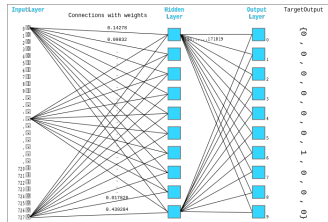
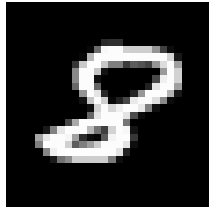


Handschrifterkennung mit CUDA und C++

Christopher Haug, Dominik Walter

University of Augsburg
Systems and Networking

26.07.2017



8

- Erkennung von handgeschriebenen Zahlen
- Neuronales Netz
- CUDA und C++

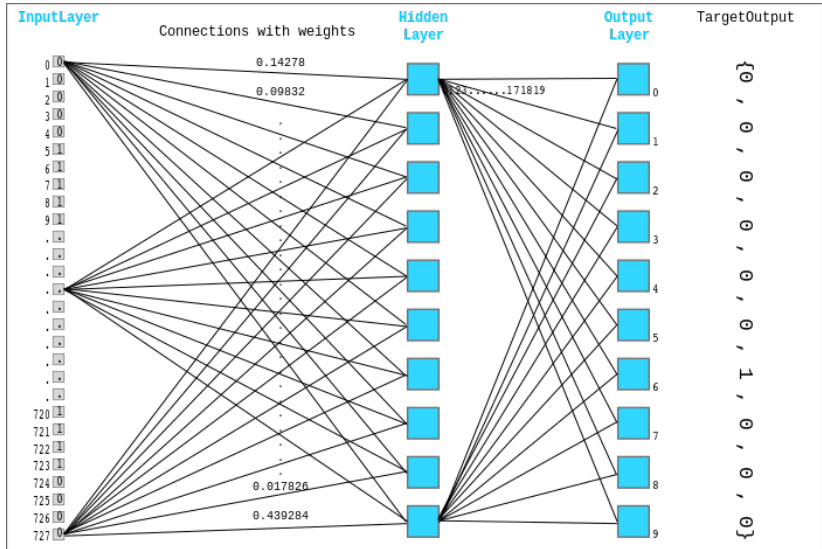
Training/Testing Dataset

THE MNIST DATABASE of handwritten digits

- 60.000 Trainings-Bilder
- 10.000 Test-Bilder
- Auflösung: 28x28
- IDX-Format
- Source: <http://yann.lecun.com/exdb/mnist/>

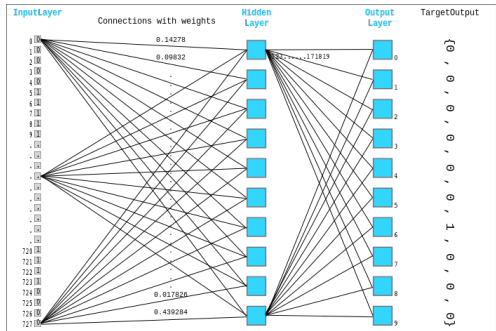


Feed-Forward / Back-Propagation



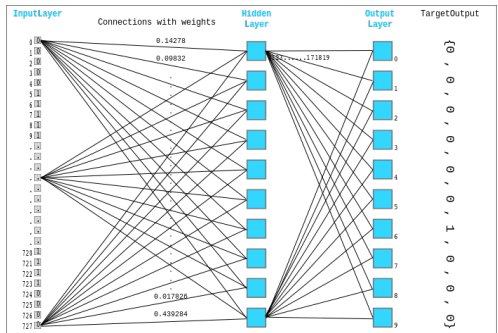
Feed-Forward:

- ▶ Eingehende-Kanten (*edges*)
 - ▶ Thread
 - ▶ Berechnet Kanten-Wert
 - ▶ Speichert in *SharedMemory*
- ▶ Knoten (*nodes*)
 - ▶ Thread-Block
 - ▶ Summiert alle Kanten-Werte
 - ▶ Berechnet Knoten-Wert (Sigmoid)
- ▶ Ausgabe
 - ▶ Index des höchsten Knoten im *OutputLayer*

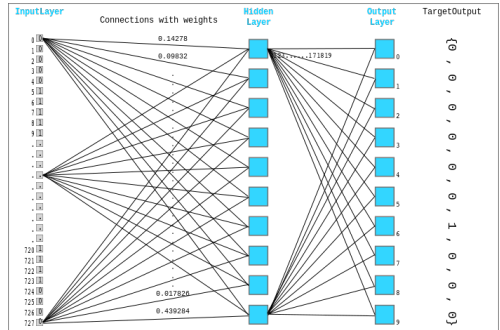


Back-Propagation:

- ▶ Ausgehende-Kanten (edges)
 - ▶ Thread
 - ▶ Berechnet Kanten-Fehler
 - ▶ Speichert in *SharedMemory*
 - ▶ Aktualisiert Kanten-Gewichte
- ▶ Knoten (nodes)
 - ▶ Thread-Block
 - ▶ Summiert alle Kanten-Fehler
 - ▶ Berechnet Knoten-Fehler



- Objektorientiert
- Aufteilung der Knoten auf n Threads
- Jeder Thread berechnet k Knoten-Werte/-Fehler
- Bulk-Synchronisation zwischen den Ebenen



Auswertung: Testsysteme

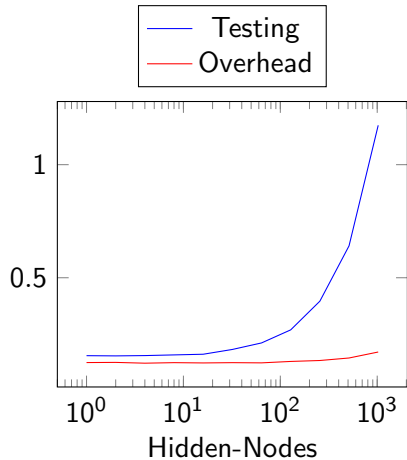
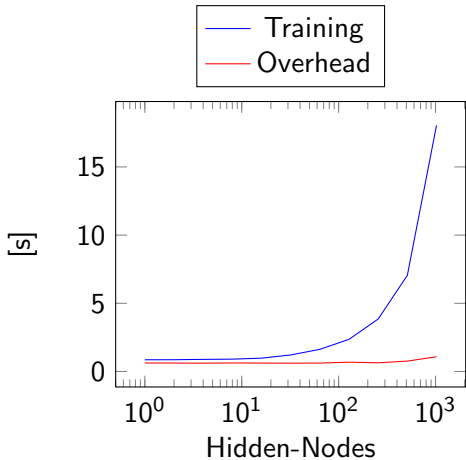
C++:

- ▶ Intel Haswell Core i7-4770 CPU @ 3.40GHz (8 *cores*)
- ▶ Ubuntu 17.04

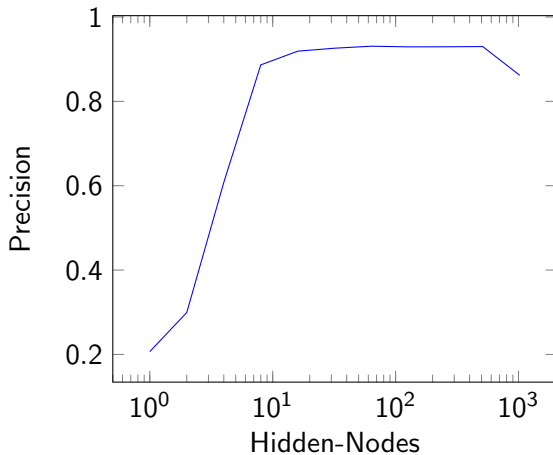
CUDA:

- ▶ Intel Skylake Core i5-6600 CPU @ 3.30GHz (4 *cores*)
- ▶ Nvidia Pascal GTX 1070 @ 1.607 GHz (1920 *cores*)
- ▶ Ubuntu 17.04

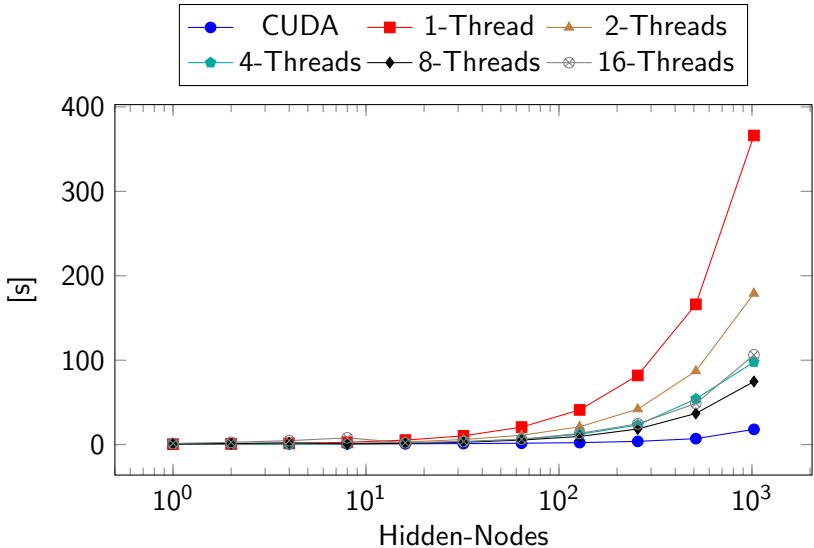
Auswertung: CUDA



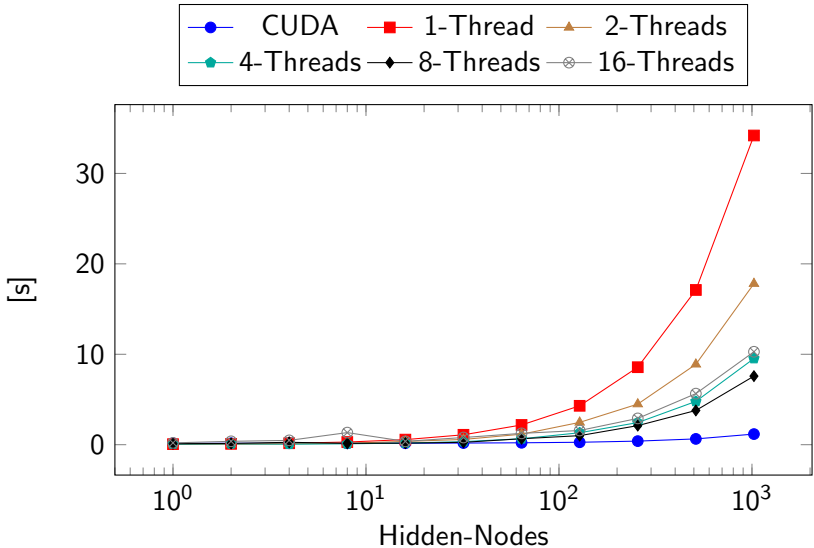
Auswertung: Precision



Auswertung: Training



Auswertung: Testing



Auswertung: Ergebnis

CUDA / C++:

- ▶ 3-layer NN 784-256-10
- ▶ Precision: 98.03%
- ▶ Dauer des Trainings (30x):
 - ▶ CUDA: 111.789sec
 - ▶ C++: 545.011sec
- ▶ Dauer des Tests:
 - ▶ CUDA: 0.40289sec
 - ▶ C++: 1.91795sec

Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition (2010):

- ▶ 6-layer NN 784-2500-2000-1500-1000-500-10
- ▶ Precision: 99.65%

Auswertung: Fazit

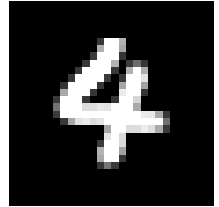
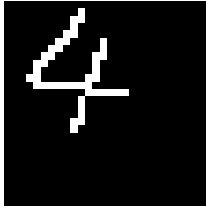
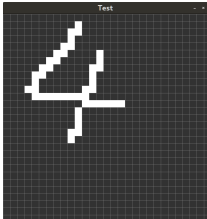
C++:

- ▶ Synchronisierungsoverhead
- ▶ Limitiert durch die Anzahl der CPU-Kerne
- ▶ Auseinanderlaufende Threads beschränken die Parallelität
- ▶ Fehlende Vektorisierung?

CUDA:

- ▶ Zu viele *Kernel*-Aufrufe
- ▶ Zu geringer *Workload*
- ▶ Datenübertragung
- ▶ Viele Speicherzugriffe

Demo



- ▶ GUI-Fenster zum Zeichnen
- ▶ Eingabe 1:1 in Bild umwandeln
- ▶ Normalisierung