Fakultät Informatik/Mathematik

Wissenschaftliches Seminar

im Wintersemester 2017/2018

Eine Einführung in TensorFlow

---

bearbeitet von:	Bierschneider Christian	3118760
	Maximilian Poeschl	3121342
	Benjamin Maiwald	3097528
	Studiengang: Informatik	
	Schwerpunkt: Software Engineering	
Betreuer:	Prof. Dr. Jan Dünnweber	
	OTH Regensburg	

Regensburg, 29. November 2017

Abstract

Abkürzungsverzeichnis

KI	Knstliche Intelligenz.
ML	Machine Learning, bzw. Maschinelles Lernen.
Neuronales Netz	Neuronales Netz, bzw. Neural Network. .
TF	TensorFlow.

Inhaltsverzeichnis

Kapitel 1

1 Eine Einführung in Maschinelles Lernen

Da Vorwissen in den Bereichen Künstliche Intelligenz (KI) und Machine Learning (ML) selbst bei Studierenden der Informatik nicht generell vorausgesetzt werden kann, gibt dieses Kapitel eine kurze Einführung in das Thema. Es wird über die Grundlagen im Bereich des Maschinellen Lernens mit dem Schwerpunkt auf informiert, die zum Verständnis der Arbeit benötigt werden. Sollte sich der Leser bereits mit dem Thema auseinander gesetzt haben und Begriffe wie „Label“, „Schichten (Layers)“ und „Gewichte“ schon bekannt sein, kann dieses Kapitel auch übersprungen werden.

Für große Teile des geschichtlichen Abrisses diente das Buch „Künstliche Intelligenz, ein moderner Ansatz“ von Stuart Russell und Peter Norvig als Quelle, welches wohl eines der bekanntesten Werke zum Thema KI sein dürfte [1]. Generell kann ich dieses Buch allen Interessierten empfehlen, gleichwohl aufgrund des doch sehr großen Umfangs je nach Situation meist nur einzelne Kapitel hilfreich sein werden.

1.1 Meilensteine des Maschinellen Lernens

In den letzten Jahren (seit ca. 2010) hat sich das Thema KI zu einem regelrechten Hype-Thema entwickelt — und das nicht ganz zu unrecht. Denn gerade durch Anwendungen in den Bereichen Bildverarbeitung, Spracherkennung, sogenannter „Recommender Systems“ oder auch des automatisierten Fahrens, gab es enorme Fortschritte. Diese machten KI für die breite Masse salonfähig und ermöglichten die Entwicklung von Produkten wie Siri, den Skype Translator, Filmempfehlungen auf Netflix oder die Fahrerassistenzsysteme im Tesla Model S, die heute von Millionen von Menschen täglich genutzt werden.

Allen voraus liegt das Hauptaugenmerk vieler Informatiker und Forscher gerade auf den sogenannten „Künstlichen Neuronalen Netzen“ (Artificial Neural Networks). Manche dieser Neuronalen Netze wurden sogar zu echten Superstars in der Szene, wie zum Beispiel „AlphaGo“, das von Google entwickelt wurde und 2016 den damaligen Vize-Weltmeister Lee Sedol in vier von fünf Runden im Brettspiel „Go“ besiegte. Aufgrund der unglaublichen

Komplexität[1] des Spiels galt der Sieg einer Maschine über einen realen Meister lange Zeit als unmöglich.

Dabei sind die meisten Grundlagen auf diesem Gebiet bereits Jahrzehnte alt. Schon in den 1940er Jahren und somit unmittelbar nach der Erfindung des modernen Computers, begannen die ersten Forscher damit, ein Modell für Künstliche Neuronale Netze zu entwickeln und behaupteten sogar bereits, dass entsprechend definierte Netze auch lernfähig seien. In demselben Artikel, in dem Alan Turing 1950 die Idee des weltbekannten Turing-Tests — in [2] „The Imitation Game“ genannt — veröffentlichte, schrieb er außerdem zum ersten Mal über „lernende Maschinen“ und philosophierte darüber, wie man einer Maschine beibringen könne, im Imitation Game zu bestehen. In dieser Niederschrift formulierte Turing auch die Grundideen zum heute als „Reinforcement Learning“ bezeichneten Lernen durch Bestrafung und Belohnung.

1956 war schließlich offiziell das Geburtsjahr der Künstlichen Intelligenz. Am Dartmouth College (Hanover, New Hampshire) veranstalteten McCarthy, Minsky, Shannon und Rochester (allesamt Größen in der Entwicklung der KI) ein „Summer Research Project on Artificial Intelligence“ zusammen mit weiteren Forschern aus ganz Amerika. Hier wurde der Begriff „Artificial Intelligence“ zum ersten Mal überhaupt benutzt. Ziel des Workshops war es, in zwei Monaten einen signifikanten Fortschritt bei der Entwicklung einer intelligenten Maschine zu erreichen. Dieses Ziel konnte zwar nicht erfüllt werden, jedoch sorgte das Treffen dafür, dass sich die wichtigsten Personen kennenlernten, die in den darauffolgenden 20 Jahren die größten Neuerungen auf diesem Gebiet entwickelten.

Wie auch heute gab es schon einmal in den 1980er Jahren einen großen Boom in der KI-Industrie. Die Investitionen stiegen von einigen Millionen Dollar im Jahr 1980 auf mehrere Milliarden Dollar im Jahr 1988. Viele der KI-Firmen konnten ihre Versprechen jedoch nicht halten, weshalb der Markt in den 90er Jahren zusammenbrach. In Folge dessen ging auch die Forschung auf dem Gebiet zurück und es kam zu keinen nennenswerten Erkenntnisgewinnen in den 90er Jahren. Deshalb wird dieses Jahrzehnt auch als „KI-Winter“ bezeichnet.

Wenn alle diese Entwicklungen im Bereich der KI aber bereits so lange zurück liegen, warum hat es dann bis heute gedauert, dass es die ersten Anwendungen zum Endkunden schaffen?

Wir leben heute in einer spannenden Zeit, denn einige wichtige Faktoren, die für den Erfolg der KI wichtig sind, wurden nahezu zeitgleich verfügbar.

[1] Ein Go-Brett besteht aus einem Raster von 19 x 19 Plätzen zum Setzen. Für den ersten Zug existieren also 361 Möglichkeiten, für den zweiten 360 und so weiter. Dies ergibt bereits für die ersten drei Züge mehr als 46 Millionen mögliche Spielabläufe.

Zum einen stieg die Leistung von Computern seit deren Erfindung stetig an. Für die meisten Berechnungen im Bereich des Maschinellen Lernens wird eine sehr hohe Rechenleistung benötigt. Vor allem die Verwendung von GPUs zur parallelen Berechnung von allgemeinen Aufgaben beschleunigt das Trainieren von neuronalen Netzen um ein Vielfaches. Rechenvorgänge, die noch vor zehn Jahren große Rechencluster für viele Monate beanspruchten, können heute in Stunden, maximal aber wenigen Tagen abgeschlossen werden – und das auf kompakten Rechnern, die sogar für Privatpersonen erschwinglich sind. Dies gibt den Forschern und Softwareingenieuren die Möglichkeit, schon nach kurzer Zeit ein Feedback zu erhalten, ob die von Ihnen gewählten Ansätze richtig sind und falls nicht, Anpassungen vorzunehmen.

Zum anderen leben wir im Zeitalter von „Big Data“. Für ML ist es extrem wichtig, dass große Datenmengen zur Verfügung stehen, die für den Trainingsprozess verwendet werden können. Firmen wie Google, Facebook, Apple oder Microsoft (und natürlich vielen anderen) steht ein schier unerschöpflicher Pool an Informationen zur Verfügung. Diese maschinell generierten Daten eignen sich aufgrund ihres großen Umfangs perfekt dazu, neuronale Netze zu trainieren und das wird von diesen Firmen natürlich auch genutzt, um neue Geschäftsideen zu entwickeln und die angebotenen Dienste für ihre Kunden kontinuierlich zu verbessern.

Noch hinzu kam der dritte große Punkt: Die Entwicklung von „hacks“, welche die bereits erforschten Algorithmen leicht abwandelten, um enorme Einsparungen in der Rechenzeit zu erreichen. So fand man zum Beispiel heraus, dass es in den meisten Fällen ausreicht, nur einen bestimmten Prozentsatz eines neuronalen Netzes zu trainieren, um trotzdem nahezu das gleiche Ergebnis zu erzielen [**?**].

Natürlich birgt die Entwicklung der Künstlichen Intelligenz auch Gefahren. Diese können ganz real sein, wie der Wegfall tausender Arbeitsplätze [3] oder aber spekulativ und in der Zukunft liegend, wie die mögliche Gefährdung der Menschheit durch eine künstliche Superintelligenz [4]. In jedem Fall wird die Weiterentwicklung und Forschung auf dem Gebiet auch die nächsten Jahre ein extrem vielfältiges und spannendes Thema bleiben.

1.2 Grundlagen von Neuronalen Netzen

Kapitel 2

2 Das Framework TensorFlow

TensorFlow (TF) ist eine plattformunabhängige Bibliothek für ML, die für große und variable Architekturen entwickelt [5] und Ende 2015 veröffentlicht wurde [6]. Um die einzelnen Verarbeitungsschritte der Daten darzustellen, werden von TF sogenannte Datenfluss Graphen verwendet. Diese bieten auch die Möglichkeit für alle Operationen festzulegen, von welcher Hardware sie berechnet werden sollen. TF unterstützt dabei CPUs, GPGPUs[1] und eigens für ML entwickelte Hardware. Besonders umfangreich unterstützt das Framework Arbeiten im Bereich der (tiefen) Neuronale Netze (NN). Schnittstellen zu den Hochsprachen Python und C++ sollen den Einstieg erleichtern und sicherstellen, dass die verfügbare Hardware immer bestmöglich genutzt werden kann. Mit dem sogenannte Tensorboard bringt TF außerdem eine Weboberfläche mit, die ohne großen Aufwand für den Entwickler viele relevante Informationen ausgibt und teilweise auch grafisch aufbereitet [5].

2.1 Eine Einführung zu TensorFlow

2.2 Die Entwicklung von Tensorflow

Bereits 2011 begann das Google Brain Projekt damit, den Nutzen von sehr großen tiefen Neuronales Netz zu erforschen. Einen Teil davon bildete der Aufbau von „DistBelief“, ein System, das Training und Vorhersage im Bereich des ML verteilt und skalierbar ermöglichte. Neben einigen Forschungsprojekten wurde DistBelief auch bereits produktive in einigen Google Produkten, wie Google Search, Google Tanslate oder Youtube eingesetzt. 2012 wurde von Google Mitarbeitern ein Paper veröffentlicht, das über die Nutzung von zentausenden CPU-Kernen durch DistBelief berichtete, wodurch auch sehr große Modelle in absehbarer Zeit trainiert werden konnten [7]. Auf Basis der Erfahrungen beim Einsatz von DistBelief arbeitete Google dann am System der zweiten Generation für groß-skalierende ML Modelle, Tensorflow. Im November 2015 wurde TF veröffentlicht [6] und ist seit dem auf github unter der Apache License 2.0 verfügbar [8]. Mit der Veröffentlichung

[1] general-purpose graphics processing units

von Version 1.0 im Februar 2017 wurde schließlich eine verlässliche API eingeführt, die auch in Zukunft sicher stellen soll, dass der geschriebene Code mit neuen Versionen von TF kompatibel ist [9]. Des weiteren wurde die Leistung weiter verbessert und die Einführung eines neuen Moduls ermöglicht seither die Nutzung von TF mit Keras[2].

2.3 Angesprochene Zielgruppe

2.4 Hard- und Software Anforderungen

2.4.1 Hardware Anforderungen

2.4.2 Software Anforderungen

2.5 Softwarearchitektur von TensorFlow

[2]Keras ist eine Deep Learning Library, die auf TF, CNTK oder Theano aufsetzt.

Kapitel 3

3 Der Allgemeine Workflow in TensorFlow

3.1 Die Tooling Pipeline

3.2 Vorgehensweise beim Trainingsprozess

3.3 Die Visualisierung mit TensorBoard

TensorBoard ist eine in der Installation von TensorFlow enthaltene Webanwendung zur Visualisierung der Abläufe in einem neuronalen Netz. TensorBoard stellt eine Vielzahl an graphischen Elementen zur Verfügung, welche dem Entwickler vorallem das Debuggen oder die Optimierung eines erstellten Modells erleichtern. Ebenso können damit insbesondere komplexe Datenstrukturen zum besseren Verständnis anschaulich visualisiert werden.

Bevor mit TensorBoard gearbeitet werden kann, muss folgender Befehl 3.1 in die Kommandozeile eingegeben werden:

$$tensorboard \quad --logdir = path/to/log\_directory \tag{3.1}$$

wobei vorher im spezifizierten Log Ordner die gewünschten Event Daten mit der Klasse *tf.summary.FileWriter('path/to/log-directory')* abgespeichert werden müssen. Nachdem TensorBoard gestart wurde, navigiert man mit dem Browser zu folgender Seite:

$$http://localhost:6006 \tag{3.2}$$

Falls keine Fehlermeldungen aufgetreten sind, sollte folgender Bildschirminhalt angezeigt werden:

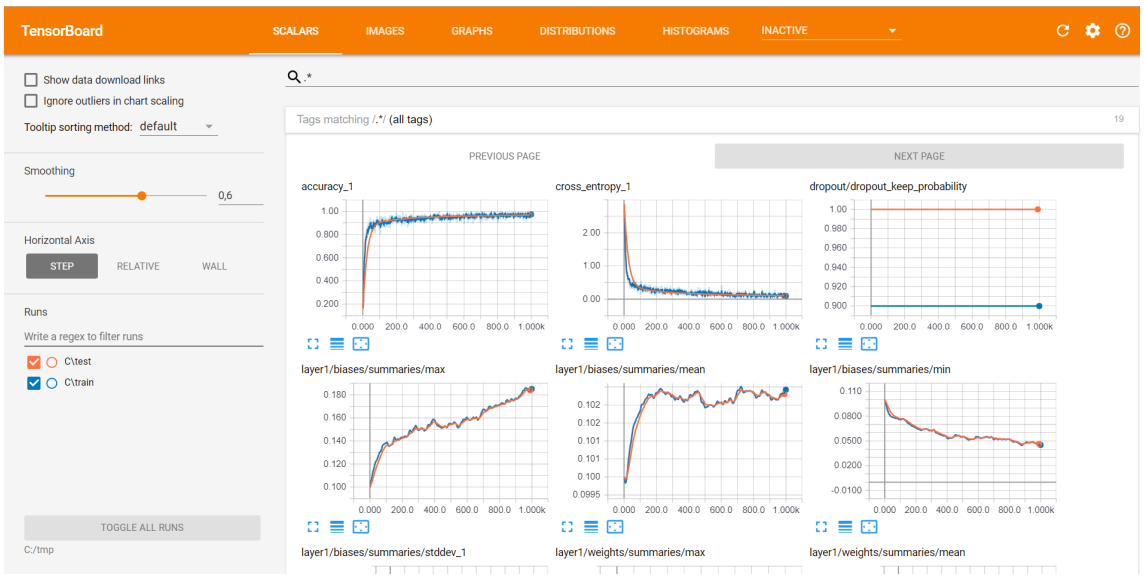


Abbildung 3.1: Die Startseite von TensorBoard

3.3.1 Die einzelnen Visualisierungsmöglichkeiten im Detail

Skalare

Unter dem Reiter Skalare, welche mit der Klasse "tf.summary.scalar“ abgespeichert werden, können verschiedenste Statistiken während eines Trainingsprozesses visualisiert werden. Dies könnten zum Beispiel die Genauigkeit oder Cross-Entropie sein, welche in Abbildung 3.2 dargestellt sind. Hierbei wird die Genauigkeit über den einzelnen Trainingsschritten aufgetragen. Wählt man mit der Maus einen bestimmten Datenpunkt aus, so werden zahlreiche weitere Informationen angezeigt. Ebenso ist hierbei ersichtlich, dass auch Trainings- und Testdaten gleichzeitig angezeigt und miteinander verglichen werden können.

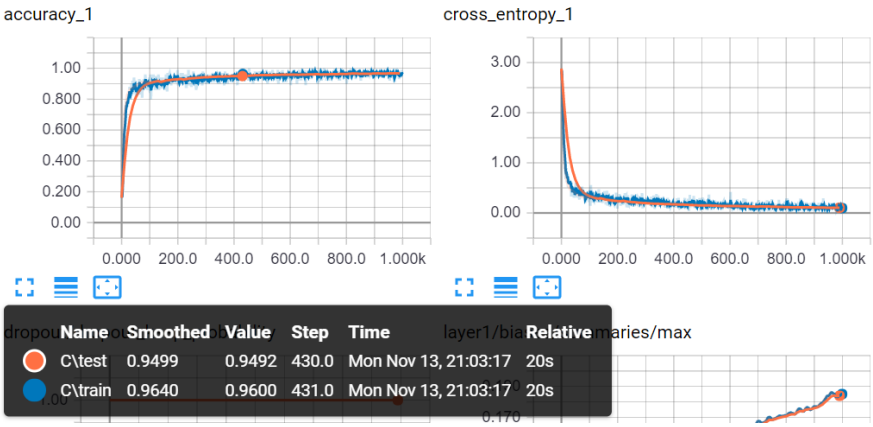


Abbildung 3.2: Visualisierung der 'Accuracy' und 'Cross_entropy' über die einzelnen Trainingschritte

Bilder

Innerhalb des Reiters Bilder, welche mit der Klasse "tf.summary.image“ abgespeichert werden, können zur genaueren Analyse die Test- und Trainingsbilder eingesehen werden. Über den Bildern ist eine Scrollbar vorhanden mit dieser können einzelne Test- und Trainingsschritte ausgewählt werden, wodurch genau ersichtlich wird, welches Bild zum aktuellen Durchlauf gehört.

Abbildung 3.3: Bild des Testschrittes 490

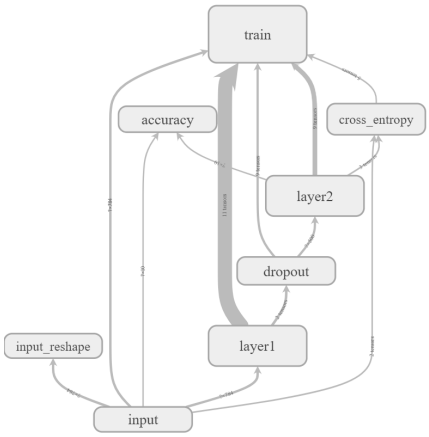


Abbildung 3.4: TensorFlow Graph mit definierten Name scopes

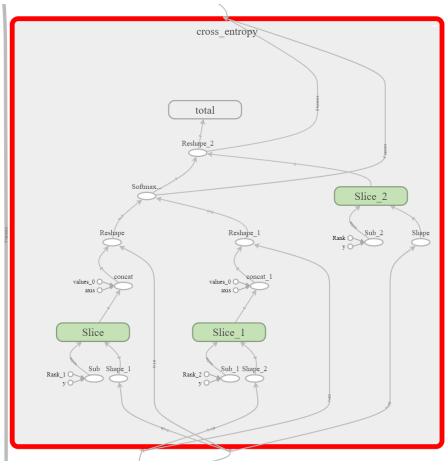


Abbildung 3.5: Name scope 'cross_entropy' unterteilt in weiteren Operationen

Graphen

Unter dem Reiter Graphen befindet sich das komplette TensorFlow Model als Graph wie in Abbildung 3.4 zu sehen ist. Um einen Graphen in Tensorboard zu erhalten, müssen im Programm die gewünschten Operationen als Name scope erstellt werden. Mit nachfolgendem Befehl 3.3 erhält man einen 'cross_entropy' Name scope:

$$with \quad tf.name\_scope('cross\_entropy'): \tag{3.3}$$

Der Name scope *cross_entropy* Abbildung 3.5 kann natürlich wiederum in mehrere Untergruppierungen aufgeteilt werden. So erhält man eine übersichtliche Visualisierung komplexer TensorFlow Modelle.

Histogramme

Unter dem Reiter Histogramme, welche mit der Klasse "tf.summary.histogram“ abgespeichert werden, wird die statistische Verteilung eines Tensors über der Zeit dargestellt. Im Histogramm sind zeitliche "Slices“ der Daten visualisiert, wobei jeder einzelne Slice ein Histogramm des Tensors in einem einzelnen Schritt darstellt. In Abbildung 3.6 ist ein einzelner Slice schwarz markiert.

Verteilungen

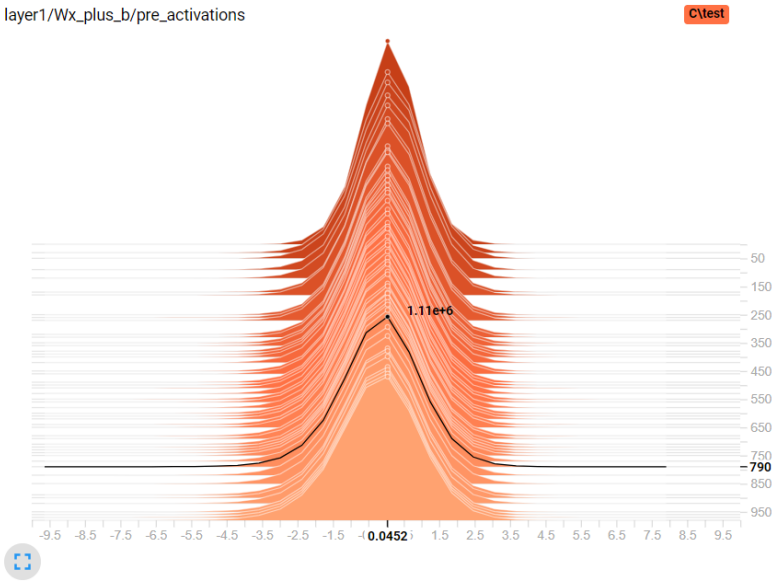
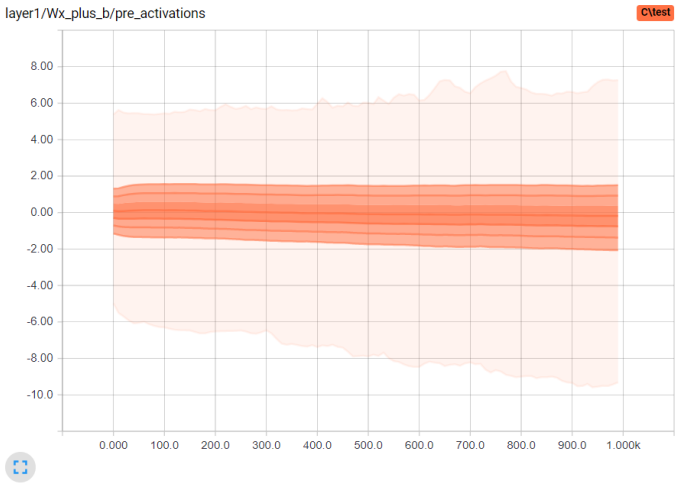


Abbildung 3.6: Visualisierung eines Histogramms über die einzelnen Trainingschritte



Audio

Projektor

Text

3.3.2 Die Graphelemente im Datenfluss

Dafür wurden zahlreiche unterschiedliche Elemente definiert, welche nachfolgend kurz erläutert werden.

Namespace

High-level Knoten repräsentiert einen definierten Namensbereich.

Unconnected series

Nummerierte Knoten, die nicht miteinander verbunden sind.

Connected series

Nummerierte Knoten, die miteinander verbunden sind.

Operation node

Ein Knoten der eine einzelne Operation darstellt.

Constant

Repräsentiert eine Konstante im Programm.

Summary node

Dieser Knoten stellt eine Zusammenfassung dar.

Dataflow edge

Durchgezogener Pfeil zeigt den Datenfluss zwischen den Operationen an.

Control edge

Gepunkteter Pfeil zeigt die Steuerungsabhängigkeit zwischen den Operationen an.

Reference edge

Gelber Pfeil bedeutet, dass die ausgehende Operation die ankommende mutieren kann

Kapitel 4

4 Ausblick

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Literaturverzeichnis

[1] Russell, Stuart ; Norvig, Peter ; Kirchner, Frank: *Künstliche Intelligenz: Ein moderner Ansatz*. 3., aktualisierte Aufl. München : Pearson Higher Education, 2012 (Always learning). – ISBN 978–3–86894–098–5

[2] TURING, A. M.: I.—COMPUTING MACHINERY AND INTELLIGENCE. In: *Mind* LIX (1950), Nr. 236, S. 433–460. http://dx.doi.org/10.1093/mind/LIX.236.433. – DOI 10.1093/mind/LIX.236.433. – ISSN 0026–4423

[3] Russell, Stuart ; Dewey, Daniel ; Tegmark, Max: Research Priorities for Robust and Beneficial Artificial Intelligence. (2015), jan. http://futureoflife.org/data/documents/research_priorities.pdf

[4] Barrat, James: *Our Final Invention: Artificial Intelligence and the End of the Human Era*. THOMAS DUNNE BOOKS, 2013 http://www.ebook.de/de/product/20253628/james_barrat_our_final_invention_artificial_intelligence_and_the_end_of_the_human_era.html. – ISBN 0312622376

[5] Abadi, Martín ; Barham, Paul ; Chen, Jianmin ; Chen, Zhifeng ; Davis, Andy ; Dean, Jeffrey ; Devin, Matthieu ; Ghemawat, Sanjay ; Irving, Geoffrey ; Isard, Michael ; Kudlur, Manjunath ; Levenberg, Josh ; Monga, Rajat ; Moore, Sherry ; Murray, Derek G. ; Steiner, Benoit ; Tucker, Paul ; Vasudevan, Vijay ; Warden, Pete ; Wicke, Martin ; Yu, Yuan ; Zheng, Xiaoqiang: *TensorFlow: A system for large-scale machine learning*. https://www.tensorflow.org/. Version: 2016. – Software available from tensorflow.org

[6] Dean, Jeff ; Monga, Rajat: *TensorFlow - Google's latest machine learning system, open sourced for everyone*. https://research.googleblog.com/2015/11/tensorflow-googles-latest-machine_9.html, . – [letzter Zugriff: 28. Nov 2017]

[7] Dean, Jeffrey ; Corrado, Greg ; Monga, Rajat ; Chen, Kai ; Devin, Matthieu ; Mao, Mark ; Ranzato, Marc'aurelio ; Senior, Andrew ; Tucker, Paul ; Yang, Ke ; Le, Quoc V. ; Ng, Andrew Y.: Large Scale Distributed Deep Networks. Version: 2012. http://papers.nips.cc/

paper/4687-large-scale-distributed-deep-networks.pdf. In: Pereira, F. (Hrsg.) ; Burges, C. J. C. (Hrsg.) ; Bottou, L. (Hrsg.) ; Weinberger, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, 1223–1231

[8] *tensorflow, Computation using data flow graphs for scalable machine learning*. https://github.com/tensorflow/tensorflow, . – [letzter Zugriff: 28. Nov 2017]

[9] Sandjideh, Amy M.: *Announcing TensorFlow 1.0*. https://developers.googleblog.com/2017/02/announcing-tensorflow-10.html, . – [letzter Zugriff: 29. Nov 2017]

[10] Jeffrey Dean, Rajat Monga Kai Chen Matthieu Devin Quoc V. Le Mark Z. Mao Marc'Aurelio Ranzato Andrew Senior Paul Tucker Ke Yang Andrew Y. N. Greg S. Corrado C. Greg S. Corrado: *Large Scale Distributed Deep Networks* . https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/40565.pdf, . – [letzter Zugriff: 28. Nov 2017]

[11] *TensorBoard*. https://github.com/tensorflow/tensorboard. – Zuletzt besucht am 25.11.2017

[12] Abadi, Martín ; Agarwal, Ashish ; Barham, Paul ; Brevdo, Eugene ; Chen, Zhifeng ; Citro, Craig ; Corrado, Greg S. ; Davis, Andy ; Dean, Jeffrey ; Devin, Matthieu ; Ghemawat, Sanjay ; Goodfellow, Ian ; Harp, Andrew ; Irving, Geoffrey ; Isard, Michael ; Jia, Yangqing ; Jozefowicz, Rafal ; Kaiser, Lukasz ; Kudlur, Manjunath ; Levenberg, Josh ; Mané, Dan ; Monga, Rajat ; Moore, Sherry ; Murray, Derek ; Olah, Chris ; Schuster, Mike ; Shlens, Jonathon ; Steiner, Benoit ; Sutskever, Ilya ; Talwar, Kunal ; Tucker, Paul ; Vanhoucke, Vincent ; Vasudevan, Vijay ; Viégas, Fernanda ; Vinyals, Oriol ; Warden, Pete ; Wattenberg, Martin ; Wicke, Martin ; Yu, Yuan ; Zheng, Xiaoqiang: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/. Version: 2015. – Software available from tensorflow.org