Distributed Tensorflow

Michael Zipperle
259564
Michael.Zipperle@hs-furtwangen.de

***Abstract*—**

I. Einführung

Eine Künstliche Intelligenz (KI) wird heutzutage immer mehr eingesetzt, um die Interaktion zwischen Mensch und Maschine zu bessern. Zusätzliche ermöglicht eine KI vollkommen neue Möglichkeiten in Bereichen wie Robotik, Daten Analyse, Gesundheitswesen, Autonomes Fahren und vielen mehr. Doch was steckt hinter einer künstlichen Intelligenz? Eine künstliche Intelligenz versucht durch maschinelles Lernen die menschliche Wahrnehmung und das menschliche Handel durch Maschinen nachzubilden. Bei der KI-Forschung gibt es viele Verbindung zur Neurologie und Psychologie, den das menschliche Denken muss erforscht und verstanden werden, bevor Maschinen dies nachahmen können. Bis heute ist dies noch nicht annähernd gelungen, die meisten KI beschränken sich auf einen bestimmten Teilbereich und sind somit optimiert für einen bestimmten Anwendungsfall [1].

Bei einer KI kommt meist maschinelles Lernen zum Einsatz, wobei von erhobenen Daten gelernt wird, um dann Entscheidung treffen zu können. Es gibt eine Reihe an vordefinierten Algorithmen die an sogenannte Entscheidungsbäume erinnern. Diese Algorithmen sind meist nicht dynamisch genug, um mehrere Variablen verarbeiten zu können. Mit der Einführung von Deep Learning als Zweig des maschinellen Lernens ist dieses Problem größtenteils behoben. Durch Deep Learning kann ein kontinuierlicher Lernprozesse umgesetzt werden, der sich stetig an neue Situation anpasst. Dabei basiert Deep Learning auf der Analyse von Big Data und gräbt sich durch riesige Mengen an Daten aus Datenbank, Internetquellen und mehr. Es nutzt neuronale Netze, um der Denkweise eines Menschen möglichst gut nachzuahmen. Deep Learning war bis vor wenigen Jahren nur von wenig Unternehmen genutzt, da der Einsatz viel Ressourcen benötigt und somit relativ teuer war. Durch den Fortschritt der Technik und der besseren Unterstützung der Grphics Processing Unit (GPU) hat sich Deep Learning durchgesetzt und kommt in heutigen Durchbrüchen wie autonomes Fahren, maschinelle Übersetzung, Spracherkennung und vieles mehr zum Einsatz [2].

Aktuell stellt der Lernprozess in Echtzeit eine große Herausforderung dar, oftmals benötigt der Lernprozess mehrere Stunden bzw. Tage. Für einige Anwendungsfälle reicht dies nicht aus und es wird ein Lernprozess nahe zu in Echtzeit benötigt. Es gibt bereits zahlreiche Deep Learning Frameworks, welche in diesem Arikel aufgezeigt werden. Sehr beliebt ist aktuell das Framework Tensorflow von Google. Dieses Framework wird im Laufe dieses Artikel genauer untersucht und es wird anhand eines Tutorials die Einrichtung einer verteilten Tensorflow Umgebung aufgezeigt. Durch diese verteilte Umgebung kann die Performance des Lernprozess verbessert werden.

II. Related Work

Bevor auf TensorFlow genauer eingegangen wird, soll dieses Kapitel eine Übersicht über folgende Deep Learning Frameworks geben:

- Torch
- Caffe
- Caffe2
- Keras
- Chainer
- CNTK
- Apache MXNet
- Amazon DSSTNE
- Eclipse Deeplearning4j

A. Torch

Torch wurde ursprünglich von Facebook entwickelt und 2017 als Open-Source Projekt veröffentlicht. Das Framework bietet zwei Hauptfunktionen: Erstens, mathematische Berechnung unter starker Unterstützung der GPU. Dabei wird es oft als Ersatz für das bekannte Python Framework Numpy eingesetzt, da es durch die GPU-Unterstützung eine wesentlich bessere Performance bietet. Zweitens, zur Bildung von neuronale Netze für Deep Learning, dabei wirbt das Framework mit maximaler Flexibilität und Geschwindigkeit [4]. Laut DL4J zeichnet sich das Framework durch viele Modulare Funktionen aus, die sich einfach kombinieren lassen. Des Weiteren lassen sich einfach neue Layer definieren und auf der GPU ausführen. Jedoch bietet Torch keinen kommerziellen Einsatz und die Dokumentation soll nicht vollständig sein [3].

B. Caffe

Caffe ist ein Open-Source Projekt, das von Yangqing Jia im Rahmen seiner Doktorarbeit beim Berkley AI Research (BAIR) initiiert wurde. Aktuell wird es vom BAIR und Community Entwicklern weiterentwickelt. Mit nur wenigen Zeilen Code lassen sich Modelle erstellen, dabei lässt sich konfigurieren ob die Berechnungen auf der CPU oder GPU durchgeführt werden. Caffe wird hauptsächlich zur Verarbeitung von Bildern eingesetzt und kann mit nur einer GPU über 60 Millionen Bilder pro Tag verarbeiten [5]. Aktuell bietet Caffe noch keine Möglichkeit die Berechnung verteilt von mehreren GPU durchzuführen [3].

C. Caffe2

Der Erfinder von Caffe Yangqing Jia arbeitet jetzt bei Facebook und entwicklte dabei an einer Erweiterung von Caffe. Diese wurde unter dem Name Caffe2 veröffentlicht und bietet mehr Skalierbarkeit und Leichtgewichtigkeit gegenüber Caffe. Zudem erlaubt Caffe2 das Verteilen von Aufgaben bzw. Berechnung auf mehrere Instanzen [6].

D. Keras

Keras ist ein High-Level Application Programming Interface (API) für neuronale Netze und basiert auf Tensorflow, welches in Kapitel III genauer erläutert wird. Die Entwicklung von Keras folgte dem Motto: ”Die Fähigkeit, mit möglichst wenig Verzögerung von der Idee zum Ergebnis zu kommen, ist der Schlüssel für eine gute Forschung” [7]. Somit ermöglicht Keras das schnell aufsetzen und testen von Neuronalen Netzen. Elephans ist eine Erweiterung für Keras, die das verteilen einer Anwendung über mehrerer Instanzen erlaubt [8].

E. Chainer

Chainer ist auch ein Open-Source Deep Learning Framework und bietet ein flexibles, intuitives und leistungsstarkes Mittel zur Implementierung einer ganzen Reihe von Deep-Learning-Modellen, einschließlich State-of-the-Art-Modellen wie z. B. rekurrenten neuronalen Netzen und variationalen Autoencodern. Mit Chainer können Anwendungen auf mehrere Instanzen verteilt werden und dadurch wie in Abbildung 1 zu sehen im Bereich Performance andere Frameworks klar hintersich lassen. Die optimale Performance wurde beim verteilen einer Anwendung auf 128 GPUs erreicht [9].

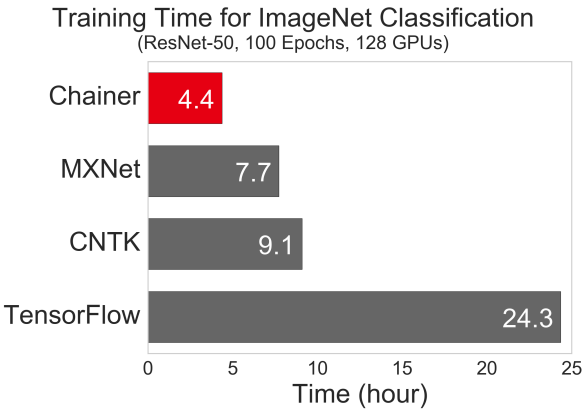

Fig. 1: Deep Learning Framework Vergleich [9]

F. Microsoft Cognitive Toolkit (CNTK)

Microsoft Cognitive Toolkit (CNTK) ist ein Open-Source Framework für das verteilen einer Deep-Learning Anwendung im kommerziellen Bereich. Es beschreibt neuronale Netze als eine Reihe von Rechenschritten über einen gerichteten Graphen. CNTK erlaubt dem Benutzer, populäre Modelltypen wie feed-forward DNNs, konvolutionelle neurale Netze (CNNs) und wiederkehrende neurale Netze (RNNs / LSTMs) leicht zu verwirklichen und zu kombinieren. CNTK implementiert stochastisches Gradientenabstiegsverfahren (SGD, Error Backpropagation) mit automatischer Differenzierung und Parallelisierung über mehrere GPUs und Server hinweg [10].

G. Apache MXNet

Apache MXNet ist ein modernes Open-Source Deep Learning Framework, mit dem neuronale Netze trainiert und implementiert werden können. Es ist skalierbar, ermöglicht schnelles Modelltraining und unterstützt ein flexibles Programmiermodell und mehrere Programmiersprachen. Die MXNet-Bibliothek ist portabel und kann auf mehreren GPUs und mehreren Instanzen skaliert werden. MXNet wird von den wichtigsten Public Cloud-Anbietern wie Amazon Web Services (AWS) und Azure unterstützt [11]. Amazon und Microsoft arbeiteten zusammen an einer API für Apache MXNet und veröffentlichten im Oktober 2017 Gluon, eine neue Open-Source Deep-Learning Schnittstelle, die es Entwicklern ermöglicht, einfach und schnell Machine-Learning-Modelle zu erstellen, ohne dabei die Leistung zu beeinträchtigen [13].

H. Amazon DSSTNE

Amazon Deep Scalable Sparse Tensor Network Engine (DSSTNE) ist eine Open-Source Deep Learning Framework zum entwicklen von Empfehlungsmodellen. DSSTNE wurde bei Amazon eingesetzt, um personalisierte Produktempfehlungen für ihre Kunden zu erstellen. Es ist für den produktiven Einsatz von realen Anwendungen ausgelegt, bei denen Geschwindigkeit und Skalierbarkeit gegenüber experimenteller Flexibilität im Vordergrund stehen müssen. Training und Vorhersagen werden beide skaliert, wobei Berechnung und Speicherung für jede Schicht modellparallel verteilt werden [14].

I. Eclipse Deeplearning4j

Eclipse Deeplearning4j (DL4J) ist die erste kommerzielle, Open-Source, verteilte Deep-Learning-Bibliothek für Java und Scala. Durch die Integration mit Hadoop und Apache Spark bringt DL4J KI zu Geschäftsumgebungen für verteilte GPUs und Central Processing Unit (CPU)s. DL4J zielt darauf ab, Plug-and-Play auf dem neuesten Stand der Technik zu sein, mehr Konvention als Konfiguration. Dies ermöglicht ein schnelles Prototyping für Data Scientists, Machine-Learning-Praktiker und Softwareentwickler. DL4J ist im Maßstab anpassbar. Unter der Apache 2.0-Lizenz veröffentlicht, gehören alle Derivate von DL4J ihren Autoren. DL4J kann Modelle aus Frameworks wie TensorFlow, Caffe und Theano importieren [15].

Wie zu sehen, gibt es aktuell viele Deep Learning Frameworks. Die meisten dieser Framework sind Open-Source Projekte und auf GitHub veröffentlicht. Heutzutage ist es wichtig, dass sich diese Technologie schnell weiterentwickeln, dazu

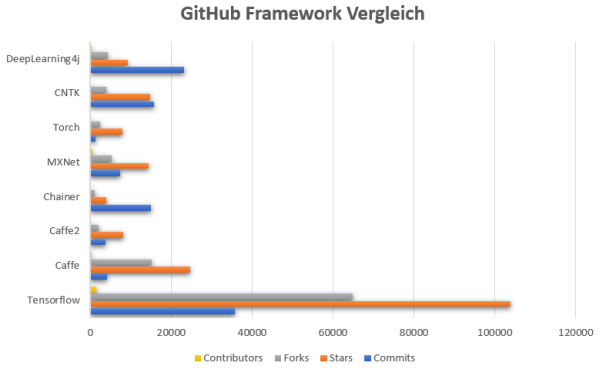

Fig. 2: GitHub: Vergleich des Interesses an o.g. Deep Learning Frameworks

ist eine große Gemeinschaft an Entwicklern notwendig. Abbildung 2 zeigt das Interesse dieser Gemeinschaft an den verschiedenen Deep Learning Frameworks vom 1. Juli 2018. Das Framework Tensorflow wird dabei besonders viel Interesse geschenkt, deshalb wollen wir dieses Framework im nächsten Kapitel genauer untersuchen.

III. TensorFlow

TensorFlow ist eine Open-Source-Softwarebibliothek für die numerische Hochleistungsberechnung. Die flexible Architektur ermöglicht die einfache Implementierung von Berechnungen auf einer Vielzahl von Plattformen (CPUs, GPUs, TPUs) und von Desktops über Cluster von Servern bis hin zu mobilen und Edge-Geräten. Ursprünglich von Forschern und Ingenieuren des Google Brain-Teams in der KI-Organisation von Google entwickelt, bietet es eine starke Unterstützung für maschinelles Lernen und Deep Learning. Der Kern der flexiblen numerischen Berechnung wird in vielen anderen wissenschaftlichen Bereichen eingesetzt [16].

A. Installation

TensorFlow kann auf den folgenden Plattformversionen und höher installiert werden:

- macOS 10.12.6
- Ubuntu 16.04
- Windows 7

Standardmäßig werden Anwendungen für TensorFlow in der Programmiersprache Python geschrieben, es lassen sich jedoch Entwicklungsumgebungen für die Programmiersprachen C++, Go, Swift und Java einrichten.

B. Unterstützung von mobilen Plattformen

Google stellt verschiedene Versionen Ihres Frameworks bereit. Unter anderem auch Versionen für mobile Endgeräte, die für diese optimiert sind und mit weniger zur Verfügung stehenden Ressourcen auskommen. Im folgenden werden die verschiedenen Versionen kurz vorgestellt.

1) TensorFlow Mobil: TensorFlow Mobil wurde für den Einsatz auf einem mobilen Gerät optimiert und enthält alle Funktionen die das Framework zu bieten hat. Jedoch bietet TensorFlow Mobil nicht die optimale Performance für mobile Geräte. Deshalb wurde TensorFlow Light entwickelt, welches sich aktuell noch im Entwicklungsstatus befindet. Dabei werden bei TensorFlow Light nur eine begrenzte Anzahl an Funktionen unterstützt, sodass die binäre Größe kleiner ist und ein Performance Steigerung erzielt werden kann. Ob TensorFlow Mobil oder Light für eine Anwendung eingesetzt werden soll, hängt von deren Anforderungen ab [16].

2) TensorFlow Light: TensorFlow Lite ist eine Lösung für mobile und eingebettete Geräte wie beispielsweise Smartphones oder Smartwatches. Es ermöglicht maschinelles Lernen auf dem Gerät mit geringer Wartezeit und einer kleinen binären Größe. TensorFlow Lite unterstützt auch Hardware-Beschleunigung mit der Android Neural Networks API. TensorFlow Lite verwendet viele Techniken, um niedrige Latenzzeiten zu erreichen, wie z. B. die Optimierung der Kernel für mobile Apps, vorfusionierte Aktivierungen und quantisierte Kernel, die kleinere und schnellere Modelle ermöglichen [16]. Gerade der Einsatz von einem Deep Learning Framework direkt auf einem mobilen Gerät ermöglicht viele neue Anwendungsfälle und erlaubt auch den Einsatz des Frameworks, wenn das mobile Gerät keine Internetverbindung hat. Meist reicht es auch aus, wenn auf dem mobilen Gerät eine bereits trainiertes Model genutzt wird, sofern kein weiteres Training des Models nötig ist.

3) TensorFlow.js: TensorFlow.js ist eine JavaScript-Bibliothek zum Trainieren und Bereitstellen von Machine Learning Modellen im Browser und auf Node.js. Es können entweder vorhandene Modell genutzt und weiter trainiert oder neue Modelle mit einer innovativen API gebaut und bereitgestellt werden [17]. Dies ermöglicht die Integration von maschinellem Lernen in Webseite, die auf jeder Plattform mit einem Browser genutzt werden kann.

IV. Distributed TensorFlow

Nachdem im vorherigen Kapitel einen kleiner Überblick über TensorFlow gegeben wurde, soll jetzt die Verteilung einer TensorFlow Umgebung auf mehrere Instanzen näher betrachtet werden. Dieses Kapitel stellt nötiges Wissen bereit, dass im nächsten Kapitel im Rahmen eines Tutorials zur Einrichtung einer verteilten TensorFlow Umgebung eingesetzt wird.

A. Architektur

Im folgenden wird die Architektur einer verteilten TensorFlow Umgebung genauer beschrieben. Dabei wird zwischen dem Paramter Server und Arbeiter unterschieden.

1) Parameterserver: Der Parameterserver verwaltet das aktuelle Model mit seinen Gewichtungen und Variablen und verteilt es an die Arbeiter. Falls durch mehrere Arbeiter zu viel Ein-/Ausgabe Anfragen an den Parameterserver erzeugt werden, können mehrere Parameterserver eingesetzt werden, um die Last zu verteilen. Mehreren Parameterserver synchronisieren das Model untereinander. Abbildung 4 zeigt, wie das Verteilen des Models an die Arbeiter aussehen kann.

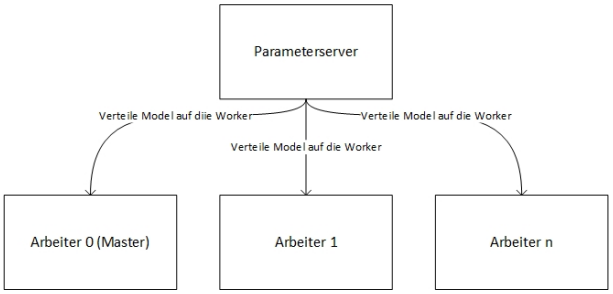

Fig. 3: Verteilen des Models auf die Arbeiter

2) Arbeiter: Arbeiter führen ressourcenintensive Berechnung durch zur Optimierung des vom Parameterserver bereitgestellten Models. Nach der Optimierung sendet der Arbeiter das neue Model mit seinen Gewichtungen und Variablen zum Parameterserver. Desto mehr Arbeiter eingesetzt werden, desto schneller und effizienter kann ein Model trainiert werden. Abbildung **??** zeigt, wie die Berechnung des Models und die Aktualisierung des Models auf dem Parameterserver aussehen kann. Der Master Arbeiter koordiniert die Trainingsvorgänge und kümmert sich um die Initialisierung des Modells, Zählen der Anzahl der ausgeführten Trainingsschritte, Speichern und Wiederherstellen von Modellkontrollpunkten.

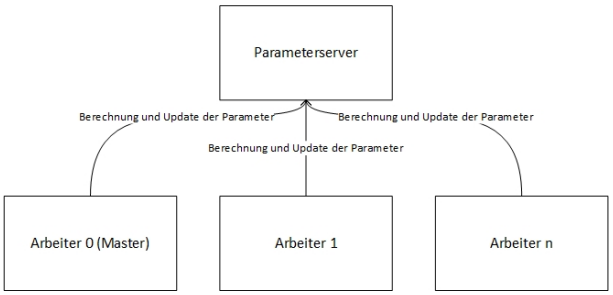

Fig. 4: Berechnung des Models auf Arbeiter + Updaten der Variablen auf dem Parameterserver

B. Datenparallelität

Es gibt zwei verschiedene Möglichkeiten für Datenparallelität:

1) Synchron: Die Arbeiter lesen gleichzeitig das Model vom Parameterserver und berechnen mit diesem ihre Trainingsoperationen und warten, bis die anderen Arbeiter fertig sind. Dann werden die Ergebnisse gemittelt und eine Aktualisierung des Models wird an den Parameterserver gesendet. Somit hat zu jedem Zeitpunkt jeder Arbeiter das gleiche Model.

2) Asynchron: Die Arbeiter lesen asynchron von den Parameterserver, berechnen ihre Trainingsoperation und senden asynchrone Aktualisierungen an den Parameterserver. Zu jedem Zeitpunkt könnten zwei verschiedene Arbeiter unterschiedliche Werte für ihr Model besitzen.

V. Tutorial: Distributed Tensorflow

VI. Evaluierung

VII. Zusammenfassung

References

[1] ”Planet Wissen - Künstliche Intelligenz” https://www.planet-wissen.de/technik/computer_und_roboter/kuenstliche_intelligenz/ Accessed 20.06.2018

[2] ”Big Data Insider - Maschinelles Lernen und Deep Learning” https://www.bigdata-insider.de/ki-maschinelles-lernen-und-deep-learning-das-sind-die-unterschiede-a-588067/ Accessed 20.06.2018

[3] ”DL4J - Deep Learning Frameworks Comparison” https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch Accessed 20.06.2018

[4] ”Torch” https://pytorch.org/about/ Accessed 20.06.2018

[5] ”Caffe” http://caffe.berkeleyvision.org/ Accessed 21.06.2018

[6] ”Caffe2” https://caffe2.ai/ Accessed 21.06.2018

[7] ”Keras” https://keras.io/ Accessed 24.06.2018

[8] ”Elephas” https://github.com/maxpumperla/elephas Accessed 24.06.2018

[9] ”Chainer” https://chainer.org/blog/ Accessed 01.07.2018

[10] ”Microsoft Cognitive Toolkit (CNTK)” https://docs.microsoft.com/de-de/cognitive-toolkit/ Accessed 01.07.2018

[11] ”Wikipedia - Apache MXNet” https://en.wikipedia.org/wiki/Apache_MXNet Accessed 01.07.2018

[12] ”AWS Apache MXNet” https://aws.amazon.com/de/mxnet/ Accessed 01.07.2018

[13] ”AWS Introducing Gluon” https://aws.amazon.com/de/blogs/aws/introducing-gluon-a-new-library-for-machine-learning-from-aws-and-microsoft/ Accessed 01.07.2018

[14] ”Amazon DSSTNE” https://aws.amazon.com/de/blogs/aws/introducing-gluon-a-new-library-for-machine-learning-from-aws-and-microsoft/ Accessed 01.07.2018

[15] ”Elipse Deeplearning4j (DL4J)” https://deeplearning4j.org/i Accessed 01.07.2018

[16] ”Tensorflow” https://www.tensorflow.org/ Accessed 01.07.2018

[17] ”Tensorflow.js” https://js.tensorflow.org/ Accessed 03.07.2018