TECHNISCHE UNIVERSITÄT DRESDEN

Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)

Analyse eines Forschungsthemas INF-D-960

Texterkennung in topographischen Karten: Untersuchungen mit dem Deep-Learning-Framework Keras in einer HPC-Systemumgebung

Jan Stephan

Betreuender Hochschullehrer: Prof. Dr. Wolfgang E. Nagel

Betreuer: Dr. Peter Winkler

02. Oktober 2018

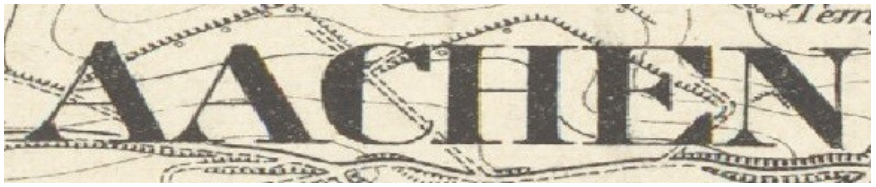
ZIH
Zentrum für Informationsdienste und Hochleistungsrechnen

Gliederung

- Einleitung
  - Motivation
  - Ausgewählte Arbeiten
  - Ziel
- Daten und Methoden
  - Reale und künstliche Daten
  - Bilderkennung
  - Textklassifizierung
  - Umsetzung auf dem HPC-System *Taurus*
- Ergebnisse
  - Trainings- und Validierungsdaten
  - Erkennungsrate und Losses
  - Performance
- Fazit & Ausblick

Motivation

- Analyse historischer topographischer Karten
  - Raum- und Landschaftsplanung
  - Ziel: Analyse der historischen Siedlungsentwicklung
- Beispiele für *Optical Character Recognition* (OCR):
  - Automatisierte Erkennung von Kennzeichen
  - Digitalisierung von Büchern
  - Informationsextraktion aus nicht digitalisierten Dokumenten
- Problem: „störende“ Bildinformationen



Ausgewählte Arbeiten

- Jaderberg et al. (2016)
  - Ziel: Texterkennung in Fotografien / „natural scenes“
  - Mehrstufiges Convolutional Network
  - Von ScaDS getestet

- Shi et al. (2017)
  - Ziel: Texterkennung in Fotografien
  - Convolutional Network + LSTM für Textsequenzen
  - eigene LSTM-Dekodierung

Ziel

- Texterkennung in und -extraktion aus topographischen Karten
- Nutzung der GPUs auf dem HPC-System *Taurus*
- Skalierungsverhalten des Netzes
  - Performance
  - Erkennungsraten

Daten und Methoden

Implementierung

- Neuronales Netz
- Implementierung mit *Keras*
- *TensorFlow*-Backend

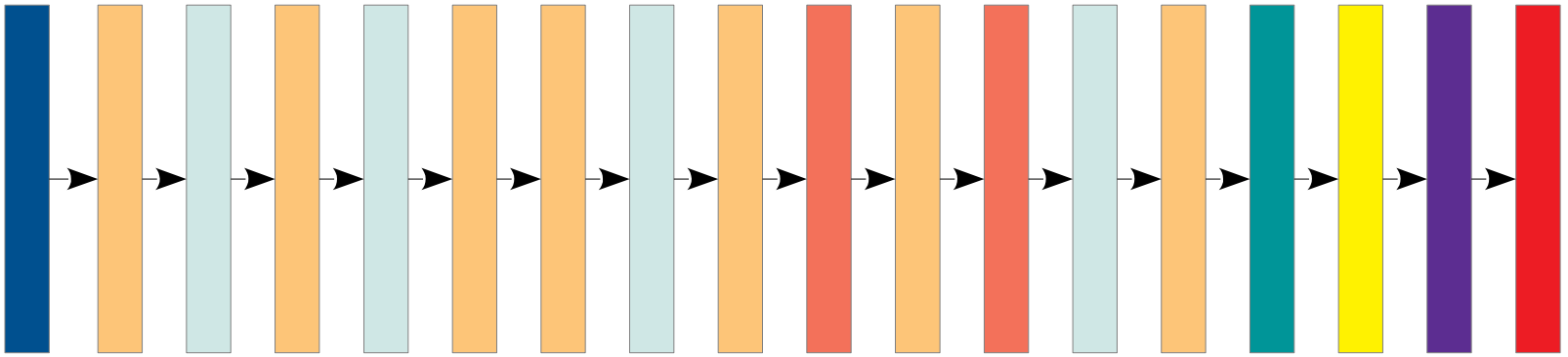
Reale und künstliche Daten

- Training mittels künstlicher Daten
- zufällige Breite und Höhe
- Datengenerator des ScaDS-Projekts (E. Schölzel)

Bilderkennung - Daten

- Beschränkung auf 6 Zeichen pro Wort
- Skalierung der Eingangsdaten
  - 192 x 64
- Greyscale
- float32

Bilderkennung - Netzwerk



Input

Convolution

MaxPool

BatchNormalization

Reshape

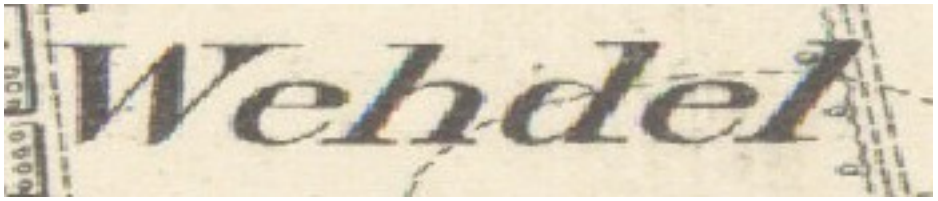
Bidirectional LSTM

Dense

Softmax

(Shi et al.)

Textklassifizierung - CTC



W W W ε e ε h h d d ε e e ε l l

W e h d e l

Wehdel

Connectionist Temporal Classifier (CTC)

(Graves et al., 2006)

Textklassifizierung - CTC

- Nicht nativ in *Keras* enthalten
- Teil von *TensorFlows* `tf.keras.backend`
- Training: `ctc_batch_cost` (ersetzt Loss-Funktion)
- Inferenz: `ctc_decode`

Umsetzung auf dem HPC-System *Taurus*

- Training und Inferenz: `gpu2`-Partition
  - 4 NVIDIA Tesla K80
  - 62 GiB Arbeitsspeicher
- Datengenerator: beliebiger CPU-Knoten
  - in der Regel `sandy`-Partition
  - 16 CPU-Kerne
  - 30 GiB Arbeitsspeicher
- SCS5-Modulumgebung (Training und Inferenz)
  - *Keras* `2.2.0-foss-2018a-Python-3.6.4`
  - *OpenCV* `3.4.1-foss-2018a-Python-3.6.4`
- `venv` (Datengenerator)
  - `/projects/p_scads/keras/new_venv2`

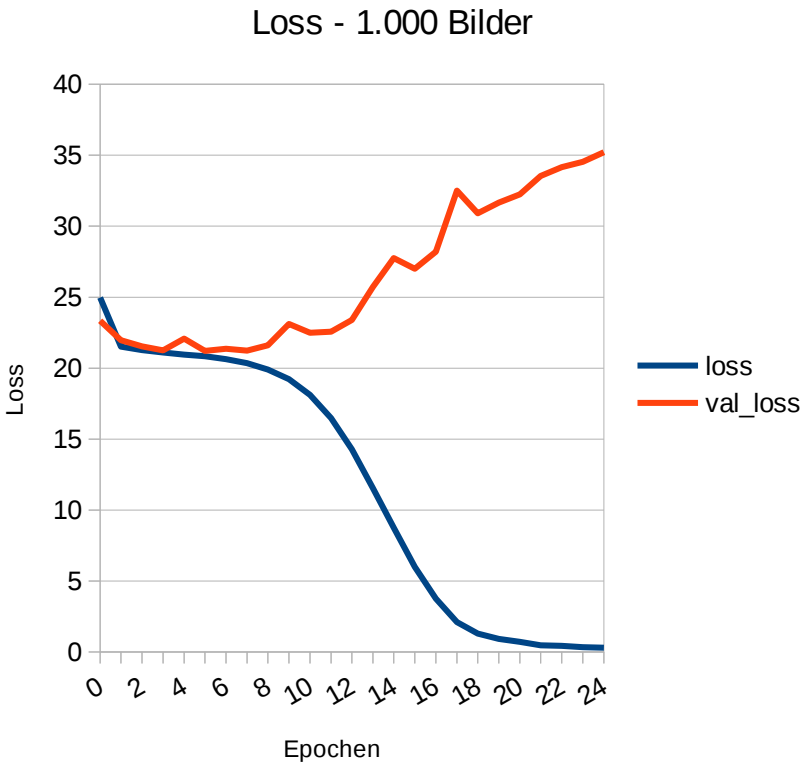
Ergebnisse

Erkennungsrate

- Wordscore W
  - 0 oder 1
- Charscore C
  - korrekte Buchstaben / Wortlänge
  - zwischen 0 und 1
- Wenn $W = 1$, dann $C = 1$
- Wenn $C < 1$, dann $W = 0$
- Beispiele:
  - Real: „Bramel“. Erkannt: „Bramel“. $W = C = 1$
  - Real: „Tannen“. Erkannt: „Taunen“. $W = 0$, $C = 0{,}8333$
- Sonderfall:
  - Real: „Wehdel“. Erkannt: „NWehdel“. $W = C = 0$

Overfitting

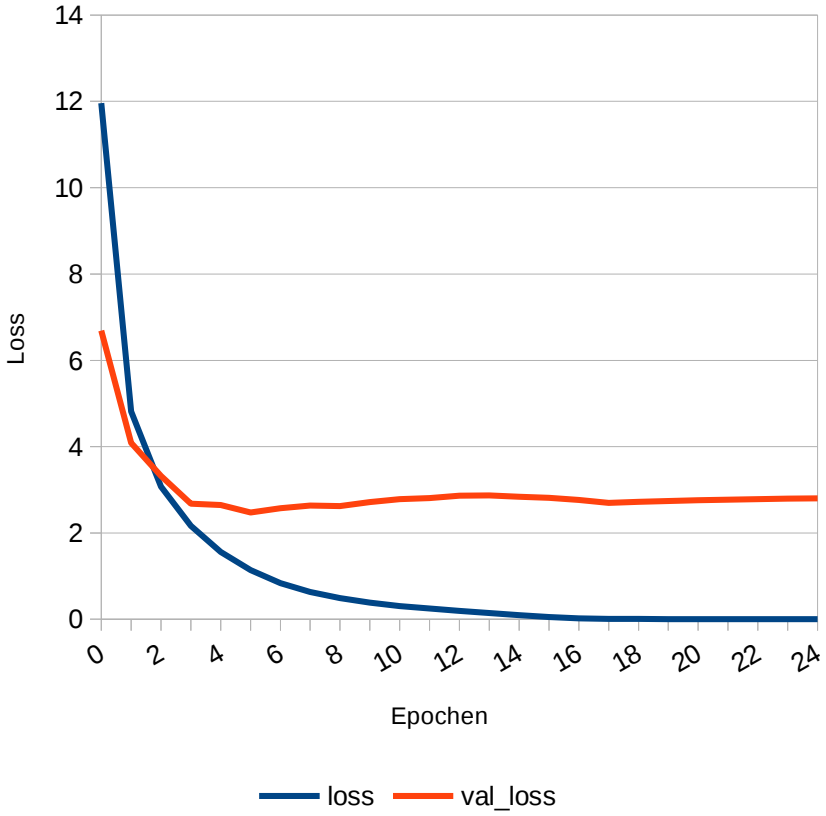
- allgemeines ML-Problem
- CTC besonders anfällig
- Kriterium: Validierungsloss

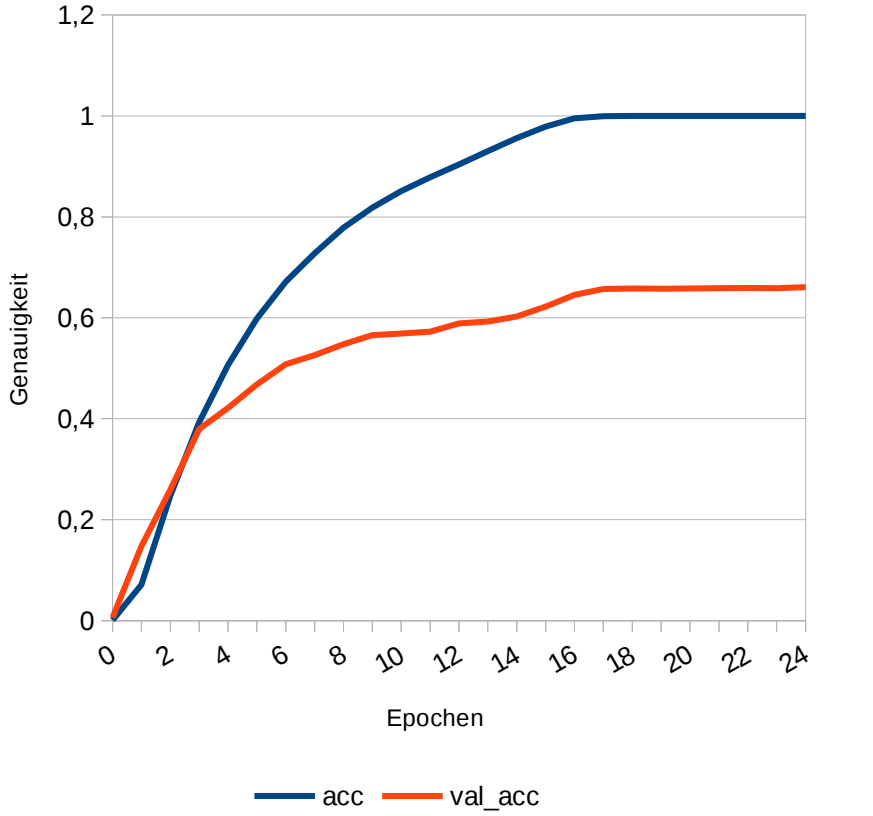


Trainings- und Validierungsdaten

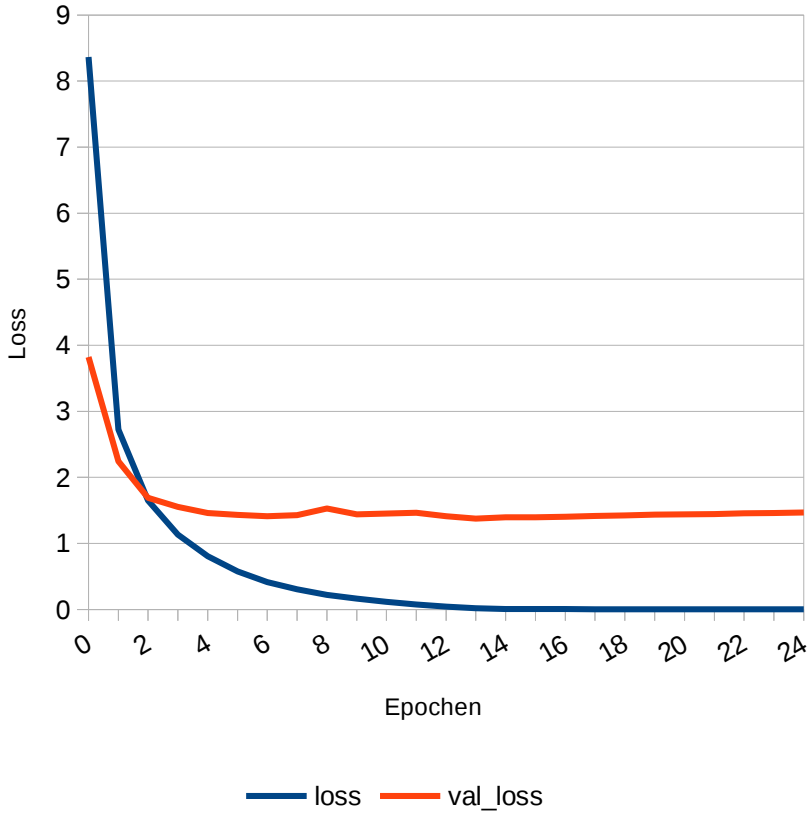
- Zwei Datensätze für das Training
  - Wortlänge 6 Buchstaben
  - `wl6_120k`: 120.164 generierte Bilder
  - `wl6_250k`: 250.000 generierte Bilder
- Zwei Datensätze für die Validierung
  - Wortlänge 6 Buchstaben
  - `wl6_1000`: 1.000 generierte Bilder
  - `wl6_real`: 27 reale Bilder (aus Kartenmaterial ausgeschnitten)
- Validierung der Netzwerke in drei Schritten
  1. Trainingsdatensatz
  2. Validierungsdatensatz: `wl6_1000`
  3. Validierungsdatensatz: `wl6_real`

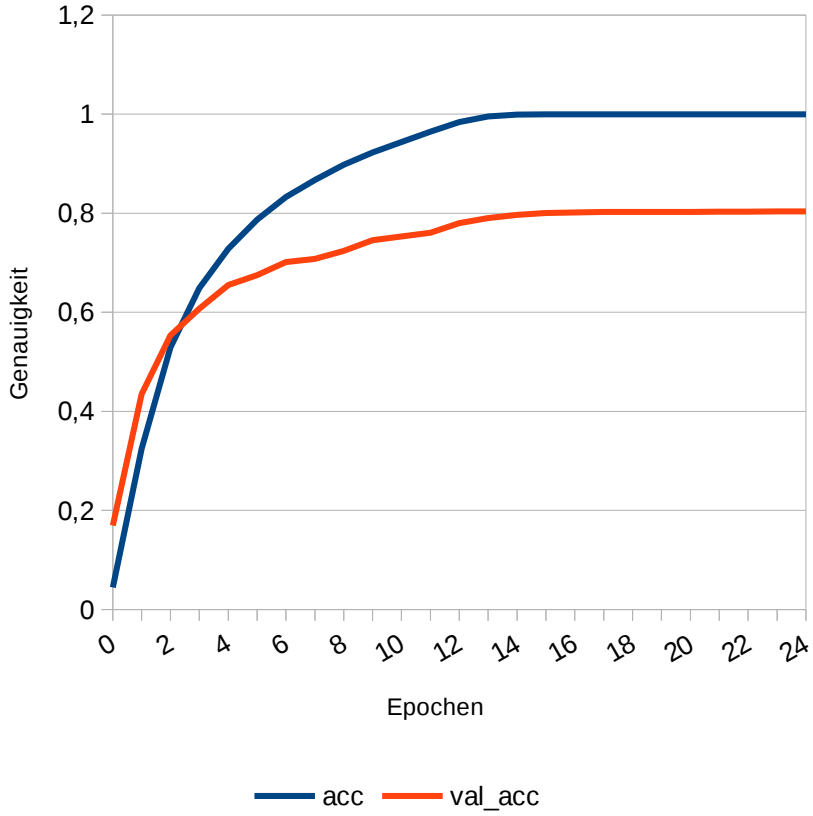
Loss & Genauigkeit: `wl6_120k`



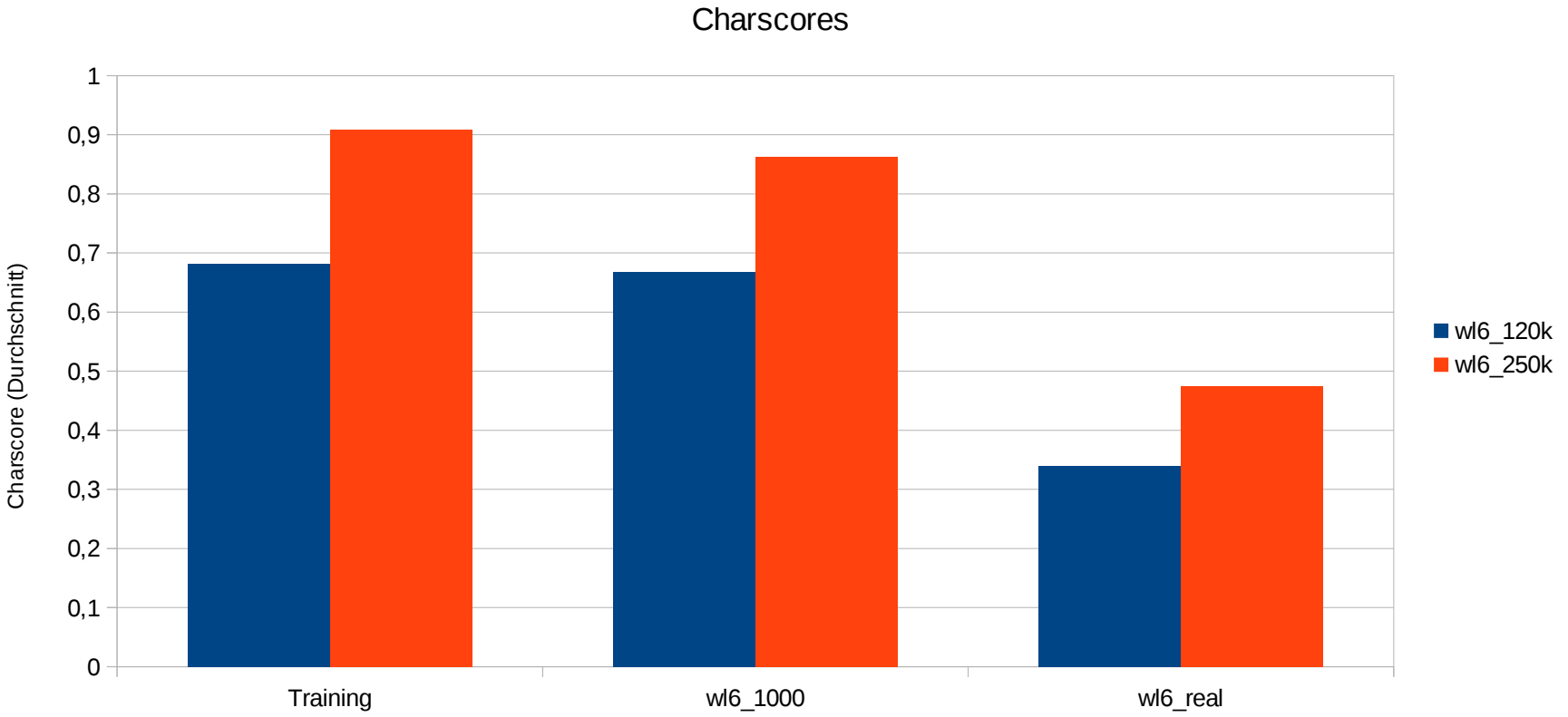


Loss & Genauigkeit: wl6_250k





Erkennungsrate



Beispiel: Haaren

`wl6_120k`: „Haaren“

$W = 1$

$C = 1$

`wl6_250k`: „Haaren“

$W = 1$

$C = 1$

Beispiel: Ostiem

`wl6_120k`: „Ostion“

$W = 0$

$C = 0{,}6666$

`wl6_250k`: „Ostiem“

$W = 1$

$C = 1$

Beispiel: Aachen

`wl6_120k`: „gAacli“

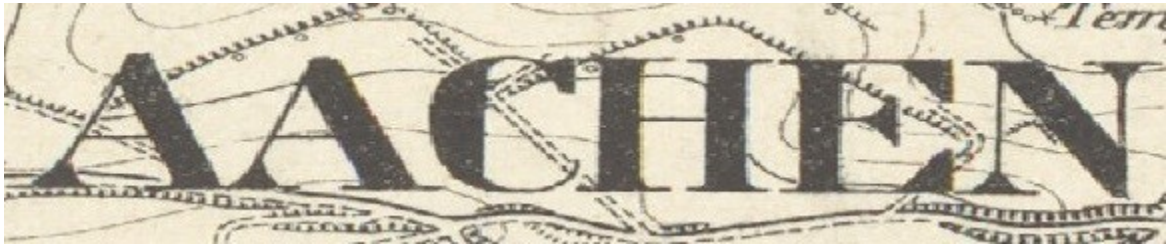
$W = 0$

$C = 0$

`wl6_250k`: „Aatiz“

$W = 0$

$C = 0{,}3333$



Beispiel: Minten

`wl6_120k`: „Mduren“

$W = 0$

$C = 0{,}5$

`wl6_250k`: „Jfnca“

$W = 0$

$C = 0{,}1666$

Beispiel: Neuhof

`wl6_120k`: „gNeuhof“

$W = 0$

$C = 0$

`wl6_250k`: „Neuhol“

$W = 0$

$C = 0{,}8333$

Performance - Training

	Gesamtlaufzeit	Epochenlaufzeit
`wl6_120k`	4h 20min 54s	10min 39s
`wl6_250k`	9h 3min 22s	21min 44s

Performance - Inferenz

	Gesamtlaufzeit	Laufzeit pro Bild
`wl6_120k`	19min 25s	9,7ms
`wl6_250k`	40min 17s	9,7ms

Weitere Ergebnisse

- Erste Tests mit:
  - Variabler Wortlänge
  - Multi-GPU-Nutzung
  - Größeren Datensätzen (500.000 Bilder)
- Keine zufriedenstellenden Ergebnisse

Fazit und Ausblick

Entwicklungsstand

- Ausgewählte Methode für Texterkennung in und -extraktion aus topographischen Karten geeignet
- Vorbedingung: Trainingsdaten in guter Qualität und hoher Menge
- einfache GPU-Nutzung durch Keras & TensorFlow

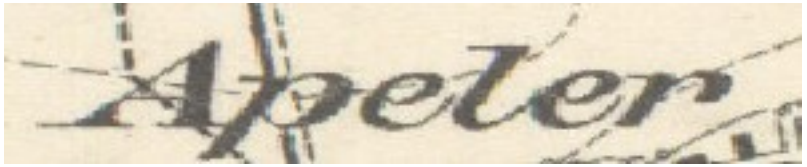
Weitere Arbeiten

- Optimierung der Trainingsdaten
- Variable Wortlängen
- Performance-Verbesserungen
- Weitere Optionen

Optimierung der Trainingsdaten

- Einführung historischer Schriftarten
  - kostenpflichtig
- Verbesserung des Trainings durch Originalschriftarten wahrscheinlich
  - *Roemisch* (Quelle: [LinotypeR])
  - *Kursivschrift* (Quelle: [LinotypeK])

grosse 

Apeler 

Variable Wortlängen

- Netzwerk grundsätzlich geeignet
  - Kleinere Quelltextänderungen nötig
- Offener Punkt: feste Bildbreite für variablen Text
  - Neue Trainings- und Validierungsdaten notwendig

Performance-Verbesserungen

- Verwendung mehrerer GPUs mit Keras prinzipiell möglich
  - `multi_gpu_model`
- Offene Punkte:
  - erhoffter Speedup bleibt aus
  - Performance-Analyse notwendig

Weitere Optionen

- Nutzbarkeit von *Transfer Learning* untersuchen
- Trennung von Text und Hintergrund
- verbesserte Score-Berechnung (Sonderfälle!)

Vielen Dank!

Quellen

A. Graves, S. Fernández, F. Gomez, J. Schmidhuber: *Connectionist Temporal Classification*: *Labelling Unsegmented Data with Recurrent Neural Networks*. In: Proceedings of the 23rd International Conference on Machine Learning, 2006. S. 369 – 376

B. Shi, X. Bai, C. Yao: *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition*. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 39, Nov. 2017. S. 2298 – 2304

M. Jaderberg, K. Simonyan, A. Vedaldi, A. Zisserman: *Reading Text in the Wild with Convolutional Networks*. In: International Journal of Computer Vision 116, 2016. S. 1 - 20

E. Schölzel: *genSet3.py*, 2017. Unveröffentlicht.

[LinotypeR] Linotype.com: *Roemisch™ Schriftfamilie*. Online abrufbar unter https://www.linotype.com/de/1410/roemisch-schriftfamilie.html, zuletzt abgerufen am 01. Oktober 2018)

[LinotypeK] Linotype.com: *Kursivschrift™ Schriftfamilie*. Online abrufbar unter https://www.linotype.com/de/915/kursivschrift-schriftfamilie.html, zuletzt abgerufen am 01. Oktober 2018