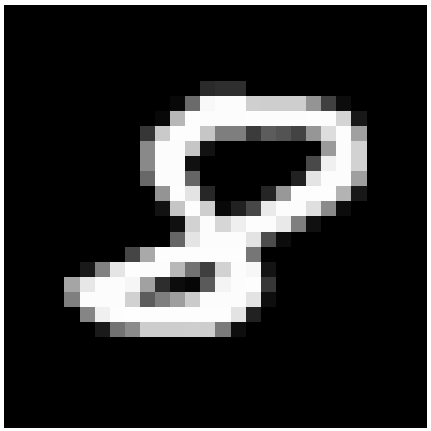
Handschrifterkennung mit CUDA und C++

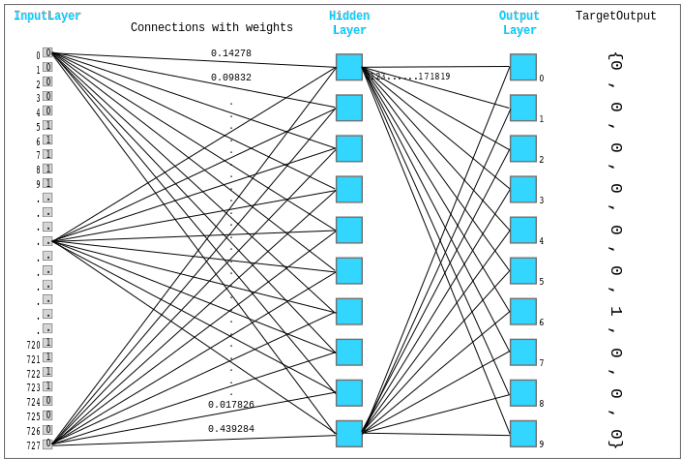
Christopher Haug, Dominik Walter

University of Augsburg
Systems and Networking

July 24, 2017

Aufgabe





8

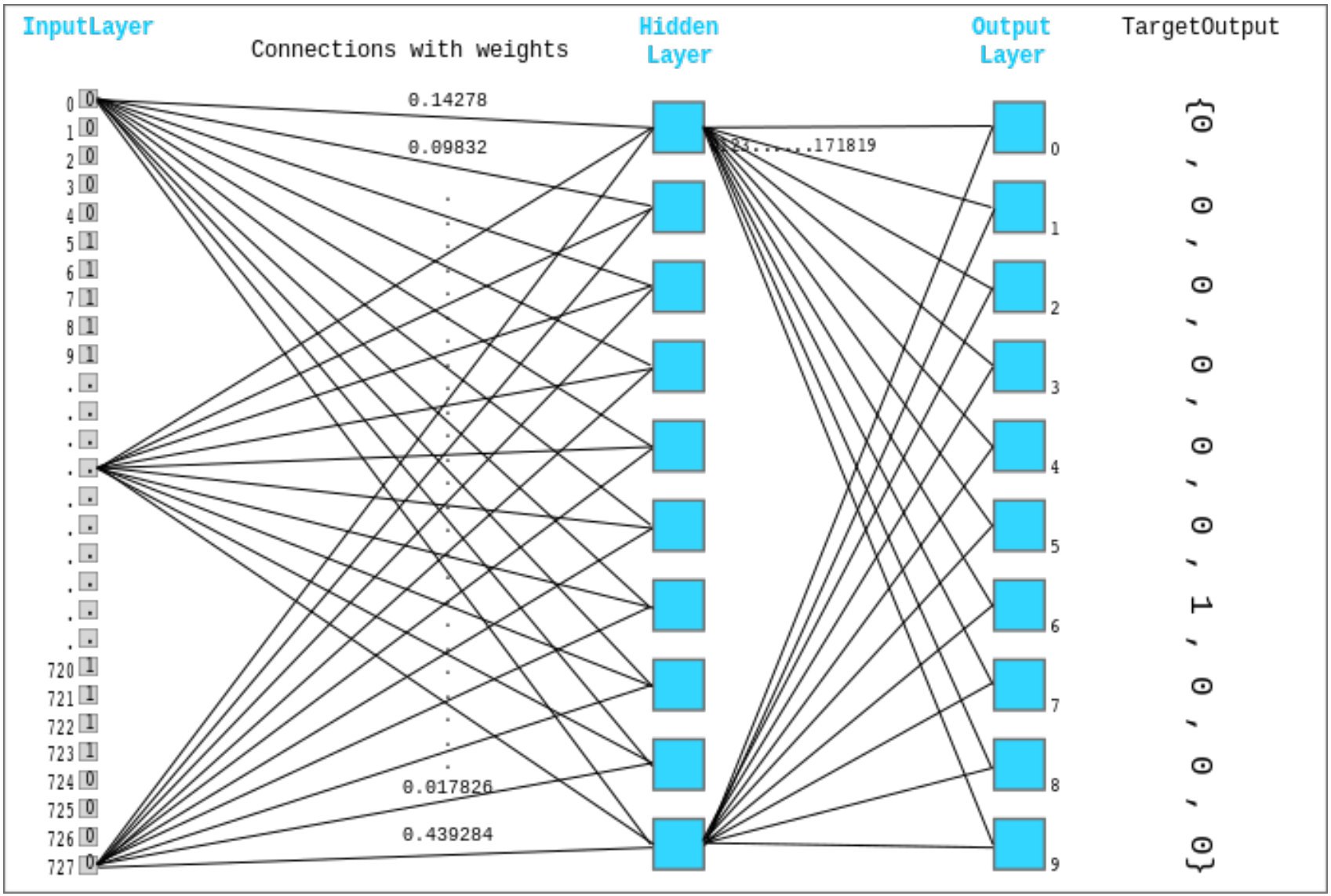
- Erkennung von handgeschriebenen Zahlen
- Neuronales Netz
- CUDA und C++

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/

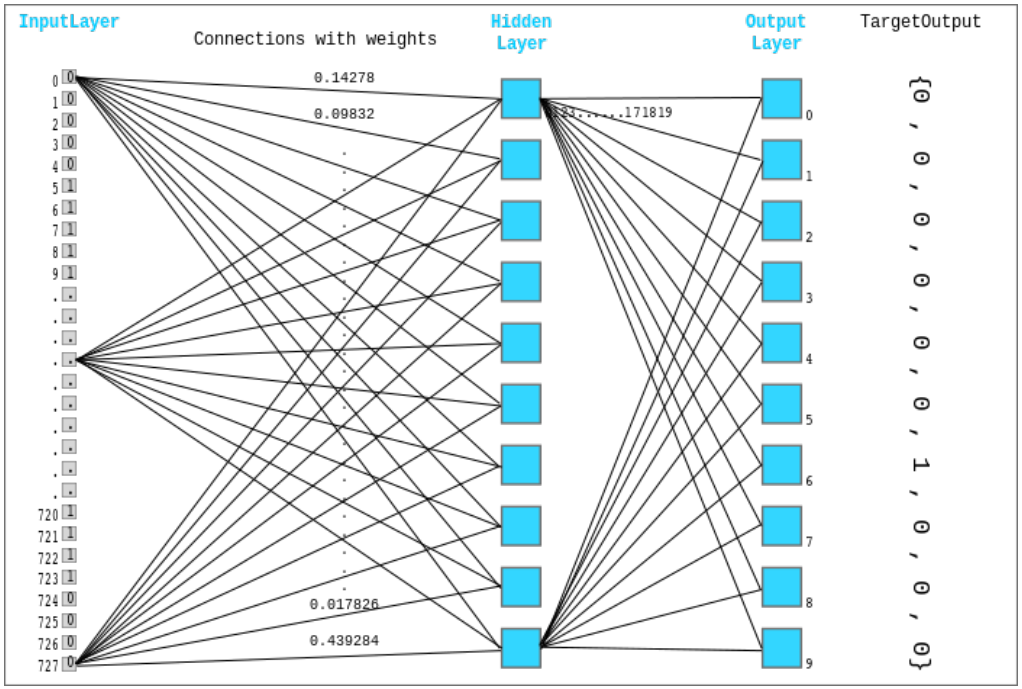
Feed-Forward / Back-Propagation



CUDA-Implementierung

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 höhsten Knoten im *OutputLayer*

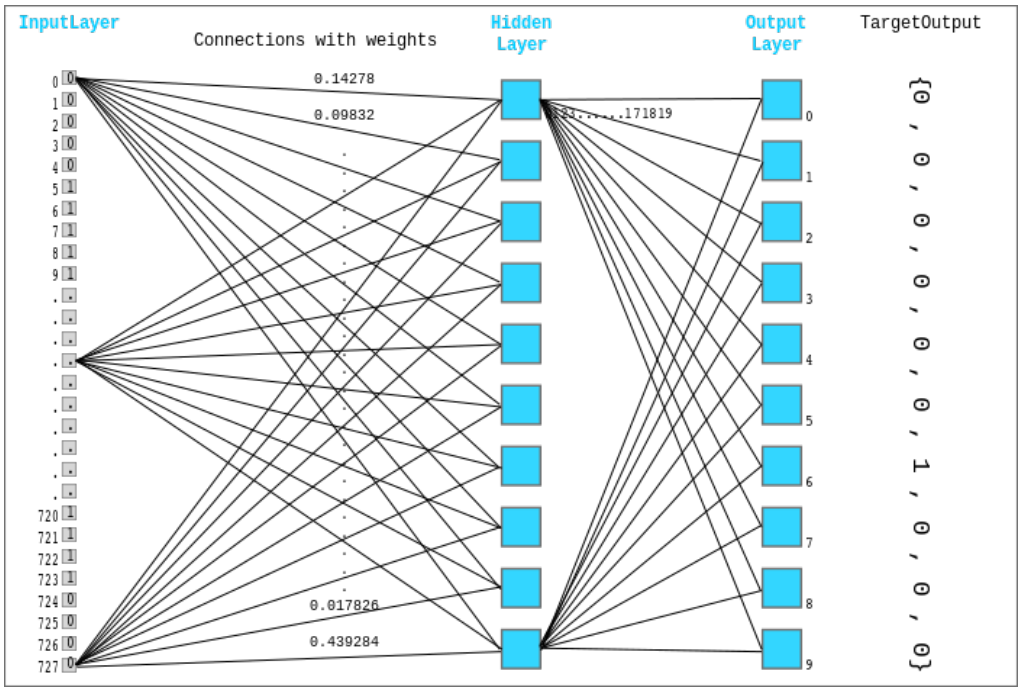


CUDA-Implementierung

Universität Augsburg University

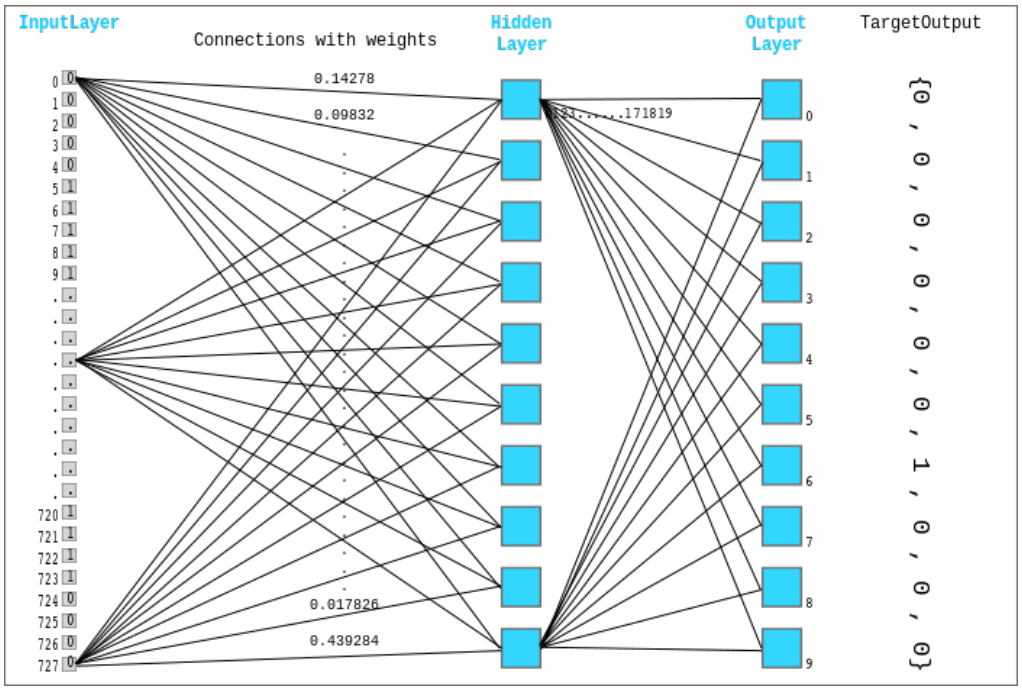
Back-Propagation:

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

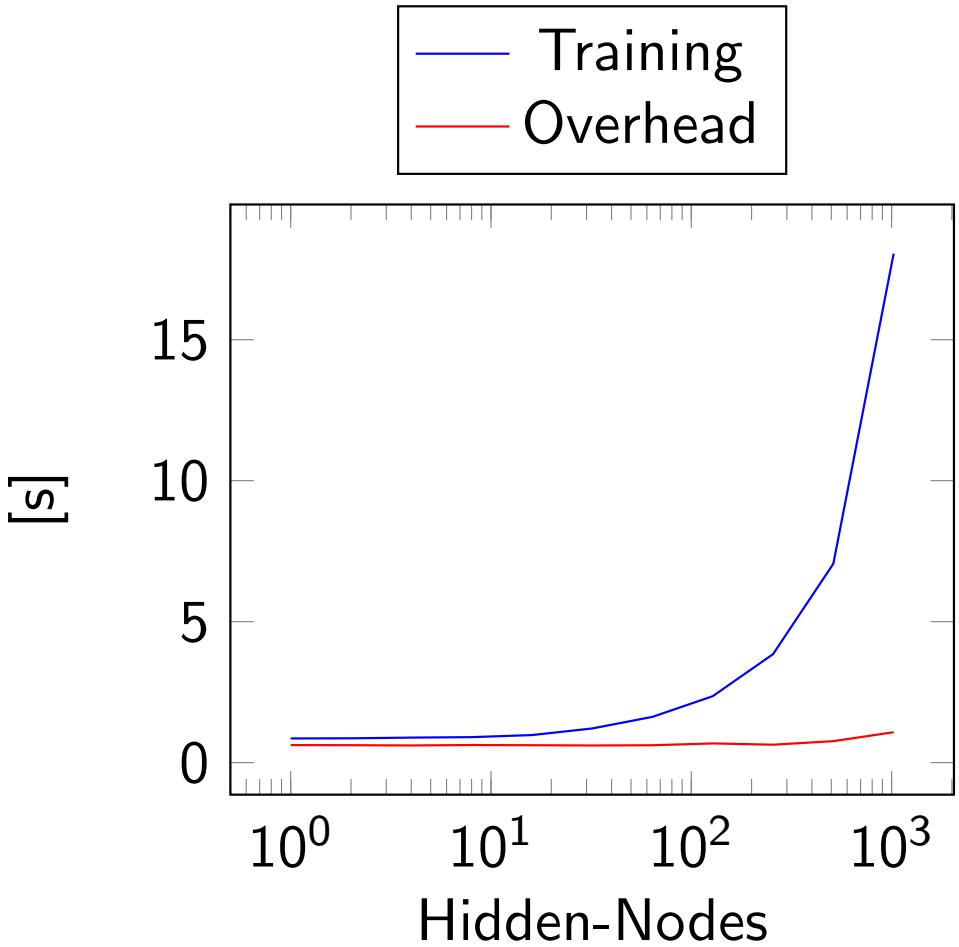


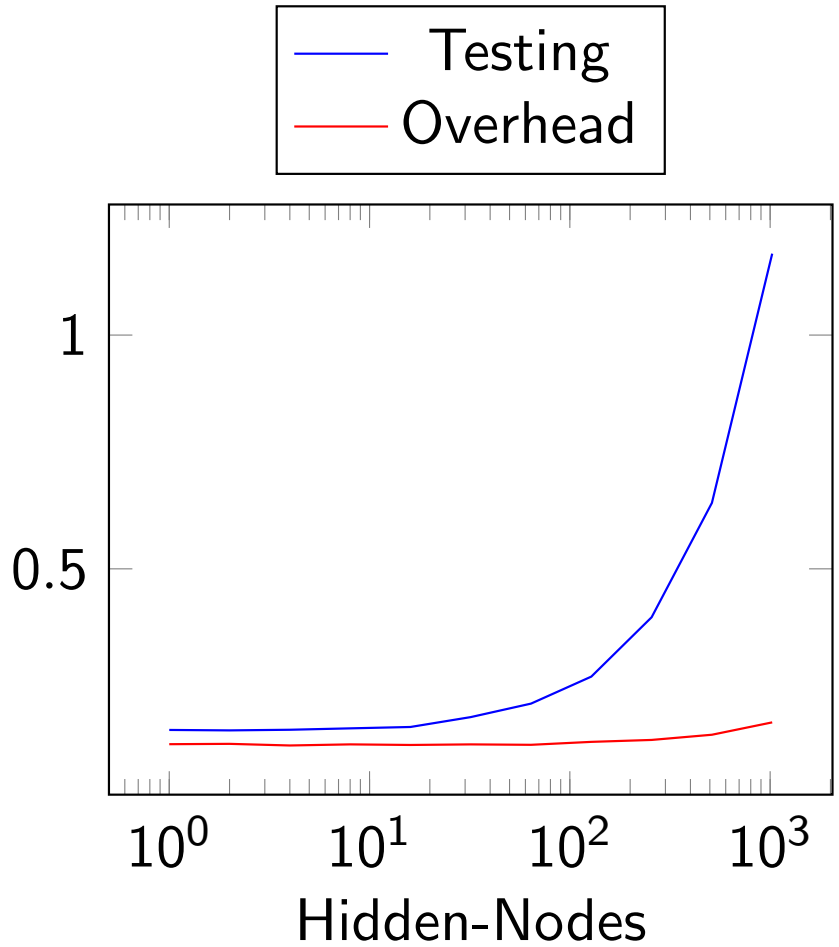
C++-Implementierung

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

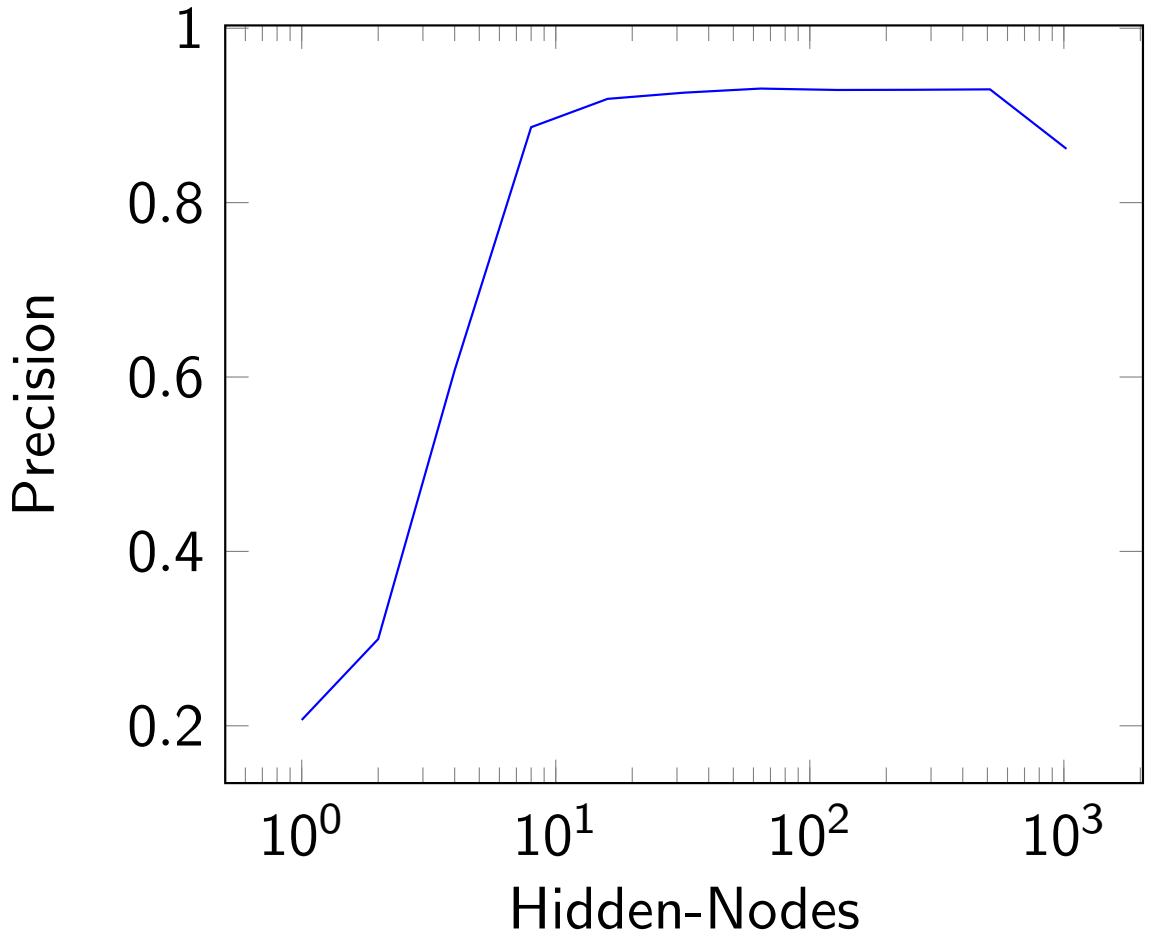


Auswertung CUDA

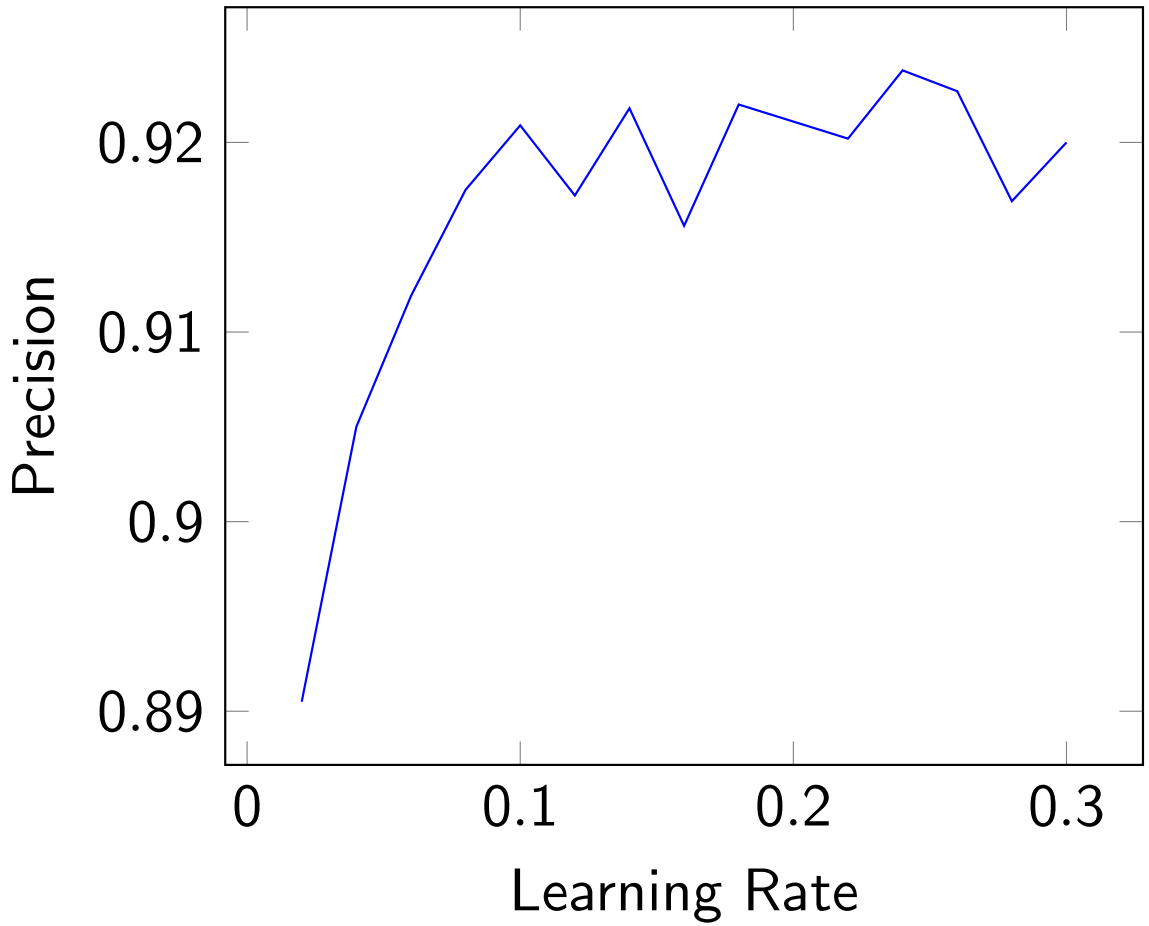




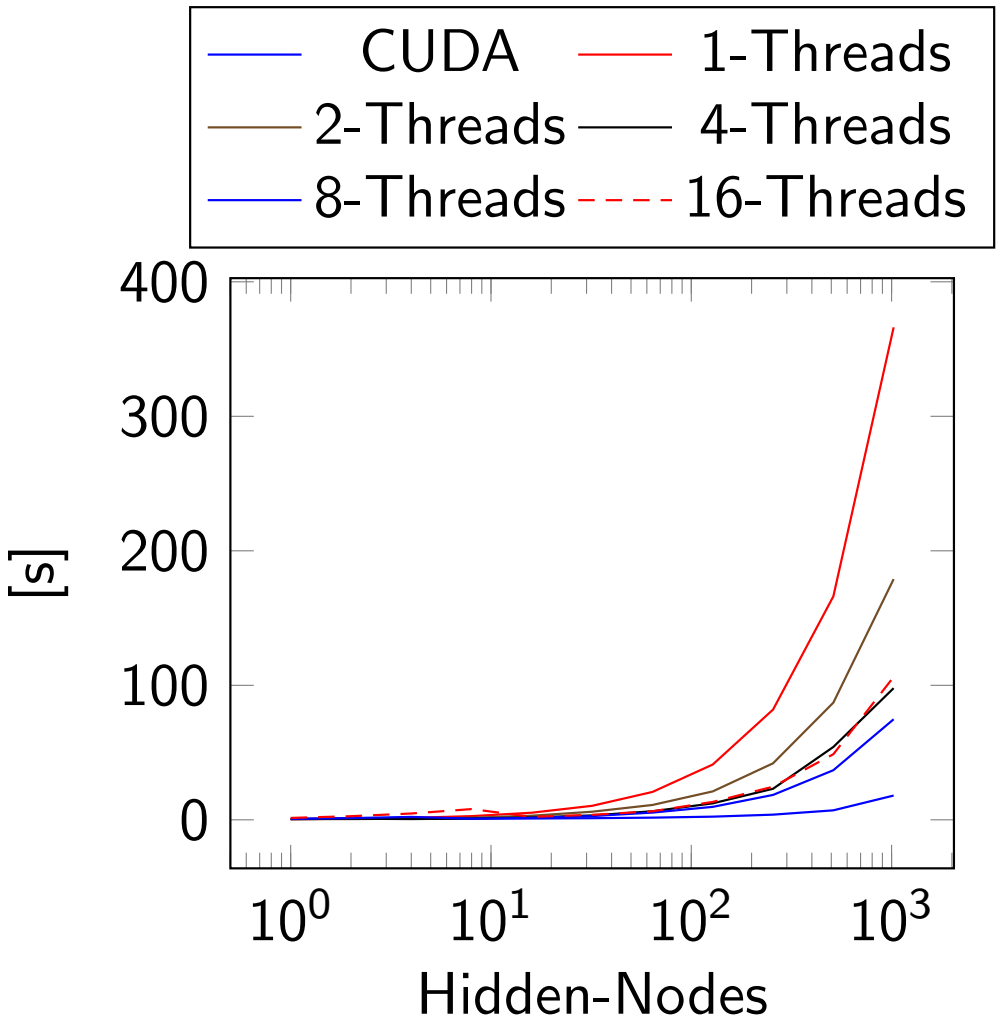
Auswertung-CUDA / C++

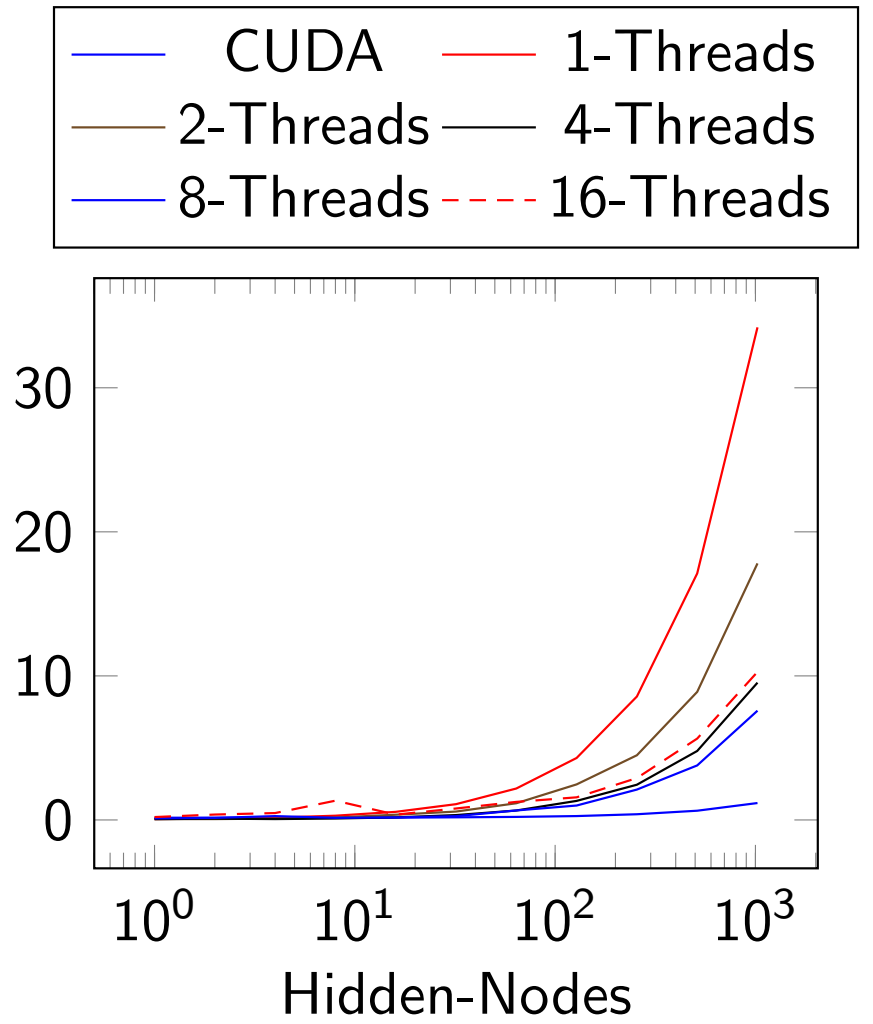


Auswertung-CUDA / C++



Auswertung-CUDA / C++





Bottle-Neck

C++:

- Synchronisierungsoverhead
- Limitiert durch die Anzahl der CPU-Kerne
- Auseinanderlaufende Threads beschränken die Parallelität

CUDA:

- Zu viele *Kernel*-Aufrufe
- Zu geringer *Workload*
- Datenübertragung