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 **ii**

1 Einleitung **1**
1.1 Motivation 1

2 Neuronale Netzwerke **3**
2.1 Neuronen 3
2.2 Schichten 3
2.3 Aktivierungen 3

3 Das Netzwerk **4**
3.1 Aktivierungsfunktion 4
3.2 Binärer Linear-Layer 4
3.3 Verluste durch binäre Linear-Layer 4

4 Binarisierung **5**

5 Export **6**
5.1 Export der Kantengewichte 6
5.2 Export der Schwellwerte 7

Literaturverzeichnis **8**

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.

In 2016 wurde von Courbariaux (TODO: ...) eine Arbeit veröffentlich, welche das Binarisierte neuronale Netzwerk (kurz *BNN*) vorstellt, einem neuronalen Netzwerk welches bis zu 32-mal schneller arbeiten kann als ein klassisches neuronales Netzwerk. Die Genauigkeit des BNN könne zudem mit der Genauigkeit von jenen klassischen Netzwerken mithalten, wenn auch mit einer leichten Verschlechterung. Das BNN ermöglicht dank seiner 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 Entwurfs eines binarisiertes Netzwerk dokumentiert. Dabei werden grundlegende Überlegungen, das Vorgehen, auftretende Probleme sowie dazu ermittelte Lösungen vorgestellt.

Kapitel 2

Neuronale Netzwerke

TODO: Jack

2.1 Neuronen

TODO: Jack

2.2 Schichten

TODO: Jack

2.3 Aktivierungen

TODO: Jack

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]