Seminararbeit über Visuelles Lernen Arithmetischer Operationen

Lara Aubele

Seminar KI: gestern, heute, morgen
Angewandte Informatik, Universität Bamberg

Zusammenfassung. Die Arbeit *Visual Learning of Arithmetic Operations* (Hoshen & Peleg, 2016) befasst sich mit dem Erlernen von arithmetischen Operationen. Dabei soll das neuronale Netz die Transformation zweier Ausgangsbilder zu einem Ergebnisbild durch Anwendung der Operation erlernen. Das Training findet dabei in einem Ende-zu-Ende Prozess statt. Das Netzwerk bekommt somit nicht die Möglichkeit die Umwandlung der Bildrepräsentation zu einer Zahlendarstellung zu erlernen, bevor es die Berechnung trainiert. Bereits zuvor wurden arithmetische Berechnungen binärer Zahlen in einer Bilddarstellung mit neuronalen Netzen durchgeführt. Dabei wurden jedoch für die Ausgangszahlen oder das Ergebnis Zahlenrepräsentationen genutzt. Der Ansatz von Hoshen und Peleg zeigt gute Resultate für die Addition und Subtraktion mit Dezimalzahlen. Schlechte Ergebnisse wies er jedoch bei dem Rechnen mit römischen Zahlen, sowie der Multiplikation mit Dezimalzahlen auf.

Schlüsselwörter: Arithmetische Operationen, Visuelles Lernen, Deep Neural Network, Ende-zu-Ende

1 Einleitung

Im Bereich des visuellen Lernens und dem Lernen von Berechnungen binärer Zahlen gab es vor dieser Arbeit bereits umfangreiche Forschungen. Jedoch wurde das Erlernen arithmetischer Operationen, bei der die Dezimalzahlen als Bild vorliegen, noch nicht genauer betrachtet.

Generell wird diese Berechnung in zwei Teilaufgaben aufgeteilt, die perzeptuelle Aufgabe, die beispielsweise die Objekterkennung beinhaltet, und die kognitive Aufgabe, die unter anderem mentale Modelle und Interpretationen der Wahrnehmungen umfasst. In dem hier vorgestellten Ansatz werden arithmetischen Operationen als Ende-zu-Ende Prozess von zwei Ausgangsbildern zu einem Ergebnisbild gelernt.

Durch das Ende-zu-Ende Lernen kann das System selbstständig Datensätze (hier die Zahlenbilder) einander zuordnen, ohne dabei die Konzepte des Menschen mit einzubeziehen. Außerdem müssen dadurch weniger Teilaufgaben einzeln erlernt werden. Arithmetische Operationen sind dabei relativ einfache Aufgaben, die jedoch für den Menschen eine große Rolle im Leben spielen. Vom Ende-zu-Ende Lernen arithmetischer Operationen lässt sich eine Übertragbarkeit

auf das Erlernen arithmetischer Operationen ohne eine gemeinsame Sprache herstellen. Im Training werden lediglich exemplarische Rechnungen inklusive des Ergebnisses gegeben, wie es auch mit einer Sprachbarriere unter Menschen möglich wäre. *Visual Learning of Arithmetic Operations* stellt außerdem das erste rein visuell gelöste Problem vor. Es muss noch erforscht werden auf welche weiteren, möglicherweise schwereren Probleme dies angewendet werden kann.

In dem Ansatz der Autoren werden zwei Eingangsbilder durch Anwendung einer arithmetischen Operation zu einem Ergebnisbild umgewandelt. Dabei werden die Teilaufgaben nicht einzeln trainiert. Das bedeutet, dass die Transformation des Bildes in die Repräsentation als Zahl nicht entkoppelt von der arithmetischen Operation erlernt werden kann. Als Lernender wird hier ein neuronales Netz eingesetzt.

In diesem Bericht gehe ich zuerst auf die Einordnung der wissenschaftlichen Arbeit, *Visual Learning of Arithmetic Operations* (Hoshen & Peleg, 2016), ein. Dann folgt der Hauptteil, in dem ich die Arbeit vorstelle und dabei auf den Aufbau des durchgeführten Experiments eingehe, die Abläufe innerhalb des Netzwerks darlege und dies an einem Beispiel erläutere. Außerdem sind dazu die Ergebnisse der Versuche aufgeführt. Abschließend folgt eine kurze Zusammenfassung und eine Diskussion. In dieser Diskussion gehe ich auf kritischen Punkte der Arbeit ein.

2 Wissenschaftlicher Hintergrund

2.1 Vorangegangene Arbeiten

Bereits vor dieser Veröffentlichung wurde die Implementierung von arithmetischen Operationen mit Binärzahlen erforscht. Bezugnehmend auf andere Zahlenrepräsentationen bei dem Lernen arithmetischen Operationen, nahm Marr schon 1982 an, dass das Rechnen mit römischen Zahlen schwieriger ist, als mit Dezimalzahlen. Ein gut erforschtes Gebiet bildet die optische Zeichenerkennung (OCR), die auch für die Auswertung in dieser Arbeit genutzt wurde.

Außerdem gab es bereits eine Arbeit zu der Addition mit Long Short Term Memorys (LSTMs), die nicht mit Bildern, sondern nur mit Textdaten lernten (Zaremba & Sutskever, 2014). Bilder wurden dann im selben Jahr mit Deep Recurrent Networks eingesetzt. Dabei wurden zwei handgeschriebene Zahlen addiert. Bei diesem Experiment war das Ergebnis aber eine Zahl und kein Bild (Zaremba, Kurach & Fergus, 2014). Arithmetische Operationen können auch als Vorhersage der Veränderung gedeutet werden, daher ist die Videobildvorhersage eine weitere vorangehende Arbeit (Vondrick, Pirsiavash & Torralba, 2015), die jedoch eine höhere Komplexität aufweist. Die Veröffentlichung von Hoshen und Peleg ist somit die erste wissenschaftliche Arbeit, in der Ende-zu-Ende das Rechnen mit visuellen Repräsentationen gelernt wird.

2.2 Weiterentwicklung

Die vorliegende Arbeit ist relativ neu, deshalb gibt es keine sehr großen Weiterentwicklungen auf diesem Gebiet. Eine aufbauende Arbeit beschreibt das Berechnen von handgeschriebenen Zahlen mithilfe eines LSTMs (Pérez, Quevedo & Caicedo, 2017). Darin wird der Operator ebenfalls von Hand geschrieben und das Netzwerk muss die verschiedenen Operatoren unterscheiden und anwenden.

2.3 Begrifferläuterungen

Für die Experimente wird von den Autoren ein *Feedforward fully connected artificial neural Network* genutzt. Dies ist ein neuronales Netz, das nur Informationen in eine Richtung weitergibt. Das heißt, dass das Netzwerk keine zirkulären Beziehungen der Informationsweitergabe aufweist. Das Netzwerk hat eine Eingangs- und eine Ausgangsschicht und kann mehrere Hidden Layer besitzen. [1]
Zum Trainieren des neuronalen Netzes wird hier auf das Training mit *Mini-Batch Stochastic Gradient Descent* zurückgegriffen. Die Trainingsdaten werden dabei in Mini-Batches (kleine Gruppe aus Datensätzen) unterteilt, für die jeweils dann der Fehler ermittelt und die Gewichte entsprechend an die Trainingsdaten angepasst werden. Durch die Mini-Batches erhält man eine höheren Effizienz, als durch das Training mit einzelnen Datensätzen. [2]
Für die Anpassung im Training wird dann der *Backpropagation Algorithmus* genutzt. Dieser sorgt dafür, dass die Berechnung des Fehlers und die Anpassung, von Ausgangs- in Richtung der Eingangsschicht, vorgenommen wird [3].

3 Das Lernen im neuronalen Netz

3.1 Aufbau des Experiments

In dem Experiment wird das Erlernen von Addition, Multiplikation und Subtraktion mit Dezimalzahlen, sowie die Addition mit römischen Zahlen getestet. Die Zahlen auf den Ausgangs- und Ergebnisbildern sind dabei auf positive Zahlen beschränkt. Für die Dezimalzahlen ist der Größenbereich dabei auf bis zu siebenstelligen Zahlen festgelegt. Somit werden für die Addition Zahlen bis 4.999.999 zugelassen, für Subtraktion bis 9.999.999 und für Multiplikation bis 3.160. Auch bei den römischen Zahlen gilt die gleiche Obergrenze der Addition, doch durch den Unterschied in der Repräsentation beläuft sich dies auf bis zu 35 Ziffern. Das Netzwerk wird im Experiment mit 30.000 Eingangs-/Ergebnisbilderpaaren mittels des *Mini-Batch Stochastic Gradient Descent* trainiert. Dabei wird versucht den Unterschied zwischen der gewünschten und erreichten Ausgabe zu minimieren. Die Gewichte werden dann mithilfe des *Backpropagation Algorithmus* angepasst. Daraufhin wird das Netzwerk an 150.000 Paaren getestet.

[1] (McGonagle, 2018)
[2] (Brownlee, 2017)
[3] (McGonagle, 2018)

Die Aufgabe wird mit einem *Feedforward fully connected Deep Neural Network* gelernt. Das Netzwerk umfasst drei Hidden Layer, eine Eingangs- und eine Ausgangsschicht (siehe Abbildung 1). Die Eingänge der Schichten werden immer zuerst durch die Gewichte ausgewertet und daraufhin wird anhand der Aktivierungsfunktion die wahrscheinlichste Klasse ausgewählt. Die Hidden Layer weisen

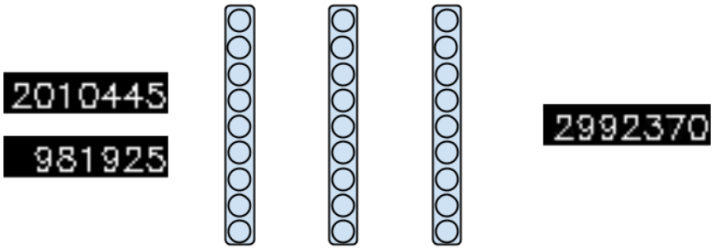

Abb. 1. Aufbau des Netzwerks
(Hoshen & Peleg, 2016)

dabei jeweils 256 Neuronen auf und verwenden die Rectified Linear Unit (ReLU) als Aktivierungsfunktion (siehe Abbildung 2).

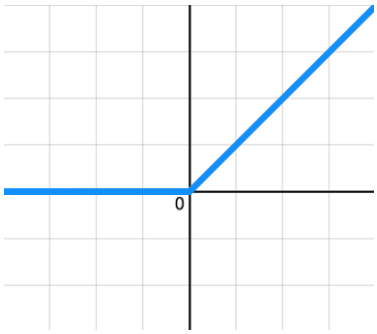

Abb. 2. Rectified Linear Unit
(Holczer, 2017)

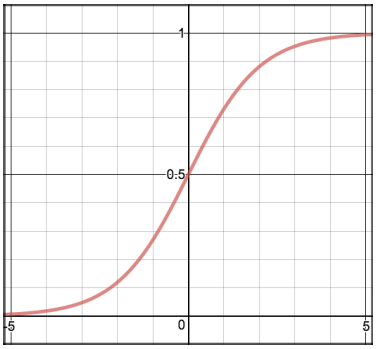

Abb. 3. S-förmige Aktivierungsfunktion
(Fortuner, 2017)

Das bedeutet, dass negative Werte durch die Funktion Null werden und positive ihren Wert behalten. Also werden dadurch negative Werte entfernt. Die ReLU ist eine gängige Aktivierungsfunktion für die Hidden Layer in neuronalen Netzen im Machine Learning. Die Ausgabeschicht weist hingegen eine S-förmige Aktivierungsfunktion auf (siehe Abbildung 3). Dadurch werden sehr kleine Werte Null und sehr große Werte Eins. Große Eingangswerte bedeuten, dass das

Gewicht zu dem vorgefundenen Wert passt. Diese werden mit der Funktion als sehr wahrscheinlich, also Eins, eingestuft.

3.2 Erklärung des Ablaufs

Die genaue Ausführung der Berechnung wird hier durch eine geringere Anzahl der Neuronen in den Schichten vereinfacht. Zuerst wird dabei jede Zahl in den Ausgangsbildern durch Gewichte bewertet. Diese repräsentieren jeweils eine Ziffer zwischen Null und Neun. Sie sind in Abbildung 4 in den *Template Weights* als schwarze Zahl zu erkennen. Fällt die Zahl in die Klasse des Gewichts, so nimmt der Wert Eins an, ansonsten wird er Null. Dadurch wird eine One-Hot-Repräsentation erreicht, die die vorgefundene Zahl mit einer Eins markiert, damit steht in dieser Repräsentation für die Zahl Sechs beispielsweise eine Eins an der Stelle Sieben. Somit erhält man durch diese Umwandlung für jede Ziffer in den Ursprungsbildern eine dieser Repräsentationen im Netzwerk (siehe HL 1 in Abbildung 4). Diese Transformation verringert die Dimension der Darstellung

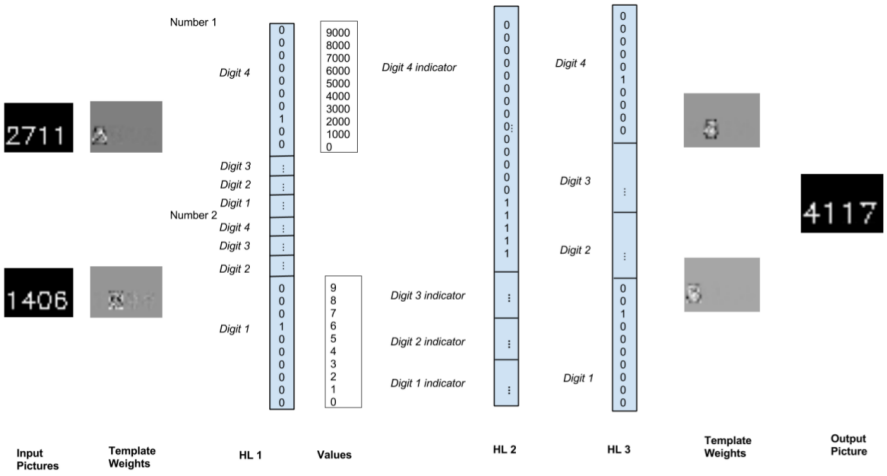

Abb. 4. Exemplarische Darstellung des Netzwerk mit der Zahlenrepräsentation (Hoshen & Peleg, 2016)

und wandelt die Bilder in eine, für die Maschine einfachere, Repräsentation um.

Für die Berechnung der zweiten Hidden Layer kommt dann die Gleichung 1 zum Einsatz.

$$v_i^m = \mathbb{1}_{\sum_{j=1}^{m}(d1^m + d2^m)*10^j \geq i*10^m} \tag{1}$$

Dabei steht i (0...19) für den Schwellenwert und gibt damit die Position im Indikator der Ziffernposition m in der Ergebniszahl an. Die Variable d1 bezeichnet

die Ziffer an der Stelle m in der ersten Zahl und d2 die Ziffer an derselben Stelle der zweiten Zahl. Dies bezieht sich nun wieder auf die Dezimalzahlen in der Bildrepräsentation. Auf das Ergebnis dieser Summe wird dann mod(10) angewendet, sodass keine Zahlen größer als Neun erreicht werden können. Hat man von der Stelle m-1 einen Übertrag, indem das Modulo die Zahl verkleinert hat, so kommt zu d1+d2 außerdem eine Addition mit Eins hinzu und man erhält in der Klammer d1+d2+1. Die Formel führt im weiteren Verlauf einen Vergleich gegen einen Schwellenwert durch und man erhält damit im Ergebnis eine Eins oder eine Null.

Die One-Hot-Repräsentation des Ergebnisses wird dann erreicht, indem die Gleichung 2 angewendet wird.

$$o_i^m = \mathbb{1}_{v_n^m - v_{n+1}^m + v_{n+10}^m - v_{n+11}^m} \tag{2}$$

N (0...19) ist dabei die Position innerhalb des jeweiligen Indikators und durch i (0...9) wird die Stelle der One-Hot-Repräsentation der jeweiligen Ziffernposition m der Ergebniszahl angezeigt. In der dritten Hidden Layer erhält man daher mit dieser Funktion eine Eins oder eine Null. Dabei gibt es in jeder Darstellung für eine Ausgabezahl eine einzige Eins, die die Ergebniszahl (0...9) markiert. Die markierte Zahl wird danach über die Gewichte, die den jeweiligen Ziffern entsprechen, von der One-Hot-Repräsentation wieder in ein Bild umgewandelt. Die Gewichte für die Ausgabewerte sind in Abbildung 4 als weiße Zahl dargestellt.

Beispiel Führt man die Berechnung für die letzte Zahl von Abbildung 4 durch (also Digit 1), so erhält man in der ersten Hidden Layer die One-Hot-Repräsentation der Eins sowie die der Sechs. Dabei ist für die erste Zahl die zweite Stelle mit einer Eins markiert und für die zweite Zahl die siebte Stelle. Interessant wird die Berechnung der zweiten Hidden Layer. Hier erhält man durch einsetzen der Zahlen in die Gleichung 1 die Formel: $v_i^1 = \mathbb{1}_{\sum_{j=1}^{1}(1+6)*10^j \geq i*10^1}$. Setzt man dort die einzelnen Indikatorstellen ein, so erhält man damit die erste Null an der Stelle v_8 (wie in Abbildung 5 dargestellt). Wendet man die Gleichung 2 auf

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
V_n	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Abb. 5. Ergebnis Indikator für Digit 1

das Beispiel an, indem man das Ergebnis aus Abbildung 5 in diese übernimmt, so erhält man nur für o_7 eine Eins. Diese One-Hot-Repräsentation, die damit in der dritten Hidden Layer entsteht, kann nun mithilfe der Gewichte zur Sieben im Bild umgewandelt werden.

3.3 Ergebnisse

Durch die Experimente kam heraus, dass die Berechnung von Addition und Subtraktion mit Dezimalzahlen sehr gut mit dieser Vorgehensweise trainiert werden kann. Selbst als ein Gauß'scher Fehler auf die Eingabezahlen gelegt wurde, konnte das neuronale Netz die Zahlen noch miteinander verrechnen. Dabei erreichte es mit sehr hoher Genauigkeit ein richtiges Ergebnis ohne Rauschen im Bild. War sich das Netzwerk hier nicht sicher welche Zahl vorlag, so zeigte sich die Ungewissheit auch im Ausgabebild und es wurde eine Mischung der Ergebnisse aus den möglichen Zahlen dargestellt. Dies war für die Auswertung mit der OCR Software problematisch. Die Ausführung dieser Aufgabe durch einen Menschen verringerte hier die Fehlerrate um ca. 65%. Anders verhielt es sich für die Multi-

Operation	Pictures	
	No. Layers	% Error
Add	3	1.9%
Subtract	3	3.2%
Multiply	5	71.5%
Roman Addition	5	74.3 %

Abb. 6. Ergebnis Indikator für Digit 1
(Hoshen & Peleg, 2016)

plikation von Dezimalzahlen, sowie die Addition römischer Zahlen. Dort führten die Wissenschaftler noch zwei weitere Hidden Layer ein. Dennoch blieb die Fehlerrate mit jeweils über 70% sehr hoch (vgl. Abbildung 6) und ist damit über 20 Mal so hoch wie die der Addition und Subtraktion. Bei der Multiplikation ist die hohe Fehlerrate auf die anspruchsvolle Aufgabe dieser arithmetischen Operation zurückzuführen. Das Netzwerk war sich bei dem Ergebnis oft nicht sicher und zeigte an einer Stelle mehrere Ergebnisse auf. Dabei wurden am Rand richtige Ergebnisse, in der Mitte jedoch zwei Zahlen an einer Stelle, abgebildet. Selbst wenn die Faktoren, sowie das Ergebnis bereits als One-Hot-Vektor vorlagen, so konnte das Netzwerk nach dem Erlernen der Berechnung keine geringe Fehlerrate erreichen. Betrachtet man die Komplexität der Multiplikation, so zeigt sich, dass diese wesentlich höher ist, als die der Addition. Während Addition eine lineare Komplexität O(n) aufweist, erhält man bei der Multiplikation eine quadratische Komplexität O(n^2), n entspricht dabei der Anzahl der Ziffern. Für die Multiplikation ist daher ein tieferes Netzwerk mit mehr Knoten nötig. Liegt die Binärdarstellung bereits vor, so kann die Multiplikation selbst durch zwei Hidden Layer gelöst werden (Franco & Cannas, 1998).
Für römische Zahlen konnte das Netzwerk die Aufgabe der Addition nicht visuell Ende-zu-Ende lernen. Hatte es die Möglichkeit die Repräsentation der römischen Zahl separat von der Addition zu lernen, erzielte es sehr gute Ergebnisse.

4 Auswertung und Kritik

4.1 Zusammenfassung

In der Arbeit *Visual Learning of Arithmetic Operations* werden neuronale Netze verwendet, deren Gewichte selbst durch das Training erlernt werden. Durch diese Gewichte, sowie vorgegebene Aktivierungsfunktionen kann das neuronale Netz die Zahlen der Bilder kategorisieren und anschließend zu einem Ergebnisbild transformieren.

4.2 Diskussion

Die Wissenschaftler hatten in ihren Versuchen strikte Bedingungen. Die Zahlen waren in Standardschrift immer in der gleichen Bildposition angeordnet. Hier wäre es interessant, ob das Netzwerk auch bei abweichenden Bedingungen gleich gute Ergebnisse erzielt hätte. Ansatzpunkte gibt dazu bereits die Arbeit *Multiple Object Recognition with Visual Attention* (Ba, Mnih & Kavukcuoglu, 2015). Dieses beruht auch auf dem gleichen Netzwerk. Jedoch wird dort keine Operation erlernt, sondern nur die Erkennung der Zahl trainiert. Die Variation der Schrift bei dem Erlernen arithmetischer Operationen wird in *Computing Arithmetic Operations on Sequences of Handwritten Digits* (Pérez et al., 2017) durch die Nutzung von handgeschriebenen Zahlen behandelt. Dennoch wird hier nicht Ende-zu-Ende von Ursprungs- zu Ergebnisbild gelernt, sondern von Eingabebildern zu einer Ergebniszahl.
Generell zeigt die Arbeit von Hoshen und Peleg, dass das visuelle Lernen Arithmetischer Operationen Ende-zu-Ende möglich ist. Handelt es sich um eine komplexere Repräsentation, so ist diese als Subtask erlernbar, jedoch nicht in einem Ende-zu-Ende Lernprozess. Bei zu komplexen Operationen wiederum stellte sich das Netzwerk als ungeeignet heraus. Hier ist die Frage, ob ein Recurrent Neural Network, wie beispielsweise das Long Short Term Memory (LSTM), diese Aufgabe bewältigen könnte. LSTMs sind gut geeignet für sequenzielle Vorgänge. Multiplikation lässt sich als eine mehrfach angewandte Addition darstellen. Daher könnte man dies als sequenziellen Vorgang ansehen. Diese Umformung müsste jedoch durch das Netzwerk erlernt werden. Es bleibt zu erforschen, ob hier ein LSTM tatsächlich ein besseres Ergebnis erzielt. Die Menschen lernen das Vereinfachen der Multiplikation, indem ihnen die Umformung in der Schule beigebracht wird. Daher ist es eine interessante Frage, ob dies durch Netzwerke auch ohne vorgegebenes Wissen, sondern allein mit Trainingssets erlernbar ist.
Die Addition römischer Zahlen konnte in dieser Arbeit ebenfalls nicht in einem Ende-zu-Ende Prozess erlernt werden. Dabei könnte die Ursache in der langen Ziffernsequenz liegen. Das Netzwerk bekommt eine deutlich längere Sequenz, die nicht gleichstellbar ist, mit der deutlich kürzeren Dezimalzahl, da es die eigentliche Bedeutung der Darstellung nicht kennt. Somit wäre hier interessant gewesen zu sehen, wie sich die Ergebnisse verhalten, wenn man die gleiche Menge an Ziffern testet, statt auf der gleichen Zahlenspanne zu verharren.

In der Erklärung der Berechnung durch das Netzwerk fällt auf, dass nicht beschrieben wird, wie das Netzwerk die Berechnung durchführt. Die Bestimmung der Zahlen in der ersten Hidden Layer ist noch plausibel für das Netzwerk durchführbar, da mit Gewichten gearbeitet und erklärt wird, wie durch diese eine Markierung der vorgefundenen Übereinstimmung stattfindet. In der Berechnung des nächsten Layers wird jedoch nicht weiter mit dieser erreichten One-Hot-Darstellung gearbeitet, sondern wieder auf die Darstellung als Dezimalzahl zugegriffen, also einer Darstellung, die das Netzwerk hier nicht kennt. Außerdem wird in der Arbeit davon berichtet, dass das Netzwerk zwei Zahlen im Ergebnis abbildet, wenn es sich zwischen zwei Zahlen in der ursprünglichen Eingabezahl unsicher ist. Dieser Fall wird in dieser Erklärung nicht berücksichtigt. Die zwei Werte der Ausgangszahl könnten durch eine weitere Eins in der Repräsentation der ersten Hidden Layer dargestellt werden, wenngleich dies dann keine One-Hot Repräsentation mehr wäre. In der weiteren Berechnung würde sie dann nicht mehr berücksichtigt werden, außer man hätte zwei Berechnungen und damit auch zwei Indikatoren der Ziffernstelle. Dies zeigt, dass die Erklärung viele Aspekte außen vor lässt.

Literatur

Ba, J., Mnih, V. & Kavukcuoglu, K. (2015). Multiple object recognition with visual attention. *CoRR*, *abs/1412.7755*.

Brownlee, J. (2017). *A gentle introduction to mini-batch gradient descent and how to configure batch size.* Zugriff am 04.01.2018 auf `https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/`

Fortuner, B. (2017). *Activation functions.* Zugriff am 07.01.2018 auf `http://ml-cheatsheet.readthedocs.io/en/latest/_images/sigmoid.png`

Franco, L. & Cannas, S. A. (1998). Solving arithmetic problems using feedforward neural networks. In *Neurocomputing* (S. 61-79).

Holczer, B. (2017). *Deep learning fundamentals – part 2.* Zugriff am 07.01.2018 auf `http://www.globalsoftwaresupport.com/wp-content/uploads/2017/12/fddffdfdgghhhh.png`

Hoshen, Y. & Peleg, S. (2016). Visual learning of arithmetic operations. In *Proceedings of the thirtieth aaai conference on artificial intelligence* (S. 3733–3739). AAAI Press.

McGonagle, J. (2018). *Feedforward neural networks.* Zugriff am 03.01.2018 auf `https://brilliant.org/wiki/feedforward-neural-networks/`

Pérez, A., Quevedo, A. & Caicedo, J. C. (2017). Computing arithmetic operations on sequences of handwritten digits. In *Progress in pattern recognition, image analysis, computer vision, and applications* (S. 393–400). Springer International Publishing.

Vondrick, C., Pirsiavash, H. & Torralba, A. (2015). Anticipating visual representations from unlabeled video. *CoRR*, *abs/1504.08023*.

Zaremba, W., Kurach, K. & Fergus, R. (2014). Learning to discover efficient mathematical identities. In *Advances in neural information processing systems 27* (S. 1278–1286). Curran Associates, Inc.

Zaremba, W. & Sutskever, I. (2014). Learning to execute. *CoRR*, *abs/1410.4615*.