Technische Universität Dortmund

Fakultät Informatik
Design Automation for Embedded Systems Group

TODO Titel

Fachprojekt

Jack Diep, Florian Köhler, Yannick Naumann

Betreuung:
M.Sc. Mikail Yayla

27. August 2021

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Neuronale Netzwerke bilden eine Unterkategorie des Machine-Learning und erlauben Auswertungen von Eingaben auf Basis von zuvor angelernten, empirischen Ergebnissen. Ein neuronales Netzwerk bildet ein System aus Neuronen ab, welche Schichtweise verbunden sind einen unidirektionalen Datenfluss erzeugen. Dieses System besteht üblicherweise aus einer Eingabeschicht, einer Ausgabeschicht und dazwischen beliebig viele *versteckte* Schichten, welche die eigentliche Arbeit des Netzwerks verrichten. Die Anzahl der Neuronen in den Eingabe- und Ausgabeschichten ist intuitiv wählbar. Die Größe der Eingabeschicht wird häufig durch die Anzahl der möglichen Eingaben bestimmt, die Größe der Ausgabeschicht durch die Anzahl der möglichen Ergebnisse. Die Größe und Anzahl der dazwischen liegenden Schichten hingegen muss je nach Anforderung und Gegebenheiten individuell ermittelt werden. Je größer das Netzwerk desto höher sind die Anforderungen an die benötigte Hardware um dieses zu betreiben. Bei schwächerer Hardware oder Einschränkungen bezüglich der Energieversorgung können kleinere Netzwerke eingesetzt werden, wenn auch häufig mit geringerer Genauigkeit verglichen mit einem größeren Netzwerk.

1.1 Motivation

Die Größe eines Netzwerks kann beim Entwurf dessen direkt beeinflusst werden. In Anwendungsgebieten, bei denen der Fokus auf geringen Hardwareanforderungen liegt, stoßen schnell das Problem der schwindenden Genauigkeit. Auf IoT-Geräten oder mobilen Platformen finden klassische neuronale Netzwerke daher nur eingeschränkt Nutzen.

2016 veröffentlichten M. Courbariaux und Y. Bengio [1] eine wissenschaftliche Arbeit und stellen dort das *binarisierte neuronale Netzwerk* (kurz BNN) vor. Dieses könne im Vergleich zu einem klassischen neuronalen Netzwerk eine theoretische Geschwindigkeitssteigerung auf das 32-fache erreichen. Die Genauigkeit des BNN liege jedoch nur knapp unter derer klassischer Netzwerke. Das BNN ermöglicht dank dieser Eigenschaften den Einsatz von neuronalen Netzen auf vergleichsweise schwacher Hardware und verspricht zugleich nur geringe Genauigkeitseinbüße.

In dieser Ausarbeitung wird die allgemeine Funktionsweise von neuronalen Netzwerken erläutert und anschließend der Entwurf eines binarisiertes Netzwerk mit Fokus auf Handschriftenerkennung auf Basis des MNIST Datensatzes dokumentiert. Dabei werden grundlegende Überlegungen, das Vorgehen, Herausforderungen sowie dazu erarbeitete Lösungen vorgestellt.

Kapitel 2

Neuronale Netzwerke

Unter einem neuronalen Netzwerk versteht man ein System aus Neuronen. Diese sind schichtweise organisiert wobei jedes Neuron einer Schicht jeweils zu allen Neuronen der direkt anliegenden Schichten verbunden ist. Abbildung 2.1 zeigt beispielhaft ein neuronales Netzwerk aus 19 Neuronen mit insgesamt 5 Schichten.

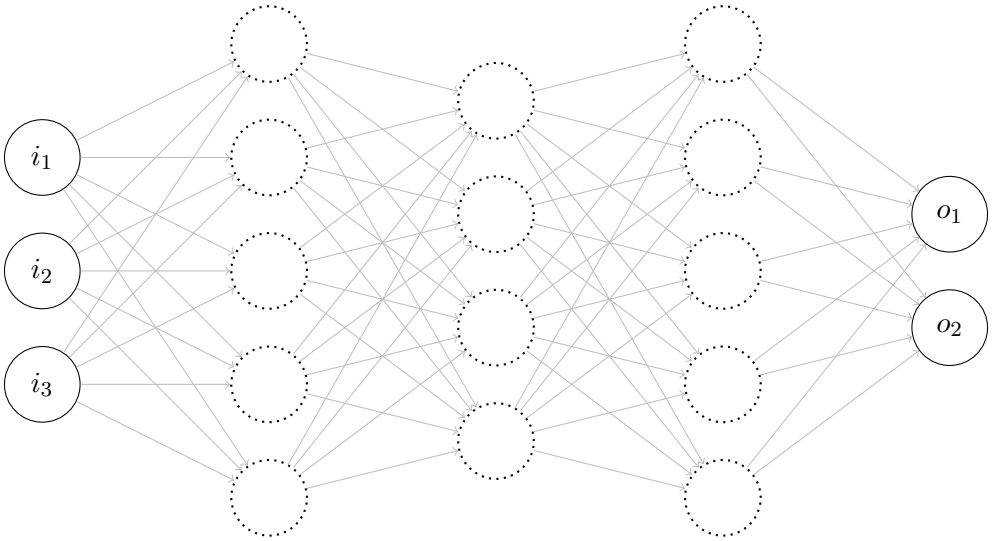

Abbildung 2.1: Neuronales Netzwerk

2.1 Neuronen

Unter einem Neuron versteht sich bei neuronalen Netzen lediglich ein Knoten, in dem üblicherweise ein 32-bit großer Wert hinterlegt ist. Häufig kommen hier Gleitkommazahlen zwischen -1.0 und 1.0 zum Einsatz, da sich dieser Wertebereich besonders gut zum Rechnen eignet und Eigenschaften bezüglich der Multiplikation besitzt, die eine Wertexplosion verhindern. Die Werte aller Neuronen, mit Ausnahme derer in der Eingabeschicht, setzen sich jeweils aus den Werten aller Neuronen der direkt davor liegenden Schicht zusammen. Das genaue Vorgehen bei der Wertermittlung hängt jeweils vom Netzwerk und den dort verwendeten Aktivierungsfunktionen ab.

2.2 Schichten und Kanten

In einem neuronalen Netzwerk ist jedes Neuron einer Schicht mit allen Neuronen der jeweiligen davor liegenden und danach liegenden Schicht über Kanten verbunden. Neuronale Netzwerke sind unidirektionale Graphen, demnach fließen Informationen über Schichten (und folglich Kanten) nur in eine Richtung.

Allen Kanten wird initial eine Gewichtung zugewiesen, welche erneut netzwerkabgängig generiert werden oder durch zuvor angelernte Daten bestimmt werden. Beim Trainieren des Netzwerks werden diese bei jeder Lerniteration (auch *Epoch* genannt) justiert, während sie beim Betrieb für gewöhnlich keine Änderungen mehr erfahren. Die Genauigkeit eines Netzwerks wird überwiegend durch diese Gewichte bestimmt, daher ist das Ziel beim Trainieren eines Netzes die Optimierung jener.

Kantengewichte wirken sich maßgeblich auf die Wertberechnung von Neuronen aus. Diese wird in zwei Schritten ausgeführt. Im ersten Schritt wird aus den Kantengewichten (e_i) aller eingehenden Kanten und den Werten der darüber verbundenen Neuronen (v_i) der Wert nach Formel (2.1) gebildet.

$$v = \sum_{i=1}^{n} (e_i * v_i) \tag{2.1}$$

2.3 Aktivierungsfunktionen

Im zweiten Schritt wird der zuvor errechnete Wert durch eine weitere Funktion modifiziert. Sogenannte *Aktivierungsfunktionen* können beliebig gewählt werden, müssen jedoch offensichtlich alle möglichen Eingaben auf einen Wert abbilden können. Die verwendete Aktivierungsfunktion kann je nach Schicht variieren. Einmal gewählt, ist diese jedoch für die jeweilige Schicht im Netzwerk fest.

Im Folgenden sind sind häufig verwendete Aktivierungsfunktionen und ihre Formeln abgebildet.

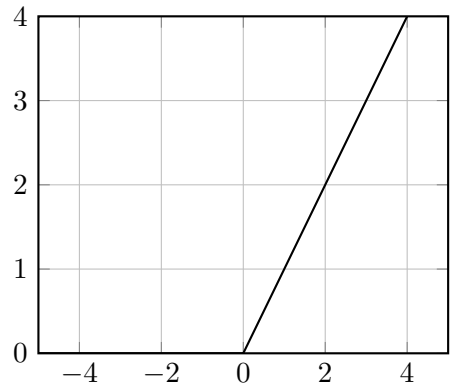

Abbildung 2.2: ReLU

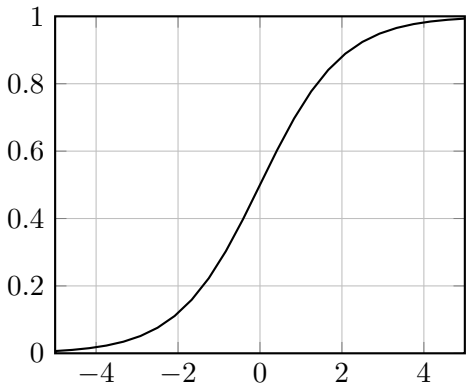

Abbildung 2.3: Sigmoid

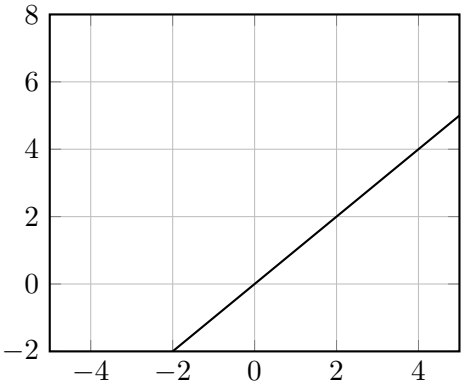

Abbildung 2.4: Softmax

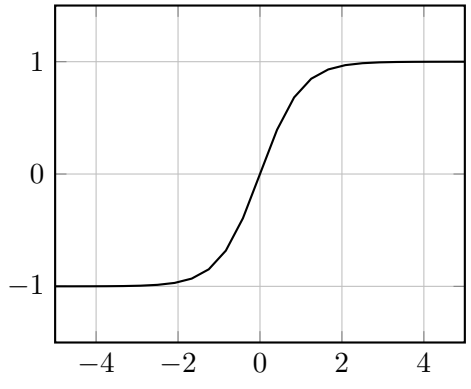

Abbildung 2.5: tanh

Die beiden erläuterten Berechnungsschritte werden schichtweise in Richtung des Datenflusses des Netzwerks für alle Neuronen durchgeführt.

2.4 Training

2.5 Backpropagation (?)

Kapitel 3

Das Netzwerk

3.1 Aktivierungsfunktion

3.2 Binärer Linear-Layer

3.3 Verluste durch binäre Linear-Layer

Durch die binarisierung der *Linear-Layer* ist zu vermuten, dass diese, im Vergleich zu normalen *Linear-Layer*, etwas schlechter performen. Dies ist der Fall, da die Anzahl der möglichen Kantengewichte stark, auf Null und Eins, eingeschränkt ist.

Trainiert wurde hier das gleiche Netzwerk, ein mal mit binären *Linear-Layern*, das andere mal mit normalen *Linear-Layer*. Jedes Netzwerk wurde für 50 Epochen trainiert, bevor die Genauigkeit ausgewertet wurde. Um sicher zu gehen, ob die Genauigkeit gegen diesen Wert konvergiert, wurde jede Messung fünf mal wiederholt.

Kapitel 4

Binarisierung

Kapitel 5

Export

Nachdem das Netzwerk nun trainiert wurde, müssen die Ergebnisse, die Kantengewichte und Schwellwerte der Neuronen, nun exportiert werden. Im Folgenden sollen diese dann in den BNN-Beschleuniger Baustein importiert und verwendet werden. Da der Import in VHDL stattfindet, eignen sich hier simple Formate, sprich eine einfache Textdatei. Diese kann dann, Zeichen nach Zeichen, von dem Import-Buffer eingelesen und in einer Matrix gespeichert werden.

5.1 Export der Kantengewichte

Da es sich bei unserem Netzwerk um ein *FullyConnected Neural Network* handelt, ist insbesondere jedes Neuron mit jedem Neuron der Folgenden Schicht verbunden. Bei unserem BNN ergibt sich also folgende Kantenanzahl

$$784 \cdot 500 + 500 \cdot 1024 + 1024 \cdot 1024 = 1.952.576$$

Diese Gewichte müssen alle, mit möglichst wenig Mehrkosten, in die Datei geschrieben werden. Da es sich bei den Gewichten lediglich um binäre Werte, Einsen und Nullen, handelt, ist kein Trennzeichen zwischen den Gewichten notwendig. Die Gewichte sind außerdem, trivialer Weise, präfixfrei und können fortlaufend in die Datei geschrieben werden.
Um die Gewichte zu extrahieren, wird zuerst über jeden *Layer* iteriert. In jedem *Layer* wird nun jedes Neuron abgelaufen. Jedes dieser Neuronen hat nun jeweils eine Kante

zu jedem Neuron in der nachfolgenden Schicht. Hier wird ebenfalls über alle Kanten iteriert und das jeweilige Gewicht wird hinten an eine Variabel an gehangen. Ist nun ein *Layer* fertig, wird der Inhalt der Variable, welche als Zwischenspeicher dient, in die Datei *weights.txt* geschrieben. So wird für jede Schicht ein IO-Zugriff gemacht.

5.2 Export der Schwellwerte

Literaturverzeichnis

[1] Beutelspacher, Albrecht: Das ist o.B.d.A. trivial!. Tipps und Tricks zur Formulierung mathematischer Gedanken. Vieweg Verlag, Braunschweig und Wiesbaden, 2004.

[1]