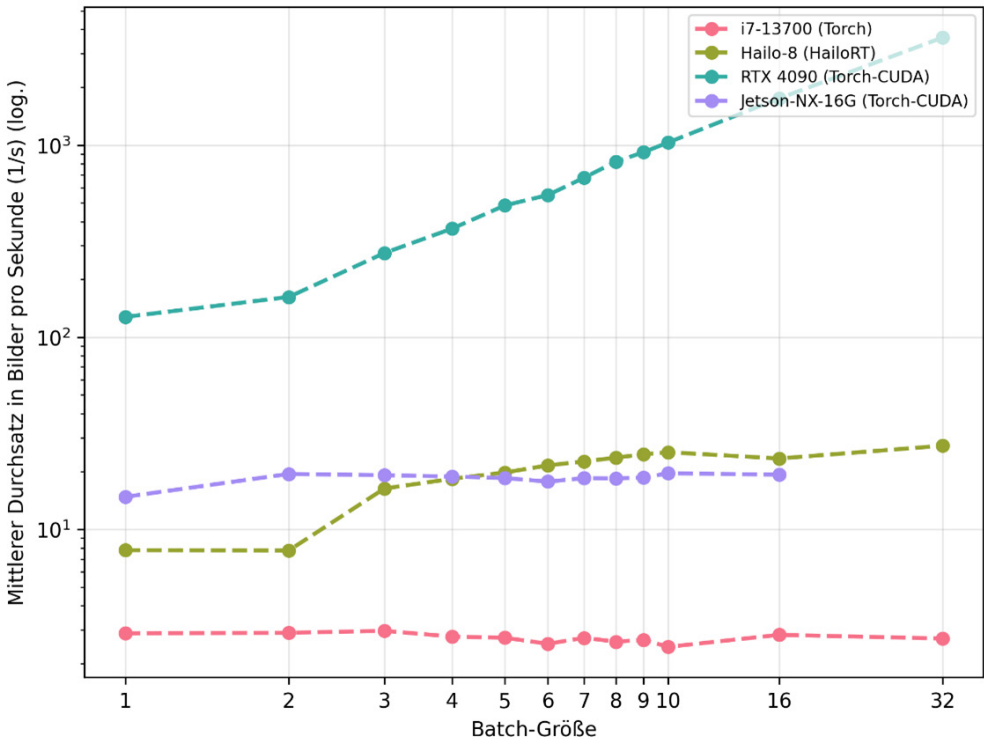


Abbildung 10: Mittlerer Durchsatz über diverse Batch-Größen für DETR-Modell, ohne Vor- und Nachbearbeitung. Der Jetson-NX-16G kann Batch-Größe 32 nicht ausführen, daher fehlt der Datenpunkt. Höhere Werte sind besser.

Ein ähnliches Bild zeigt auch Abbildung 10, allerdings sind die Verläufe hier flacher; so erreicht der Jetson den Durchsatz des Hailo-Chips bereits bei Batch-Größe 4 und der i7-CPU geht bereits bei Batch-Größe 1 in die Sättigung. Beim Jetson schlägt die Inferenz des Modells bei Batch-Größe 32 fehl, da der geteilte Arbeitsspeicher von 16 GB nicht ausreicht. In der Realität würden auf dieser Hardware aber voraussichtlich eher Batch-Größen von 1-8 eingesetzt werden, bspw. für 1-8 gleichzeitige Kamera-Videostreams.

Als Erklärung für die flacheren Kurven kann der Rechenaufwand aus Tabelle 6 für das DETR-Modell herangezogen werden: DETR benötigt ca. 86 GFLOPs, während ResNet-50 lediglich 4,1 GFLOPs fordert. Somit ist DETR ein wesentlich rechenintensiveres Modell und führt schneller zur Sättigung der KI-Hardware.

4.2.2. Energieeffizienz

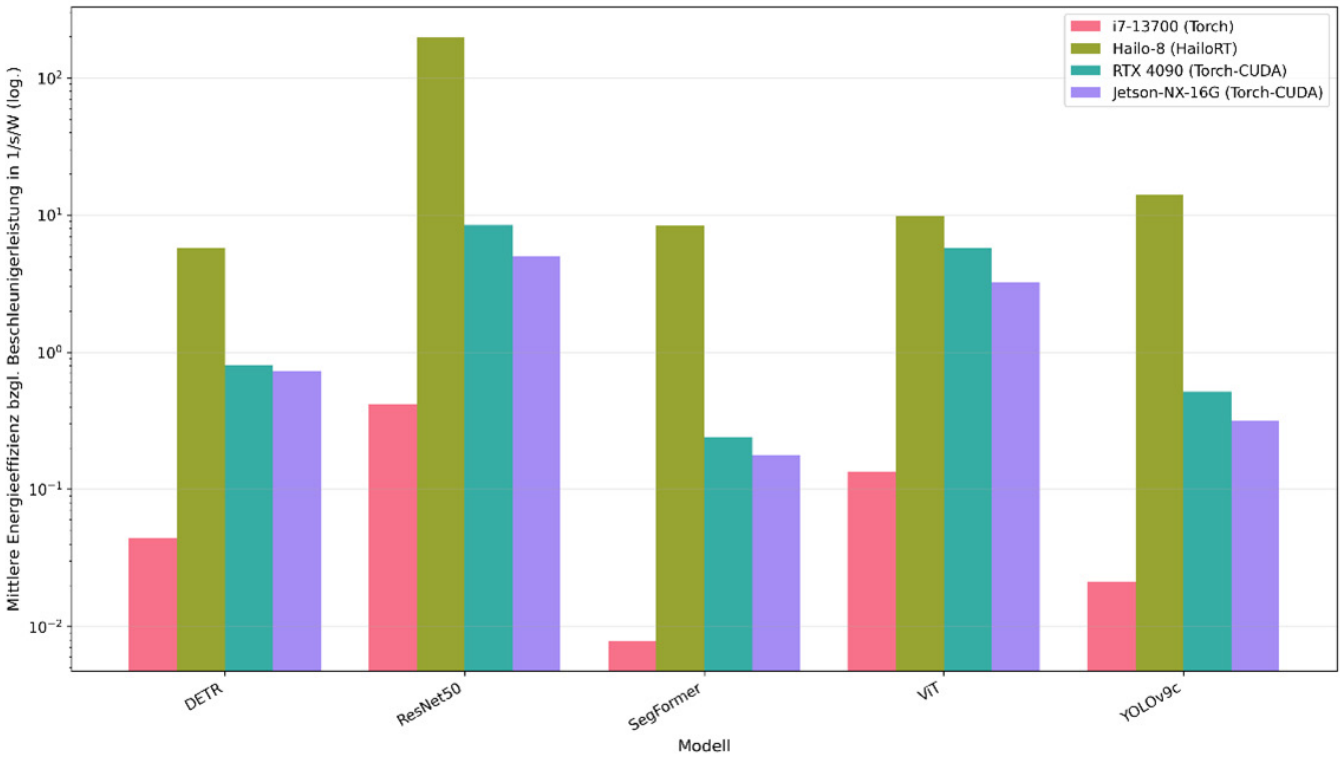


Abbildung 11: Mittlere Energieeffizienz bei Batch-Größe 1 bezogen auf Beschleunigerleistung. Anmerkung: Die Werte des Jetsons inkludieren CPU-Power mit ca. 5 Watt, also ist die tatsächliche Energieeffizienz als ca. 20-30 % besser anzunehmen als hier dargestellt. Höhere Werte sind besser.

Abbildung 11 vergleicht die mittlere Energieeffizienz für fünf Modelle auf den vier ausgewählten KI-Hardware-Beschleunigern. Insgesamt zeigt der Edge Beschleuniger Hailo 8 die geringste Leistungsaufnahme pro Durchsatz, während die CPU durchgängig am ineffizientesten ist. Die RTX 4090 und der Jetson-NX-16GB sind sehr ähnlich in ihrer Effizienz und sind immer deutlich besser als die CPU, erreichen jedoch nicht die Energieeffizienz des Edge Beschleunigers Hailo-8. Insgesamt bekräftigen die Ergebnisse die Eignung von Hailo 8 und Jetson für energieeffiziente Edge Inferenz, während die CPU Inferenz energetisch nachteilig bleibt.

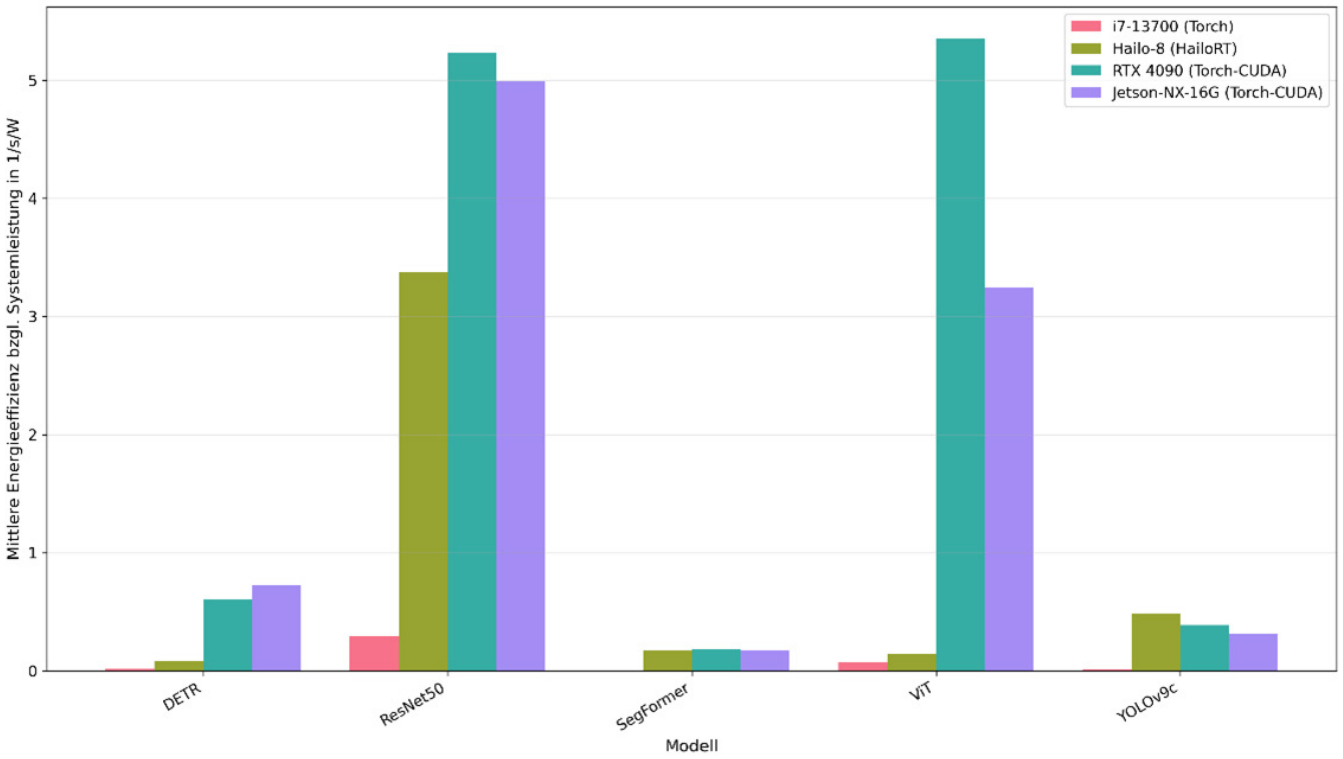


Abbildung 12: Mittlere Energieeffizienz bei Batch-Größe 1 bezogen auf Systemleistung (nicht nur Beschleunigerleistung). Höhere Werte sind besser.

Abbildung 12 vergleicht die mittlere Energieeffizienz bezogen auf die Systemleistung, das heißt, hier zählt auch die Leistungsaufnahme der CPU, RAM, Festplatten und sonstiger Komponenten. Über alle Modelle hinweg ist hier, wie bereits in Abbildung 11, die CPU am ineffizientesten, während spezialisierte Hardware deutlich höhere Effizienzwerte erzielt. Beim Hailo-Chip fällt deutlich auf, dass CNN-basierte Architekturen sichtbar besser ausfallen als Transformer-basierte. Dies deutet auf eine bessere Auslegung der Hardware für genau diese CNN-Architekturen hin. Da dieser Test sich auf die Systemleistung bezieht, ergeben sich für die RTX 4090 relativ zu den restlichen Beschleunigern bessere Werte.

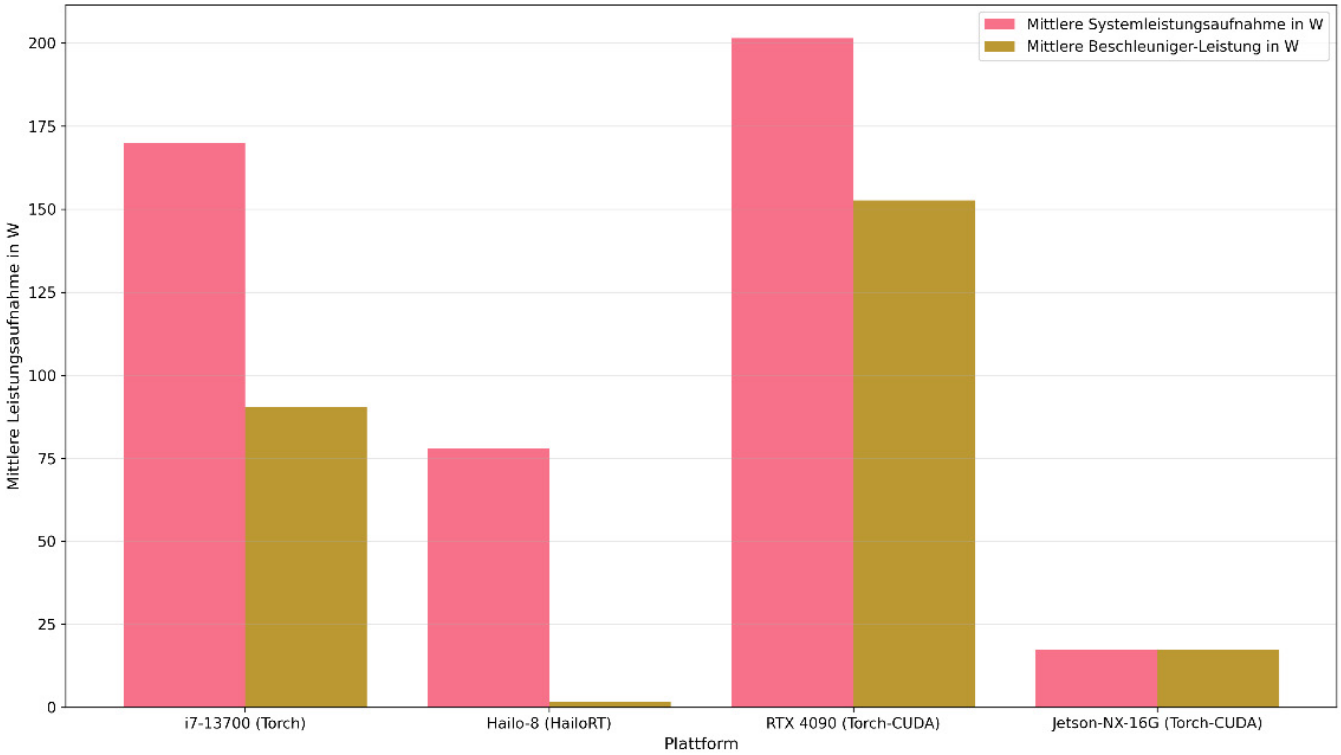


Abbildung 13: Mittlere Leistungsaufnahme für Gesamtsystem und Beschleuniger, gemittelt über alle fünf Modelle. Anmerkung: Für Jetson sind CPU und GPU kombiniert, daher keine Unterscheidung bei Leistungsaufnahme.

Abbildung 13 vergleicht die mittlere Leistungsaufnahme in Watt, getrennt nach Gesamtsystem und Beschleuniger. Der i7 13700 liegt bei 170 W Systemleistung, davon 90 W für den Prozessor an sich, was auf einen hohen Plattform Overhead hinweist. Beim Hailo 8 beträgt die Systemleistung 78 W, während der Beschleuniger selbst nur rund 1,5 W im Mittel verbraucht; die Leistungsaufnahme wird somit klar vom Host-CPU dominiert. Die RTX 4090 erreicht mit rund 202 W die höchste Systemleistungsaufnahme; 153 W davon entfallen auf die GPU. Der Jetson NX 16G bleibt bei ca. 17 W Leistungsaufnahme.

Insgesamt zeigen die Messungen hohe absolute Leistungsaufnahmen bei der diskreten High End GPU, deutlich geringere Werte bei Edge-/Embedded Lösungen sowie eine vergleichsweise hohe Leistungsaufnahme bei der CPU. Da RTX 4090 und Hailo-8 die CPU für die Koordination von KI-Berechnungen benötigen, steigt die Systemleistung entsprechend stark an.

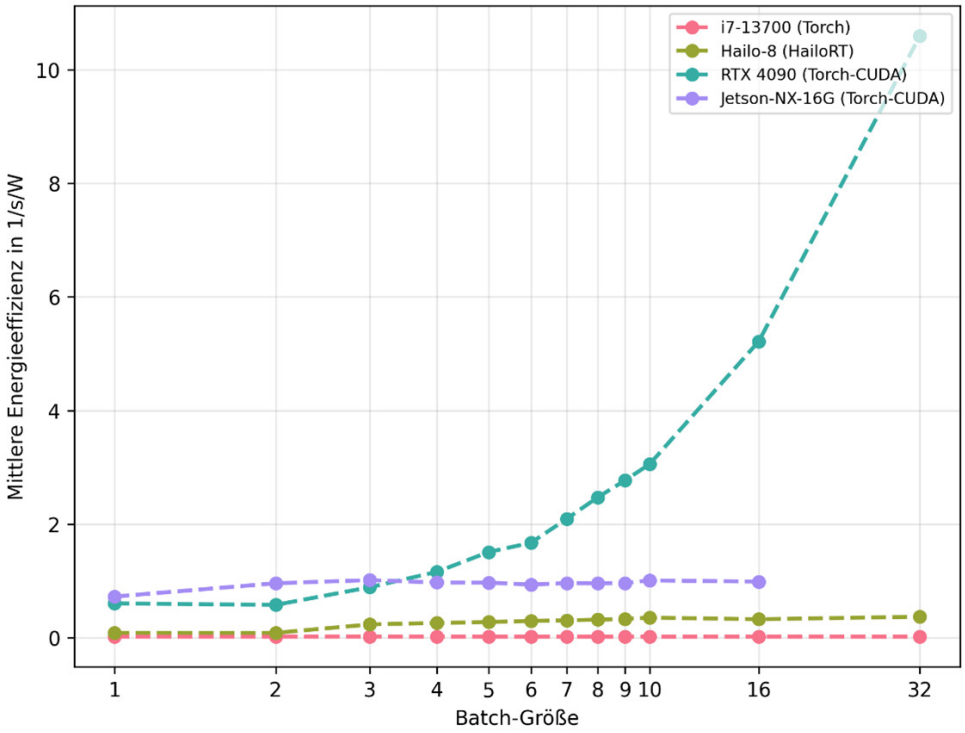


Abbildung 14: DETR-Energieeffizienz bezogen auf Systemleistung. Höhere Werte sind besser.

Abbildung 14 zeigt die mittlere Energieeffizienz für DETR in Abhängigkeit von der Batch-Größe für alle vier Plattformen. Mit steigender Batch-Größe skaliert die RTX 4090 am stärksten und übertrifft ab mittleren Batch-Größen alle anderen Systeme deutlich; bei großen Batches steigt ihre Effizienz steil an. Der Jetson-NX-16G verbessert sich nur moderat und erreicht früh ein Plateau. Hailo-8 und die CPU bleiben über alle Batches hinweg nahezu konstant auf niedrigem Niveau und profitieren kaum von größeren Batches. Die sehr ähnliche Effizienz von Hailo-8 und CPU lässt sich dadurch erklären, dass hier die Gesamtsystemleistungsaufnahme betrachtet wird und in beiden Fällen die CPU für Koordination und Berechnungen (Vor- und Nachbearbeitung) eingesetzt wird.

Insgesamt verdeutlicht die Grafik, dass die Batch-Größe ein zentraler Hebel für Energieeffizienz ist und dass eine diskrete High-End-GPU insbesondere bei hohen Batches die mit Abstand beste Effizienz erzielt, während Edge-/CPU-Lösungen nur begrenzt skalieren.

4.2.3. CPU-Auslastung und CPU-Bottlenecks

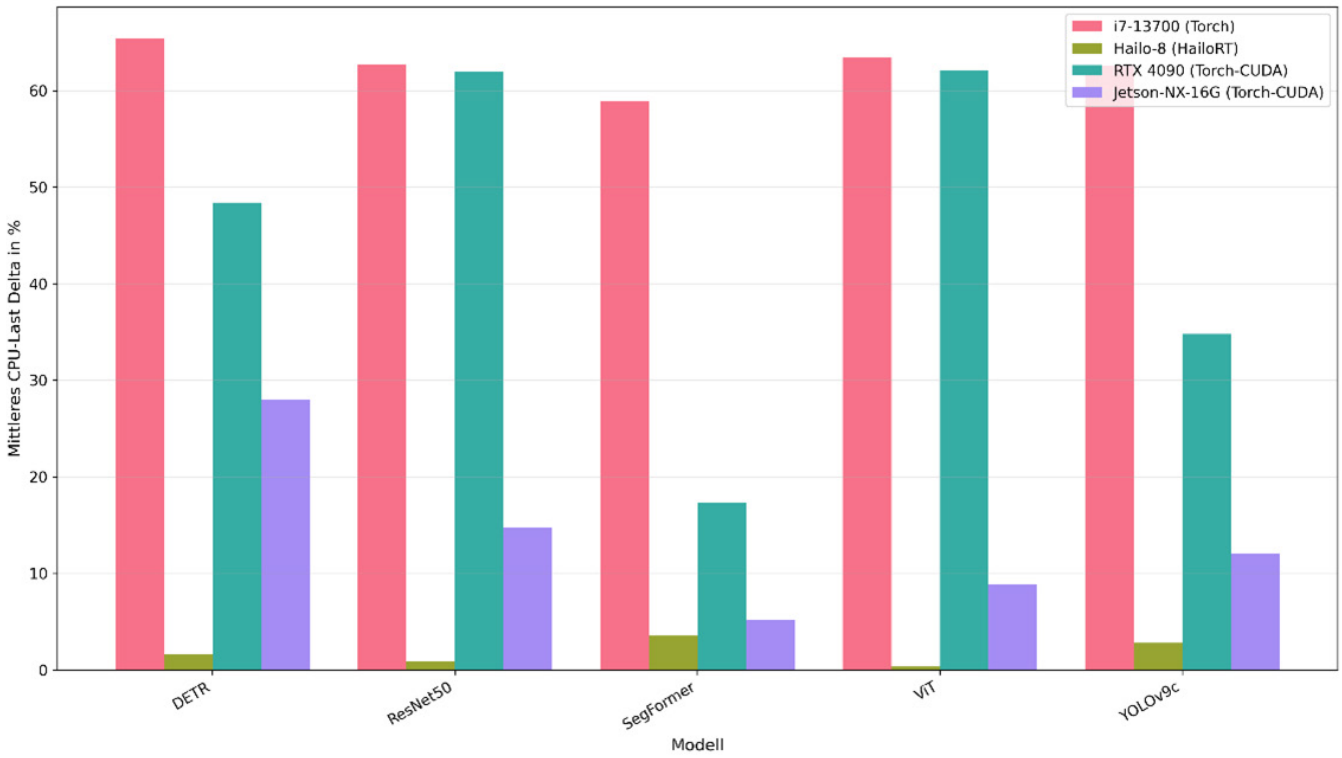


Abbildung 15: CPU-Last-Delta zwischen Benchmark- und Idle-Leistungsaufnahme bei Batch-Größe 1.

Abbildung 15 zeigt den Unterschied (=Delta) in Prozentpunkten zwischen Idle und Benchmark des jeweiligen Modells der CPU. Bei der Inferenz auf dem i7-13700 liegt das CPU-Last-Delta konsistent bei rund 59–65 Prozent über alle Modelle hinweg. Die RTX 4090 weist einen stark modellabhängigen Anteil auf: Hohe Werte (~62 Prozent) bei ResNet-50 und ViT, moderat bei DETR (~48 Prozent) und niedrig bei Segformer (~17 Prozent). Diese Werte lassen sich durch den wesentlich höheren Durchsatz mancher Modelle auf der RTX 4090 erklären, sodass mehr CPU-Last u.a. durch Vor- und Nachbearbeitung bzw. Host-Device-Kommunikation und Kopiervorgänge entsteht. Beim Hailo-8 zeigt sich, dass die CPU nur geringfügig zusätzlich belastet wird, was durch weniger Vor-/Nachbearbeitung bei der Implementierung der Hailo-Inferenz verursacht wird. Der Jetson befindet sich im Mittelfeld, was das CPU-Last-Delta angeht. Insgesamt ist die CPU-Last stark abhängig von Art und Umfang der Vor- und Nachbearbeitung.

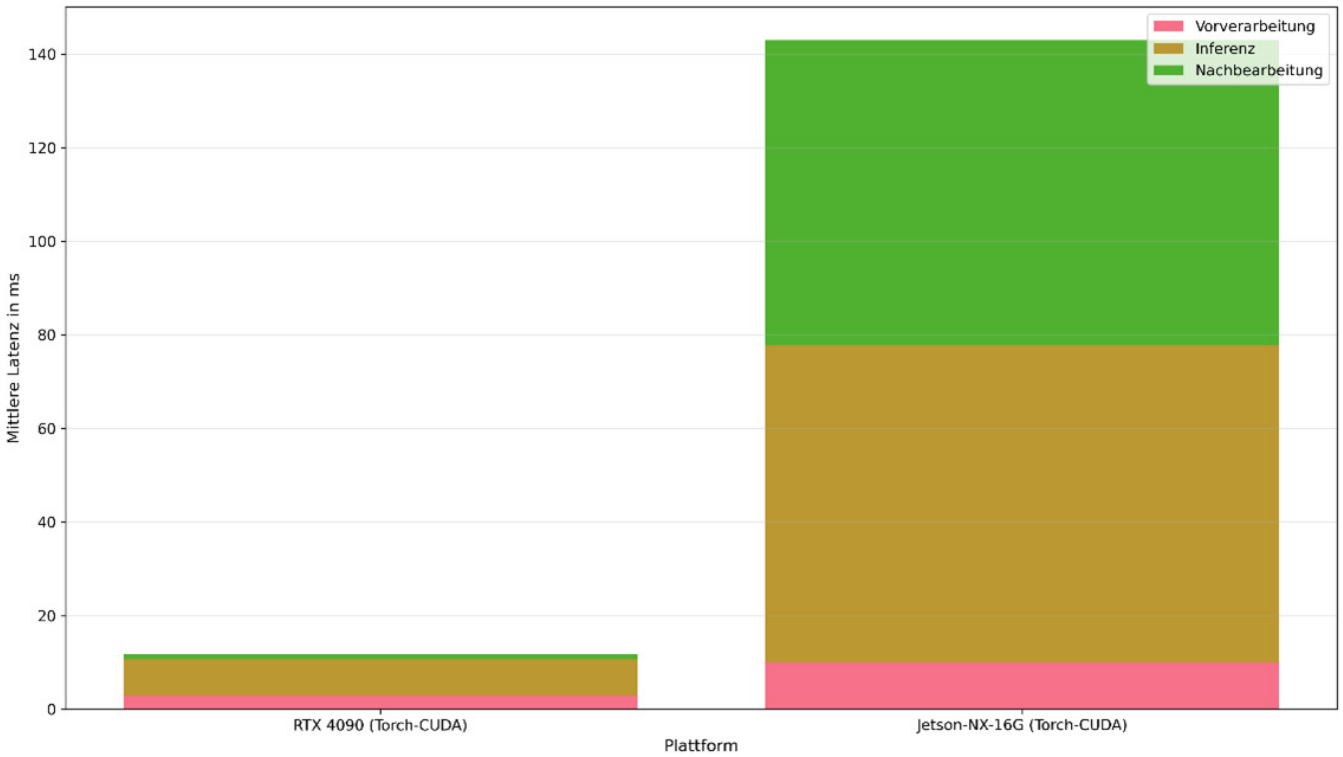


Abbildung 16: Mittlere Latenz für Inferenz, Vor- und Nachbearbeitung im Detail bei Batch-Größe 1 für Jetson und RTX 4090 bei DETR-Modell.

Abbildung 16 vergleicht die End-to-End-Latenz und ihre Anteile (Vorverarbeitung, Inferenz, Nachbearbeitung) für zwei GPU-Plattformen: RTX 4090 und Jetson-NX-16G. Die RTX 4090 erreicht eine sehr niedrige Gesamtlatenz im niedrigen zweistelligen Millisekundenbereich mit 11,7 ms, wobei die Inferenz am meisten zur Gesamtzeit beiträgt. Der Jetson-NX-16G weist mit 143,0 ms eine rund zwölfmal höhere Latenz auf. Sie wird deutlich von Inferenz und Nachbearbeitung dominiert, während die Vorverarbeitung nur einen kleinen Anteil ausmacht.

Diese Gegenüberstellung veranschaulicht, dass die Vor- und Nachbearbeitung einen signifikanten Anteil der Ausführungszeit in Anspruch nimmt, insbesondere auf Systemen mit schwächeren CPUs und niedrigen Leistungsbudgets wie dem Jetson-NX-16G.

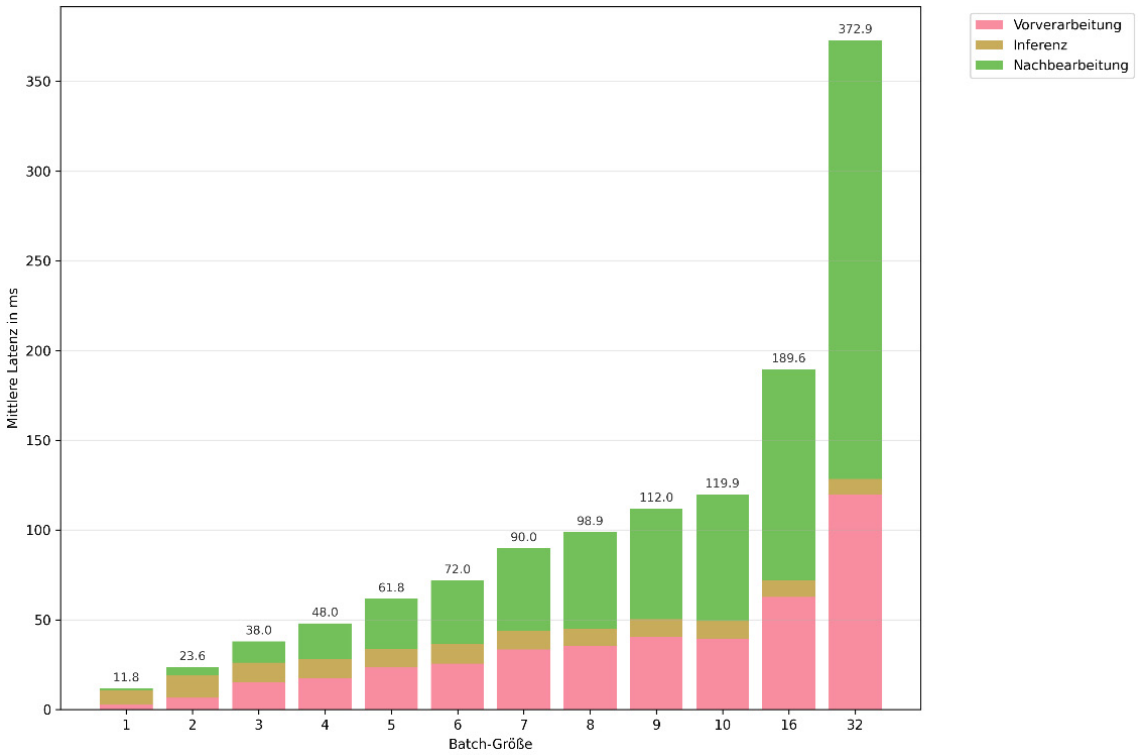


Abbildung 17: Detaillierte Aufschlüsselung der Pipelinebestandteile über diverse Batch-Größen auf RTX 4090 mit DETR.

Abbildung 17 zeigt die komplette Pipeline-Latenz bei verschiedenen Batch-Größen und zerlegt sie in Vorverarbeitung, Inferenz und Nachbearbeitung. Die Gesamtdauer steigt mit der Batch-Größe deutlich an; von 11,8 ms bei Batch-Größe 1 bis 372,9 ms bei Batch-Größe 32. Mit zunehmender Batch-Größe wächst insbesondere die Nachbearbeitung überproportional und wird ab mittleren bis großen Batches zum dominierenden Anteil, während die Vorverarbeitung moderat, annähernd linear zunimmt. Die reine Inferenz bleibt über alle Batches hinweg der kleinste Zeitanteil, da die RTX 4090 GPU eine sehr große Rechenkapazität und damit starke Skalierungsfähigkeiten aufweist. Insgesamt wird die Gesamt-Latenz bei größeren Batches primär durch Vor- und Nachbearbeitungszeiten bestimmt. Dies liegt unter anderem daran, dass die Vor- und Nachbearbeitung in dieser Implementierung sequenziell und nicht vektorisiert abläuft. Daher skaliert sie wesentlich schlechter.

4.2.4. Vergleich Optimierungslevel von PyTorch und TensorRT

In diesem Experiment werden die Auswirkungen von Genauigkeitsreduktion und optimierten Software-Bibliotheken am Beispiel PyTorch und TensorRT gezeigt. In Abbildung 18 ist zu sehen, dass die Reduktion von FP32 auf FP16 eine Verbesserung von 7,5 Prozent mit reinem PyTorch ohne TensorRT erzielt. Durch das Konvertieren und Optimieren mit TensorRT verbessert sich die mittlere Latenz weiter und erreicht eine bessere Latenz als FP16 mit nativem PyTorch. Den größten Sprung erreicht man, wenn TensorRT in Kombination mit FP16-Genauigkeit eingesetzt wird. Hierbei wird die Latenz um 71,8 Prozent reduziert, ohne jegliche manuelle Anpassung am Modell oder dem Hardware-Aufbau.

Hier wird, im Gegensatz zu den Benchmarks aus dem vorherigen Abschnitt bis 4.2.3, das Original-Repository[1] von YOLOv9 genutzt, da es umfangreiche Unterstützung für diverse Optimierungs-Frameworks (wie TensorRT) bietet und den Export erleichtert. Dennoch kann TensorRT auch mit wenig Aufwand in bestehende Projekte integriert werden. Ein besonders hilfreiches Tool dafür ist bspw. torch2trt[2]. Im YOLOv9-Repository wird zunächst ein Export ins ONNX-Format und anschließend die Konvertierung in ein serialisiertes TensorRT-Format durchgeführt.

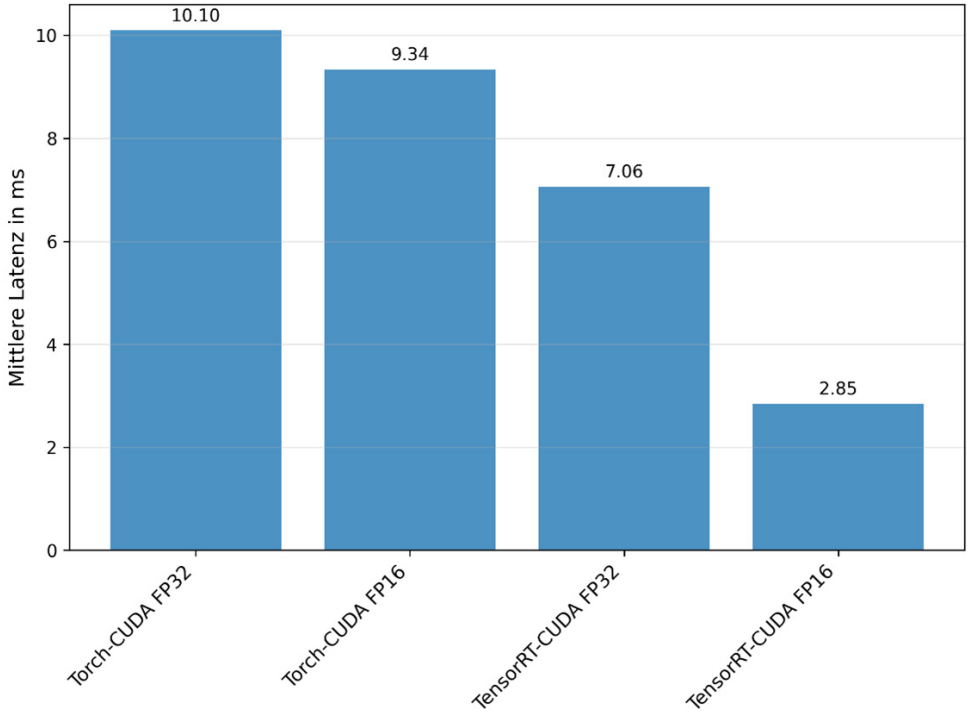


Abbildung 18: Vergleich der mittleren Latenz über 200 Iterationen der reinen Modellinferenz mit unterschiedlichen Genauigkeiten zwischen PyTorch 2.3.0 und TensorRT 8.6.3 mit Nvidia RTX 4090 innerhalb des nvcr.io/pytorch:24.03 Docker-Containers. Niedriger ist besser.

[1] https://github.com/WongKinYiu/yolov9
[2] https://github.com/NVIDIA-AI-IOT/torch2trt

4.2.5. Linux vs. Windows

Da Windows auf Industrie-PCs weit verbreitet ist, wird ein Teil der Benchmarks auf Windows mit identischer Hardware erneut ausgeführt. Hierzu wird Windows 10 (Version 22H2, Build 19045.6216) mit der identischen CUDA- und PyTorch-Version eingesetzt. Der Nvidia-Treiber ist jeweils in Version 560 installiert. Die Ergebnisse in Abbildung 19 stellen dar, dass für reine CPU-Ausführung kein signifikanter Unterschied besteht. Das geringfügig bessere Ergebnis für Windows liegt im Bereich der Messungenauigkeit. Bei Nutzung der Nvidia RTX 4090 GPU ist allerdings eine wesentlich niedrigere Latenz zu beobachten, die Verbesserung liegt zwischen 27,0 Prozent und 51,5 Prozent.

Zudem bieten manche Frameworks mehr Funktionen und bessere Kompatibilität auf Linux-Systemen. So unterstützt TensorFlow Windows seit Version 2.11 nur noch in der CPU-Variante, die GPU-Funktionalität ist ausschließlich über das Windows Subsystem for Linux (WSL) auf Windows möglich.

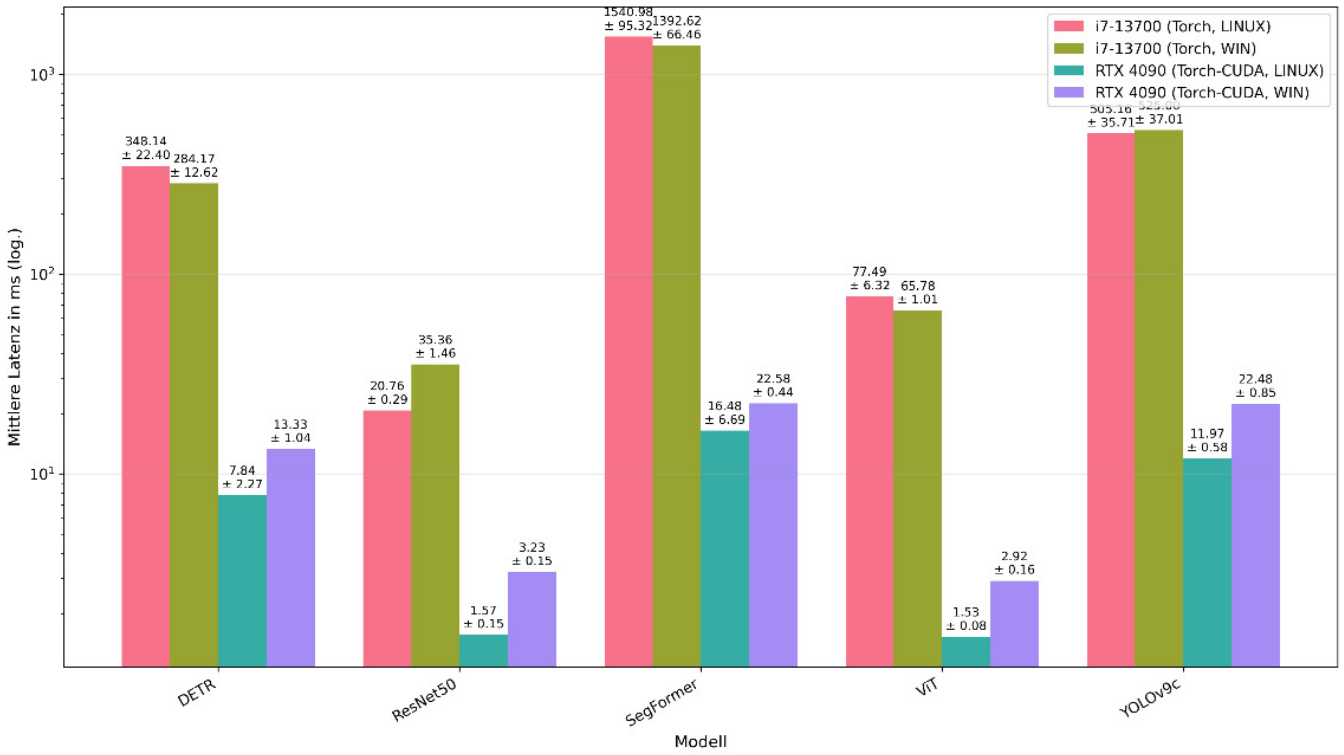


Abbildung 19: Vergleich der mittleren Latenz mit Batch-Größe 1 über alle fünf Modelle auf Windows und Ubuntu 22.04 (Linux).

4.3. Diskussion

Die Ergebnisse zeigen folgende zentrale Punkte:

1. CPU-Inferenz ist in Latenz, Durchsatz und Effizienz weit abgeschlagen im Vergleich zum Einsatz von KI-Beschleunigern.
2. Die spezialisierten KI-Beschleuniger (Hailo-8, Jetson, RTX 4090) werden bei Batch-Größe 1 meist nicht voll ausgelastet und bieten ein hohes Skalierungspotenzial für höhere Batch-Größen oder Ausführung mehrerer simultaner Inferenzen, insbesondere bei der RTX 4090.
3. Beim Betrachten der Effizienz sollte nicht nur die reine Beschleunigerleistungsaufnahme, sondern auch die Gesamtsystemleistung betrachtet werden, da diese signifikante Auswirkungen auf die Effizienz des gesamten KI-Systems haben kann.
4. Die Performance der CPU ist essenziell für Vor- und Nachbearbeitung, sofern diese nicht auf die KI-Beschleuniger ausgelagert werden können. Bei hohen Durchsätzen oder aufwendiger Vor- und Nachbearbeitung kann dies zu einem Ausbremsen bzw. Bottleneck der KI-Beschleuniger führen.
5. Spezielle, hardware-spezifische Runtimes und Software-Optimierungen wie Präzisionsreduktion können die Inferenz signifikant und mit wenig Implementierungsaufwand beschleunigen.

Auf Basis der Ergebnisse aus Abschnitt 4.2.1 zum Durchsatz kann ein Vergleich mit der theoretischen Performance der Beschleuniger getätigt werden: Als grobe Referenz bieten die Hardware-Plattformen folgende Spitzenwerte: Hailo 8 ≈26 TOPS (INT8), Jetson NX 16G 100 TOPS (INT8), RTX 4090 ≈83 TOPS (FP32) bzw. ≈1321 TOPS (INT8). Solche Zahlen sind nicht direkt vergleichbar, sie liefern aber eine obere Grenze. Die effektive Rechenrate lässt sich praxisnäher aus den benötigten Modellrechenoperationen und dem gemessenen Durchsatz abschätzen:

$$\text{Effektive Rechenrate} \approx \text{Rechenoperationen pro Bild} \times \text{Durchsatz}$$

Setzt man dabei ausschließlich den Durchsatz basierend auf der Inferenzzeit ohne Vor- und Nachbearbeitung an, ergibt sich für Batch-Größe 1 beim ResNet-50-Modell auf der RTX 4090 eine effektive Rechenrate von 2,61 TOPS. Das liegt deutlich unter den Peak-Rechenraten der Hersteller und bestätigt die Beobachtung der Unterauslastung bei kleinen Batches. Mit steigender Batchgröße steigt die Rechenrate spürbar, bei Batch-Größe 32 werden bspw. 51,72 TOPS erreicht.

Generell wird bei Batch-Größe 1 praktisch nie die volle Auslastung der RTX 4090 GPU erreicht. Für latenzkritische Einzelbild Workloads reduziert sich der Vorsprung einer RTX 4090 gegenüber der günstigeren Hardware daher deutlich. Wer primär BS1 Inferenz ausführt, kann Kosten und Energie sparen und zu günstigeren Beschleunigern bzw. kleineren Varianten der jeweiligen GPU (bspw. RTX 4070 anstatt 4090) greifen. Für hohe Batch-Größen, mehrere parallele Videostreams und große Eingabedaten bleibt die RTX 4090 klar überlegen, was die absolute Performance betrifft.

Der Vergleich der Latenzen und Durchsätze zwischen den Modellen ist nur zwischen solchen Modellen aussagekräftig, die die gleiche Input-Größe nutzen. Dies sollte beim Betrachten der Ergebnisse berücksichtigt werden. Daher wurden in den Ergebnissen nur Vergleiche in Bezug auf unterschiedliche Hardware für das gleiche Modell durchgeführt. Des Weiteren unterscheidet sich die Implementierung von Vor- und Nachbearbeitung pro Modell stark, weshalb sich die Experimente größtenteils auf die reine Modellinferenz ohne Vor- und Nachbearbeitung konzentriert haben. Eine Ausnahme ist die detaillierte Pipeline-Betrachtung in Abschnitt 4.2.3, da hier explizit auf die Vor- und Nachbearbeitung eingegangen wird, allerdings nur für ein bestimmtes Modell.

Bei den genutzten Methoden zur Leistungsaufnahmemessung und Effizienzbewertung sind folgende Limitierungen zu berücksichtigen: Der Hailo-8-Beschleuniger wirkt aufgrund der Kombination mit dem i7-CPU »unterbewertet« bei Betrachtung der Gesamtsystemeffizienz, weil der CPU vergleichsweise hohe Idle-Leistungsaufnahmen besitzt. Auf einem sehr schlanken Host-System wie einem Raspberry Pi würden die Effizienzwerte für das Gesamtsystem signifikant besser ausfallen, während die reine Modellinferenzlatenz voraussichtlich ähnlich bleiben würde. Weiterhin erzeugt unser Monitoring für Auslastung und Leistungsaufnahme via nvidia smi, tegrastats, Intel RAPL und Co. zusätzliche CPU Last durch Parsing und API-Abfragen und systematische Unterschiede. Daher wird bei der Betrachtung der CPU-Auslastung das Delta zwischen Idle und Last betrachtet und nicht die absolute Leistungsaufnahme. Zudem wird beim Jetson keine extreme Leistungsaufnahmemessung durchgeführt und ausschließlich auf die tegrastats-API zugegriffen. Hier kann außerdem keine Aufteilung in CPU- und GPU-Leistungsaufnahme erfolgen, weshalb die Effizienz bezogen auf die Beschleunigerleistung nicht exakt bestimmbar ist.

Insgesamt ist wichtig zu erwähnen, dass nicht allein die Auswahl der Hardware bzw. Modelle von Bedeutung ist. In Form von Software-Optimierungen, wie durch das Beheben von Pipeline Bottlenecks (Vor-/Nachverarbeitung) durch Auslagern dieser Operationen auf Beschleuniger statt Ausführung auf dem CPU, ist eine vielfache Beschleunigung der Inferenz möglich. Hier lohnt es sich, detaillierte Benchmarks und exaktes Profiling durchzuführen, um die Bottlenecks zu identifizieren. Auch der Einsatz von optimierten Videostreaming-Frameworks wie Gstreamer kann für Bildverarbeitungs-Workloads eine weitere Beschleunigung liefern.

5. Praxisbeispiele

In diesem Kapitel werden konkrete Beispiele aus der Praxis beschrieben, die das Deployment auf KI-Hardware behandeln.

5.1. Konvertierung von YOLOv9e auf Hailo-8

Da im Hailo-Modell Zoo[1] nur ausgewählte Netze verfügbar sind, muss man bei Nichtvorhandensein eines gewünschten Modells (oder gewünschter Hyperparameter wie z. B. Eingabegröße) manuell in ein Hailo-kompatibles Format konvertieren. Dazu sind mehrere Schritte nötig, die im Folgenden erläutert werden.

Als Beispiel wird die Konvertierung von YOLOv9e in ein Hailo-Format betrachtet. Grundlage sind das offizielle YOLOv9-Repository von WongKinYiu[2] sowie der Dataflow Compiler von Hailo[3].

1.Schritt: Erzeugen einer .onnx[4] Datei
Der offene Standard zur Repräsentation von neuronalen Netzen (Open Neural Network Exchange, ONNX) eignet sich ideal als hardwareunabhängiger Startpunkt der Konvertierung in das hardwareabhängige .hef-Format des Hailo-Chips. Viele Repositories, so auch das YOLOv9-Repository, bieten vorgefertigte Skripte zur Konvertierung in ONNX.

2. Schritt: Erzeugen einer .har-Datei
Mit der Erzeugung einer .har-Datei aus einer .onnx-Datei erhält man eine Hailo-spezifische Modellrepräsentation, die allerdings noch auf CPU ausführbar ist. Mögliche Anpassungen, die man in diesem Konvertierungsschritt vornehmen kann, sind z. B. die Anpassung der Start- und Endknoten des Modells sowie der Inputgröße.

[1] https://github.com/hailo-ai/hailo_model_zoo
[2] https://github.com/WongKinYiu/yolov9
[3] https://hailo.ai/developer-zone/documentation/dataflow-compiler-v5-0-0/?sp_referrer=overview/overview.html#introduction
[4] https://onnx.ai/

3. Schritt: Modelloptimierung
Im dritten Schritt wird ein Kalibrierungsdatensatz für die INT8-Quantisierung benötigt. Es lohnt sich, hierbei einen für den jeweiligen Anwendungsfall repräsentativen Datensatz zu verwenden. Weiterhin werden unter anderem folgende Parameter festgelegt:

- Batchgröße
- Optimierungslevel
- Konfiguration der Auslagerung von Vorverarbeitungs-Operationen wie Resize

Viele der in Abschnitt 2.4 erwähnten Optimierungsschritte werden vom Hailo-Dataflow Compiler auf dem resultierenden Graphen aus Schritt 2 im Rahmen der Modelloptimierung ausgeführt, so zum Beispiel: Graphoptimierungen, Operator-Fusion und Layer-Transformationen und Quantisierung auf INT8. Anschließend wird die Modellausführung auf dem Hailo-8 emuliert und die Genauigkeit mit dem Original-Modell verglichen, da insbesondere die INT8-Quantisierung zu Genauigkeitsverlusten führen kann.
Der Hailo Dataflow Compiler unterstützt CUDA-GPUs, was die Optimierung signifikant beschleunigen kann.

4.Schritt: Erzeugen einer .hef-Datei (Kompilierung für Hailo-8)
Die erzeugte .har-Datei aus dem vorherigen Schritt ist für Menschen lesbar und lässt sich untersuchen. Zudem lassen sich hilfreiche Plots erstellen.
Die in diesem Schritt erzeugte .hef-Datei liegt als Binärdatei vor und ist für den Hailo-8 kompiliert und optimiert. Das .hef-Modell kann nun für die Inferenz auf dem Hailo-8-Chip eingesetzt werden.

Fazit:
Eine manuelle Konvertierung in das Hailo-Format ist machbar und folgt einem klaren Prozess (ONNX → HAR → Optimierung → HEF). Der Optimierungsschritt kann selbst auf einer GPU je nach Konfiguration Stunden bis Tage dauern. Außerdem kann die Konvertierung fehlschlagen, falls bestimmte Operationen oder Architekturen des Modells nicht unterstützt werden. Diese müssen dann auf die CPU ausgelagert werden. Empfehlung: Wenn ein passendes Modell im Modell-Zoo existiert, sollte dieses bevorzugt werden und gegebenenfalls mit dem vorhandenen, verlinkten GitHub-Repository auf eigenen Daten nachtrainiert werden. Ansonsten lohnt sich die manuelle Pipeline für Spezialanforderungen (z.B. mehr Genauigkeit und damit größere Modelle notwendig, spezielle Vor- und/oder Nachbearbeitung) mit ausreichender Zeit- und Ressourcenplanung.

5.2. Multi-Kamera-Personenerkennung für Sicherheitsanwendungen

Im Rahmen eines Kundenprojekts im Jahr 2024 wurde am Fraunhofer IPA vom Forschungsbereich KI und Maschinelles Sehen ein Multi-Kamera-System mit vier Kameras entwickelt, um das Betreten von Gefahrenzonen zu erkennen (vgl. Abbildung 20).

Abbildung 20: Schema des Multi-Kamerasystems mit Nvidia Jetson Hardware. Quelle: Fraunhofer IPA/Foto: Timo Leitritz.

Im Projekt gab es eine harte Anforderung von 170 ms an die Verarbeitungszeit bzw. Latenz von Kamerabildaufnahme aller 4 Kameras bis zum Ergebnis der Auswertungslogik. Im Prototyp wurde eine Latenz von ca. 500 ms auf der Jetson Hardware erreicht, was im Verlauf des Projekts durch gezielte Optimierungen, wie in Abschnitt 2.4 behandelt, auf 50 ms reduziert werden konnte. Dabei spielte der Einsatz von TensorRT und die Nutzung von C++ anstatt Python zur Beschleunigung der Vor- und Nachbearbeitung eine elementare Rolle. Als Hardware wurde die industrielle Jetson-Variante von Seeed Studio aus Abbildung 6 eingesetzt und in einem Schaltschrank verbaut. Dieses Jetson-Modell besitzt eine passive Kühlung und geringe Leistungsaufnahme von 15 Watt. Die Hardware kommuniziert per Generel Purpose Input/Output (GPIO) Schnittstelle mit der SPS der Maschine.

Fazit:
Durch spezialisierte KI-Hardware lässt sich aufwändige, KI-basierte Bildverarbeitung im Maschinenumfeld mit niedriger Latenz effizient umsetzen. Essenziell ist dabei aber nicht nur die Wahl der KI-Hardware, sondern auch die softwareseitigen Optimierungen und der Einsatz passender, hardwarespezifischer Softwarebibliotheken wie TensorRT für Nvidia-Hardware. Diese Bibliotheken können die Ausführung teilweise um das Dreifache beschleunigen, wie in Abschnitt 4.2.4 gezeigt.

6. Schlussbemerkung

Leistungsfähige und verlässliche KI-Systeme ergeben sich aus der kohärenten Auslegung von Modell, Beschleunigerhardware, Software Stack und einer optimierten End to End Pipeline. Einzelne Spitzenkennzahlen (z. B. TOPS) aus den Hersteller-Spezifikationen sind mit Vorsicht zu vergleichen und bieten meist keine belastbare Aussage über die reale Systemperformance oder Effizienz. Hier sollte die Zielanwendung konkret auf der KI-Hardware evaluiert werden. Zu diesem Zweck hat das Fraunhofer IPA eine große Bandbreite an KI-Beschleunigern zur Evaluation entlang konkreter Anforderungen vorliegen. So bietet das Fraunhofer IPA im Rahmen des EU-Projekts AI-MATTERS[1] Unternehmen einen Service zur Evaluation, Benchmarking und Optimierung von existierenden KI-Systemen auf diverser KI-Hardware an.

Bei den Benchmarks wird ersichtlich, dass diskrete GPUs wie die Nvidia RTX 4090 den höchsten Durchsatz liefern und stark von wachsenden Batch-Größen profitieren. EdgeSoCs (z. B. Jetson Orin Serie) bieten ein günstiges Verhältnis aus Performance, Integrationsfähigkeit und Energiebedarf, skalieren aber weniger bei steigender Batch-Größe und Modellkomplexität. Hochspezialisierte Inferenz-ASICs (z. B. Hailo-8) setzen Maßstäbe bei der Energieeffizienz, entfalten ihre Stärken jedoch insbesondere in Kombination mit schlanken Host-Systemen. Reine CPU Ausführung von KI-Modellen ist für moderne Computer Vision in aller Regel nicht praktikabel, die CPU ist aber essenziell für Vor-/Nachbearbeitung und Steuerlogik. Sehr kleine Netze können unter Umständen mit ausreichend Performance auf einer herkömmlichen CPU ausgeführt werden.

Die Benchmarks dieser Studie wurden mit Computer-Vision-Modellen ausgeführt. Die Ergebnisse lassen sich aber zum Großteil auf KI-Modelle aus Bereichen wie Zeitreihen, Sprach- und Textverarbeitung etc. übertragen. Zudem wurden bewusst Modelle gewählt, die hinsichtlich der Operatoren eine hohe Übereinstimmung mit LLMs aufgrund der gleichen Basis-Transformer-Architektur (ViT, DETR, SegFormer) besitzen. Large Language Models benötigen dabei vor allem große Speichermengen, die immer mehr KI-Hardwarehersteller berücksichtigen.

Ein Hauptansatzpunkt für die Beschleunigung von KI-Systemen ist die Optimierung der Software. Präzisionsreduktion (INT8/FP16), Graph/Operator Fusion und optimierte Runtimes (z. B. TensorRT, OpenVINO, ONNX Runtime) reduzieren Latenz und Energiebedarf substanziell, häufig ohne architekturverändernde Modellanpassungen oder größere Genauigkeitseinbußen. Für KI-Workloads erweist sich Linux als Betriebssystem aufgrund Tool-, Treiber- und Ökosystemreife in der Praxis als vorteilhaft.

[1] https://ai-matters.eu/how-it-works/

Industrielle Rahmenbedingungen sind beim Einsatz von KI-Hardware nicht zu vernachlässigen: Bauraum, thermische Limits, Echtzeitfähigkeit, Cybersecurity und Erklärbarkeit (XAI) sind zentrale Auswahlkriterien. Die zunehmende Verfügbarkeit von NPUs in Industrie-PCs durch moderne CPU-SoCs wird die Möglichkeiten zur effizienten und sicheren Ausführung von KI-Modellen in naher Zukunft weiter verbessern.

Zusammengefasst: Wer heute Computer-Vision-KI in der Industrie wirtschaftlich betreiben will, sollte nicht nur »den schnellsten Chip« kaufen, sondern die gesamte Lösung systematisch optimieren – von der Vorverarbeitung über die Modellarchitektur bis zur Laufzeitumgebung – und dabei industrielle Rahmenbedingungen berücksichtigen.

7. Transparenzerklärung

Für die Verfeinerung des Entwurfes dieser Studie wurde das Fraunhofer-eigene KI-Tool FhGenie (Version GPT-5) verwendet. Es wurden jedoch keine Textausschnitte kopiert, ohne dass diese zuvor von den Autoren gründlich Korrektur gelesen und überarbeitet worden wären. Alle zitationsfähigen Passagen sind als solche gekennzeichnet.

8. Literaturverzeichnis

[1] S. Jahnel et al., „Generative KI in der deutschen Wirtschaft 2025“, KPMG AG Wirtschaftsprüfungsgesellschaft, Berlin, 2025.

[2] I. Gartner, „Gartner forecasts worldwide AI chips revenue to grow 33% in 2024“, 29. Mai 2024. [Online]. Verfügbar unter: https://www.gartner.com/en/newsroom/press-releases/2024-05-29-gartner-forecasts-worldwide-artificial-intelligence-chips-revenue-to-grow-33-percent-in-2024. [Zugriff: 15. Jan. 2026].

[3] I. Goodfellow et al., „Deep learning“, MIT Press, 2016.

[4] D. Guo et al., „DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning“, Nature, Bd. 645, Nr. 8081, 2025.

[5] V. Nair et al., „Rectified linear units improve restricted Boltzmann machines“, in Proc. International Conference on Machine Learning (ICML), 2010.

[6] D. E. Rumelhart et al., „Learning representations by back-propagating errors“, in Neurocomputing: Foundations of Research, Vol. 1, MIT Press, 1988.

[7] Y. LeCun et al., „Gradient-based learning applied to document recognition“, Proceedings of the IEEE, Bd. 86, Nr. 11, 1998.

[8] A. Vaswani et al., „Attention is all you need“, in Proc. NeurIPS, 2017.

[9] D. P. Kingma et al., „Adam: A method for stochastic optimization“, arXiv preprint arXiv:1412.6980, 2014.

[10] V. Sze et al., „Efficient processing of deep neural networks: A tutorial and survey“, Proceedings of the IEEE, Bd. 105, Nr. 12, 2017.

[11] T. Dao et al., „FlashAttention: Fast and memory-efficient exact attention with IO-awareness“, in Proc. NeurIPS, 2022.

[12] K. Simonyan et al., „Very deep convolutional networks for large-scale image recognition“, arXiv preprint arXiv:1409.1556, 2014.

[13] K. He et al., „Deep residual learning for image recognition“, in Proc. CVPR, 2016.

[14] A. Dosovitskiy et al., „An image is worth 16×16 words: Transformers for image recognition at scale“, in Proc. ICLR, 2021.

[15] S. Ren et al., „Faster R-CNN: Towards real-time object detection with region proposal networks“, in Proc. NeurIPS, 2015.

[16] J. Redmon et al., „You only look once: Unified, real-time object detection“, in Proc. CVPR, 2016.

[17] N. Carion et al., „End-to-end object detection with transformers“, in Proc. ECCV, 2020.

[18] O. Ronneberger et al., „U-Net: Convolutional networks for biomedical image segmentation“, in Proc. MICCAI, 2015.

[19] A. Kirillov et al., „Segment anything“, in Proc. ICCV, 2023.

[20] H. Touvron et al., „LLaMA: Open and efficient foundation language models“,, arXiv preprint arXiv:2302.13971, 2023.

[21] E. Xie et al., „SegFormer: Simple and efficient design for semantic segmentation with transformers“, in Proc. NeurIPS, 2021.

[22] T. Liang, „Pruning and quantization for deep neural network acceleration: A survey“, Neurocomputing, vol. 461, 2021.

[23] W. Niu et al., „DNNFusion: Accelerating deep neural networks execution with advanced operator fusion“, in Proc. PLDI, 2021.

[24] Z. Jiang et al., „Efficient deep learning inference on edge devices“, in Proc. SysML, 2018.

[25] T. Chen et al., „TVM: An automated end-to-end optimizing compiler for deep learning“, in Proc. OSDI, 2018.

[26] C.-Y. Wang et al., „YOLOv9: Learning what you want to learn using programmable gradient information“, in Proc. ECCV, 2024.

[27] Y. Bengio, P. Simard und P. Frasconi, „Learning long-term dependencies with gradient descent is difficult“, IEEE Transactions on Neural Networks, Bd. 5, Nr. 2, 1994.

KI-Fortschrittszentrum

Das KI-Fortschrittszentrum »Lernende Systeme und Kognitive Robotik« unterstützt Firmen dabei, die wirtschaftlichen Chancen der Künstlichen Intelligenz und insbesondere des Maschinellen Lernens für sich zu nutzen. In anwendungsnahen Forschungsprojekten und in direkter Kooperation mit Industrieunternehmen arbeiten die Stuttgarter Fraunhofer-Institute für Produktionstechnik und Automatisierung IPA sowie für Arbeitswirtschaft und Organisation IAO daran, Technologien aus der KI-Spitzenforschung in die breite Anwendung der produzierenden Industrie und der Dienstleistungswirtschaft zu bringen. Unterstützt werden sie dabei vom Institut für Arbeitswissenschaft und Technologiemanagement IAT der Universität Stuttgart. Finanzielle Förderung erhält das Zentrum vom Ministerium für Wirtschaft, Arbeit und Tourismus Baden-Württemberg.

Mission
Das KI-Fortschrittszentrum ist der anwendungsorientierte Zweig von Cyber Valley, Europas größter Forschungskooperation im Bereich der Künstlichen Intelligenz. Darüber hinaus ist das KI-Fortschrittszentrum Bestandteil von S-TEC, dem Stuttgarter Technologie- und Innovationscampus: www.s-tec.de

Das KI-Fortschrittszentrum schlägt die Brücke von der KI-Spitzenforschung in den Mittelstand und macht KI-Technologien für die Wirtschaft in Baden-Württemberg und darüber hinaus nutzbar. Als führender Innovationspartner für den Mittelstand arbeitet das Zentrum an Themen, die für den Einsatz von KI und Robotik branchenübergreifend von zentraler Bedeutung sind, beispielsweise Autonomie, Effizienz und Nachhaltigkeit, Mensch-Maschine-Interaktion sowie Vertrauen. Das KI-Fortschrittszentrum informiert Unternehmen über Technologietrends und deren Einsatzpotenziale und unterstützt sie bedarfsgerecht und niedrigschwellig bei der Entwicklung und Umsetzung von ambitionierten KI-Innovationen, damit sie die wirtschaftlichen Chancen der KI künftig noch besser nutzen können.

Vision
Das KI-Fortschrittszentrum ist ein Leuchtturm für erfolgreichen Technologietransfer in den Mittelstand und ermöglicht Unternehmen einen wirtschaftlichen und verantwortungsvollen Einsatz von Künstlicher Intelligenz und Robotik für unternehmerischen Erfolg sowie individuellen und gesellschaftlichen Nutzen.

Studienreihe »Lernende Systeme und Kognitive Robotik«
Die Studienreihe »Lernende Systeme und Kognitive Robotik« gibt Einblick in die Potenziale und die praktischen Einsatzmöglichkeiten von KI. Nähere Informationen und die aktuellen Versionen der Studien finden Sie unter: https://www.ki-fortschrittszentrum.de/veroeffentlichungen/

Fraunhofer

Die Fraunhofer-Gesellschaft mit Sitz in Deutschland ist eine der führenden Organisationen für anwendungsorientierte Forschung. Im Innovationsprozess spielt sie eine zentrale Rolle – mit Forschungsschwerpunkten in zukunftsrelevanten Schlüsseltechnologien und dem Transfer von Forschungsergebnissen in die Industrie zur Stärkung unseres Wirtschaftsstandorts und zum Wohle unserer Gesellschaft. Seit ihrer Gründung als gemeinnütziger Verein im Jahr 1949 nimmt sie eine einzigartige Position im Wissenschafts- und Innovationssystem ein.

Knapp 32 000 Mitarbeitende an 75 Instituten und selbstständigen Forschungseinrichtungen in Deutschland erarbeiten das jährliche Finanzvolumen von 3,6 Mrd. €. Davon entfallen 3,1 Mrd. € auf das zentrale Geschäftsmodell von Fraunhofer, die Vertragsforschung. Im Vergleich zu anderen öffentlichen Forschungseinrichtungen bildet die Grundfinanzierung durch Bund und Länder lediglich das Fundament des jährlichen Forschungshaushalts. Sie ist die Basis für wegweisende Vorlaufforschung, die in den kommenden Jahren für Wirtschaft und Gesellschaft bedeutend wird. Das entscheidende Alleinstellungsmerkmal ist der hohe Anteil an Wirtschaftserträgen, der Garant ist für die enge Zusammenarbeit mit Wirtschaft und Industrie und die stetige Marktorientierung der Fraunhofer-Forschung: 2024 beliefen sich die Wirtschaftserträge auf 867 Mio. € des laufenden Haushalts. Ergänzt wird das Forschungsportfolio durch im Wettbewerb eingeworbene öffentliche Projektmittel, wobei eine ausgewogene Balance zwischen öffentlichen und wirtschaftlichen Erträgen angestrebt wird.

Wir produzieren Zukunft
Das Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, kurz Fraunhofer IPA, realisiert hoch innovative und nachhaltige Lösungen in der Produktionstechnik und Automatisierung für verschiedenste Zukunftsbranchen. Dies können Methoden, Komponenten und Geräte bis hin zu kompletten Maschinen und Anlagen sein. Die Lösungen stehen stets in Verbindung mit den strategischen Eckpfeilern des Instituts »Mass Sustainibility« und »Mass Personalization«. Seine Hauptaufgabe sieht das Institut im Wissens-, Innovations- und Technologietransfer von Forschungsergebnissen in Applikationen, um die Wettbewerbsfähigkeit der Unternehmen zu stärken. Dabei versteht es sich als unabhängiger Ansprechpartner, der neutral berät und Unternehmen mit genau auf deren Bedürfnisse zugeschnittenen Projektteams unterstützt.

Autoren

Timo Leitritz, M. Sc.
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart

Timo Leitritz ist wissenschaftlicher Mitarbeiter im Forschungsbereich Künstliche Intelligenz und Maschinelles Sehen am Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. Sein Schwerpunkt liegt auf der Entwicklung von menschzentrierten Assistenzsystemen in industriellen Anwendungen mithilfe von kamerabasierter Aktivitätserkennung und deren Deployment auf spezialisierter KI-Hardware. Des Weiteren administriert und optimiert er die KI-Trainingsserver und -Workstations des Forschungsbereichs. Im Rahmen des KI-Fortschrittszentrums war er an einigen Quick Checks und Exploring Projects beteiligt, unter anderem zum Thema »Machine Learning mit Embedded Devices«, bei dem ein detaillierter Vergleich mehrerer KI-Beschleuniger erfolgte.

Jonas Heinle, M. Sc.
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Stuttgart

Jonas Heinle ist wissenschaftlicher Mitarbeiter im Forschungsbereich Künstliche Intelligenz und Maschinelles Sehen am Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. Dabei entwickelt er hardwarebewusste, effiziente Vision-Modelle und End-to-End-Pipelines, die von der Prototypisierung bis zum Edge-Deployment im industrienahen Kontext laufen. Sein Schwerpunkt liegt auf der Inferenz kamerabasierter Modelle auf ressourcenlimitierter Hardware sowie auf der Etablierung von Software-Engineering-Praktiken für kamerabasierte KI mit Fokus auf Reproduzierbarkeit, Testbarkeit und langfristige Wartbarkeit.

Impressum

Kontaktadresse
Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA
Nobelstraße 12, 70569 Stuttgart

Timo Leitritz
Telefon +49 711 970-1859
timo.leitritz@ipa.fraunhofer.de

Jonas Heinle
Telefon +49 711 970-3566
jonas.heinle@ipa.fraunhofer.de

Fraunhofer-Publica
http://dx.doi.org/10.24406/publica-7085

Titelbild
© Klyra Work – Adobe Stock

Gefördert durch das Ministerium für Wirtschaft, Arbeit
und Tourismus Baden-Württemberg

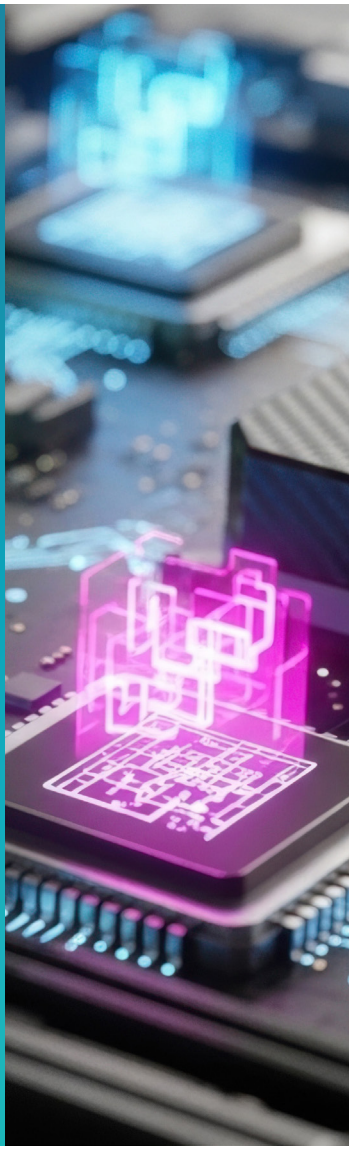
Kontakt

KI-Fortschrittszentrum
Fraunhofer IAO und Fraunhofer IPA
Nobelstraße 12
70569 Stuttgart
www.ki-fortschrittszentrum.de

Timo Leitritz
Telefon +49 711 970-1859
timo.leitritz@ipa.fraunhofer.de

Jonas Heinle
Telefon +49 711 970-3566
jonas.heinle@ipa.fraunhofer.de

Fraunhofer IPA
Nobelstraße 12
70569 Stuttgart
www.ipa.fraunhofer.de



Gefördert durch

Baden-Württemberg
Ministerium für Wirtschaft,
Arbeit und Tourismus

Partner

Cyber Valley

S·TEC

Universität Stuttgart
Institut für Arbeitswissenschaft und
Technologiemanagement IAT