dena
Deutsche Energie-Agentur

Future Energy Lab

dena-ANALYSE

Auf dem Weg zu energieeffizienter künstlicher Intelligenz

Welche Energieeinsparpotenziale bieten KI-Anwendungen?

Impressum

Herausgeber
Deutsche Energie-Agentur GmbH (dena)
Chausseestraße 128a
10115 Berlin
Tel.: +49 (0)30 66 777-0
Fax: +49 (0)30 66 777-699
E-Mail: futureenergylab@dena.de
Internet: www.dena.de
www.future-energy-lab.de

Autorinnen und Autoren:
Lisa Kratochwill, dena, Projektleitung
Linda Babilon, dena
Karsten Müller, Fraunhofer HHI
Roman Rischke, Fraunhofer HHI
Wojciech Samek, Fraunhofer HHI
Benno Stabernack, Fraunhofer HHI
Fritjof Steinert, Fraunhofer HHI

Stand:
März 2022

Bitte zitieren als:

Deutsche Energie-Agentur (Hrsg.) (dena, 2022) „Auf dem Weg zu energieeffizienter künstlicher Intelligenz: Welche Energieeinsparpotenziale bieten KI-Anwendungen?“

Bundesministerium für Wirtschaft und Klimaschutz

Die Veröffentlichung dieser Publikation erfolgt im Auftrag des Bundesministeriums für Wirtschaft und Klimaschutz. Die Deutsche Energie-Agentur GmbH (dena) unterstützt die Bundesregierung in verschiedenen Projekten zur Umsetzung der energie- und klimapolitischen Ziele im Rahmen der Energiewende.

Inhaltsverzeichnis

Kurzfassung

Der Trend zu künstlicher Intelligenz (KI) ist weltweit ungebrochen. KI-basierte Systeme kommen in immer mehr Anwendungsfeldern zum Einsatz und unterstützen zunehmend auch Entscheidungsprozesse in der Energiewirtschaft, z. B. für Bedarfsprognosen, prädiktive Wartung, Kundenclusterung und die Abwehr von Cyberangriffen (Kratochwill et al., 2020). Für eine erfolgreiche Anwendung müssen diese komplexen Systeme mit Millionen von Beispieldaten „angelernt“ werden. Dieser rechenintensive Prozess wird üblicherweise in großen Rechenzentren durchgeführt. Der CO_2-Fußabdruck eines State-of-the-Art-KI-Modells ist vergleichbar mit dem CO_2-Ausstoß, den eine Person mit über 300 Hin- und Retourflügen zwischen San Francisco und New York verursacht (Strubell, Ganesh, & McCallum, 2019). Beim verteilten Training – einer Technik des maschinellen Lernens – von KI-Modellen entsteht zusätzlich noch ein nicht zu vernachlässigender Kommunikationsoverhead, welcher die Energiebilanz weiter verschlechtert. Demnach sollten austrainierte KI-Modelle, die mehrfach zur Anwendung kommen, so implementiert werden, dass ihre Ausführung energetisch effizient vollzogen werden kann. Die vorliegende Analyse fasst die Problematik mangelnder Energieeffizienz heutiger künstlicher Intelligenz zusammen und zeigt neue Möglichkeiten zur Entwicklung energieeffizienter KI und energiesparender Datenübertragung auf.

1 Einleitung

Künstliche Intelligenz (KI) ist unangefochten eines der Topthemen der Digitalisierung. Gleichzeitig ist auch die zwingende Notwendigkeit des Klimaschutzes insbesondere in den letzten Jahren immer deutlicher geworden. Während zumeist von dem enormen Potenzial der Technologie für eine erfolgreiche Energiewende als Grundvoraussetzung für die Eindämmung des Klimawandels gesprochen wird, geht diese Diskussion jedoch oftmals auch mit der Frage nach dem Energieverbrauch von KI einher.

Eine Studie der University of Massachusetts Amherst (Strubell, Ganesh, & McCallum, 2019) setzt das Trainieren eines einzigen neuronalen Netzes mit dem CO_2-Ausstoß von fünf Autos während ihrer gesamten Lebenszeit inklusive des dabei verbrauchten Kraftstoffs gleich. Auf den ersten Blick ergibt sich eine erschreckend hohe Zahl. Bezieht man aber die Anzahl der Nutzenden mit ein, im Falle der Fahrzeuge sind das vielleicht zwischen fünf und zehn Personen, im Falle des trainierten KI-Modells je nach Anwendung dagegen mehrere Hunderttausend, relativiert sich dieser Wert sehr schnell. Neben dem Energieverbrauch muss natürlich auch der Nutzen der Anwendung in die Betrachtung mit einbezogen werden. Thorsten Staake, Professor für Wirtschaftsinformatik an der Universität Bamberg und Spezialist für energieeffiziente Systeme, gibt hier eine Einordnung: „Wenn, als Beispiel, für das Training eines neuronalen Netzes für einen Spurhalte-Assistenten für eine ganze Fahrzeugflotte eines großen Herstellers Energie in der Höhe des Verbrauchs eines Pkws aufgewendet wird und damit auch nur ein Unfall verhindert wird, wäre das energetisch schon kompensiert – vom zusätzlichen Wert für die Unfallbeteiligten ganz abgesehen“ (Lobe, 2019).

Zudem hängt der Energiebedarf einer KI-Anwendung von unterschiedlichen Attributen beispielweise der Komplexität des Use Cases oder der Menge der zu verarbeitenden Daten ab. Auch der im Rechenzentrum verwendete Strommix spielt natürlich eine erhebliche Rolle in Bezug auf den CO_2-Ausstoß des Trainings bzw. der Ausführung des KI-Modells.

Gleichzeitig führen immer komplexer werdende KI-Ansätze und der Anstieg an zu verarbeitenden Daten zu zusätzlichem Rechenleistungs- und damit zu erhöhtem Energiebedarf. Wenngleich KI ein großes Potenzial zur Energieeffizienzsteigerung bietet, muss daher verhindert werden, dass es bei ihrer Anwendung zu Rebound-Effekten kommt, also die Nutzeneffekte durch den erhöhten Energieverbrauch wieder aufgehoben werden. Insofern ist eine wirksame Lösungsstrategie zur energieeffizienten Ausgestaltung der Technologie zwingend erforderlich. Dabei muss die Effizienz entlang der gesamten Wertschöpfungskette, sowohl hardware- als auch softwareseitig, betrachtet werden.

1.1 Die Problematik des steigenden Strombedarfs

Der Strombedarf für Rechenzentren betrug bereits im Jahr 2016 mit 12,4 TWh (Hintemann R. , 2016) 2,4 Prozent des gesamten Stromverbrauchs der Bundesrepublik Deutschland (insgesamt 518 TWh laut Umweltbundesamt). Zu diesem Verbrauch in Rechenzentren tragen 2,3 Mio. Server mit 4,5 TWh bei, was bei einer Nutzung des deutschen Strommix für 2016 einem Ausstoß von 2,4 Mio. Tonnen CO_2 entspricht. Für den Wachstumsmarkt Rechenzentren wird bis zum Jahr 2025 ein weiterer Anstieg des Strombedarfs um 32,3 Prozent auf 16,4 TWh erwartet (Hintemann & Clausen, 2016). Dazu trägt auch eine um 55.6 Prozent steigende Serverleistung bei. Durch dieses starke Wachstum ist damit selbst im Jahr 2025 noch mit stagnierenden oder sogar steigenden CO_2-Emissionen (in Abhängigkeit des zukünftigen CO_2-Ausstoßes pro kWh) zu rechnen. Wie bereits

erwähnt, sind KI-Anwendungen bereits heute mit ihren Millionen benötigten Einzelberechnungen sehr rechenintensiv, wobei mit einer weiteren Steigerung der Anzahl der Operationen zu rechnen ist. Auch insgesamt wird die Bedeutung von KI-Anwendungen voraussichtlich weiterhin stark steigen. Für den europäischen Markt wird allein von 2019 bis 2022 ein Wachstum um 7 Mrd. auf insgesamt 10 Mrd. Euro prognostiziert (Bitcom, kein Datum). Die steigende Bedeutung von KI-Anwendungen sowie deren steigende Komplexität bzw. immer höhere benötigte Rechenleistung führen zu einem erhöhten Stromverbrauch und infolgedessen zu einem Anstieg der CO_2-Emissionen. Vor dem Hintergrund der Zielsetzung, die CO_2-Emissionen bis 2030 um 60 Prozent gegenüber 1990 zu senken und bis 2045 komplett treibhausgasneutral zu werden, stellt die energieeffiziente Ausgestaltung digitaler Technologien eine Grundvoraussetzung zum Erreichen der Zielmarken dar.

1.2 Künstliche neuronale Netze

Die Entwicklung künstlicher Intelligenz verfolgt das Ziel, Maschinen in die Lage zum selbstständigen Lösen von Problemen und Aufgaben zu versetzen. Das KI-Teilgebiet des maschinellen Lernens befasst sich mit dem Trainieren von Modellen für das Erkennen sowie Vorhersagen statistischer Muster und Eigenschaften in Daten. Angelehnt an das biologische Vorbild eines natürlichen neuronalen Netzes zur Informationsverarbeitung werden beim Deep Learning künstliche neuronale Netze (KNN) als Modelle verwendet. KNN sind üblicherweise in sequenziell angeordneten Schichten aufgebaut, wobei jede Schicht den Output der vorherigen Schicht transformiert bzw. filtert und wiederum als Input an die nächste Schicht weiterreicht (siehe Abbildung 1). Durch diese Eigenschaften sind KNN in der Lage, natürliche neuronale Systeme nachzubilden. Sie werden beispielsweise in der Bildklassifikation eingesetzt. Hier sorgen die verschiedenen Schichten zumeist für die Detektion geometrischer Objekte in unterschiedlichen Abstraktionslevels: Während der Input-Schicht nachgelagerte Schichten versuchen, einfache Strukturen wie etwa Kanten im Bild zu detektieren, setzen spätere Schichten diese einfachen Strukturen zu immer komplexeren Gefügen zusammen, wodurch schließlich ganze Szenen im Bild klassifiziert werden können.

Realisiert wird eine Schicht eines KNN üblicherweise durch eine lineare Transformation der Aktivierungen der Input-Neuronen mit anschließender nichtlinearer Aktivierung der Output-Neuronen. Die lineare Transformation der Aktivierung der Input-Neuronen kann als Matrix-Vektor-Multiplikation implementiert werden und stellt analog zur Synapse als biologische Verknüpfung zweier Nervenzellen, die Verbindung zwischen den Input- und Output-Neuronen einer Schicht her.

Die Aktivierungsfunktion kann beispielsweise dafür sorgen, dass Neuronen, deren Aktivierung einen gewissen Schwellenwert nicht überschreitet, gar nicht erst aktiviert werden und somit keinen Einfluss auf nachfolgende Schichten haben. Gerade nichtlineare Aktivierungsfunktionen, wie etwa die häufig verwendeten Funktionen ReLU (Rectified Linear Unit) und GELU (Gaußian Error Linear Unit), ermöglichen die Verwendung beliebig komplexe Transformationsschichten, was wiederum KNN ihre vielseitige Einsetzbarkeit verleiht. Wichtig ist auch, dass es keine allgemeingültige Konvention gibt, ab welcher Anzahl von Schichten ein KNN als tiefes neuronales Netz bezeichnet wird, da die Komplexität bzw. Wirkmächtigkeit eines KNN nicht nur von seiner Tiefe abhängt, sondern auch von seiner Breite (der Anzahl von Neuronen in einer Schicht).

Die Verbindungen zwischen den Input- und den Output-Neuronen einer KNN-Schicht stellen also Gewichte oder Gewichtungsfaktoren in einer Transformationsmatrix dar. Diese Gewichte werden innerhalb eines Trainings beispielsweise auf einem vorklassifizierten Datensatz iterativ mittels Optimierungsverfahren wie dem stochastischen Gradientenabstiegsverfahren so lange angepasst, bis sich eine gewünschte

Klassifikationsgüte einstellt. Alle Gewichte bzw. Parameter eines KNN zusammen bilden ein (KI-)Modell. Auch der Trainingsprozess selbst bildet ein biologisches neuronales Netz nach, indem sich mit jedem Trainingsschritt die Stärke der Verbindungen verändert, also die Gewichtswerte für das KNN.

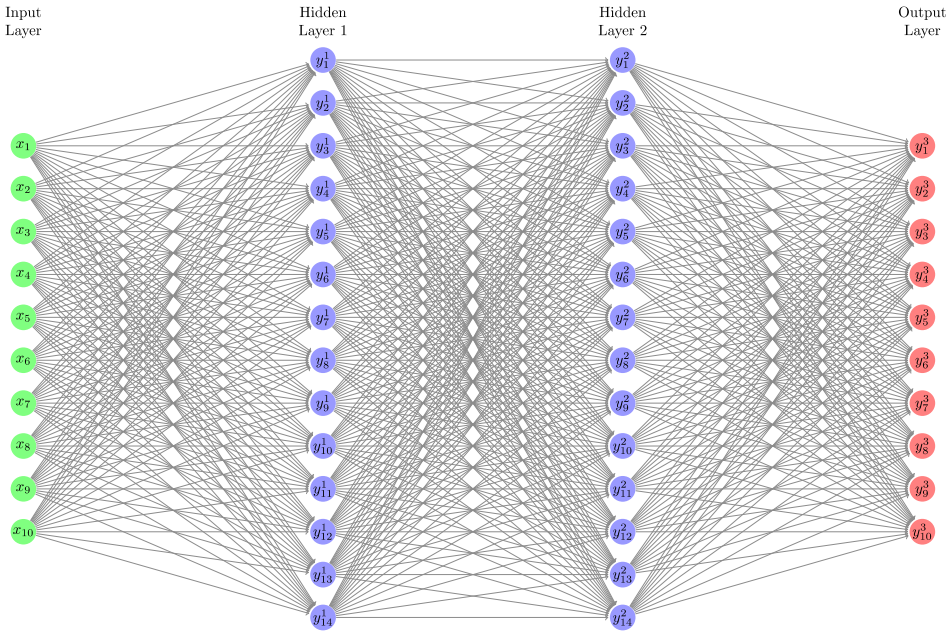

Abbildung 1: Schematische Darstellung eines KNN mit vier Schichten

Die Trainingsphase zur Einstellung der Gewichte ist sehr rechenintensiv. Daher konnten erst in den letzten Jahrzehnten über die stetig steigende Verfügbarkeit von Rechenleistung und Verarbeitbarkeit großer Datenmengen infolge der Digitalisierung immer komplexere KNN konzipiert werden, welche die bis dahin verwendeten Technologien bei Weitem übertrafen. Gerade die Verwendung tiefer neuronaler Netze kann Lösungen für komplexe Probleme in Anwendungsbereichen wie Computer Vision, Sprachverarbeitung, Autonomes Fahren, Medikamentenforschung und viele weitere liefern. Durch diese Erfolge sind neuronale Netze inzwischen ein essenzieller Bestandteil vieler Anwendungen und aktuell geht die Entwicklung hin zu immer komplexeren und zuverlässigeren neuronalen Netzen (siehe Abbildung 2). Entsprechend wird auch deren Topologie immer komplexer, also die Anzahl ihrer Schichten, Verknüpfungen und Parameter.

Ein typischer Vertreter eines solchen KI-Modells ist VGG16, ein neuronales Netz mit 21 Schichten und über 138 Mio. trainierbaren Parametern. Im Vergleich dazu hat ein einfaches lineares Modell, welches den Input lediglich linear transformiert, lediglich ~50000 trainierbare Parameter. Die zunehmende Komplexität von KI-Modellen scheint einerseits eine zwingende Voraussetzung für die Schlüsselstellung zu sein, die KI mittlerweile einnimmt, führt aber andererseits auch zu einem immer höheren Energieverbrauch für KI-Anwendungen. Neben der bereits angesprochenen rechenintensiven Trainingsphase betrifft die Problematik des höheren Energieverbrauchs insbesondere auch (1) die **Ausführung** (sog. Inferenz) dieser komplexen Modelle in KI-Anwendungen und aufgrund der zunehmenden Popularität dezentraler KI-Anwendungen, wie dem Internet of Things (IoT) oder dem Federated Learning, auch (2) die **Übertragung** von KI-Modellen von einem Sender zu einem Empfänger. Die Phasen (1) und (2) werden in den Kapiteln 2 und 3 jeweils genauer betrachtet.

1.3 Rechenbeschleunigersysteme

Bezüglich spezieller Anwendungen und Algorithmen aus dem Bereich der Signalverarbeitung, der Computergrafik und neuerdings des maschinellen Lernens (ML) hat sich herausgestellt, dass normale prozessorbasierte Rechnersysteme sehr ineffizient arbeiten. Sie weisen eine geringe spezifische Rechenleistung pro Rechenknoten mit weniger FLOP pro Sekunde auf (FLOP: Floating Point Operations, wird als Vergleichsmaßstab für Rechenoperationen verwendet). Dies macht prozessorbasierte Rechensysteme sehr ineffizient in Hinblick auf ihren Energieverbrauch. Dies gilt für einfache Arbeitsplatzrechner genauso wie für ganze Rechnersysteme, die typischerweise in Rechenzentren zu finden sind.

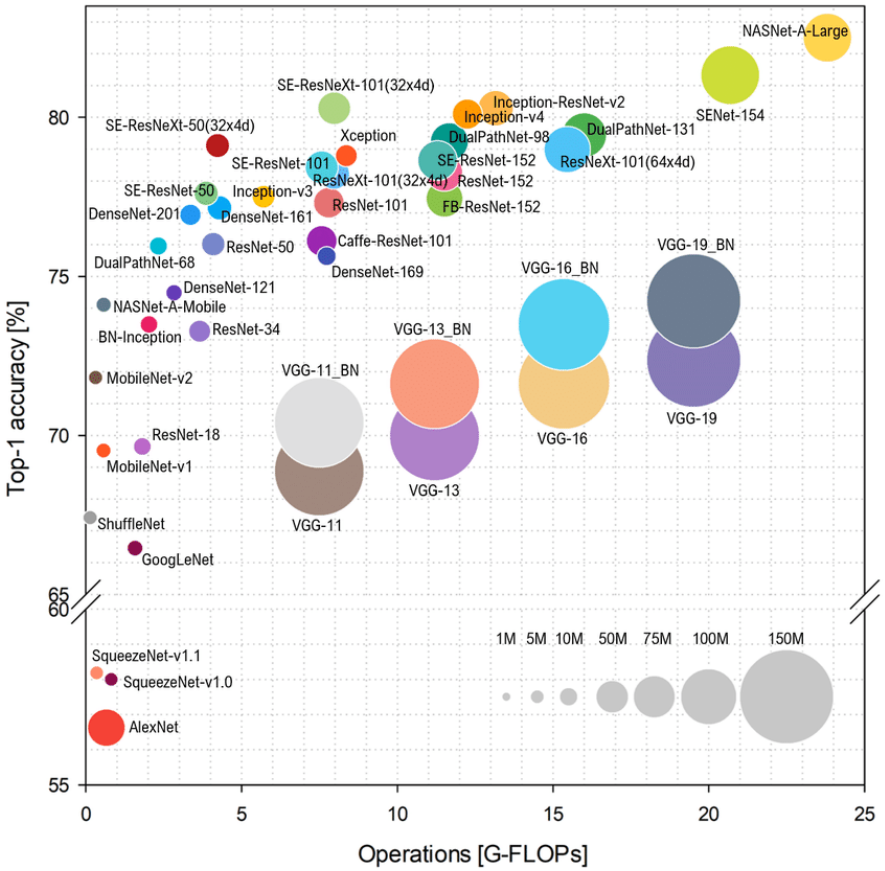

Abbildung 2: Ball-Chart mit Top-1-Erkennungsrate vs. Berechnungskomplexität für typische KI-Modelle
Erkennungsrate: Korrekt erkannte Objektklasse an erster Stelle der wahrscheinlichen Objektklassen)
Berechnungskomplexität: zur Klassifikation notwendigen Berechnungen in G-FLOPs (10^9 Floating Point Operations)
Quelle: (Bianco, Cadene, Celona, & Napoletano, 2018)

Um die spezifische Rechenleistung zu erhöhen, werden daher typischerweise zwei unterschiedliche Arten spezieller Rechenbeschleuniger eingesetzt. Während sich im Bereich der Grafikverarbeitung eigens darauf ausgerichtete Grafikkarten (GPU – graphics processing unit) etabliert haben, findet man insbesondere für Anwendungen der Signalverarbeitung, wie Edge Computing in Mobilfunkanwendungen, in großem Maße FPGA-basierte Systeme (FPGA – field programmable gate array). Im Bereich des maschinellen Lernens sind beide Arten von Beschleunigersystemen anzutreffen (Falsafi et al., 2017). So bieten z. B. kommerzielle Anbieter von Cloudcomputing-Ressourcen, wie Amazon Cloud AWS, der Kundschaft entsprechende Systeme, die entweder FPGA- (Amazon-a) oder GPU-basierte Rechnerinstanzen (Amazon-b) zur Verfügung stellen. Der Cloudanbieter Microsoft Azure setzt teilweise FPGA-basierte Systeme ein, um z. B. Suchmaschinen wie Microsoft Bing (Putnam, Caulfield, Chung, & Burger, 2015) massiv zu beschleunigen.

Neben großen Anbietern von Public-Cloud-Systemen wie Amazon, Google und Microsoft setzen zunehmend auch nationale Anbieter in großem Maße Rechenbeschleuniger ein, um den zukünftig drastisch steigenden Bedarf an Rechenleistung für KI-Anwendungen decken zu können. Die Nachfrage nach privaten Anbietern entsprechender Dienstleistungen in Deutschland steigt zunehmend, da aus rechtlichen und sicherheitsrelevanten Erwägungen die Nutzung globaler Anbieter wie Amazon, Google und Microsoft ausgeschlossen ist oder zumindest aus Datenschutzgründen immer problematischer wird.

Abbildung 3 stellt eine typische Konfiguration einer Private-Cloud-Installation dar, wie sie bei Bedarf von entsprechenden Dienstleistern angemietet werden kann. Die hier dargestellte Konfiguration verfügt bereits über spezielle Rechenbeschleuniger (Compute Nodes), wie sie für KI-Anwendungen eingesetzt werden. Neben einer hochratigen Verbindungsstruktur besitzt sie ein Speicher- und ein Managementsystem, über das die Lastverteilung der Rechner vorgenommen werden kann.

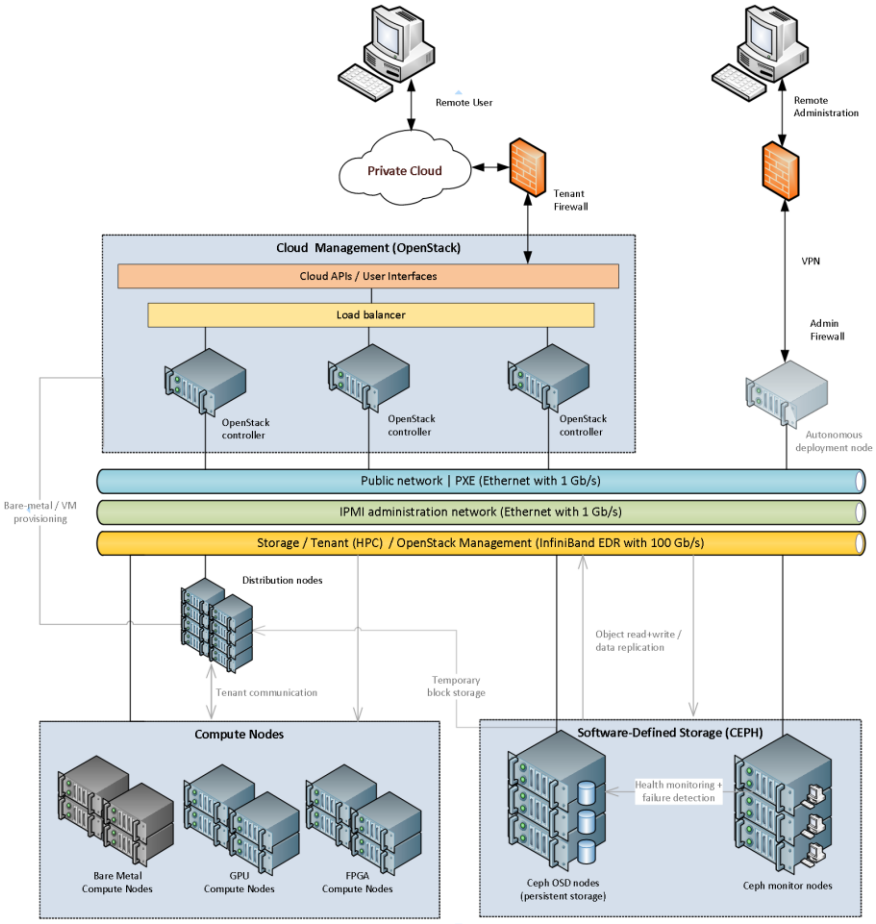

Abbildung 3: Typische Private-Cloud-Architektur mit GPU- und FPGA-Compute Nodes für KI-Anwendungen

2 Energieeffizienz bei der Ausführung von KI-Modellen

Die in diesem Kapitel beschriebene Innovation besteht in einer neuartigen Art und Weise, wie KI-spezifische Rechenbeschleuniger in Rechensysteme integriert werden können (siehe auch Abbildung 3). Auf Basis dieser Kerntechnologie kann auf einen großen Teil der normalerweise benötigten Serverrechner verzichtet werden. Durch die so zu erreichende drastische Reduzierung der Gesamtverlustleistung kann die Energieeffizienz erheblich gesteigert und damit der KI-spezifische CO_2-Ausstoß wesentlich verringert werden.

KI-Verarbeitungsarchitekturen können auf unterschiedliche Weise (z. B. als Software auf Prozessoren bzw. GPU oder als Hardwarearchitektur für FPGA) implementiert werden. Dabei verursachen die für KI-Anwendungen erforderlichen Kernalgorithmen immer die Hauptrechenlast des Gesamtsystems und bestimmen damit den Energiebedarf. Zur effizienteren Verarbeitung dieser Hauptrechenlast bietet sich eine Reihe von Ansätzen an, die sich in Hinblick auf Flexibilität, Energieverbrauch, Kosten und Leistungsfähigkeit der spezifischen Rechenleistung unterscheiden. Abbildung 4 stellt die Ausprägung der Parameter in Bezug auf die üblichen Implementierungsformen dar.

In der internationalen Forschung wird im Wesentlichen an Lösungen gearbeitet, die FPGA für die Beschleunigung von KI-Algorithmen verwenden oder eigene Chips (ASIC – application-specific integrated circuit) entwickeln. Bei ASIC-Implementierungen handelt es sich allerdings um extrem kostspielige Entwicklungen, die zwar das Optimum bezüglich des Energieverbrauchs pro Rechenoperation darstellen, jedoch aufgrund ihrer geringen Flexibilität immer das Problem bergen, für zukünftige Algorithmen möglicherweise nicht mehr anwendbar zu sein.

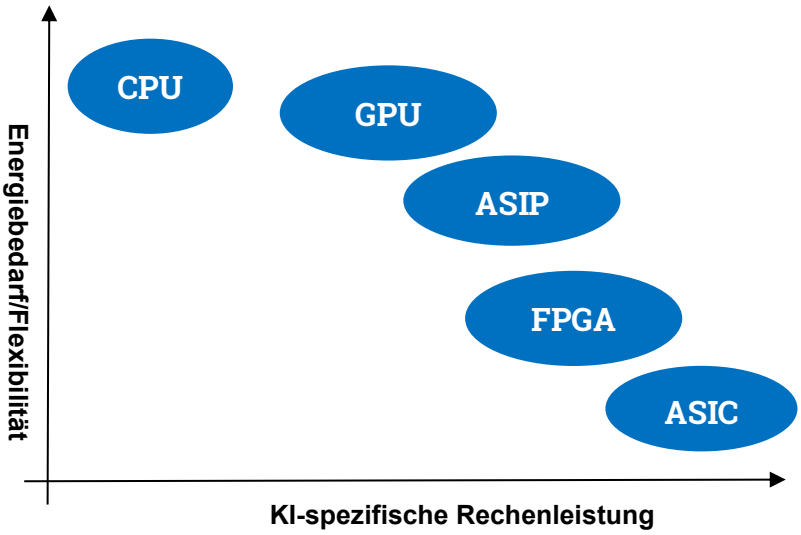

Abbildung 4: Das Zusammenwirken zwischen KI-spezifischer Leistungsfähigkeit und Energiebedarf/Flexibilität

FPGAs stellen dabei unter den in Abbildung 4 dargestellten Verfahren den besten Kompromiss aus Flexibilität, Energiebedarf und Leistungsfähigkeit dar. Eine Steigerung der Energieeffizienz wird in der Regel durch einen höheren Spezialisierungsgrad der jeweiligen Ausführungsplattform erzielt.

Verfügt ein Prozessor z. B. über spezielle Instruktionssatzerweiterungen[1], um Matrixmultiplikationen abarbeiten zu können, ist er entsprechend besser für Anwendungen des maschinellen Lernens geeignet, da er die Kernoperationen optimaler, d. h. schneller in Bezug auf die Anzahl benötigter Takte einer gegebenen Taktfrequenz ausführen kann (Nurvitadhi et al., 2016).

General Purpose GPU (GPGPU), die über sehr viele kleine, spezielle Verarbeitungseinheiten verfügen, sind besonders effizient bei Problemstellungen mit hoher Datenparallelität (Daten können unabhängig voneinander und damit gleichzeitig bearbeitet werden), da sie nach dem SIMD[2]-Prinzip arbeiten. Zusätzlich profitieren GPGPU vom lokalen Speicher der Grafikkarte und damit verbundenen kürzeren Zugriffszeiten gegenüber einem entfernten Speicher wie dem Hauptspeicher eines Computers sowie einem breitbandigen Speicherinterface mit hoher Datenrate zum externen Speicher.

Eine weitere Effizienzsteigerung stellen applikationsspezifische Prozessoren, sogenannte ASIPs (application-specific instruction-set processors) dar. Bei diesen wird die grundsätzliche Programmierbarkeit eines speziellen Prozessors zugunsten der effektiveren Abarbeitung von KI-Algorithmen eingeschränkt. Ein typischer ASIP-Vertreter ist die Tensor Processing Unit (TPU) von Google (Google), welche speziell auf Anwendungen des maschinellen Lernens optimiert wurde und nur für ausgewählte ML-Verfahren programmierbar ist. Auf den bisher beschriebenen Plattformen werden in der Regel Algorithmen unter Einsatz einer spezialisierten Programmiersprache implementiert, die dann unter Anwendung der entsprechenden Entwicklungssoftware in einen auf dem jeweiligen Prozessor ausführbaren Code überführt werden.

Eine weitere Steigerung der Energieeffizienz lässt sich über die Entwicklung spezieller Hardwarebeschleuniger für bestimmte Halbleitertechnologien erreichen. Hier nehmen FPGA eine Sonderrolle ein, da sie zum einen den Vorteil besitzen, spezielle Hardwareimplementierungen abzubilden, und zum anderen ähnlich wie Central Processing Units (CPU) oder GPU während des Betriebs rekonfigurierbar sind, womit eine eigene Art der Programmierbarkeit gegeben ist. Durch ihre spezielle Hardwareimplementierung können FPGA auch irreguläre Verarbeitungsstrukturen gut abbilden. Gleichzeitig enthalten sie Tausende Hardwaremultiplizierer, welche für neuronale Netze eine große Bedeutung haben. In Xia et al. (2019) wird ein FPGA-basiertes System beschrieben, das der umfangreichen parallelen Verarbeitung von tiefen neuronalen Netzen dienen soll. Der Fokus dieser Forschungsarbeit liegt auf der Flexibilität in Hinblick auf unterschiedliche Netzarchitekturen, wobei jedoch eine Kopplung der FPGA über ein Peripheral Component Interconnect Express (PCIe) Interface[3] vorgenommen wird. Bei den untersuchten Netzen zeigt sich ein leicht optimierter Datendurchsatz bei verbesserter Energieeffizienz. In Wang et al. (2020) wird die massive Parallelisierung des Trainings von Deep Convolutional Neural Networks (DCNN) mithilfe von FPGA gezeigt und eine Energieeinsparung um den Faktor vier beschrieben. Zudem verdeutlicht das beschrieben Verfahren zwar die Bedeutung der schnellen Netzwerkkommunikation, bietet aber keine Integration in Rechenzentren. Im Projekt Brainwave (Microsoft) verwendet Microsoft einen mittels PCIe eng an den Server gekoppelten FPGA für ML der Suchmaschine Bing. In dem Bericht dazu werden insbesondere die geringe Latenz bei zugleich hohem Durchsatz sowie die geringen Kosten von 21 US-Cent pro 1 Million Bildern hervorgehoben. Intel (Nurvitadhi et al., 2019) zeigt die Bedeutung von eng an Server gekoppelten FPGA-Beschleunigern für komplexe neuronale Netze.

---

[1] Erweiterung des Befehlssatzes einer CPU für spezielle Aufgaben, beispielsweise für Matrixmultiplikationen.

[2] Single Instruction Multiple Data (SIMD) beschreibt die parallele Verarbeitung von Datenwörtern, die sich in einem Prozessorregister befinden, jedoch eine getrennte arithmetische Verarbeitungseinheit verwenden.

[3] Standard für eine Erweiterungsschnittstelle eines PC, über die man z. B. Grafikkarten anbinden kann.

Die höchste Energieeffizienz ergibt, wie in Sze et all (2017) und Chen et al. (2016) beschrieben, die Umsetzung des Rechenbeschleunigers als Halbleiterchip (ASIC), die jedoch mit hohen Kosten verbunden ist, da die Fertigung eines Halbleiterbausteins erforderlich ist. Bei aktuellen Halbleiterprozessen belaufen sich allein die reinen Einrichtungskosten für die Fertigung auf mehrere Millionen Euro.

Für eine genauere Betrachtung in Hinblick auf den Einsatz energieeffizienter Rechenbeschleuniger sollen nachfolgend drei wesentliche Aspekte diskutiert werden.

- Zunächst wird die Frage erörtert, ob die Art und Weise, in der ML-Rechenbeschleuniger zurzeit in aktuelle Rechnerinfrastrukturen integriert werden, hinsichtlich der verwendeten Schnittstellen und erforderlichen Datenübertragungsraten optimal gelöst ist.
- Sodann wird eine exemplarische Verlustleistungsbetrachtung als Grundlage für die Beurteilung unterschiedlicher Verarbeitungsarchitekturen angestellt.
- Schließlich wird der Frage nachgegangen, ob es alternative Methoden der Integration von Rechenbeschleunigern in aktuelle Rechnerinfrastrukturen gibt.

2.1 Qualität der Integration von Rechenbeschleunigern in (aktuelle) Rechnerinfrastrukturen

Rechenbeschleuniger, wie z. B. GPGPU oder FPGA, die in Servern untergebracht sind, werden als Rechenknoten bezeichnet. Diese werden zurzeit ausschließlich auf der Basis interner Rechnerschnittstellen an das Trägersystem angekoppelt. Hierzu steht momentan alleinig das PCIe-Bussystem[4] zur Verfügung, das in nahezu allen gängigen Rechnersystemen vorhanden ist und für eine hohe Bandbreite von Anwendungen konzipiert wurde. Im Laufe einer evolutorischen Entwicklung wurde damit eine Schnittstelle geschaffen, die immer höhere Datenraten erlaubt, jedoch eine hochgradig enge physikalische Kopplung des Hostsystems mit der Zusatzkarte erfordert.

Auf Basis der in Abbildung 5 dargestellten Konfiguration wurden unterschiedliche Untersuchungen in Bezug auf die tatsächlich benötigten Datenraten von KI-Anwendungen angestellt.

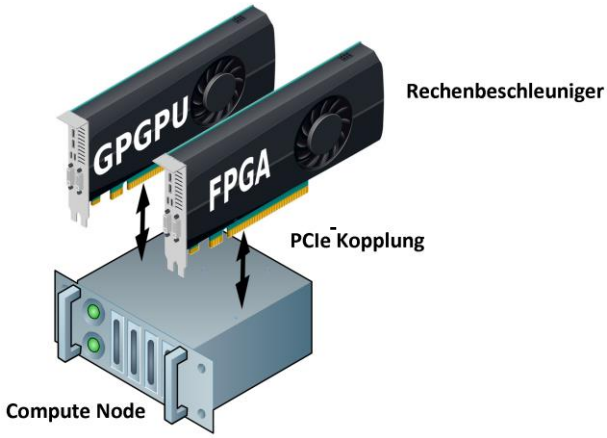

Abbildung 5: PCIe-basierte Anbindung an FPGA- bzw. GPGPU-basierte Rechenbeschleuniger

[4] Standardisiertes Bussystem zur Anbindung von Peripheriekomponenten an den Prozessor.

	PCI Express Version 4.0 x16	Ethernet 100 GE	Ethernet 10 GE
Theoretische Datenrate in GByte/s **Nutzdatenrate in GByte/s (50%)**	31,508 15,754	12,5 6,25	1,25 0,625
Mögliche Bildraten der Bildklassifikation 224 × 224 RGB Bildpunkte in FPS	104.658,2	41.520,5	4.152,05

Tabelle 1: Mit der Datenrate verschiedener Schnittstellen erzielbare Bildklassifikationsrate

Die Bildraten ergeben sich dabei als Quotient aus Nutzdatenrate und Bilddatenmenge. Jedes Bild benötigt eine Bilddatenmenge von 224 × 224 × 3 (RGB) Byte = 150.528 Byte pro RGB-Bild. Damit ergibt sich z.B. für Ethernet 10 GE: $0{,}625 \times 10^9$ Byte / 150.528 Byte = 4.152,05

In Tabelle 1 sind die Datenraten dargestellt, die für die Bildklassifikation im Falle einer Inferenzanwendung, z. B. mit MobileNetV2 , anfallen. Die Auflösungen der zu klassifizierenden Bilder betragen 224 × 224 Bildpunkten und entsprechen damit der mit vertretbarem Rechenaufwand zu bewältigenden Datenmenge.

Wie der letzten Zeile der Tabelle 1 zu entnehmen ist, ermöglicht die Nutzdatenrate die Übertragung mehrerer tausend Bilder (FPS – Frames per Second) selbst mit der 10 GE Schnittstelle. Dies übersteigt realistische Bildklassifikationsraten von Inferenzbeschleunigern um ein Vielfaches (Bianco, Cadene, Celona, & Napoletano, 2018). . Die dargestellten Berechnungen bestätigen das Funktionsprinzip von Rechenbeschleunigern im Allgemeinen, da die Kommunikationszeit sehr viel niedriger sein muss als die erforderliche Rechenzeit (Wang, Zhao, Li, Hou, & Gu, 2018).

2.2 Verlustleistungsbetrachtung

Im Folgenden soll die Frage geklärt werden, wie viel Energie eingespart werden kann, wenn man einen FPGA-Beschleuniger ohne Serverrechner betreibt, d. h. über eine Netzwerkschnittstelle direkt in einen Rechnerverbund integriert. Da eine Grafikkarte bzw. ein GPU-Beschleuniger diese Möglichkeit nicht bietet, sind die entsprechenden Messexperimente mit dem erforderlichen Trägerrechner durchzuführen. Weiterhin soll ermittelt werden, welche Energie benötigt wird, wenn überhaupt kein Rechenbeschleuniger eingesetzt, also eine reine Rechnerapplikation auf einem Server ausgeführt wird.

Wie der nachfolgenden Tabelle 2 entnommen werden kann, entfällt ein großer Teil der Verlustleistung eines Rechenknotens auf den Serverrechner selbst, der nur als Trägersystem und intelligente Schnittstelle für die eigentlichen Beschleuniger dient. Um die einzelnen Komponenten (Server, GPU, FPGA) separat untersuchen zu können, wurde ein spezielles Messkonzept eingesetzt, mit dem die in der Tabelle aufgeführten Ergebnisse ermittelt wurden. Dabei wurden sowohl die Gesamtleistung des Servers als auch die Verlustleistung einzelner PCIe-Karten gemessen (Ruloff, 2019).

In Tabelle 2 wird der Energiebedarf für die Bildklassifikation mit dem KNN-Modell MobileNetV2 dargestellt. Die Durchschnittsleistung bezeichnet die Leistung, die ein Rechenknoten durchschnittlich verbraucht hat, um die Operation der Bildklassifikation durchzuführen. Die Leerlaufleistung ist derjenige Anteil des Leistungsbedarfs, der durch den Rechenknoten verbraucht wird, während nicht am Problem gerechnet wird. Für die Messungen wurden folgende Systeme verwendet:

- Workstation: Xeon Gold 6128, 64 GByte DDR4 RAM mit einer Quadro P2000 GPU
- Server: Xeon Sliver 4114, 8 GByte DDR4 RAM mit zwei Tesla V100
- FPGA: Bittware 385A, Standalone-Messung

Test	Durchschnitts-leistung	Anteil der Leerlauf-leistung	Energiebedarf pro Bildklassifikation
Leerlauf Workstation	84 W	100 %	
Betrieb Workstation mit CPU	245 W	34,3 %	3,7 J
Betrieb Workstation mit GPU (PCIe-Kopplung)	217 W	38,7 %	1,3 J
Leerlauf Server	148 W	100 %	
Betrieb Server mit GPU (PCIe-Kopplung)	229 W	64,6 %	0,84 J
Leerlauf FPGA (Netzwerk, kein Trägerrechner erforderlich)	20 W	100 %	
Betrieb FPGA (Netzwerk, kein Trägerrechner erforderlich)	35 W	57,1 %/0 %	0,092 J

Tabelle 2: Energiebedarf für Bildklassifikationen

Die Bildklassifikation durch CPU bietet sich aufgrund des hohen Energiebedarfs pro Bild nicht an. Bei der Klassifikation durch GPU werden mindestens 38 Prozent der benötigten Leistung bereits im Leerlauf durch den Server benötigt. Da diese Leistung durchgehend benötigt wird, bietet sich hier ein erhebliches Einsparpotenzial. Auch die Nutzung von FPGA als energieeffiziente Hardwarebeschleuniger bietet Einsparpotenzial (Fallahlalehzari). Die Implementierung von MobileNetV2 auf Basis rekonfigurierbarer Hardware (FPGA) benötigt verglichen mit der effizientesten GPU-Lösung nur 11 Prozent der Energie. Damit sinkt der Energieeinsatz zur Lösung der Problemstellung um den Faktor 9,1. Die Nutzung rekonfigurierbarer Hardware ermöglicht durch deren sehr geringe Startzeiten (< 1 Sekunde) das komplette Abschalten der Hardware, wodurch der Stromverbrauch des Rechenbeschleunigers in der Zeit der Abschaltung komplett entfällt.

2.3 Alternative Methoden der Integration von Rechenbeschleunigern

Neben den oben erwähnten PCIe-Schnittstellen bieten herkömmliche Rechnersysteme eine Vielzahl weiterer Schnittstellen, deren wichtigster Vertreter die hochratige (> 10 GBit/s), standardisierte Ethernet-Schnittstelle ist. Ethernet-Schnittstellen werden in Rechenzentren, neben Infiniband[5], für die Vernetzung von Rechnersystemen verwendet, um auf die bestehende Infrastruktur zuzugreifen. Insofern bietet sich Ethernet als Schnittstelle für die Kommunikation mit Hardwarebeschleunigern an. Jedoch besteht der Nachteil, dass die entsprechenden Protokolle zumeist auf der Basis von Softwareimplementierungen umgesetzt werden. Dies führt zu einer indirekten Abwicklung der Kommunikation über ein Hostsystem, ohne direkte Hardwareverbindung mit dem FPGA-basierten ML-Algorithmus.

Die ideale Lösung wäre eine Konfiguration, in der die Schnittstelle direkt auf dem Beschleuniger untergebracht ist und kein zusätzlicher Hostrechner für den Datentransfer benötigt wird. Dafür ist der Einsatz eines komplett hardwarebasierten Netzwerkprotokollstapels erforderlich, der es erlaubt, beliebige Rechenbeschleuniger über standardisierte Netzwerkschnittstellen anzusprechen und zu betreiben. Die Schnittstelle ist dann, je nach verwendeter Technologie, mit bis zu 100 Gbit/s betreibbar und liefert damit ähnlich hohe Datenraten wie herkömmliche Bussysteme. Diese ideale Lösung als reines Hardwaresystem hat weitere Vorteile, wie z. B. niedrige Latenz, schnelle Reaktionszeiten und eine Optimierung der Ausfallsicherheit, da auf jegliche Art von Software verzichtet werden kann.

Zusammenfassend lässt sich hier feststellen: Die Umsetzung hardwarebasierter Netzwerkprotokolle erlaubt es, Hardwarebeschleuniger direkt mit einer hochratigen Netzwerkschnittstelle auszustatten, wie sie für die Integration in Rechenzentren benötigt wird, ohne dabei ein Rechnersystem einsetzen zu müssen.

[5] Hochgeschwindigkeitsnetzwerk zur Nutzung in Rechenzentren mit geringer Latenz.

3 Energieeffizienz bei der Übertragung von KI-Modellen

Nachfolgend werden Energieeinsparpotenziale bei Prozessen der Übertragung von KI-Modellen von einem Sender zu einem Empfänger erörtert und eine passende Untersuchungsumgebung zur Analyse dieser Potenziale vorgestellt.

Wie bereits in der Einleitung beschrieben, kommen als KI-Modelle üblicherweise tiefe neuronale Netzwerke zur Anwendung, die in sequenziell angeordneten Schichten aufgebaut sind. Dabei stellen die Verbindungen zwischen den Input- und den Output-Neuronen einer KNN-Schicht trainierbare Gewichte in einer Transformationsmatrix dar. Im Folgenden wird davon ausgegangen, dass ein konkretes Modell mit seinen Parametern vorliegt, welches möglichst effizient von einem Sender zu einem Empfänger übertragen werden soll. Ein Beispiel hierfür könnte sein, dass für ein autonom fahrendes Fahrzeug ein durch eine verbesserte Datenlage aktualisiertes Modell auf einem Server vorliegt, welches in das Fahrzeug zur Optimierung von dessen Entscheidungsfindung übermittelt werden soll. Eine weitere konkrete Anwendung, bei der KI-Modelle bzw. deren Aktualisierungen von einem Sender zu einem Empfänger verschickt werden, ist der verteilte Lernansatz des Föderierten Lernens (engl. Federated Learning), bei dem mehrere Teilnehmende parallel ein globales Modell trainieren. Dafür trainiert jeder Teilnehmende mit seinen Trainingsdaten ein lokales Modell und schickt dieses sodann an einen zentralen Server, der alle empfangenen Modelle zu einem neuen globalen Modell für die nächste Trainingsiteration zusammenfasst (z. B. durch Mittelwertbildung pro Parameter). Dieser verteilte Lernansatz eignet sich sehr gut, um die Energieeffizienz bei der Übertragung von KI-Modellen systematisch zu untersuchen. Bevor jedoch auf die konkrete Anwendung des Föderierten Lernens näher eingegangen wird, müssen zunächst Möglichkeiten zur effizienten Übertragung von KI-Modellen betrachtet werden.

3.1 Direkte Übertragung ohne Vorverarbeitung als Benchmark

Der Trend zu immer größeren KI-Modellen wirkt sich nicht nur negativ auf die Energiebilanz bei deren Training, sondern aufgrund der steigenden dezentralen Verbreitung von KI auch zunehmend auf den Energiebedarf für die Übertragung aus. Das in Kapitel 1.2 bereits erwähnte KI-Modell VGG16, ein neuronales Netz mit 21 Schichten, von denen 16 Schichten zusammen über 138 Mio. Gewichte bzw. Parameter enthalten. Die restlichen 5 Schichten, die sog. Max-Pooling-Layer, haben keine eigenen Parameter, sondern leiten nur jeweils einen maximalen Wert aus vier Aktivierungen der vorangegangenen Schicht weiter. Das VGG16-Modell kommt auf eine Größe von mehr als 550 MB.

Bei der Übertragung von KI-Modellen sind jedoch neben der zu übertragenden Datenmenge noch weitere Dimensionen von Energieeffizienzbetrachtungen zu berücksichtigen. So haben Latenzanforderungen und die zur Verfügung stehende Bandbreite einen wesentlichen Einfluss auf die Gesamtbilanz bei der Übertragung, denn sie geben die Rahmenbedingungen vor, unter denen ein KI-Modell vom Sender zum Empfänger verschickt wird. Ist die Latenzanforderung hoch, jedoch die Bandbreite gering, wie etwa in kritischen Anwendungen wie dem autonomen Fahren, so muss gegebenenfalls zunächst Energie aufgewendet werden, um das Modell so stark zu komprimieren, dass die Latenzanforderung unter der gegebenen Bandbreite eingehalten werden kann. Dieses Beispiel verdeutlicht, dass die gesamtheitliche Energiebilanz eines KI-Modells nicht nur beim Training, sondern auch bei der Übertragung maßgeblich beeinflusst werden kann, wodurch sich auch dort Möglichkeiten zur Optimierung mit entsprechender Energieeinsparung bieten.

3.2 Parameterreduktion

Standardmodelle, wie z. B. das oben vorgestellte VGG16-Modell, weisen Millionen von trainierbaren Parametern auf, und gerade bei kleinen Trainingsdatensätzen führt diese Überparametrisierung häufig zum sog. Overfitting, also einer Art Auswendiglernen bzw. Codierung des Trainingsdatensatzes. Aber auch Modelle, die über eine gute Verallgemeinerung auf ungesehenen Daten verfügen, sind meist überparametrisiert; d. h., es existieren oft kleinere Modelle mit wesentlich weniger Parametern, die dieselbe Aufgabe mit annähernd gleicher Genauigkeit erledigen. Um dieses Problem zu lösen, werden Methoden zur Parameterreduktion verwendet, welche komplexe überparametrisierte KNN in kleinere Modelle überführen. Hierfür wurden unterschiedliche Verfahren entwickelt, die im Folgenden näher vorgestellt werden.

Sparsifikation. Bei diesem Verfahren versucht man diejenigen Gewichte bzw. Parameter eines KNN zu Null zu setzen, die eine geringe Relevanz für das Gesamtverhalten des KNN (z. B. die Klassifikationsgüte) haben. Aufgrund der komplexen Zusammenhänge innerhalb eines tiefen KNN ist es bei diesem Verfahren schwierig, die Relevanz eines jeden einzelnen Parameters oder einer Gruppe von Parametern (z. B. eines konkreten Filters) effizient zu ermitteln, da zunächst kein direkter Zusammenhang zwischen dem absoluten Zahlenwert eines Parameters und seinem Einfluss auf das Gesamtverhalten des KNN besteht. Daher führen insbesondere Methoden, die betragsmäßig kleine Parameter zu Null setzen, häufig nicht zu akzeptablen Ergebnissen. Zudem nimmt auch ein zu Null gesetzter Wert zunächst noch Speicherplatz in Anspruch und muss schlussendlich für die Übertragung codiert werden.

Pruning. Im Gegensatz zur Sparsifikation werden beim Pruning alle irrelevanten Parameter aus dem KNN gelöscht, sodass sie auch keinen Speicher mehr einnehmen. Allerdings ergibt Pruning nur dann Sinn, wenn es im Ausgangsmodell in regelmäßigen oder strukturierten Teilen erfolgt, z. B. indem ganze Filter oder Neuronen mit ihren Verbindungen zu anderen Neuronen gelöscht werden. Die unstrukturierte Löschung einzelner Gewichte bzw. Parameter ist nicht sinnvoll, da die Position der unstrukturiert gelöschten Parameter wiederum gespeichert und codiert werden muss. Häufig geht das Pruning mit signifikanten Einbußen bei der Güte des KNN einher, insbesondere dann, wenn man nach dem strukturierten Löschen kein Retraining des KNN vornimmt.

Niedrigrang-Approximation und -Zerlegung. Das Ziel dieser Methode ist es, die Transformationsmatrix einer Schicht durch ein Produkt von zwei Matrizen mit wesentlich weniger Parametern zu approximieren, um die Ausgangsmatrix effizienter übertragen zu können. Wenn also z. B. eine $(n \times m)$-dimensionale Ausgangsmatrix in zwei Matrizen mit den Dimensionen $(n \times k)$ und $(k \times m)$ zerlegt werden kann, so hat man anstatt $n \times m$ Parameter nun $k \times (n + m)$ Parameter, was sich gerade bei einem kleinen k bemerkbar macht. Im Allgemeinen ist eine solche Zerlegung gerade für kleine k nur dann möglich, wenn anstelle der Ausgangsmatrix eine Approximation verwendet wird. Bei diesem Verfahren muss üblicherweise ein rechenintensives Optimierungsproblem gelöst werden.

Batchnorm-Faltung. Wie bereits für das VGG16-Modell beschrieben, sind nicht alle Schichten eines Netzwerks mit Parametern behaftet, sondern einige führen lediglich die Maximierung oder Normalisierung von Aktivierungen vorangegangener Schichten durch. Im speziellen Fall der sog. Batchnorm-Schicht findet dort eine Normalisierung für die Mittelwerte und Standardabweichungen vorangegangener Aktivierungen statt. Bei der Batchnorm-Faltung wird diese Normalisierung direkt mit den Parametern der davorliegenden Schicht fusioniert; es werden also zwei Transformationsmatrizen miteinander multipliziert, wodurch die ursprünglichen zwei Schichten für die Codierung zu einer zusammengefasst werden. Allerdings ist zu beachten, dass dieser Schritt nicht reversibel ist.

3.3 Quantisierung

Generell wird unter Quantisierung die Approximation einer großen Menge durch eine kleine Menge verstanden. Dementsprechend besteht das Ziel der Quantisierung von KNN darin, die Parameter bzw. Gewichte in niedrigerer Präzision darzustellen und zu codieren. So ist es beispielsweise möglich, die Parameter nicht in der während des Trainings üblicherweise verwendeten 32-Bit- oder 64-Bit-Gleitkommazahlen (engl. Floating Point) darzustellen, sondern in Festkommazahlen oder ganzzahlig mit entsprechender Skalierung. Im Folgenden werden die gebräuchlichsten Formen der Quantisierung näher vorgestellt. Es sei aber bereits hier erwähnt, dass eine Quantisierung der Netzparameter nach dem Training oftmals zu erheblichen Einbußen in der Güte des KNN führt, was wiederum durch Retraining oder Quantisierungsbewusstes Training (engl. Quantization Aware Training – QAT) ausgeglichen werden kann. Beim QAT wird die vorgesehene Quantisierung des KNN bereits während des Trainings durch sog. Fake-Quantisierung (Quantisierung mit sofortiger Dequantisierung) berücksichtigt, sodass sich das KNN bereits in der Trainingsphase darauf einstellen kann.

Gleichmäßige skalare Quantisierung. Bei diesem Verfahren wird der durch Kalibrierung ermittelte Wertebereich der zu quantisierenden Parameter bzw. Aktivierungen durch abstandsgleiche Rekonstruktionswerte repräsentiert, sodass sich die Rekonstruktion jedes quantisierten Ausgangswerts durch einfache Multiplikation des Quantisierungsindexes mit der Quantisierungsschrittweite ergibt (sog. Dequantisierung). Die Differenz zwischen Ausgangswert und dazugehörigen Rekonstruktionswert entspricht dem Quantisierungsfehler. Diese Methode führt oftmals zu großen Fehlern, da sie die Verteilung über den Wertebereich vernachlässigt. Sie wird aber aufgrund ihrer einfachen Implementierbarkeit dennoch sehr oft eingesetzt.

Codebuch-Quantisierung. Bei dieser Form der Quantisierung werden die Ausgangswerte beliebig über den Wertebereich verteilten Clustern als Rekonstruktionswerte zugewiesen, wodurch man der tatsächlichen Verteilung der Werte über dem Wertebereich besser gerecht werden und somit den Quantisierungsfehler verringern kann. Da bei diesem Verfahren die Lage der Clusterzentren meist durch Optimierungsmethoden aufwendig ermittelt wird und zudem zusätzlich codiert werden muss, wird in der Praxis oft auf die oben beschriebene einfachere gleichmäßige Quantisierung zurückgegriffen.

Abhängige Quantisierung. Diese spezielle Vektorquantisierungsmethode verbessert die klassische gleichförmige skalare Quantisierung. Dazu werden anstatt nur einem zwei Quantisierer verwendet und die jeweilige Zuordnung zu einem der beiden findet auf Basis der zuvor getroffenen Quantisierungsentscheidungen statt. Damit erreicht dieses Verfahren kleinere Quantisierungsfehler, geht jedoch mit einer Abhängigkeit zwischen zwei aufeinanderfolgenden Quantisierungs- bzw. Rekonstruktionsschritten einher.

3.4 Verlustfreie Komprimierungsformate und Codierung

Bevor Modelle vom Sender zum Empfänger verschickt werden, durchlaufen sie einen oder mehrere der oben beschriebenen Vorverarbeitungsschritte zur Parameterreduktion und Quantisierung. Anschließend werden die resultierenden Modelle meist noch verlustfrei komprimiert, entweder durch Überführung in ein effizienteres Format (z. B. dem Compressed-Sparse-Row/Column (CSR/CSC)-Format), oder durch Verwendung der Entropiecodierung zur Erzeugung eines Bitstroms für die effiziente Modellübertragung. Die Entropiecodierung setzt einen Encoder auf Senderseite mit entsprechendem Decoder auf Empfängerseite voraus und basiert auf der Grundidee, für Symbole, die öfter auftreten, also weniger Informationsgehalt repräsentieren, entsprechend weniger Bits zuzuweisen. Ein sehr bekanntes Verfahren zur Entropiecodierung ist die Huffman-Codierung, die jedem Quellsymbol ein Codewort zuordnet. Als sehr effiziente Erweiterung der Huffman-Codierung

hat sich die arithmetische Codierung herausgestellt, bei der mittels Wahrscheinlichkeitsschätzungen für die Quellsymbole die gesamte Nachricht oder ein Teil von ihr in Form einer rationalen Zahl codiert werden kann.

DeepCABAC. Das Basisverfahren für diese Methode bildet das am Fraunhofer HHI entwickelte CABAC (Context-Based Adaptive Binary Arithmetic Coding), welches eine Weiterentwicklung der klassischen arithmetischen Codierung ist. CABAC ist als effiziente Methode zur verlustfreien Komprimierung Bestandteil weltweiter Videocodierstandards. Das Verfahren wurde für die Kompression neuronaler Netzwerke angepasst (Wiedemann et al., 2020). DeepCABAC erlaubt mit seiner adaptiven, kontextbasierten Ratenmodellierung eine optimale Quantisierung und Codierung der Parametermatrizen eines neuronalen Netzes und somit eine sehr effiziente Kompression ohne Güteverluste.

3.5 NNC-Standard als Werkzeugkasten zur effizienten Übertragung Neuronaler Netze

Der Zweck des Neural-Network-Coding (NNC)-Standards (ISO/IEC 15938-17, für einen Überblick siehe auch Kirchhoffer et al. (2021)) ist es, eine hohe Kompressionsrate für neuronale Netze möglichst ohne Qualitätsverluste zu erzielen. Dabei bezieht sich hier die Qualität auf die Klassifizierungsgenauigkeit. Um dieses Ziel zu erreichen, verwendet NNC eine Reihe von Codierwerkzeugen und -methoden, unter anderem zur Parameterreduktion, Quantisierung und Entropiecodierung. Je nach Bedarf und Anwendungsszenario können diese Methoden individuell miteinander kombiniert werden. Zudem kann der Standard problemlos in gängige Industrieformate für neuronale Netze integriert werden. Wie bereits andere Multimedia-Standards zur Datenkompression spezifiziert auch der NNC-Standard nur den Bitstrom und den Decoder als normative Bestandteile, während Methoden zur Encodierung als nicht-normative Beispiele beschrieben sind. Damit kann der Encoder von jeder nutzenden Person oder jedem Hersteller selbst implementiert und gegebenenfalls optimiert werden. Die allgemeine NNC-Struktur mit Encoder und Decoder ist in Abbildung 6 dargestellt.

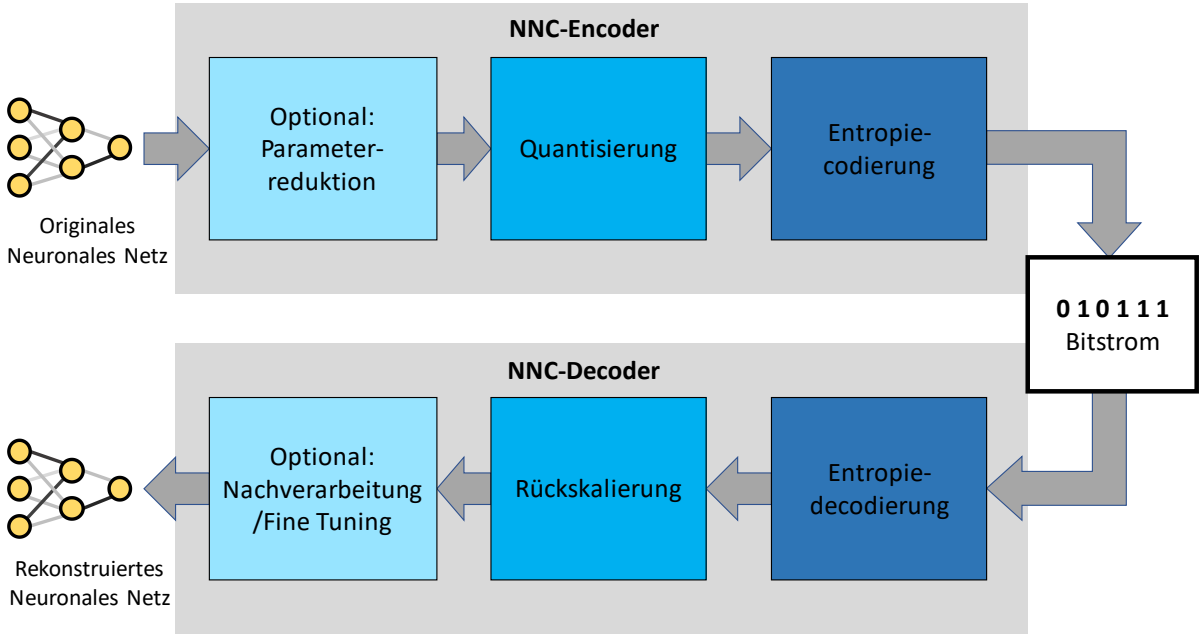

Abbildung 6: Schematische Darstellung der NNC-Struktur mit Encoder und Decoder

Zur Veranschaulichung der Effizienz des neuen NNC-Standards sind in Tabelle 3 Codierergebnisse für unterschiedliche neuronale Netze aufgeführt. Dabei sind die ersten drei Netze, VGG16, ResNet50 und MobileNetV2, für die Bildklassifizierung und das vierte Netz, DCase, für die Audioklassifizierung trainiert worden.

Model	Kompression	Top-5-Genauigkeit, rekonstruiertes NN	Top-5-Genauigkeit, originales NN	Originalgröße
VGG16	2,98 %	89,54 %	89,85 %	553,43 MB
ResNet50	6,54 %	91,80 %	92,15 %	102,55 MB
MobileNetV2	12,18 %	90,06 %	90,27 %	14,16 MB
DCase	4,12 %	92,35 %	91,85 %	467,26 kB

Tabelle 3: Codierergebnisse ohne Qualitätsverlust für unterschiedliche neuronale Netze

Für jedes neuronale Netz sind in Tabelle 3 die erreichte Kompression, die Top-1- und Top-5-Genauigkeiten bei der Bild- bzw. Audioklassifikation für das rekonstruierte und decodierte neuronale Netz sowie die Originalgröße in Byte angegeben. VGG16 kann beispielweise auf 2,98 Prozent seiner Originalgröße komprimiert werden, also von 553 auf ca. 16,5 Mbyte, bei nahezu gleicher Klassifizierungsgenauigkeit von > 70 % für Top-1 und > 89 % für Top-5. Die aufgeführten Genauigkeiten beziehen sich auf den Prozentsatz an gewünschten Ergebnissen (z. B. alle Hunde aus einer Tierbilder-Datenbank), den das neuronale Netz in dem ersten Prozent (Top-1) oder den ersten 5 Prozent (Top-5) der ähnlichsten Daten findet. In der Tabelle 3 sind diese Genauigkeiten sowohl für das originale als auch für das rekonstruierte neuronale Netz angegeben, um die gleichbleibende Klassifizierungsqualität bei den jeweiligen Kompressionen zu zeigen. Diese wiederum geben an, auf welchen Prozentsatz seiner Originalgröße das jeweilige neuronale Netz ohne Qualitätsverlust komprimiert werden kann. Als Beispiel sei hier wieder VGG16 herausgegriffen: das originale Netz hat bei einer Auflösung von 4 Byte pro Parameter eine Gesamtgröße von mehr als 552 Mbyte, was etwa dem durchschnittlichen Mobilfunk-Datenvolumen pro Monat in Deutschland im Jahr 2016 entspricht (Statista). Mit einer Kompression von 2,98 Prozent kann der Bitstrom auf 16,5 Mbyte reduziert werden. Zudem erreicht NNC wesentlich höhere Kompressionen von unter 1 Prozent bei verlustbehafteter Codierung, d. h., wenn ein Verlust an Klassifizierungsgenauigkeit in Kauf genommen wird.

3.6 Federated Learning als Anwendungsfall zur systematischen Untersuchung

Federated Learning (FL; deutsch Föderiertes Lernen) bezeichnet eine maschinelle Lernumgebung, in der viele Teilnehmer (z. B. mobile IoT-Geräte oder ganze Organisationen) kollaborativ ein globales KI-Modell unter der Orchestration eines zentralen Servers trainieren, während die Trainingsdaten dezentralisiert bleiben (Kairouz et al., 2021). Beim Federated Learning wird also die komplexe und zeitaufwendige Trainingsphase auf die Teilnehmer verteilt und damit parallelisiert. Dafür vereinbaren die Teilnehmer zunächst in dem unter ihnen gebräuchlichsten FL-Protokoll eine Netzarchitektur für das globale Modell. Jeder Teilnehmer trainiert dann eine lokale Version des Modells (mit dem globalen Modell gleichender Architektur) mit seinen lokalen Trainingsdaten. Da sich die Trainingsdaten der einzelnen Teilnehmer in der Praxis üblicherweise statistisch unterscheiden, werden hierdurch auch leicht unterschiedliche Versionen des neuronalen Netzes erzeugt.

Diese lokalen Versionen werden anschließend an das lokale Training von allen Teilnehmern an einen zentralen Server geschickt, welcher eine neue einheitliche Version des neuronalen Netzes aus allen erhaltenen Varianten erzeugt, z. B. als parameterweise Mittelwerte aller Versionen. Diese neue Version wird wiederum an alle Beteiligten verteilt, woraufhin eine neue Lern- oder Kommunikationsrunde beginnt. Einen schematischen Überblick über das beschriebene Vorgehen gibt Abbildung 7.

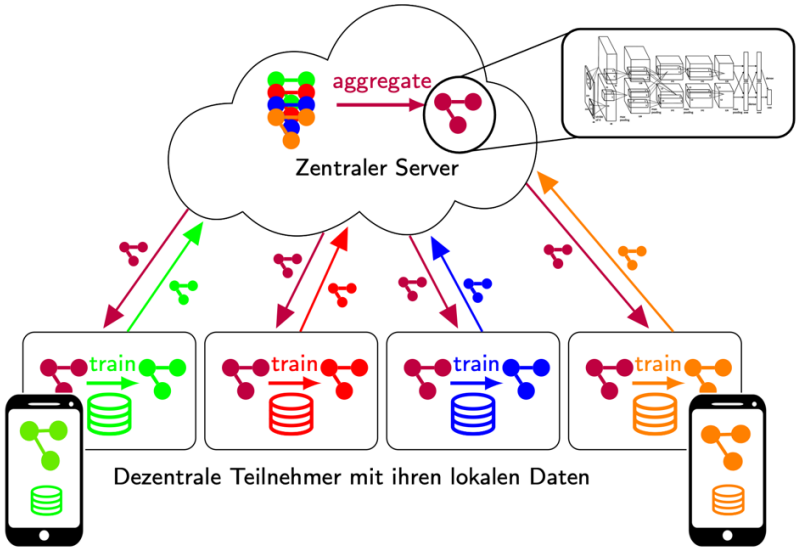

Abbildung 7: Schematische Übersicht des Federated Learning

Damit ergeben sich beim FL durch die wiederholte Übertragung neuronaler Netzwerk-Updates Echtzeitanforderungen, die die Anwendung von Methoden zur Komprimierung der übertragenen Modellparameter unerlässlich machen. Dies gilt insbesondere für Anwendungsfälle, in denen Millionen von mobilen Geräten gleichzeitig am Lernverfahren teilnehmen und dabei oftmals nur über eine langsame Verbindung mit dem Server kommunizieren. Diese Anwendungsfälle werden in der Literatur häufig Cross-Device-FL genannt wird. Der Ansatz des FL mit seinen spezifischen Anforderungen an Latenz und seinen Herausforderungen bezüglich der Bandbreite stellt damit eine ideale Umgebung für die systematische Untersuchung von Energieeinsparpotenzialen bei der Übertragung von KI-Modellen dar.

Im Folgenden werden erste Untersuchungen zur Energieeffizienz und Vorarbeiten dazu für die Untersuchungsumgebung des Federated Learning vorgestellt. Die Arbeit von Sattler et al. (2019) untersucht, wie sich die kombinierte Anwendung von Parameterreduktion, beabsichtigter Verzögerung in der Kommunikation mit dem Server, starker Quantisierung (Binarisierung) und Codierung auf die Kommunikationskosten innerhalb von FL auswirkt. Die Arbeit zeigt, wie auch die Untersuchungen zum in Abschnitt 3.5 beleuchteten NNC-Standard belegen, dass bei der Übertragung von Parametern von KI-Modellen oder deren Aktualisierungen erstaunlich hohe Kompressionsraten möglich sind. Diese Erkenntnis bestätigte sich erneut in der kürzlich erschienen Studie von Sattler et al. (2021) zu einem alternativen Ansatz für das FL mittels Wissensdestillation (siehe Abschnitt 3.2). Tabelle 4 liefert einen Überblick über die damit erzielten Einsparungen.

Szenario	Kommunikationskosten (bei gleichbleibender Accuracy)	Einsparung in der Kommunikation
Federated Learning Parameterübertragung): ResNet50, ImageNet	Federate Averaging (Google): ≈ 125.000 GB pro Client Sparse Binary Compression: ≈ 3,35 GB pro Client (Sattler, Wiedemann, Müller, & Samek, 2019)	> 99 %
Federated Learning (neuer Distillation-Ansatz): VGG16, CIFAR-10	Federate Averaging (Google): ≈ 1.300 MB pro Client Compressed Federated Distillation: ≈ 0,2 MB pro Client (Sattler, Marban, Rischke, & Samek, 2021)	> 99 %
Einmalige (vollständige) Übertragung eines trainierten Models: VGG16	Unkomprimiert: ≈ 550 MB DeepCABAC: ≈ 16,5 MB	> 97 %

Tabelle 4: Überblick über erzielte Einsparungen

Diese hohen Kompressionsraten erfordern jedoch zunächst einen zusätzlichen Energieeinsatz auf der Senderseite, womit sich die folgenden Forschungsfragen stellen:

- Welche Energiebilanz weisen die einzelnen Kompressionsmethoden auf und welche Methoden sind daher bei gegebener Latenz und Bandbreite aus Energieeffizienzgründen vorzugswürdig?
- Kann durch Kompression insgesamt Energie eingespart werden?

Die erste Frage ist vergleichbar mit der Frage nach dem effizientesten Transportmittel, um in einer vorgegebenen Zeit von A nach B zu kommen. Die zweite Frage kann mit der Frage verglichen werden, ab wann sich ein Investment (hier die für die Kompression erforderliche Energie) lohnt bzw. (durch Energieeinsparungen bei der Gesamtübertragung) amortisiert.

Es gilt also herauszufinden, ab wann die enormen Einsparungen bei der Kommunikation in tatsächliche Energieeinsparungen münden.

Einen ersten Schritt zur Beantwortung dieser Fragen hat die Arbeit von Qiu et al. (2021) geleistet. Sie beschreibt erstmalig eine systematische Untersuchung eines CO_2-Fußabdrucks für den dezentralen Lernansatz des Federated Learning und vergleicht diesen mit dem des klassischen zentralen Lernansatzes, bei dem alle Daten zentralisiert auf einem Server vorliegen. Die Arbeit zeigt, dass in Abhängigkeit vom verwendeten Energiemix (also dem Verhältnis umweltfreundlicher gegenüber nicht umweltfreundlicher Energieerzeugung) der dezentrale Lernansatz eine bessere Bilanz aufweisen kann als der zentrale Lernansatz. Darüber hinaus zeigt die Studie, dass der Anteil des Energieeinsatzes für die Kommunikation stark von der Größe des KI-Modells, der Größe des Trainingsdatensatzes bei jedem Teilnehmer und der von den Teilnehmern während des Trainings verbrauchten Energie abhängt. Dabei kann die Kommunikation im Durchschnitt sehr einen sehr unterschiedlichen Anteil der Gesamtemission ausmachen, z. B. lediglich 0,4 Prozent bei ImageNet und im Gegensatz dazu 95 Prozent (bei CIFAR10). Der Anteil der Kommunikation an den Gesamtemissionen schwankt somit stark und bedarf weiterer Untersuchungen hinsichtlich möglicher Energieeinsparungen.

4 Ausblick

Verfahren der KI bzw. des maschinellen Lernens konnten durch stetig steigende Rechenleistung, die Verfügbarkeit großer Datenmengen infolge der Digitalisierung und den Einsatz von tiefen neuronalen Netzwerken in den letzten Jahren entscheidend verbessert werden. KI gilt als Schlüsseltechnologie, die in vielen Bereichen bereits zu einem Umbruch führt bzw. führen wird. Die zunehmende Bedeutung von KI-Anwendungen geht allerdings gerade durch die steigende Komplexität und die damit erforderliche immer höhere Rechenleistung mit einem erhöhten Stromverbrauch und entsprechend erhöhten CO_2-Emissionen und Kosten einher. Diese gilt es dringend zu reduzieren. Die Analyse und Demonstration der Energieeffizienz bei der Ausführung und Übertragung von KI-Modellen und die Untersuchung weiterführender Energieeffizienzpotenziale mit Fokus auf die speziellen Bedürfnisse der Digital- und Energiewirtschaft sind daher von hoher Relevanz. Die vorliegende Analyse gibt einen Überblick über bestehende Ansätze der beiden Hauptbereiche Ausführung und Übertragung neuronaler Netze.

Die Studie bildet die Grundlage für die Untersuchungsumgebung für das Pilotprojekt Energieeffiziente KI (EEKI), welches im Rahmen des Future Energy Lab, des Pilotierungslabors für die Digitalisierung der Energiewende der Deutschen Energie-Agentur GmbH im Auftrag des Bundesministeriums für Wirtschaft und Klimaschutz (BMWK) durchgeführt wird.

4.1 Pilotprojekt Energieeffiziente künstliche Intelligenz

In dem Projekt werden beispielhafte Untersuchungen zur energieeffizienten Ausgestaltung von KI-Modellen durchgeführt. Dabei wird auf die beiden in dieser Analyse näher beschriebenen Ansätze für energieeffiziente Gestaltung, bei der Ausführung sowie bei der Übertragung von KI-Modellen, fokussiert. Im Folgenden wird ein Ausblick der im Rahmen des Projekts zu bearbeitenden Forschungsfragen und des Versuchsaufbaus gegeben.

Energieeffizienz bei der Ausführung von KI-Modellen

Die Ausführungen in Kapitel 2 machen deutlich, dass die Art und Weise, in der Rechenbeschleuniger zurzeit verwendet und in Rechenzentren integriert werden, eine Reihe von Eigenschaften zeigt, die zu hochgradig ineffizienten Systemen in Bezug auf ihren Energieverbrauch führen. FPGA-basierte Systeme bieten eine Möglichkeit, den Stromverbrauch zu reduzieren. Allerdings wird für die Kommunikation mit den FPGA zurzeit das PCIe-Bussystem verwendet, weswegen eine Vielzahl der hierfür benötigten Rechnersysteme ausschließlich als intelligente Netzwerkschnittstelle für die Rechenbeschleuniger dient.

Auf Basis der in Kapitel 2 vorgestellten Protokollimplementierung in Form reiner Hardware ist es möglich, einen FPGA-basierten KI-Hardwarebeschleuniger direkt mit einer Ethernetschnittstelle auszustatten. Das nachfolgende Blockschaltbild verdeutlicht die zugrundeliegende Hardwarearchitektur.

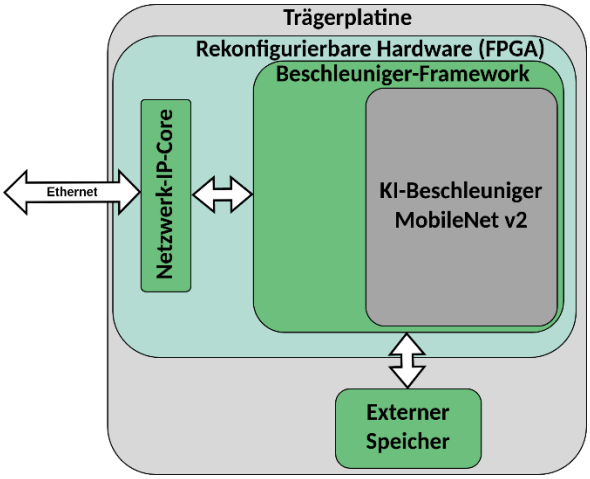

Abbildung 8: Blockschaltbild

Folgende Vorteile würden sich als Alleinstellungsmerkmale gegenüber den etablierten Technologien ergeben:

- Geringerer Energiebedarf: Der Einsatz der FPGA-Technologie in Form netzwerkgekoppelter KI-Beschleuniger zieht im Vergleich zu etablierten Technologien niedrigere Stromkosten nach sich. Durch die geringere Leistungsaufnahme wird darüber hinaus die Abwärme der betreffenden Systeme stark reduziert, was wiederum Energie für ihre Kühlung einspart.
- Niedrige Betriebskosten: Durch die Absenkung des Energiebedarfs wird zugleich eine Senkung der Betriebskosten und der CO_2-Emission erzielt.
- Geringerer Platzbedarf: Das in Abbildung 8 wiedergegebene Blockschaltbild zeigt vereinfacht die notwendigen Komponenten eines netzwerkgekoppelten KI-Beschleunigers. Die Bauform entspricht einer 3,5-Zoll-Festplatte, im Gegensatz zu einem sehr viel größeren Serverrechner.
- Einfache Systemintegration: Durch die Verwendung von Netzwerkschnittstellen wird es sehr einfach sein, die KI-Beschleuniger in bestehende Netzwerkinfrastrukturen zu integrieren. Der Vorteil besteht dabei in der direkten Hardwareverbindung und damit dem Wegfall der Notwendigkeit systemspezifischer Software wie z. B. PCIe-Treiber. Für die Installation kann daher auch weniger geschultes Personal eingesetzt werden.
- Niedrigere Latenz und hohe Resilienz: Durch den Einsatz hardwarebasierter Netzwerkprotokolle wird eine höhere Reaktionsgeschwindigkeit bzw. niedrigere Latenz erzielt, als sie von softwarebasierten Netzwerkstacks erreicht werden kann. Weiterhin bietet die reine Hardwareimplementierung eine durch Software nicht erzielbare sehr hohe Ausfallsicherheit.

Energieeffizienz bei der Übertragung von KI-Modellen

Zur systematischen Untersuchung von Energieeffizienzpotenzialen bei der Übertragung von KI-Modellen adressiert das Projekt die in Kapitel 3 beschriebenen Forschungsfragen innerhalb der Untersuchungsumgebung Federated Learning.

In dem Projekt werden verschiedene Konfigurationen des Federated Learning betrachtet, um möglichst viele Anwendungsfälle mit ihren individuellen Herausforderungen abdecken zu können. Zur Variation der Konfigurationen können z. B. folgende Parameter verändert werden:

- Anwendungsaufgabe (z. B. Bildklassifikation oder Spracherkennung)
- Trainingsdatensatz (z. B. ImageNet, Cifar, Google Speech Commands)
- KI-Modellarchitektur (z. B. VGG, ResNet)
- Anzahl der Teilnehmer
- FL-Protokoll (z. B. FedAVG, FedADAM, FedDIST, Scheduling der Teilnehmer)
- Systemarchitektur (z. B. Cross-Device-FL [IoT] mit sehr vielen Teilnehmern oder Cross-Silo-FL mit eher wenigen Teilnehmern und den entsprechenden Herausforderungen bei beiden Ansätzen)
- statistische Verteilung der Trainingsdaten (iid vs. non-iid)
- Anforderungen an die Latenz
- Bandbreite

Für jede der ausgewählten Konfigurationen des Federated Learning werden die Kommunikationskosten bei annähernd gleichbleibender Güte der KI-Modelle für verschiedene Ansätze zur Modell-Kompression während des iterativen und des dezentralen Lernansatzes gemessen. Dabei orientiert sich das Projekt weitestgehend an den standardisierten Testbedingungen, die auch für die Entwicklung des NNC-Standards gemäß Common Test Conditions for Incremental Compression of Neural Networks (MPEG, 2021) zur Anwendung kommen. Dessen Hauptaufgabe war allerdings die Entwicklung von Kompressionsverfahren mit minimaler Bitrate bei maximaler Genauigkeit. Eine Energiebilanzbetrachtung hat dabei nicht stattgefunden.

Insofern wird das Projekt hier neuartige Untersuchungen über die reine Kompressionsfähigkeit hinaus anstellen, indem die eingesparten Kommunikationskosten in Energieeinsparungen (z. B. gemessen in CO_2-Äquivalenten) umgerechnet und anschließend dem vom jeweiligen entsprechenden Ansatz zur Modellkompression benötigten Energieaufwand gegenübergestellt werden. Bei Konfigurationen mit expliziten Vorgaben hinsichtlich Latenz und Bandbreite genügt für ein anwendungsabhängiges Energieeffizienz-Ranking der verschiedenen Kompressionsansätze die Ermittlung der beim jeweiligen Ansatz investierten Energie zur Erfüllung der gesetzten Rahmenbedingungen für die Kommunikation zwischen Sender und Empfänger.

Als Ergebnis soll das Projekt erstmals ein anwendungsabhängiges Effizienzranking für die untersuchten Kompressionsmethoden liefern, daraus Handlungsempfehlungen und Entscheidungshilfen ableiten und einen Demonstrator zur Visualisierung der Energieeinsparpotenziale bei der Übertragung von KI-Modellen entwickeln.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Amazon-a. (kein Datum). Abgerufen am 15. 12 2021 von https://aws.amazon.com/de/ec2/instance-types/f1/

Amazon-b. (kein Datum). Abgerufen am 15. 12 2021 von https://aws.amazon.com/de/ec2/instance-types/p2/

Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access, 6*, 64270-64277.

Bitcom. (kein Datum). Abgerufen am 15. 12 2021 von https://www.bitkom.org/Presse/Presseinformation/Europaeischer-KI-Markt-verfuenffacht-sich-binnen-fuenf-Jahren

Chen, Y.-H., Krishna, T., Emer, J., & Sze, V. (2016). 14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, 262-263.

Fallahlalehzari, F. (kein Datum). Abgerufen am 15. 12 2021 von https://www.aldec.com/en/company/blog/167--fpgas-vs-gpus-for-machine-learning-applications-which-one-is-better

Falsafi, B., Dally, B., Singh, D., Chiou, D., Yi, J. J., & Sendag, R. (2017). FPGAs versus GPUs in Data centers. *IEEE Micro, 37*(1), 60-72.

Google. (kein Datum). Abgerufen am 15. 12 2021 von https://cloud.google.com/tpu/docs/tpus

Hintemann, R. (2016). Abgerufen am 15. 12 2021 von https://www.borderstep.de/wp-content/uploads/2017/03/Borderstep_Rechenzentren_2016.pdf

Hintemann, R., & Clausen, J. (2016). Green Cloud? The current and future development of energy consumption by data centers, networks and end-user devices. *Proc. of ICT for Sustainability*.

Kairouz, P., McMahan, H. B., Avent, B., ..., & Zhao, S. (2021). Advances and Open Problems in Federated Learning. *arxiv:1912.04977*.

Kirchhoffer, H., Haase, P., Samek, W., Müller, K., Rezazadegan-Tavakoli, H., Cricri, F., . . . Bailer, W. (2021). Overview of the Neural Network Compression and Representation (NNR) Standard. *IEEE Trans. Circuits Syst. Video Technol.*

Kratochwill, L., Richard, P., Babilon, L., Rehmann, F., Mamel, S., & Fasbender, S. (2020). *Künstliche Intelligenz – vom Hype zur energiewirtschaftlichen Realität.* Deutsche Energie-Agentur GmbH (dena).

Lobe, A. (2019). Abgerufen am 07. 01 2022 von https://www.spektrum.de/news/kuenstliche-intelligenz-verbraucht-fuer-den-lernprozess-unvorstellbar-viel-energie/1660246

Microsoft. (kein Datum). Abgerufen am 15. 12 2021 von https://www.microsoft.com/en-us/research/project/project-catapult/

MPEG. (2021). *Common Test Conditions for Incremental Compression of Neural Networks.* MPEG document WG04N0123, ISO/IEC JTC 1/SC 29/WG.

Nurvitadhi, E., Boutros, A., Budhkar, P., Jafari, A., Kwon, D., Sheffield, D., . . . Naik, M. (2019). Scalable Low-Latency Persistent Neural Machine Translation on CPU Server with Multiple FPGAs. *Proc. of International Conference on Field-Programmable Technology (ICFPT)*, 307-310.

Nurvitadhi, E., Sheffield, D., Sim, J., Mishra, A., Venkatesh, G., & Marr, D. (2016). Accelerating Binarized Neural Networks: Comparison of FPGA, CPU, GPU, and ASIC. *Proc. of International Conference on Field-Programmable Technology (FPT)*, 77-84.

Putnam, A., Caulfield, A. M., Chung, E. S., & Burger, D. (2015). A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services. *IEEE Micro, 35*(3), 10-22.

Qiu, X., Parcollet, T., Fernandez-Marques, J., Gusmao, P. P., Beutel, D. J., Topal, T., . . . Lane, N. D. (2021). A First Look into the Carbon Footprint of Federated Learning. *arxiv:2102.07627*.

Ruloff, S. (2019). Konzept und Realisierung zur Erfassung der elektrischen Leistung von hardwarebeschleunigten Rechnerplattformen am Beispiel von FPGA PCI-Express Karten. Masterarbeit, HTW Berlin.

Sattler, F., Marban, A., Rischke, R., & Samek, W. (2021). CFD: Communication-Efficient Federated Distillation via Soft-Label Quantization and Delta Coding. *IEEE Trans. Netw. Sci. Eng.*

Sattler, F., Wiedemann, S., Müller, K.-R., & Samek, W. (2019). Sparse binary compression: Towards distributed deep learning with minimal communication. *Proc. of International Joint Conference on Neural Networks (IJCNN)*, 1-8.

Statista. (kein Datum). Abgerufen am 01. 02 2022 von https://de.statista.com/statistik/daten/studie/3506/umfrage/monatliches-datenvolumen-pro-mobilfunknutzer-in-deutschland/

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *arxiv:1906.02243*.

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. (2017). Efficient processing of deep neural networks: A tutorial and survey. *arxiv:1703.09039*.

Wang, T., Geng, T., Li, A., Jin, X., & Herbordt, M. (2020). FPDeep: Scalable Acceleration of CNN Training on Deeply-Pipelined FPGA Clusters. *IEEE Trans. Comput., 69*(8), 1143 - 1158.

Wang, Y., Zhao, T., Li, L., Hou, Z., & Gu, J. (2018). Roofline Model Based Performance-Aware Energy Management for Scientific Computing. *Proc. of 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*.

Wiedemann, S., Kirchhoffer, H., Matlage, S., Haase, P., Marban, A., Marinč, T., . . . Samek, W. (2020). DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks. *IEEE J. Sel. Top. Signal Process., 14*(4), 700 - 714.

Xia, L., Diao, L., Jiang, Z., Liang, H., Chen, K., Ding, L., . . . Lin, W. (2019). PAI-FCNN: FPGA Based Inference System for Complex CNN Models. *Proc. of IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 107-114.

Abkürzungsverzeichnis

ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instruction-Set Processor
AVC	Advanced Video Codec
CABAC	Context-Based Adaptive Binary Arithmetic Coding
CPU	Central Processing Unit
CSR/CSC	Compressed Sparse Row/Column
(D)CNN	(Deep) Convolutional Neural Network
dena	Deutsche Energie-Agentur
EEKI	Energieeffiziente Künstliche Intelligenz (Projektname)
FL	Föderiertes Lernen/Federated Learning
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GB	Gigabyte
GE	Gigabit Ethernet
GELU	Gaußian Error Linear Unit
(G-)FLOPs	(Giga) Floating Point Operations
(GP)GPU	(General Purpose) Graphics Processing Unit
HEVC	High Efficiency Video Codec
HHI	Heinrich-Hertz-Institut
IoT	Internet of Things
KD	Knowledge Distillation
KI	Künstliche Intelligenz
KNN	Künstliches Neuronales Netz
kWh	Kilowattstunde
MB	Megabyte
ML	Maschinelles Lernen/Machine Learning
NAS	Netzwerkarchitektursuche
NNC	Neural Network Coding
PCIe	Peripheral Component Interconnect Express

QAT	Quantization-aware Training
ReLU	Rectified Linear Unit
SIMD	Single Instruction Multiple Data
TPU	Tensor Processing Unit
TWh	Terawattstunde
VVC	Versatile Video Codec

www.dena.de

dena
Deutsche Energie-Agentur