FOM Hochschule für Oekonomie & Management

Hochschulzentrum Hamburg

Bachelor Thesis

im Studiengang Informatik

zur Erlangung des Grades eines

Bachelor of Science (B.Sc.)

über das Thema

Implementierung des Equilibrium Propagation Algorithmus auf analogen Computern für das Trainieren neuronaler Netze

von

Merlin Moelter

Betreuer :	Dipl.-Phys.Ing. Stefan Scharr
Matrikelnummer :	575141
Abgabedatum :	12. Dezember 2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Symbolverzeichnis

Glossar

Backpropagation Lernalgorithmus für MLP basierend auf dem Gradienten-Verfahren. VII, 4

Bias-Neuron Zusätzliches Neuron, welches Konstant den Wert 1 ausgibt. Durch gewichtete Verbindungen kann jedem Neuron der nachfolgenden Ebene ein Bias-Wert zugewiesen werden. 3

CNN Convolutional Neural Network; Neuronales Netz mit Convolutional-Ebenen, welche nur mit jeweils einem Ausschnitt der vorherigen Ebene verbunden sind. 3

Dense Layer Ebene eines neuronalen Netzes, in der jedes Neuron mit jedem Neuron der vorherigen Ebene verbunden ist. 3

DNN Deep Neuronal Network; Neuronales Netz mit vielen versteckten Ebenen. 3

FNN Feedforward Neural Network; Neuronales Netz, in dem Signal nur in eine Richtung geleitet werden. 3

Forward Pass Erster Schritt im Backpropagation, in dem die Trainingsdaten das Modell durchlaufen. 4

Fully Connected Layer Siehe "‘Dense Layer"’. 3

Gradienten-Verfahren Gradient Descent; Ein generischer Optimierungs-Algorithmus. VII, 4

Hebbian learning Eine Lernregel die beschreibt, dass sich Gewichtungen zwischen Neuronen verstärken, die gleichzeitige Aktivierungen aufweisen. 4

MLP Multilayer-Perceptron; Eine Zusammensetzung an Perceptrons mit einer Eingabe-Ebene, mindestens einer versteckten Ebene und einer Ausgabe-Ebene. VII, 3, 4

Perceptron Sammlung an TLUs auf einer einzelnen Ebene. 2, 3, 4

Reverse Pass Zweiter Schritt im Backpropagation, in dem das Gradienten-Verfahren auf die Gewichtungen zwischen Neuronen angewendet wird. 4

RNN Recurrent Neural Network; Neuronales Netz mit rückwärtigen Verbindungen, Hauptbestandteil ist hier die Speicherzelle. 3

TLU Threshold Logic Unit; Berechnet die gewichtete Summe seiner Eingabewerte und gibt abhängig von der Überschreitung eines Schwellenwerts entweder 0 oder 1 aus. 2, 3

1 Einleitung

2 Grundlagen

2.1 Neuronale Netze

2.1.1 Aufbau und Arten neuronaler Netze

> „Birds inspired us to fly, burdock plants inspired Velcro, and nature has in- spired countless more inventions. It seems only logical then, to look at the brain's architecture for inspiration on how to build an intelligent machine“ (*Géron*, *A.*, 2019, S. 279)

Neuronale Netze wurden erstmals 1943 von den Wissenschaftlern Warren S. McCulloch und Walter Pitts in ihrer gemeinsamen Arbeit „A logical calculus of the ideas immanent in nervous activity“ *McCulloch*, *W. S.*, *Pitts*, *W.*, 1943 eingeführt. Sie stellten ein vereinfachtes Modell eines künstlichen Neurons vor, das lediglich aus binären Eingaben und einer binären Ausgabe besteht und seine Ausgabe aktiviert, sobald sich eine bestimmte Anzahl an Eingabewerten aktiviert. McCulloch und Pitts zeigten, dass dieser einfache Baustein ausreicht, um jeden möglichen logischen Ausdruck als neuronales Netz darzustellen.

Neuronale Netze zeichnen sich mittlerweile durch ihre Vielseitigkeit, Leistungsfähigkeit und Skalierbarkeit aus und haben damit maßgeblich zur Gründung eines neuen Forschungsfeldes, dem „Deep Learning“, beigetragen. *Géron*, *A.*, 2019 Die neu gewonnen Aufmerksamkeit im Zusammenhang mit Deep Learning brachte Innovationen wie das von OpenAI entwickelte Sprachmodell „GPT-4“ oder der Bildgenerierungs-KI „Midjourney“ hervor, wodurch bewiesen wurde, dass neuronale Netze zur Lösung komplexer Aufgaben geeignet sind und sogar teilweise ähnlich brauchbare Ergebnisse wie ein Mensch liefern können. *OpenAI* et al., 2024

Eines der einfachsten neuronalen Netze ist das von *Rosenblatt*, *F.*, 1958 vorgestellte Perceptron, dieses basiert auf dem Konzept der TLU. Die Eingabe- und Ausgabewerte einer TLU sind numerisch und den eingehenden Verbindungen ist jeweils eine Gewichtung zugewiesen. Die TLU berechnet nun die gewichtete Summe der Eingabewerte $z = w_1x_1 + w_2x_2... + w_nx_n = xtw$. Unter Anwendung einer Aktivierungsfunktion $hw(x) = step(z)$ kann nun berechnet werden, ob die TLU den Wert 0 oder 1 ausgibt. Eine Aktivierungsfunktion für diese Art der Neuronen ist i. d. R. eine Heaviside-Funktion oder eine Vorzeichen-Funktion. *Géron*, *A.*, 2019, vgl. S. 284 ff.

Aktivierungsfunktionen aus Buch einfügen

Abbildung aus Buch nachstellen

Eine alleinstehende TLU kann ausschließlich für lineare binäre Klassifikation genutzt werden, es berechnet eine gewichtete Summe anhand der Eingabewerte und gibt einen positiven oder negativen Ausgabewert, abhängig von der Überschreitung eines Schwellenwertes. Ein Perceptron stellt nun eine Sammlung dieser Einheiten auf einer einzelnen Ebene dar, wobei jede TLU mit jeder Eingabe verbunden ist. Dies wird als Dense Layer oder Fully Connected Layer bezeichnet. Die Eingabewerte des Perceptron werden durch Eingabe-Neuronen geschleust, wozu zusätzlich ein Bias-Neuron gezählt wird. Dieses Neuron gibt konstant den Wert 1 aus und dient dazu, jedem Neuron des Perceptron eine Bias-Gewichtung zuzuweisen. Die Ausgabe eines Perceptron berechnet sich durch $h_W, b(X) = \rho(XW + b)$ *Géron, A.*, 2019, vgl. S. 284 ff.

Das Perceptron kann nun in mehreren Ebenen genutzt werden, um ein MLP zu erzeugen. Dieses besteht aus einer Eingabe-Ebene, mindestens einer versteckten Ebene und einer Ausgabe-Ebene. Jede dieser Ebenen ist im MLP mit der jeweils nächsten Ebene vollständig verbunden. Das MLP kann durch die vorwärts gerichteten Verbindungen auch als FNN bezeichnet werden, eines mit vielen versteckten Ebenen wird DNN genannt. ebd., vgl. S. 284 ff.

Das bisher beschriebene MLP kann zur Klassifikation genutzt werden. Um dieses auch auf Regressionen anwenden zu können, muss die Aktivierungsfunktion entweder entfernt oder durch z.B. ReLU ausgetauscht werden, damit die Ausgabeneuronen willkürliche Werte annehmen können. Die Anzahl der Ausgabe-Neuronen muss in dem Fall angepasst werden, sodass jeder geforderte Ausgabewert durch ein Neuron abgebildet ist. ebd., vgl. S. 292 ff.

Eine weitere Art des neuronalen Netz ist das CNN, dessen Hauptbestandteil die Convolutional-Ebenen sind. Diese Ebenen sind nur jeweils mit einem Ausschnitt der vorherigen Ebene verbunden, wodurch das daraus entstehende neuronale Netz abstrakte Eigenschaften der Eingabe-Neuronen erlernen kann. Da die einzelnen Ebenen hier nicht, wie beim MLP, vollständig verbunden sind, erlaubt ein CNN eine große Anzahl an Neuronen pro Ebene, ohne den Rechenaufwand für Training und Inferenz exponentiell zu steigern. Durch diese Eigenschaften eignet sich das CNN besonders für die Bildbearbeitung, da hier als Eingabe die Pixel genutzt und darin Eigenschaften des Bildes gefunden werden können. ebd., vgl. S. 447 f.

Das RNN ähnelt vom Aufbau dem FNN unterscheidet sich aber durch rückwärts gerichtete Verbindungen. Die Ausgabe eines Neuron wird damit Teil seiner Eingabe, womit es sich Informationen "merken"kann. Ein RNN kann als Sequenz-zu-Sequenz Netzwerk genutzt werden, es erhält also eine Sequenz an Eingabe-Werten und produziert eine Sequenz an Ausgabe-Werten. Genauso kann jeder Ausgabewert bis auf den letzten ignoriert werden, was als Sequenz-zu-Vektor Netzwerk bezeichnet wird. Andersherum kann ein RNN

auch einen Vektor als Eingabe erhalten und eine Sequenz ausgeben, wodurch z.B. eine Beschreibung für ein Bild generiert werden könnte. Letztlich ist auch ein sog. Encoder-Decoder Modell möglich, wobei ein Sequenz-zu-Vektor Netzwerk zuerst eine Repräsentation der Eingabesequenz erzeugt, und ein Vektor-zu-Sequenz Netzwerk daraufhin eine Ausgabe aus dieser Repräsentation erzeugt. Ein Anwendungsfall dafür sind Sprachanwendungen wie ein Übersetzer, da jedes Wort eines Textes im Vornherein bekannt sein muss, um eine korrekte Ausgabe zu erzeugen. *Géron*, *A.*, 2019, vgl. S. 497 ff.

2.1.2 Training neuronaler Netze mit Backpropagation und Gradient Descent

Das Gradienten-Verfahren ist ein Optimierungs-Algorithmus, der mithilfe einer Fehlerfunktion die Parameter eines Modells so anpasst, dass die Fehlerfunktion minimiert wird. Um das zu erreichen, muss das Verfahren erst die Steigung der Fehlerfunktion berechnen können, um dann iterativ die Parameter anzupassen. ebd., vgl. S. 118

> „Neurons wire together if they fire together'"*Lowel*, *S.*, *Singer*, *W.*, 1992

Dieses Zitat prägt das sog. Hebbian learning, eine Regel die beschreibt wie sich die Gewichtungen zwischen Neuronen relativ zu deren Aktivierungen verändern. Ein Perceptron wird mit einer Abwandlung dieser Regel trainiert, die außerdem den Fehler der Ausgabe mit einbezieht und diejenigen Gewichtungen verstärkt, die zu einer Verringerung des Fehlers führen. *Géron*, *A.*, 2019, vgl. S. 289 ff. Die Lernregel lautet damit:

$$w_{i,j}{}^{(nextstep)} = w_{i,j} + \eta(y_j - \hat{y}_j)$$

Ein Verfahren zum Trainieren eines MLP stellt das von *Rumelhart*, *D. E.*, *Hinton*, *G. E.*, *Williams*, *R. J.*, 1986 vorgestellte Backpropagation dar. Dieses basiert auf dem Gradienten-Verfahren, mit der Eigenschaft die Gradienten der Gewichtungen in allen Ebenen des MLP mit Bezug auf jeden einzelnen Parameter effizient berechnen zu können. Die Trainingsdaten werden in mehreren Epochen im folgenden Ablauf durchlaufen:

Zuerst wird für jede Instanz der Trainingsdaten das MLP im Forward Pass durchlaufen. Die Ausgabe jedes Neurons der versteckten Ebenen wird dabei zwischengespeichert. Nun wird die Ausgabe des MLP anhand der Fehlerfunktion bestimmt. Die Gradienten aller Gewichtungen zwischen Neuronen der versteckten Ebenen werden im Reverse Pass bestimmt. Dazu wird der Beitrag jedes Ausgabe-Neuronen zum Fehler berechnet und gleiches rekursiv für die Neuronen der versteckten Ebenen wiederholt. Mit den berechneten Werten kann abschließend das Gradienten-Verfahren angewendet werden. *Géron*, *A.*, 2019, S. 286

2.2 Analoge Computer

2.2.1 Geschichte zu analogen Computern

Die Geschichte des analogen Computers reicht bis in die Antike zurück. So wurde im Jahr 1900 der Antikythera-Mechanismus in einem Schiffswrack entdeckt, der aus etwa 100 v. Chr. stammt. Er gilt als eines der komplexesten mechanischen und mathematischen Geräte der Antike und konnte die Bewegungen von Himmelskörpern modellieren sowie Sonnen- und Mondfinsternisse vorhersagen. *Ulmann*, *B.*, 2022, S. 9 f.

Im Verlauf der technischen Entwicklung wurden Gleitkomma-Rechner konstruiert, mechanische Geräte, die zur Lösung komplexer mathematischer Probleme, wie der Berechnung von Bombenflugbahnen und Feuerleitsystemen, eingesetzt wurden. Diese fanden auch in friedlichen Anwendungen wie der Gezeiten-Berechnung Verwendung. ebd., S. 9

Mit dem Fortschritt der Technik entstanden die ersten elektronischen analogen Computer, die von Helmut Hoelzer in Deutschland und George A. Philbrick in den USA entwickelt wurden. Diese Computer wurden primär für militärische Anwendungen wie Flugbahn- und Feuerleitsysteme genutzt. Ein Beispiel für einen frühen elektronische Analogrechner ist Hoelzers Mischgerät, das während des zweiten Weltkriegs in Deutschland entwickelt wurde und Elektrohnenröhren zur Durchführung von Berechnungen verwendete. ebd., S. 41 f.

Ein weiteres bedeutendes Gerät war der Caltech-Computer, ein elektronischer Analogrechner, der zwischen 1946 und 1947 entwickelt wurde und etwa 15 Tonnen wog. Dieser Computer wurde zur Lösung komplexer mathematischer und wissenschaftlicher Probleme, einschließlich Differenzialgleichungen, entwickelt. ebd., S. 69

George A. Philbrick spielte eine bedeutende Rolle in der Weiterentwicklung und Verbreitung elektronischer Analogrechner. Er führte kommerzielle Operationsverstärker ein und entwickelte in den 1950er Jahren modulare elektronische Analogrechner, was einen großen Einfluss auf die Standardisierung und Verbreitung dieser Technologie hatte. ebd., S. 136

Anhang

Literaturverzeichnis

Géron, *Aurélien* (2019): Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2. Aufl., o. O.: O'Reilly Media, 2019

Lowel, *Siegrid*, *Singer*, *Wolf* (1992): Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity, in: Science, 255 (1992), Nr. 5041, S. 211

McCulloch, *Warren S.*, *Pitts*, *Walter* (1943): A logical calculus of the ideas immanent in nervous activity, in: The Bulletin of Mathematical Biophysics, 5 (1943), Nr. 4, S. 115–133

OpenAI, *Achiam*, *Josh*, *Adler*, *Steven*, *Agarwal*, *Sandhini*, *Ahmad*, *Lama*, *Akkaya*, *Ilge*, *Aleman*, *Florencia Leoni*, *Almeida*, *Diogo*, *Altenschmidt*, *Janko*, *Altman*, *Sam*, *Anadkat*, *Shyamal*, *Avila*, *Red*, *Babuschkin*, *Igor*, *Balaji*, *Suchir*, *Balcom*, *Valerie*, *Baltescu*, *Paul*, *Bao*, *Haiming*, *Bavarian*, *Mohammad*, *Belgum*, *Jeff*, *Bello*, *Irwan*, *Berdine*, *Jake*, *Bernadett-Shapiro*, *Gabriel*, *Berner*, *Christopher*, *Bogdonoff*, *Lenny*, *Boiko*, *Oleg*, *Boyd*, *Madelaine*, *Brakman*, *Anna-Luisa*, *Brockman*, *Greg*, *Brooks*, *Tim*, *Brundage*, *Miles*, *Button*, *Kevin*, *Cai*, *Trevor*, *Campbell*, *Rosie*, *Cann*, *Andrew*, *Carey*, *Brittany*, *Carlson*, *Chelsea*, *Carmichael*, *Rory*, *Chan*, *Brooke*, *Chang*, *Che*, *Chantzis*, *Fotis*, *Chen*, *Derek*, *Chen*, *Sully*, *Chen*, *Ruby*, *Chen*, *Jason*, *Chen*, *Mark*, *Chess*, *Ben*, *Cho*, *Chester*, *Chu*, *Casey*, *Chung*, *Hyung Won*, *Cummings*, *Dave*, *Currier*, *Jeremiah*, *Dai*, *Yunxing*, *Decareaux*, *Cory*, *Degry*, *Thomas*, *Deutsch*, *Noah*, *Deville*, *Damien*, *Dhar*, *Arka*, *Dohan*, *David*, *Dowling*, *Steve*, *Dunning*, *Sheila*, *Ecoffet*, *Adrien*, *Eleti*, *Atty*, *Eloundou*, *Tyna*, *Farhi*, *David*, *Fedus*, *Liam*, *Felix*, *Niko*, *Fishman*, *Simón Posada*, *Forte*, *Juston*, *Fulford*, *Isabella*, *Gao*, *Leo*, *Georges*, *Elie*, *Gibson*, *Christian*, *Goel*, *Vik*, *Gogineni*, *Tarun*, *Goh*, *Gabriel*, *Gontijo-Lopes*, *Rapha*, *Gordon*, *Jonathan*, *Grafstein*, *Morgan*, *Gray*, *Scott*, *Greene*, *Ryan*, *Gross*, *Joshua*, *Gu*, *Shixiang Shane*, *Guo*, *Yufei*, *Hallacy*, *Chris*, *Han*, *Jesse*, *Harris*, *Jeff*, *He*, *Yuchen*, *Heaton*, *Mike*, *Heidecke*, *Johannes*, *Hesse*, *Chris*, *Hickey*, *Alan*, *Hickey*, *Wade*, *Hoeschele*, *Peter*, *Houghton*, *Brandon*, *Hsu*, *Kenny*, *Hu*, *Shengli*, *Hu*, *Xin*, *Huizinga*, *Joost*, *Jain*, *Shantanu*, *Jain*, *Shawn*, *Jang*, *Joanne*, *Jiang*, *Angela*, *Jiang*, *Roger*, *Jin*, *Haozhun*, *Jin*, *Denny*, *Jomoto*, *Shino*, *Jonn*, *Billie*, *Jun*, *Heewoo*, *Kaftan*, *Tomer*, *Kaiser*, *Łukasz*, *Kamali*, *Ali*, *Kanitscheider*, *Ingmar*, *Keskar*, *Nitish Shirish*, *Khan*, *Tabarak*, *Kilpatrick*, *Logan*, *Kim*, *Jong Wook*, *Kim*, *Christina*, *Kim*, *Yongjik*, *Kirchner*, *Jan Hendrik*, *Kiros*, *Jamie*, *Knight*, *Matt*, *Kokotajlo*, *Daniel*, *Kondraciuk*, *Łukasz*, *Kondrich*, *Andrew*, *Konstantinidis*, *Aris*, *Kosic*, *Kyle*, *Krueger*, *Gretchen*, *Kuo*, *Vishal*, *Lampe*, *Michael*, *Lan*, *Ikai*, *Lee*, *Teddy*, *Leike*, *Jan*, *Leung*, *Jade*, *Levy*, *Daniel*, *Li*, *Chak Ming*, *Lim*, *Rachel*, *Lin*, *Molly*, *Lin*, *Stephanie*, *Litwin*, *Mateusz*, *Lopez*, *Theresa*, *Lowe*, *Ryan*, *Lue*, *Patricia*, *Makanju*, *Anna*, *Malfacini*, *Kim*, *Manning*, *Sam*, *Markov*, *Todor*, *Markovski*, *Yaniv*, *Martin*, *Bianca*, *Mayer*, *Katie*, *Mayne*, *Andrew*, *McGrew*, *Bob*, *McKinney*, *Scott Mayer*, *McLeavey*, *Christine*, *McMillan*, *Paul*, *McNeil*, *Jake*, *Medina*, *David*, *Mehta*, *Aalok*, *Menick*, *Jacob*, *Metz*, *Luke*, *Mishchenko*, *Andrey*, *Mishkin*, *Pamela*, *Monaco*, *Vinnie*,

Morikawa, *Evan*, *Mossing*, *Daniel*, *Mu*, *Tong*, *Murati*, *Mira*, *Murk*, *Oleg*, *Mély*, *David*, *Nair*, *Ashvin*, *Nakano*, *Reiichiro*, *Nayak*, *Rajeev*, *Neelakantan*, *Arvind*, *Ngo*, *Richard*, *Noh*, *Hyeonwoo*, *Ouyang*, *Long*, *O'Keefe*, *Cullen*, *Pachocki*, *Jakub*, *Paino*, *Alex*, *Palermo*, *Joe*, *Pantuliano*, *Ashley*, *Parascandolo*, *Giambattista*, *Parish*, *Joel*, *Parparita*, *Emy*, *Passos*, *Alex*, *Pavlov*, *Mikhail*, *Peng*, *Andrew*, *Perelman*, *Adam*, *de Avila Belbute Peres*, *Filipe*, *Petrov*, *Michael*, *de Oliveira Pinto*, *Henrique Ponde*, *Michael*, *Pokorny*, *Pokrass*, *Michelle*, *Pong*, *Vitchyr H.*, *Powell*, *Tolly*, *Power*, *Alethea*, *Power*, *Boris*, *Proehl*, *Elizabeth*, *Puri*, *Raul*, *Radford*, *Alec*, *Rae*, *Jack*, *Ramesh*, *Aditya*, *Raymond*, *Cameron*, *Real*, *Francis*, *Rimbach*, *Kendra*, *Ross*, *Carl*, *Rotsted*, *Bob*, *Roussez*, *Henri*, *Ryder*, *Nick*, *Saltarelli*, *Mario*, *Sanders*, *Ted*, *Santurkar*, *Shibani*, *Sastry*, *Girish*, *Schmidt*, *Heather*, *Schnurr*, *David*, *Schulman*, *John*, *Selsam*, *Daniel*, *Sheppard*, *Kyla*, *Sherbakov*, *Toki*, *Shieh*, *Jessica*, *Shoker*, *Sarah*, *Shyam*, *Pranav*, *Sidor*, *Szymon*, *Sigler*, *Eric*, *Simens*, *Maddie*, *Sitkin*, *Jordan*, *Slama*, *Katarina*, *Sohl*, *Ian*, *Sokolowsky*, *Benjamin*, *Song*, *Yang*, *Staudacher*, *Natalie*, *Such*, *Felipe Petroski*, *Summers*, *Natalie*, *Sutskever*, *Ilya*, *Tang*, *Jie*, *Tezak*, *Nikolas*, *Thompson*, *Madeleine B.*, *Tillet*, *Phil*, *Tootoonchian*, *Amin*, *Tseng*, *Elizabeth*, *Tuggle*, *Preston*, *Turley*, *Nick*, *Tworek*, *Jerry*, *Uribe*, *Juan Felipe Cerón*, *Vallone*, *Andrea*, *Vijayvergiya*, *Arun*, *Voss*, *Chelsea*, *Wainwright*, *Carroll*, *Wang*, *Justin Jay*, *Wang*, *Alvin*, *Wang*, *Ben*, *Ward*, *Jonathan*, *Wei*, *Jason*, *Weinmann*, *CJ*, *Welihinda*, *Akila*, *Welinder*, *Peter*, *Weng*, *Jiayi*, *Weng*, *Lilian*, *Wiethoff*, *Matt*, *Willner*, *Dave*, *Winter*, *Clemens*, *Wolrich*, *Samuel*, *Wong*, *Hannah*, *Workman*, *Lauren*, *Wu*, *Sherwin*, *Wu*, *Jeff*, *Wu*, *Michael*, *Xiao*, *Kai*, *Xu*, *Tao*, *Yoo*, *Sarah*, *Yu*, *Kevin*, *Yuan*, *Qiming*, *Zaremba*, *Wojciech*, *Zellers*, *Rowan*, *Zhang*, *Chong*, *Zhang*, *Marvin*, *Zhao*, *Shengjia*, *Zheng*, *Tianhao*, *Zhuang*, *Juntang*, *Zhuk*, *William*, *Zoph*, *Barret* (2024): GPT-4 Technical Report, o. O., 2024, arXiv: 2303.08774 [cs.CL], URL: https://arxiv.org/abs/2303.08774

Rosenblatt, *F.* (1958): The perceptron: A probabilistic model for information storage and organization in the brain. In: Psychological Review, 65 (1958), Nr. 6, S. 386–408

Rumelhart, *D. E.*, *Hinton*, *G. E.*, *Williams*, *R. J.* (1986): Learning internal representations by error propagation, in: *Rumelhart*, *D. E.*, *McClelland*, *J. L.* (Hrsg.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Bd. 1, o. O.: MIT Press, 1986, S. 318–362

Ulmann, *Bernd* (2022): Analog Computing, o. O.: DeGruyter, 2022

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit einverstanden/nicht einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Hamburg, 12.12.2024
(Ort, Datum)

(Eigenhändige Unterschrift)