Ende zu Ende Spracherkennung

Moritz Wolter

21.09.2017

Inhalt

Worum geht es?

Das Problem

- Was wird gesagt?
- Wie können Text und Ton zugeordnet werden?
- Mit welcher Modell-Architektur?
- Was sind gute Optimierungsparameter?
- Wie lässt sich gute Generalisierung sicherstellen?

ConvLSTM

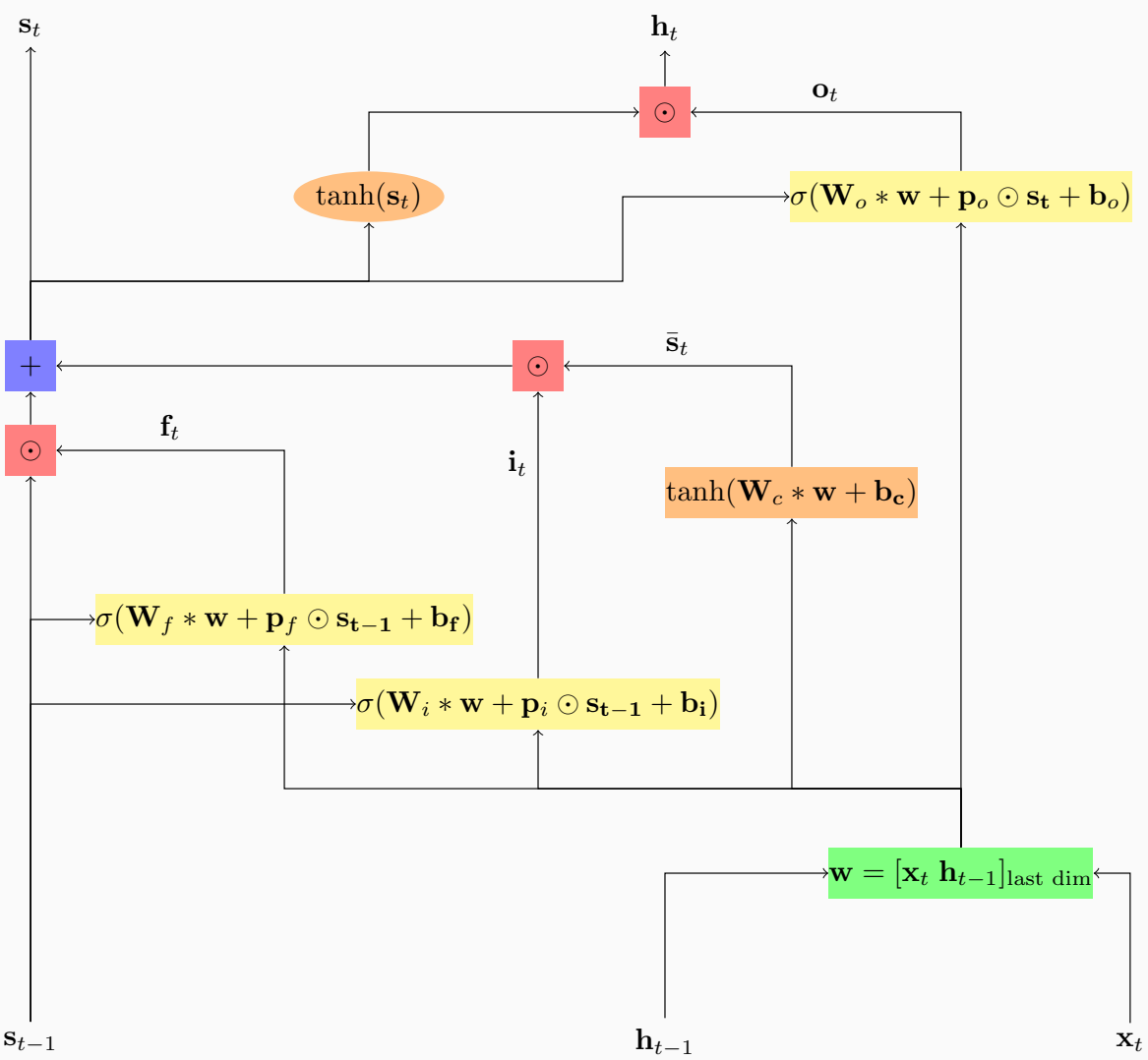

Abbildung 1: Faltungs-LSTM Zelle.

Die LAS-Architektur

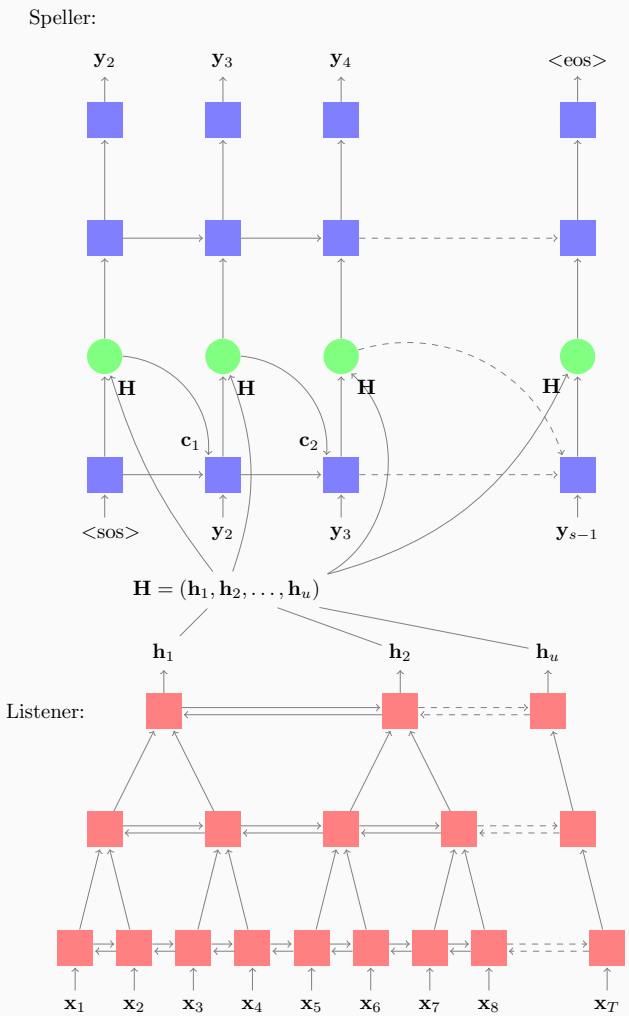

Abbildung 2: Listener und speller.

Die LAS-Gleichungen

Attend und spell Zell-Berechnungen:

$$\mathbf{s}_i = \text{RNN}(\mathbf{s}_{i-1}, \mathbf{y}_{i-1}, \mathbf{c}_{i-1}), \tag{1}$$

$$\mathbf{c}_i = \text{AttentionContext}(\mathbf{s}_i, \mathbf{H}), \tag{2}$$

$$P(\mathbf{y}_i|\mathbf{x}, \mathbf{y}_{<i}) = \text{CharacterDistribution}(\mathbf{s}_i, \mathbf{c}_i). \tag{3}$$

AttentionContext Berechnungen:

$$e_{i,u} = \phi(\mathbf{s}_i)^T \psi(\mathbf{h_u}), \tag{4}$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})}, \tag{5}$$

$$\mathbf{c}_i = \sum_u \alpha_{i,u} \mathbf{h}_u. \tag{6}$$

Implementierung

Herausforderungen

- Attend und spell Zelle
- Dekodier-Schleifen-Logik
  - Greedy decoding
  - Beam search
    - Mehrere Hypothesen.
    - Betrachtet die individuelle Wahrscheinlichkeit einzelner Sequenz-Elemente, den Status und die Gesamtlänge.
- Effizienz

Attend und spell Zellenlayout

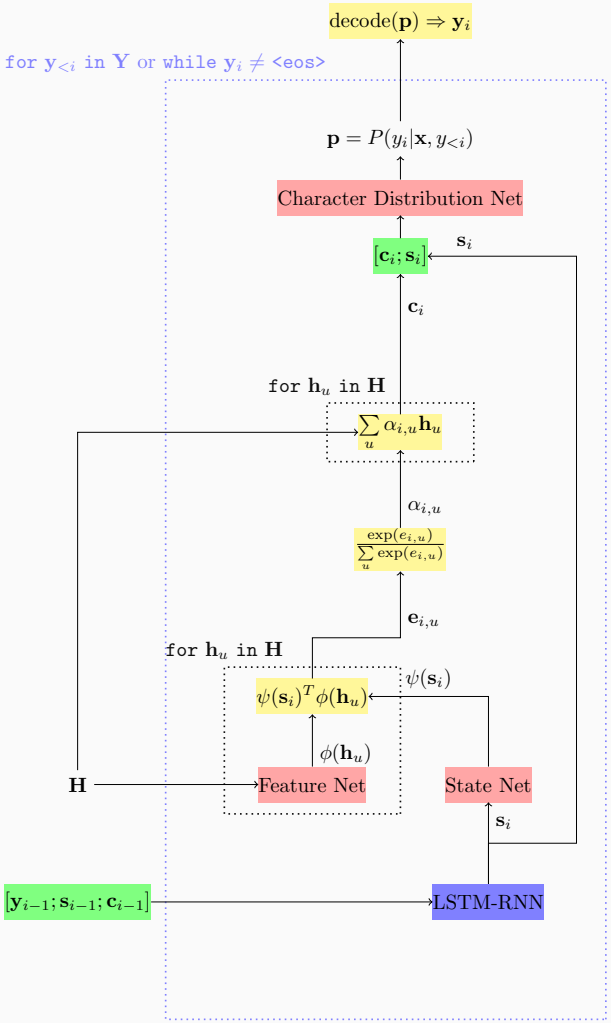

Abbildung 3: Attend und spell Zellen-Flow-Chart.

Effiziente Implementierung

- Evaluiere das feature Netz außerhalb der Attend und Spell Zelle.
- Tensoren mit konstanter Größe für effizienter Speicherallokation.
- Sequenzlängen beachten gerade auf dem Sprachsignal.
- Sequenzlängen verwenden um keine Nullen zu verarbeiten.

Dropout

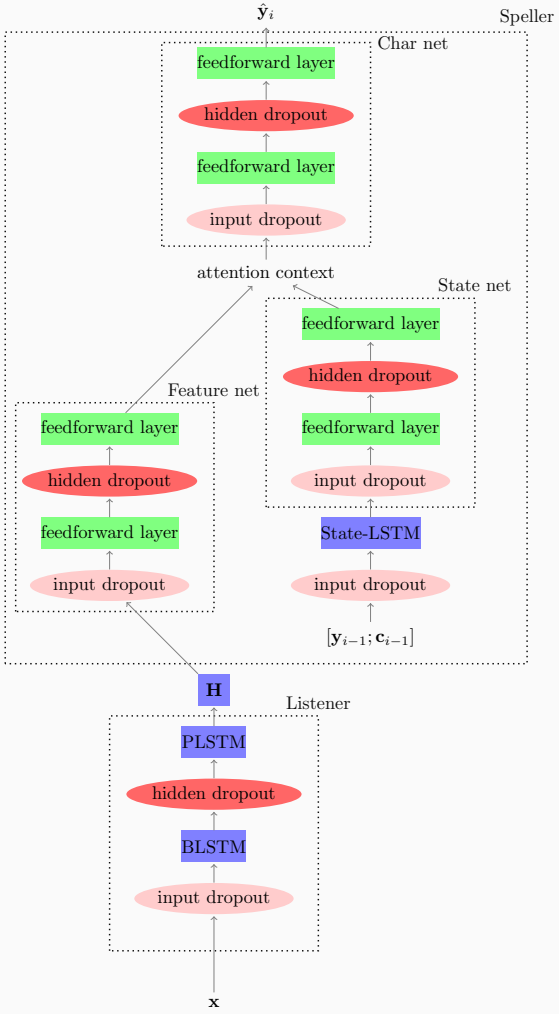

Abbildung 4: Input dropout - hellrot. Hidden dropout - dunkelrot.

Ergebnisse

Experimentalparameter

- TIMIT Sprachkorpus.
- Schrittweite: 0.001.
- batch_größe: 16 Sätze, Im Trainings-Datensatz insgesamt 3696.
- Validierungsdatensatzgröße: 4 Sätze.
- Testdatensatzgröße: 192 Sätze.
- Alle 20 schritte ein Validierunsschritt.

Aufmerksamkeitsplots

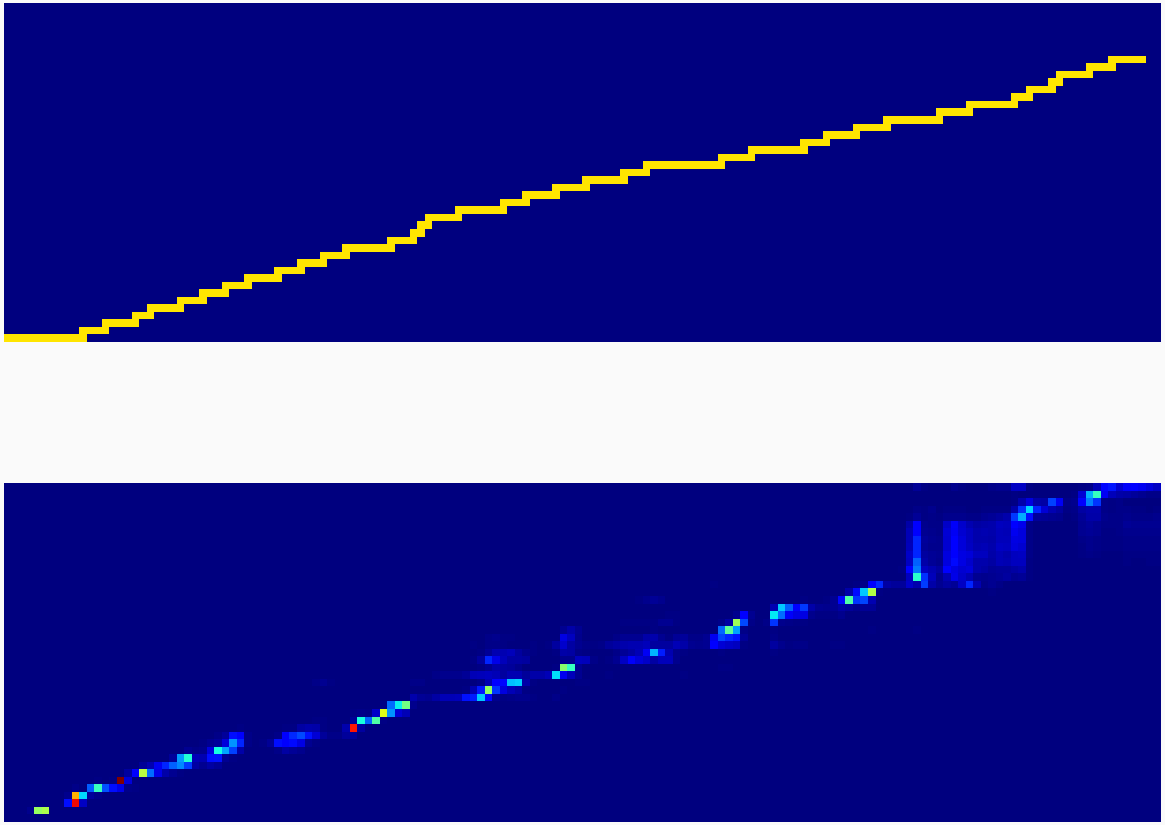

Abbildung 5: Plot der Aufmerksamkeitsvektoren für alle 45 Phoneme, die dem Timit sample `fmld0_sx295` (unten) zugeordnet wurden und Phonemdauer-Zuordnung eines menschlichen Zuhörers (oben).

Aufmerksamkeitsplots

Ziel labels

<sos> sil ih f sil k eh r l sil k ah m z sil t ah m aa r ah
hh ae v er r ey n jh f er m iy dx iy ng ih sil t uw sil <eos>

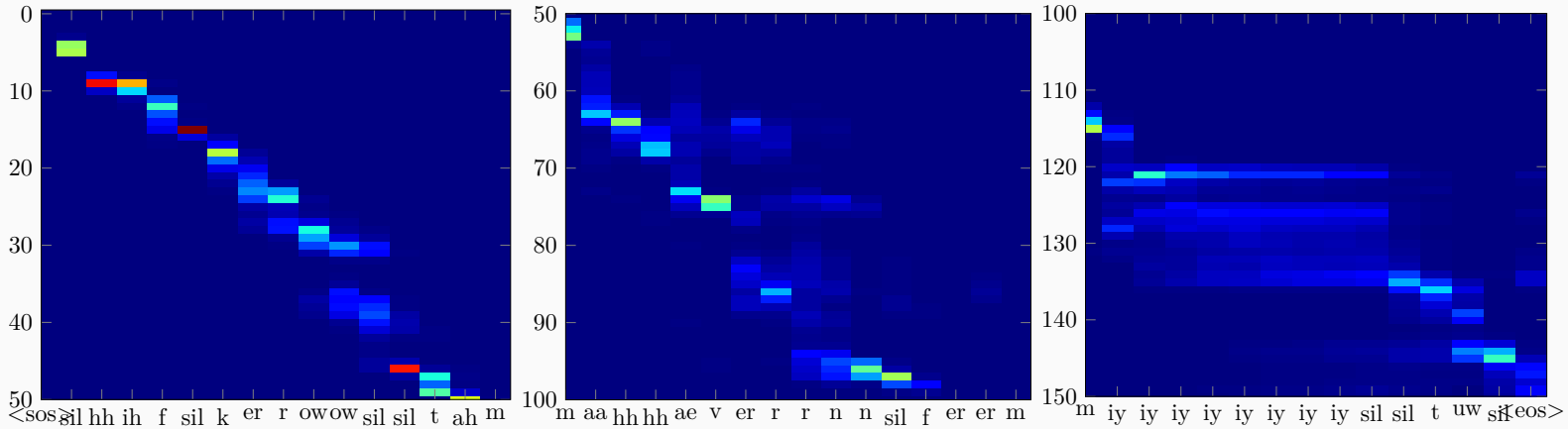

Abbildung 6: Aufmerksamkeitsgewichte α und Netzwerk output.

Rückkopplungswahrscheinlichkeit

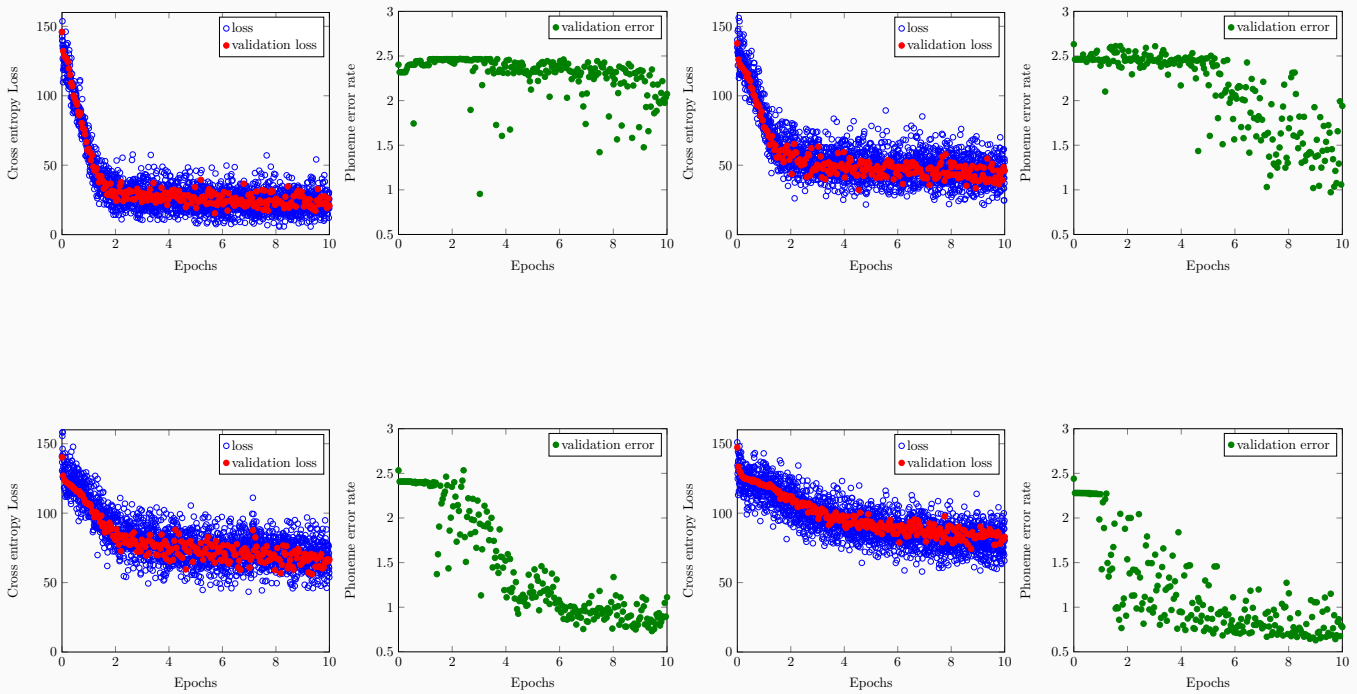

Abbildung 7: Wiederholungen des selben Experimentes mit sich erhöhender Rückkopplungs-Wahrscheinlichkeit in der LAS-Zelle 0.2, 0.4, 0.6, 0.8

Dropout-LAS mit beam search

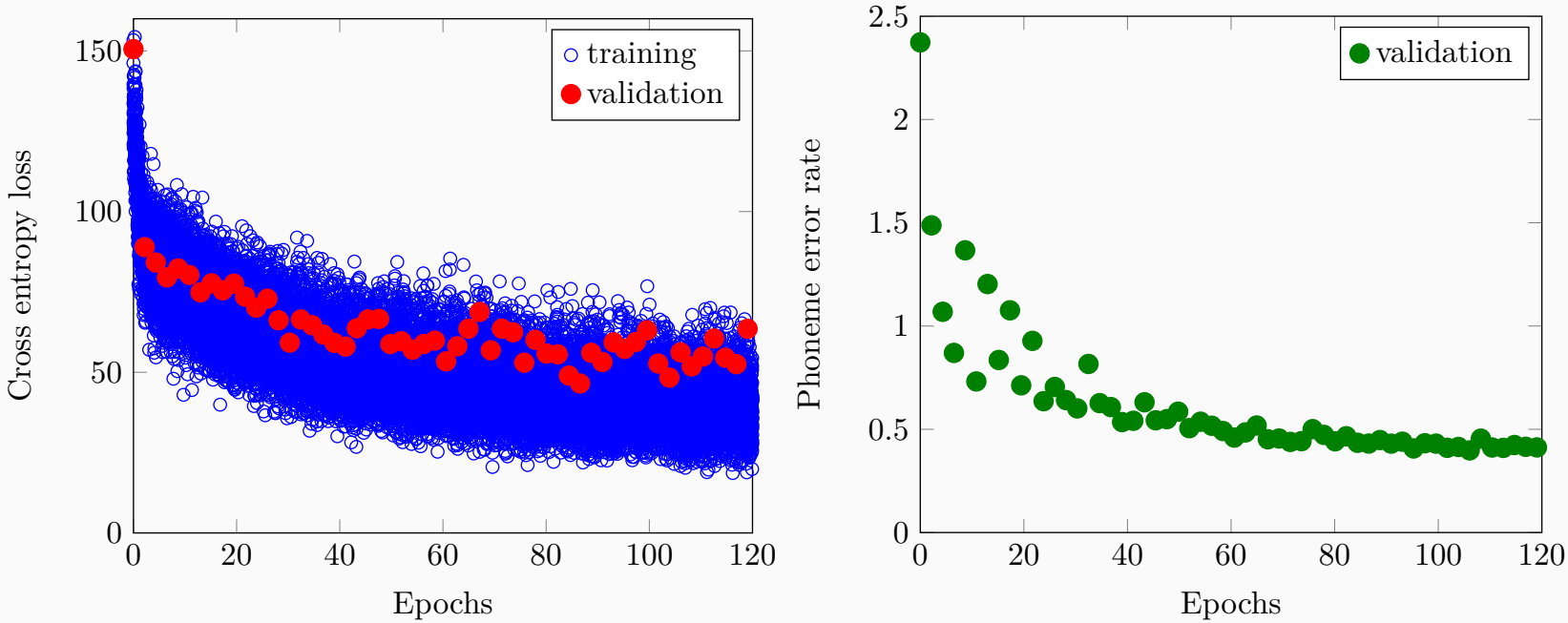

Abbildung 8: Dropout-LAS Ergebnisse.

Zusammenfassung

Zusammenfassung

Experiment		listener		speller				epochs	error
name	reg	dim	layers	dim	net	layers	p reuse		
BLSTM-CTC	$\sigma_i = .65$	64	2	-	-	-	-	10	29%
Listener-CTC	$\sigma_i = .65$	64	2	-	-	-	-	10	26.8%
Greedy-LAS	$\sigma_i = .65$	64	2	128	64	1,2	.7	40	55%
Greedy-LAS	$\sigma_i = .65$	64	2	128	64	1,2	.5	40	54%
Big-Beam-LAS	$p_i = .8, p_h = .6$	128	2	256	128	1,2	.6	40	45%

Tabelle 1: Ausgewählte Parameter und Fehler einiger interessanter Experimente.

Zusammenfassung

- Kleine Erfolge mit re-implementierter LAS-Architektur.
- Verbesserungsmöglichkeiten:
  - Tensorflow hat mittlerweile eine eigene Aufmerksamkeits Funktionen.
  - Mehr Gewichte.
  - Nutzung eines größeren Datensatzes (TIMIT sind nur 700 MB!).

Diskussion

Diskussion

Vielen dank für eure Aufmerksamkeit. Fragen?

Jetzt oder später `moritz@wolter.tech`.

`https://github.com/v0lta/Listen-attend-and-spell`