

Building an end-to-end speech recognizer

Moritz Wolter

01.02.2017

Outline

Introduction

Implementation

Results

Conclusion

Questions

Introduction

The problem

- What is being said?
- How is the data aligned?
- What model architecture?
- What are good hyper-parameters?
- How to ensure generalization?

The LAS-Architecture

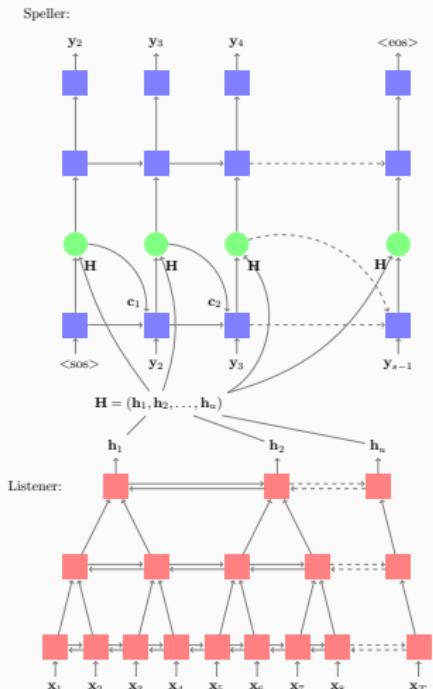


Figure 1: Listener and speller.

The LAS-Equations

Attend and spell cell computations:

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

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

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

AttentionContext computations:

$$e_{i,u} = \phi(\mathbf{s}_i)^T \psi(\mathbf{h}_u), \quad (4)$$

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

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

Implementation

Major obstacles

- Attend and spell cell
- Decoding loop logic
 - greedy decoding
 - works with multiple utterances
 - beam search
 - multiple hypotheses
 - keeps track of hypotheses' probabilities, sequence elements, states, status (done?) and sequence lengths
- Efficiency

Attend and spell cell layout

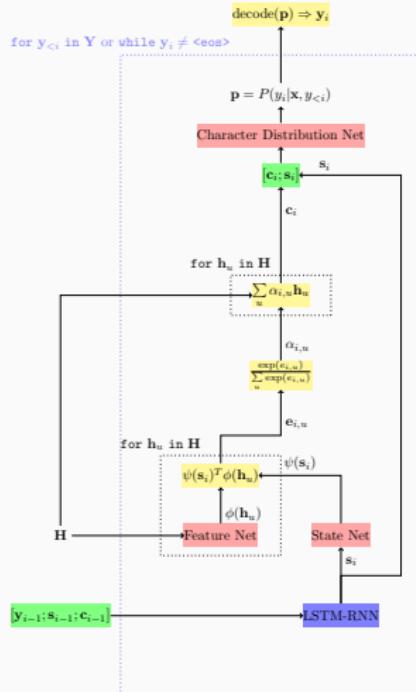


Figure 2: Attend and spell cell flow chart.

Efficiency improvements

- Evaluate the feature net output outside the attend and spell cell.
- Keep track of sequence lengths.
- Zero padding makes it possible to work with fixed size tensors.
- Sequence length information must be used to avoid processing the padding.

Dropout

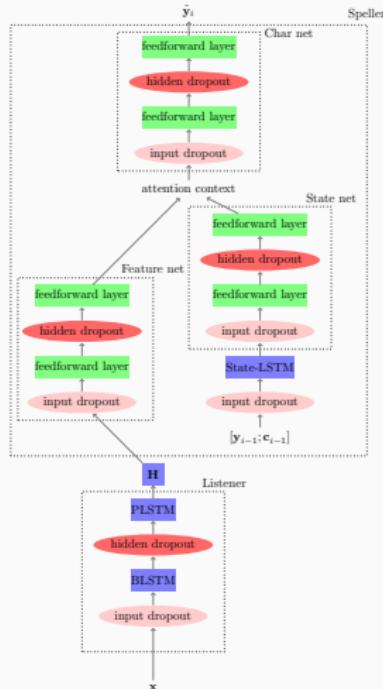


Figure 3: Input dropout - light red. Hidden dropout - dark red.

Results

Experimental settings

- TIMIT Speech corpus.
- learning rate: 0.001.
- batch_size: 16 utterances, Training set total 3696.
- validation set size: 4 utterances.
- test set size: 192 utterances.
- validation every 20 weight updates.

Greedy Decoding

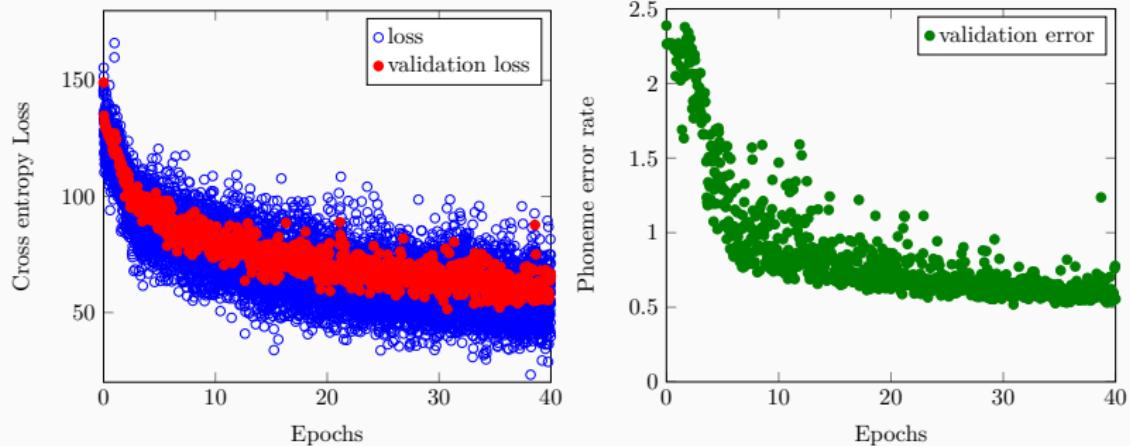
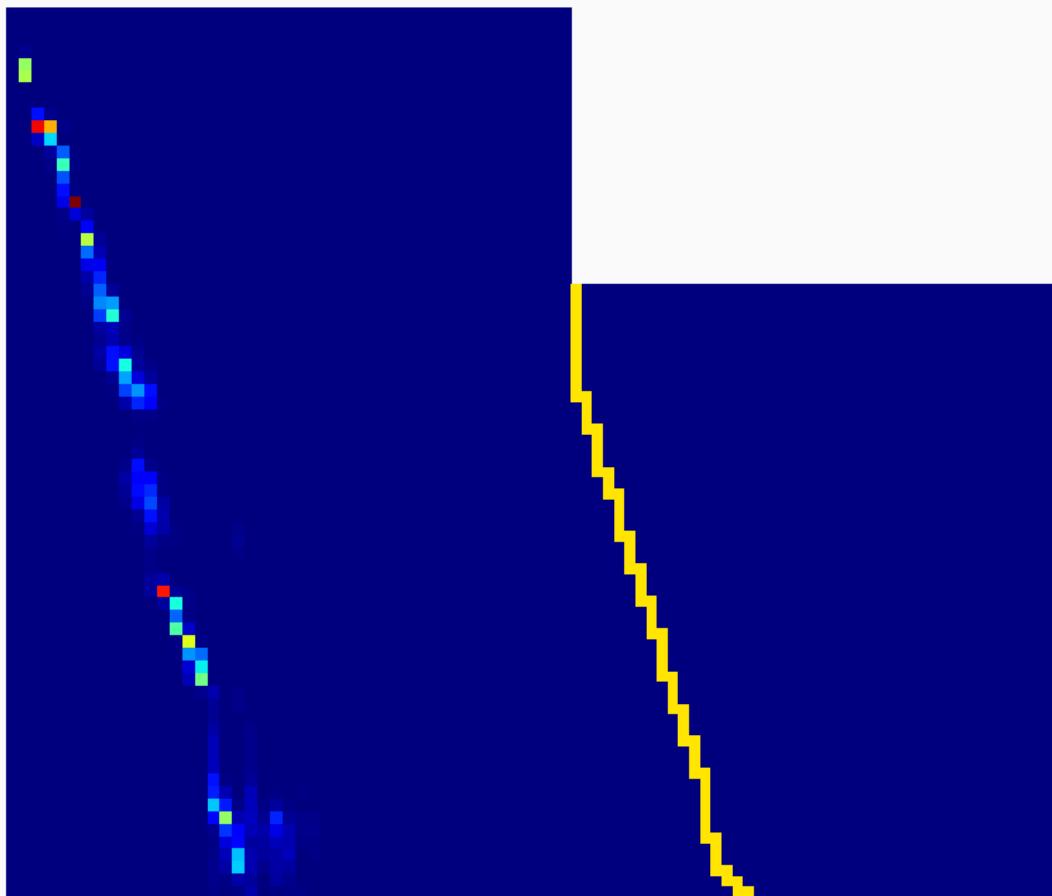


Figure 4: The training progress shown for the full las architecture with greedy decoding, over 40 epochs, network output reuse probability 0.7 and input noise standard deviation 0.65.

Alignment plots



Alignment plots

Target 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>

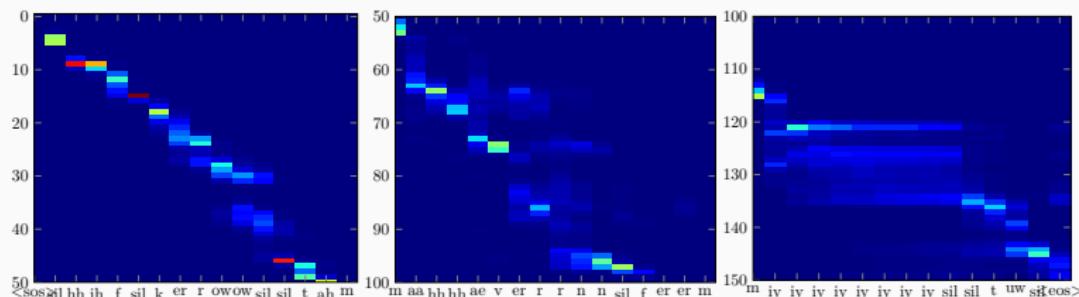


Figure 6: Attention weights α and network output.

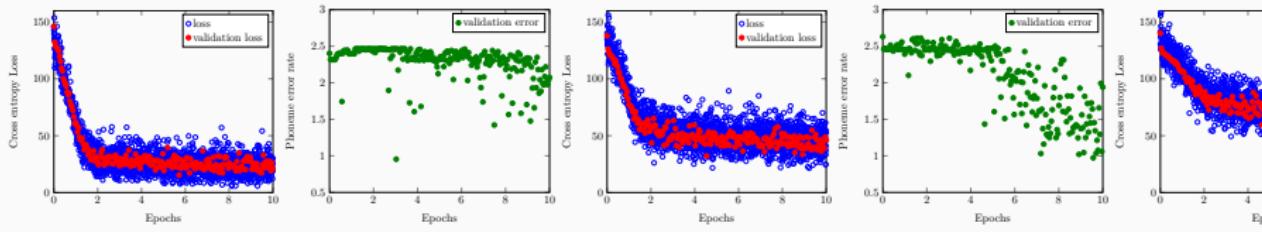


Figure 7: Repetitions of the same experiment with increasing network output reuse probabilities 0.2, 0.4, 0.6, 0.8, one experiment per row.

Dropout las with beam search

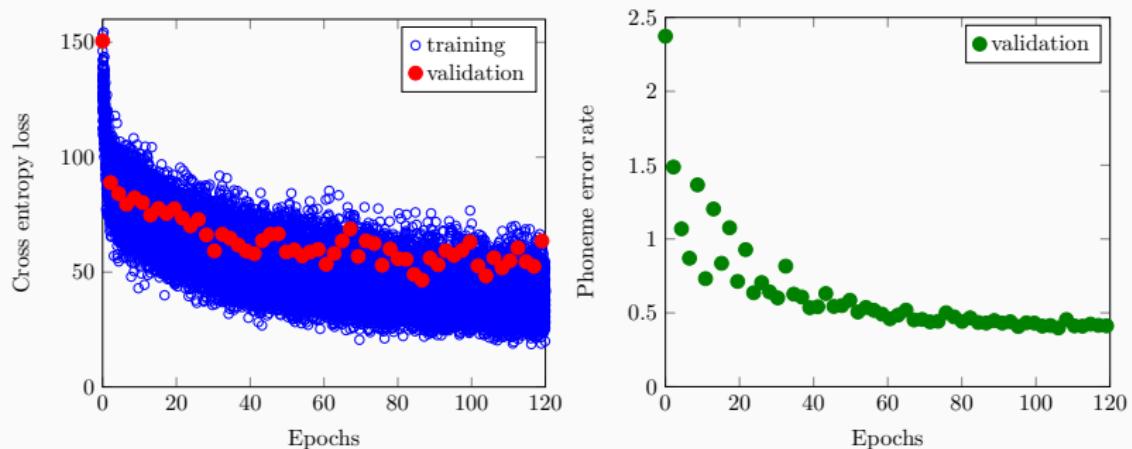


Figure 8: Dropout las results.

Conclusion

Conclusion

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%

Table 1: Selected important experiment parameters and test set phoneme error rate.

Conclusion

- Some success with a reimplemented LAS-architecture.
- Independent validation of beam search and dropout code required.
- Ways to improve:
 - tensorflow's default attention mechanism.
 - run larger networks.
 - try other optimization parameters (learning rate, decay, . . .)

Questions

Questions

Thank's for your attention. Questions?

Now, or later moritz@wolter.tech.