Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen

Machine Learning Hardware

Report

Neuste Trends in Big Data Analytics WS 2017/18

Dominik Scherer
Matr.Nr. 6540167

Betreuer: Dr. Julian Kunkel

Hamburg, 31.03.2018

Abstract

Anwendungen basierend auf künstlicher Intelligenz und Machine Learning haben in den letzten Jahren extrem an Popularität gewonnen und sind mittlerweile allgegenwärtig. Die Fülle an Daten, die uns heute zu Verfügung steht, bietet ideale Voraussetzungen für die Anwendung neuronaler Netze und anderer Machine Learning Verfahren auf verschiedenste Gebiete und Tätigkeiten. Allerdings ist das Ausführen dieser Algorithmen auf solch großen Datenmengen und komplexen Netzen mit einem nicht unerheblichen Zeitaufwand verbunden und es besteht eine hohe Nachfrage an schneller, effizienter Hardware, welche die Trainings- und Inferenzzeit minimieren kann. In dieser Arbeit werden die zwei aktuell bekanntesten Technologien dieser Art vorgestellt: die Tensor Processing Unit von Google und die Tensor Cores von Nvidia. Desweiteren wird ein kurzer Ausblick auf zwei weitere, noch in der Entwicklung befindlichen Lösungen von Intel gegeben.

Inhaltsverzeichnis

1 Einführung

Machine Learning (im Folgenden mit ML abgekürzt) wird heutzutage in vielen Bereichen angewendet und bestimmt einen großen Teil unseres Alltags. Sprach- und Texterkennung sowie Übersetzung kommt in unzähligen Produkten und Dienstleistungen zum Einsatz. Suchmaschinen, die wohl fast jeder täglich benutzt, basieren zum Großteil auf ML Algorithmen, sowie auch sämtliche Empfehlungen, die dem Kunden in Online Shops oder Streaming Portalen angezeigt werden. Roboter werden mithilfe von ML immer intelligenter und haben bereits auf vielen Gebieten menschliche Arbeitskräfte ersetzt. Tabelle 1 zeigt die Ergebnisse einer Umfrage der Crisp Research AG Kassel. 168 Firmen wurden befragt, wofür sie ML verwenden.

Abbildung 1: Machine Learning Umfrage, Crisp Research AG Kassel, [4]

Während die Theorie hinter ML Algorithmen schon lange bekannt und gut erforscht ist, sind wir heute zum ersten mal an einem Punkt, in der wir auch die technischen Möglichkeiten haben, ML Algorithmen effektiv zu nutzen. Diese benötigen in erster Linie zwei Dinge: große Datenmengen und schnelle Hardware, um diese Daten in akzeptabler Zeit zu verarbeiten. An Daten herrscht im Zeitalter von Internet, globaler Vernetzung und günstigem Speicher kein Mangel, bei der Hardwareleistung gibt es jedoch noch viel Potential für Verbesserungen. Dies liegt vor allem daran, dass die Entwicklungen auf dem Gebiet der künstlichen Intelligenz früher nicht abzusehen waren und erst in den letzten Jahren das Potential von ML wirklich vollends erkannt wurde. Deshalb muss auch heute noch klassische

Hardware Architektur (GPU und CPU) für ML herhalten, die zwar zweckmäßig, aber nicht spezialisiert und daher bei weitem nicht optimal ist. CPUs eignen sich besser für sequentielle Probleme und müssen ein breites Spektrum an Aufgaben bewältigen können. GPUs sind aufgrund ihrer höheren parallelen Rechenleistung besser geeignet und werden deshalb bevorzugt, jedoch sind diese in erster Linie auf Grafikanwendungen ausgelegt und daher auch in vielerlei Hinsicht nicht für ML angepasst. Firmen wie Google, Nvidia und Intel haben sich dieses Problems angenommen und forschen an spezialisierter ML Hardware bzw. haben diese bereits veröffentlicht.
Die Frage drängt sich auf, was man eigentlich genau unter Machine Learning versteht. Tom Mitchell gibt in seinem Buch *Machine Learning* folgende Definition:

> "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience."[5]

Obwohl ML mehrere Unterkategorien umfasst, wie z.B. Entscheidungsbäume, Clustering, Reinforcement Learning und genetische Algorithmen, sind die neuronalen Netze aufgrund ihrer Flexibilität die Methode der Wahl bei einem Großteil von Anwendungen und am weitesten verbreitet. Daher ist auch die heute schon verfügbare ML Hardware auf neuronale Netze zugeschnitten, welche vor allem auf Matrixmultiplikationen basieren. Das Training dieser Netze, insbesondere wenn es sich um komplexere handelt, ist mit großen Datenmengen, welche nötig sind um gute Ergebnisse zu erhalten, extrem rechen- und damit zeitintensiv. Hardware, die für diesen Zweck entworfen wird, muss gewisse Anforderungen erfüllen. Zum einen spielt Skalierbarkeit eine wichtige Rolle. Eine einzelne Hardware Einheit ist für professionelle Anwendungen nicht ausreichend. Es müssen sich beliebig viele Einheiten gut miteinander vernetzen lassen und dann auch eine deutliche Leistungssteigerung bringen. Da viele dieser Einheiten gleichzeitig und meistens auch 24 Stunden am Tag laufen sollen, spielt außerdem geringer Stromverbrauch eine große Rolle. In vielen Fällen ist auch eine niedrige Latenz wichtig, man denke z.B. an autonom fahrende Autos. Des Weiteren ist erwähnenswert, dass eine hohe arithmetische Präzision nicht nötig ist. Es hat sich in der Vergangenheit gezeigt, dass neuronale Netze mit 8 Bit oder 16 Bit Arithmetik keine schlechteren Ergebnisse liefern als mit float oder double Werten. In dieser Arbeit wird eine kleine Auswahl der bekanntesten Hardware vorgestellt, die speziell dafür entworfen wurde, diese Anforderungen zu erfüllen. Da es sich dabei jedoch ausschließlich um proprietäre Technologien handelt, sind Informationen zur Funktionsweise und Architektur nur eingeschränkt vorhanden. Die verfügbaren Informationen wurden so gut wie möglich zusammengefasst. Als Quellen dienten dabei in erster Linie die offiziellen Webseiten der jeweiligen Hersteller.
Zunächst wird aber noch ein kurzer Einblick in die Theorie neuronaler Netze gegeben, was das Nachvollziehen der Hardwaredesigns im Anschluss einfacher machen soll.

2 Grundlagen von Neuronalen Netzen

Im Allgemeinen durchläuft ein neuronales Netz (im Folgenden mit NN abgekürzt) immer zwei Phasen. Eine Trainingsphase, in der das NN mit einem Trainingsdatensatz trainiert wird, und eine Testphase, in der das NN Daten verarbeitet, die nicht Teil des Trainingsdatensatzes waren und mit dem bestimmt werden soll, wie akkurat das Netz letzten Endes wirklich arbeitet und ob es für den Einsatz in der tatsächlichen Anwendung bereit ist. Das klassifizieren von Daten außerhalb des Trainings wird Inferenz genannt. Abbildung 2 zeigt ein Schema dieses Prozesses.

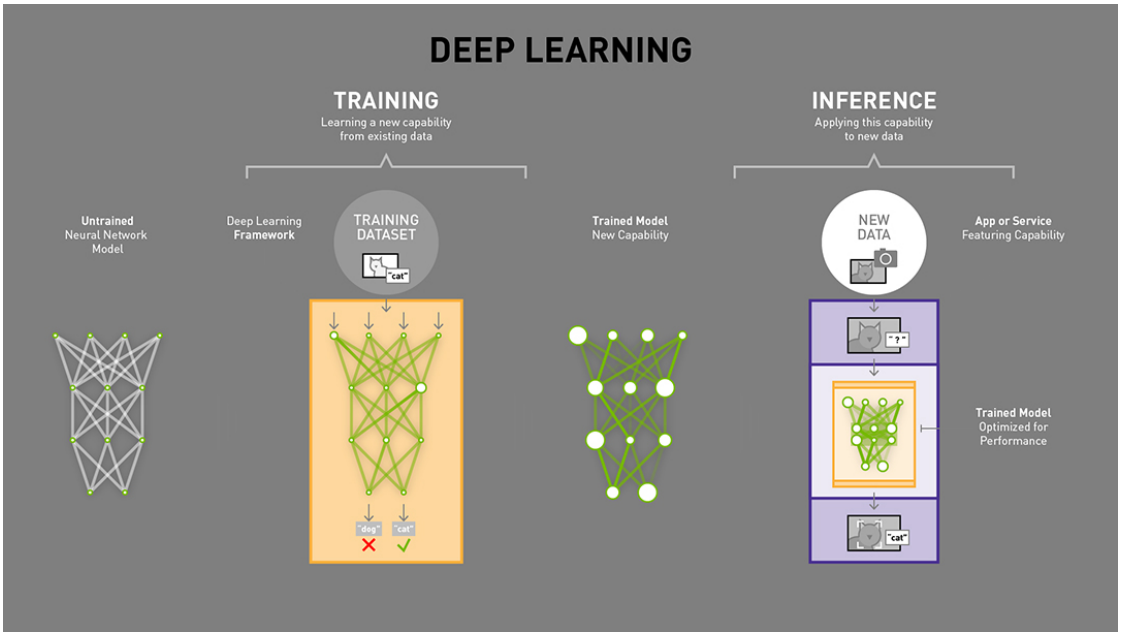

Abbildung 2: Training und Inferenz von NNs, Nvidia, [6]

Links sehen wir das untrainierte Netzwerk, welches aus mehreren Schichten von Neuronen besteht. Handelt es sich um ein sehr komplexes NN mit vielen Schichten und Neuronen, wird häufig der Begriff *Deep Learning* verwendet, wie hier der Fall. Im nächsten Schritt, dem Training, bekommt das NN ein Bild aus dem Trainingsdatensatz als Input, welches korrekt erkannt werden soll. Sind die Daten wie hier gelabelt, handelt es sich um überwachtes Lernen (*Supervised Learning*). In diesem Beispiel ist dem Bild das Label *cat* zugeordnet. Das bedeutet, dass das NN überprüfen kann, ob es falsch gelegen hat und Gewichte angepasst werden müssen. Die Gewichte bestimmen, welche Neuronen bei einem gewissen Input aktiviert werden. Dies ist im nächsten Bild mit dem trainierten Modell durch die unterschiedlich großen Neuronen symbolisiert. Im letzten Schritt wird das NN nun in der Anwendung benutzt (Testphase ist in diesem Beispiel ausgelassen). Wir sehen, dass das NN im letzten Schritt nun ein Bild einer anderen Katze (dieses Bild war nicht Teil des Trainingssatzes) korrekt identifiziert.

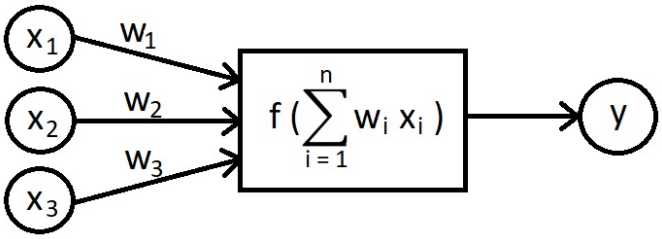

Abbildung 3: Aktivierung eines Neurons, Dominik Scherer

In Abbildung 3 ist ein minimales NN dargestellt, welches ein Output Neuron *y* mit 3 Input Neuronen *x1*, *x2* und *x3* zeigt. Die Gewichte sind mit *w1*, *w2* und *w3* dargestellt und es gibt eine Aktivierungsfunktion *f*. Die Gewichte der einzelnen Input Neuronen werden jeweils mit ihrem eigenen Wert multipliziert und anschließend mit den anderen Produkten aufaddiert. Dann wird die Aktivierungsfunktion auf das Ergebnis angewandt und wird dabei ein gewisser Schwellwert erreicht, wird das Output Neuron *y* aktiviert. Dies kann mathematisch als Matrixmultiplikation A*B dargestellt werden, wobei Matrix A aus den Input Neuronen und Matrix B aus den Gewichten besteht. In diesem simplen Beispiel würden man also eine 1x3 Matrix mit einer 3x1 Matrix multiplizieren. In realen Anwendungen bestehen diese Matrizen aus unzähligen Einträgen, es müssen also extrem viele Produkte berechnet werden. Daher liegt auch der Hauptfokus bei für ML entwickelter Hardware darauf, Matrixmultiplikationen so effizient wie möglich zu realisieren. Da die einzelnen Produkte unabhängig voneinander sind, bietet sich hier sehr viel Potential für Leistungsgewinn durch Parallelisierung.

3 Tensor Processing Unit

3.1 Überblick

Die Tensor Processing Unit (TPU) wurde von Google entwickelt und wird seit ca. 2015 in deren Data Centers verwendet. Es handelt sich um einen ASIC (*application-specific integrated circuit*), der speziell zum Verarbeiten neuronaler Netze mit Tensorflow entworfen wurde. TPUs sind nicht kommerziell verfügbar. Im Mai 2017 hat Google die zweite Generation von TPUs angekündigt. In diesem Bericht wird, aufgrund der nur spärlich verfügbaren Informationen, nur auf die erste Generation von TPUs eingegangen.

Die Spezifikationen der TPU lauten wie folgt:

- CISC Instruction Set
- 700 MHz Taktrate
- 65.536 8-Bit Integer Multiply-and-Add Einheiten

- 28 MB On-Chip Activation Memory
- 4MB On-Chip Accumulator Memory
- 8GB Off-Chip DRAM mit 34 GB/s
- PCIe 3.0 Bus
- 75W Stromverbrauch

Wie bereits erwähnt, reicht ein niedrige Präzision für NNs vollkommen aus, wie hier noch einmal deutlich wird: Die TPU arbeitet mit nur 8 Bit.

3.2 Architektur und Instruction Set

In Abbildung 4 ist ein Blockdiagramm der TPU Architektur zu sehen.

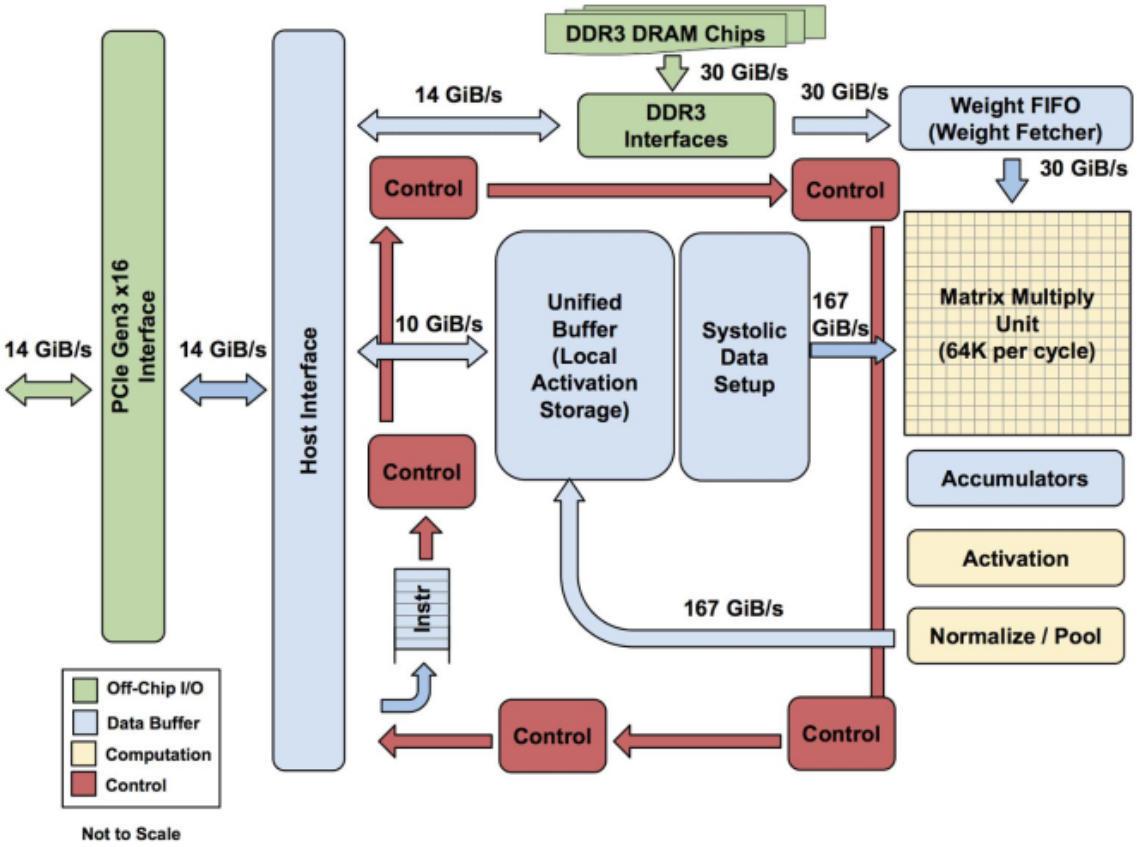

Abbildung 4: TPU Block Diagram, Google, [3]

Das Herzstück der TPU ist die Matrix Multiply Unit. Hier werden die ca. 64000 Matrix Multiplikationen pro Taktzyklus durchgeführt. Die Gewichtsmatrizen aus dem DRAM werden hier mit den Ergebnissen des letzten Zyklus verrechnet und die neuen Ergebnisse akkumuliert. Anschließend wird die Aktivierungsfunktion angewandt. Nach dem sogenannten Pooling, bei dem überflüssige Informationen herausgefiltert werden und das Datenvolumen verkleinert wird, werden die aktuellen Werte gebuffert und für die weitere Berechnung im nächsten Taktzyklus vorbereitet. Die Ergebnisse können auch zum Host transferiert und ausgelesen werden.
Die Matrix Multiply Unit besteht aus einem systolischen Array mit 256x256 ALUs (*arithmetic logic unit*). Systolische Arrays bestehen aus eng vernetzen Einheiten, die jeweils eine Funktion berechnen, das Ergebnis in sich selbst speichern und direkt an ihre Nachbarn weitergeben. Dadurch gibt es keine Umwege über den RAM, im Fall der TPU also ca. 128.000 Operationen in einem Zyklus ohne Speicherzugriffe, und der Durchsatz sowie die Effizienz wird deutlich erhöht. Die 65.536 Multiply-and-Add Operationen in jedem Zyklus mal die Takrate von 700 MHz ergeben insgesamt ca. 92 Tera-Operationen (8 Bit) pro Sekunde. Zum Vergleich, die schnellsten heute verfügbaren Desktop Prozessoren erreichen nicht mehr als 1 TFLOPS (Intel Core i9 XE).
Das Instruction Set der TPU besteht aus CISC Befehlen. Die mit den beschriebenen Funktionen korrespondierenden Befehle lauten wie folgt:

Read_Host_Memory Lese Daten/Input vom Host Speicher

Read_Weights Lese Gewichte ein

MatrixMultiply Führe die Multiplikationen aus und akkumuliere die Ergebnisse

Activate Wende Aktivierungsfunktion auf Ergebnisse an

Write_Host_Memory Schreibe die Ergebnisse in den Host Seicher

CISC Befehle erlauben im Vergleich zu RISC Befehlen deutlich mehr Operationen pro Instruktion. Wie bereits erwähnt, können mit einem einzigen MatrixMultiply Befehl ca. 128000 Operationen ausgeführt werden.

3.3 Performance und Stromverbrauch

Abbildung 5 zeigt die Leistung pro Watt einer TPU im Vergleich zu einer zeitgenössischen CPU und GPU. Durch die im vorigen Kapitel beschriebene Implementation der ALUs als systolisches Array, erreicht die TPU im Mittel eine 83 mal besser Performance pro Watt als aktuelle CPUs und eine ca. 29 mal bessere Performance pro Watt als GPUs.
Abbildung 6 zeigt den Durchsatz pro Sekunde für Inferenzen, die in 7 ms oder weniger berechnet wurden, wieder im Vergleich zu aktuellen CPUs und GPUs. Hier

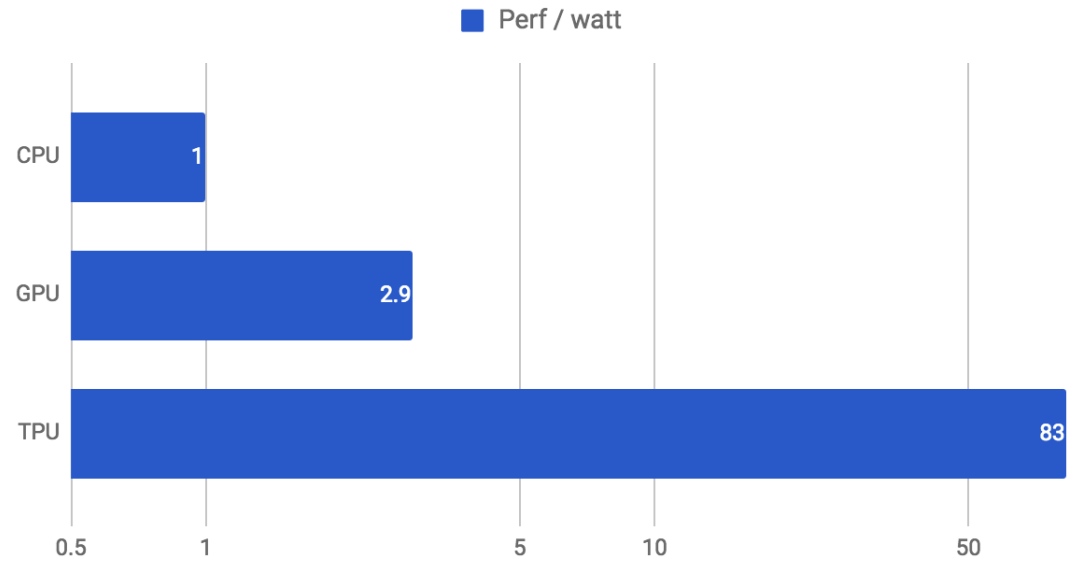

Abbildung 5: Performance/watt of a TPU, relative to contemporary CPUs and GPUs (in log scale)(Incremental, weighted mean), Google, [1]

erkennt man deutlich den Leistungszuwachs, welche eine spezialisierte Architektur bringt. Die TPU erreicht ca. 41 mal mehr Durchsatz als CPUs und ca. 17 mal mehr Durchsatz als GPUs.

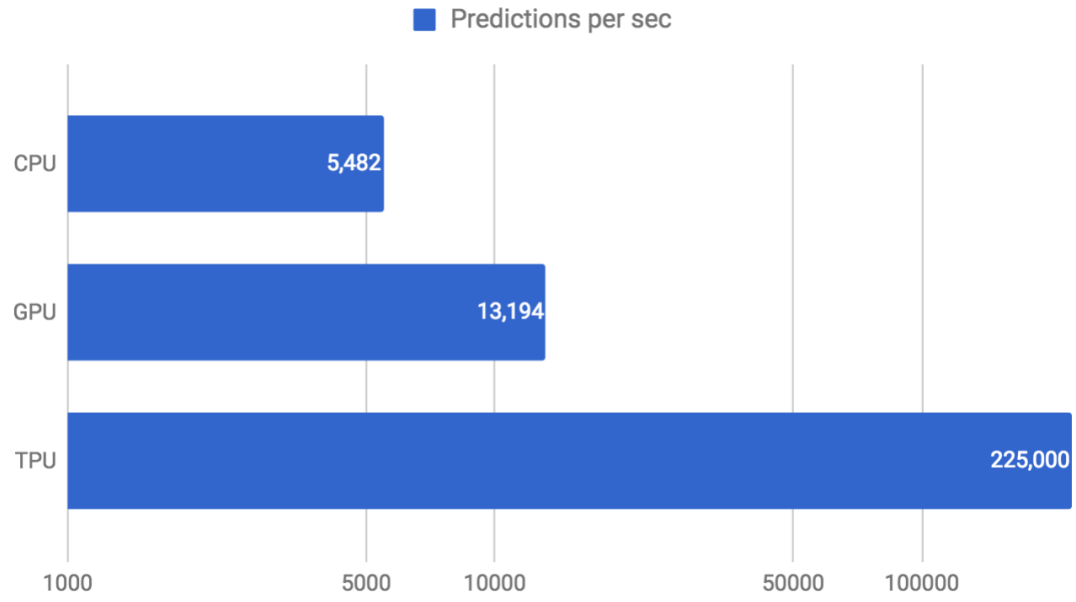

Abbildung 6: Throughput under 7 ms latency limit (in log scale)(99th% response with MLP0: CPU = 7.2 ms, GPU = 6.7 ms, TPU = 7.0 ms), Google, [2]

4 Tensor Cores

4.1 Überblick

Tensor Cores sind ein von Nvidia entwickelter, speziell auf Deep Learning ausgerichter, Teil der Volta GPU Mikroarchitektur. Sie sind bisher nur auf dem 2017 veröffentlichten GPGPU Chip Tesla V100 vorhanden. Erste Endbenutzer Chips mit Tensor Cores sind Mitte 2018 zu erwarten.
Der Tesla V100 Chip ist mit 21 Milliarden Transistoren auf 815 mm^2 der bisher größte GPU Chip der Welt. Die Spezifikationen lauten wie folgt:

- 80 Streaming Multiprozessoren (SM)
- 64 FP32 Cores / SM, 5120 pro GPU
- 32 FP64 Cores / SM, 2560 pro GPU
- **8 Tensor Cores / SM, 640 pro GPU**
- 1530 MHz Taktrate
- 16GB HBM2 mit 900GB/s
- PCIe 3.0 Bus
- 300W Stromverbrauch

Die Leistung des Chips beläuft sich auf ca. 7.5 TFLOPS Double Precision Performance und ca. 15 TFLOPS Single Precision Performance. Die für ML wichtige Tensor Performance, welche durch die Tensor Cores realisiert wird, beträgt etwa 120 TFLOPS.
Wie bereits erwähnt sind GPUs auch heute noch das Mittel der Wahl um NNs zu trainieren, sind jedoch nicht speziell für diesen Zweck konzipiert. Mit dem Tesla V100 gibt es nun erstmals eine GPU, die dedizierte ML Hardware auf dem Chip verbaut hat. Um die Leistungssteigerung klar erkennbar zu machen, wird im Folgenden der Tesla V100 Chip mit dem Tesla P100 Chip der vorherigen Generation, welche keine spezielle ML Hardware wie Tensor Cores vorzuweisen hat, verglichen.

4.2 Volta vs. Pascal: Performance

Abbildung 7 zeigt einen Vergleich der Performance des V100 mit dem P100 Chip für Matrix-Matrix Multiplikationen mit verschieden großen Matrizen. Im linken Diagramm ist zu erkennen, dass die reguläre Performance des V100 ohne Tensor Cores im besten Fall mit Single Precision ca. 1,8 mal so hoch ist wie die des P100. Im rechten Diagramm wird deutlich, dass die Leistungssteigerung, die durch

Tensor Cores gewonnen werden kann, weit über die zu erwartende Performancesteigerung von einer Generation zur nächsten hinaus geht. Für große Matrizen ist die Performance des V100 etwa 9,3 mal höher als die des P100.

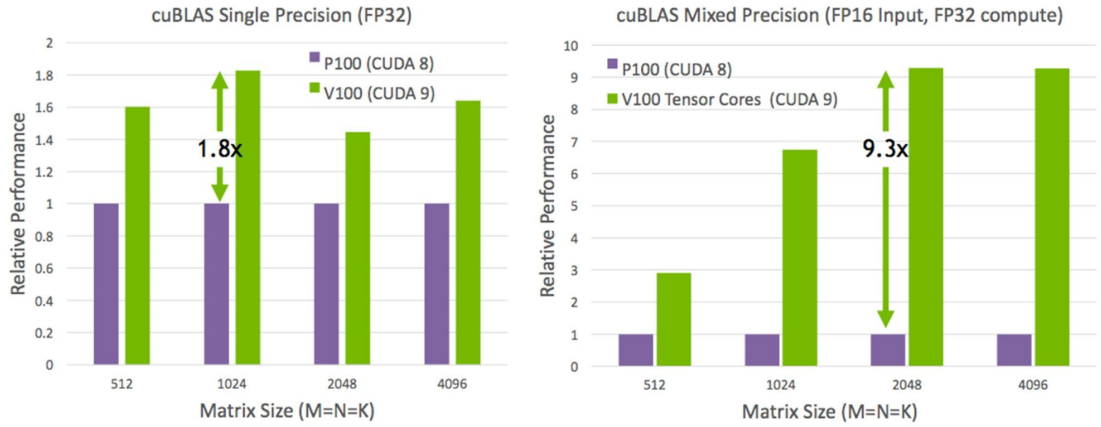

Abbildung 7: Tesla V100 Tensor Cores and CUDA 9 deliver up to 9x higher performance for GEMM operations. (Measured on pre-production Tesla V100 using pre-release CUDA 9 software.), Nvidia, [7]

4.3 Operationsweise eines Tensor Cores

Folgendes ist bekannt über die Funktionsweise der Tensor Cores:
Tensor Cores verarbeiten 4x4 Matrizen und können als 4x4x4 Matrix Processing Arrays angesehen werden. Die Multiplikationen der einzelnen Komponenten von zwei 4x4 Matrizen produzieren 64 Produkte (4x4x4 Array), von denen jeweils 4 akkumuliert werden müssen (16 Summen). Alle Produkte können im Tensor Core mit nur einer Instruktion parallel berechnet und anschließend akkumuliert werden. Die Multiplikation arbeitet dabei mit 16 Bit Inputs, die Addition unterstützt aber bis zu 32 Bit (*mixed precision*). Zum Vergleich, die Pascal Architektur unterstützt nur 4 Produkte mit Akkumulation pro Instruktion. Ein einzelner Tensor Core erreicht also 64 FMA (*fused multiply and add*) Operationen pro Takt. Mit 8 Tensor Cores ergeben sich 1024 einzelne Operationen pro Takt pro SM. Der Tesla V100 Chip hat 80 SMs, insgesamt beläuft sich die Leistung also auf 81920 Mixed Precision Operationen pro Chip.

5 Zukünftige Technologien

5.1 Nervana

Nervana ist der Name einer Prozessor Familie von Intel, welche speziell für neuronale Netze entwickelt wird, und erstmals 2016 unter dem Namen *Lake Crest Deep*

Learning Architecture angekündigt wurde. Eine Veröffentlichung ist nicht vor 2019 zu erwarten. Die heute namensgebende, auf künstliche Intelligenz spezialisierte Software Firma Nervana wurde 2016 von Intel gekauft. Außerdem ist bekannt, dass Facebook an der Entwicklung beteiligt ist.
Bekannte technische Details sind, dass eigens entwickelte bi-direktionale High Bandwidth Links verwendet werden sollen, um einzelne Hardware-Einheiten direkt miteinander verbinden zu können (6 pro Prozessor), und dass es keine Standard Cache Hierarchie geben wird. Stattdessen soll der vorhandene On-Chip Speicher (32GB HBM2) per Software verwaltet werden, um mehr Flexibilität zu ermöglichen. Außerdem soll ein neues numerisches Format namens *Flexpoint* zum Einsatz kommen, welches von Intel als ein Mischung aus Floating und Fixed Point beschrieben wurde.

5.2 Loihi

Der Loihi Chip ist ein ebenfalls von Intel entwickelter Prototyp eines selbst lernenden Chips, welcher die Funktionsweise des menschlichen Gehirns imitieren soll. Das bedeutet, dass Training und Inferenz lokal auf dem Chip ausgeführt werden, welcher aus einem Netz aus 128 neuromorphischen Cores besteht, von denen jeder eine eigene Learning Engine besitzt. Der Chip kann planar in 4 Richtungen mit weiteren Chips verbunden werden. Der Loihi Chip wurde erstmals im September 2017 angekündigt und im Februar/März 2018 beim *Neuro Inspired Computational Elements Workshop* in Oregon offiziell vorgestellt.

6 Zusammenfassung

Es wurden zwei der bekanntesten, speziell für Machine Learning entwickelten, Technologien vorgestellt, die Tensor Processing Unit von Google und die Tensor Cores von Nvidia. Es hat sich gezeigt, dass die Leistung der TPU bei etwa 92 Tera-Operationen (8 Bit) mit 75 Watt Stromverbrauch liegt, und die Leistung des Tesla V100 Chip bei etwa 120 Tera-Operationen mit 300 Watt Stromverbrauch liegt. Desweiteren wurde ein kurzer Ausblick auf zukünftige Technologien gegeben. Die Machine Learning Branche ist stark am Wachsen und es gibt eine hohe Nachfrage an Hardware, die das extrem zeitaufwendige Trainieren von KI-Anwendungen beschleunigen kann. Allgemein kann man sagen, dass durch die unvorhersehbaren Entwicklungen auf dem Gebiet der künstlichen Intelligenz, die Forschung auf Hardware Ebene noch nicht ausgereift ist und wir in Zukunft mit an Sicherheit grenzender Wahrscheinlichkeit große Innovationen und Leistungssteigerungen im Bereich Machine Learning sehen werden.

Literaturverzeichnis

[1] Google. *Performance/watt of a TPU, relative to contemporary CPUs and GPUs (in log scale)(Incremental, weighted mean).* `https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu`, 2017.

[2] Google. *Throughput under 7 ms latency limit (in log scale)(99th% response with MLP0: CPU = 7.2 ms, GPU = 6.7 ms, TPU = 7.0 ms).* `https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu`, 2017.

[3] Google. *TPU Block Diagram.* `https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu`, 2017.

[4] Crisp Research AG Kassel. *Diese Machine-Learning-Funktionen nutzen die Anwender.* `https://www.computerwoche.de/a/machine-learning-die-technik,3330420`, 2017.

[5] Tom Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[6] Nvidia. *Inference is where capabilities learned during deep learning training are put to work.* `https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai`, 2016.

[7] Nvidia. *Tesla V100 Tensor Cores and CUDA 9 deliver up to 9x higher performance for GEMM operations. (Measured on pre-production Tesla V100 using pre-release CUDA 9 software.).* `https://devblogs.nvidia.com/parallelforall/inside-volta`, 2017.

Abbildungsverzeichnis