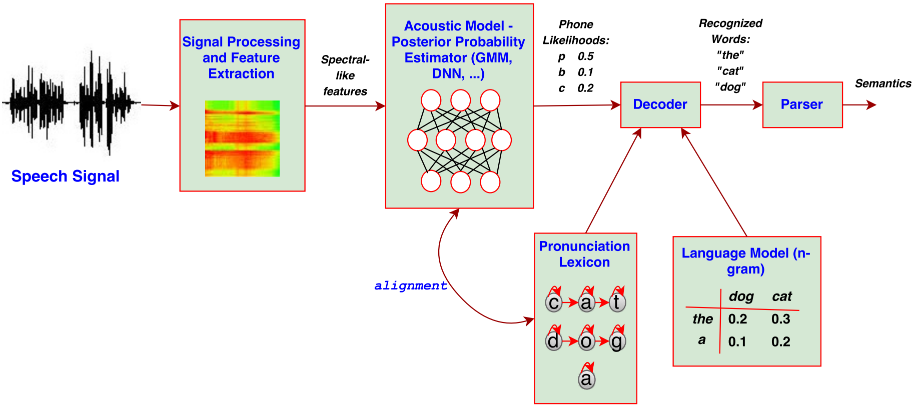
Connectionist Temporal Classification for Robust Speech Recognition Applications

Nadim Ghaddar
École Polytechnique Fédérale de Lausanne, Switzerland

August 29, 2016

Speech Recognition Overview

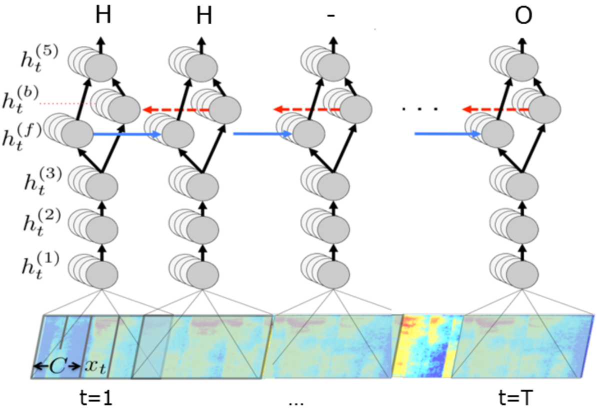


> "There is at least one fundamental difficulty with supervised training of a (purely) connectionist network for continuous speech recognition: a target function must be defined, even though the training is done for connected speech units where the segmentation is generally unknown."

H. Bourlard and N. Morgan, 1994, in *"Connectionist Speech Recognition: A Hybrid Approach"*

HOW CTC?

- Alignment of inputs with outputs are not known ⇒ CTC considers all possible alignments.
- In addition to all label characters, a special blank label (-) is defined



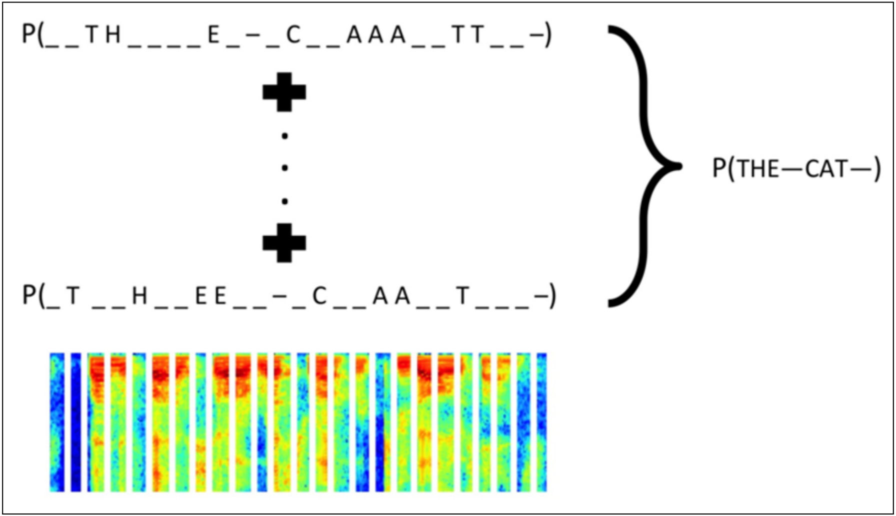
- Let L' = L $\cup$ {blank}
- Let y_k^t: probability of seeing label k at time t
- Define function $\mathcal{B} : \mathrm{L'}^{\mathrm{T}} \to \mathrm{L}^{\leq \mathrm{T}}$, that removes blank labels and consecutive characters
- To find probability of a certain label, sum over all possible alignments:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} \prod_{t=1}^{T} y_{\pi_t}^t$$

- Define cost function:

$$E_{CTC} = -\sum_{(\mathbf{x},\mathbf{l}) \in S} \log p(\mathbf{l}|\mathbf{x})$$

HOW CTC?



Forward-backward Algorithm

- l' is a modified version of the target label sequence l by adding blanks between every other label ("aab" $\rightarrow$ "-a-a-b-")
- Set $V(t, u)$ is defined as:

$$V(t,u) = \left\{\pi \in L'^t : \mathcal{B}(\pi) = \mathbf{l}_{1:u/2}, \pi_t = l'_u\right\}$$

- Forward variables are defined:

$$\alpha(t,u) = \sum_{\pi \in V(t,u)} p(\pi|\mathbf{x}) = \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} y^i_{\pi_i}$$

Forward-backward Algorithm

- l' is a modified version of the target label sequence l by adding blanks between every other label ("aab" $\rightarrow$ "-a-a-b-")
- Set $V(t,u)$ is defined as:

$$V(t,u) = \left\{\pi \in L'^t : \mathcal{B}(\pi) = \mathbf{l}_{1:u/2}, \pi_t = l'_u\right\}$$

- Forward variables are defined:

$$\alpha(t,u) = \sum_{\pi \in V(t,u)} p(\pi|\mathbf{x}) = \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} y^i_{\pi_i}$$

- Finally:

$$p(\mathbf{l}|\mathbf{x}) = \alpha(\mathrm{T}, |\mathbf{l'}|) + \alpha(\mathrm{T}, |\mathbf{l'}| - 1)$$

Forward-backward Algorithm

- Backward variables:

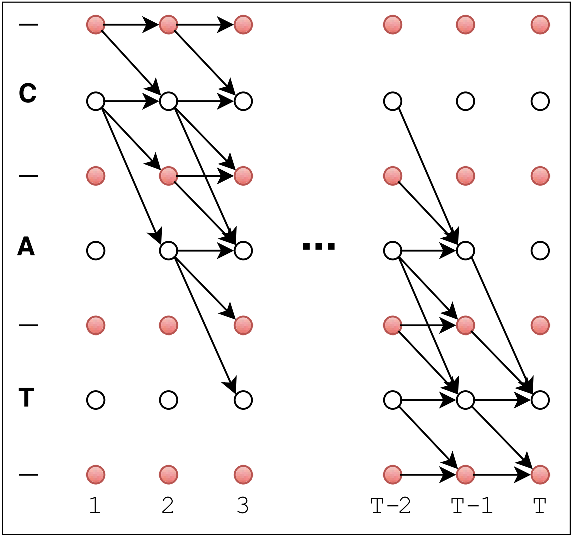
$$\beta(t,u) = \sum_{\pi \in W(t,u)} \prod_{i=1}^{T-t} y_{\pi_i}^{t+i}$$

where:

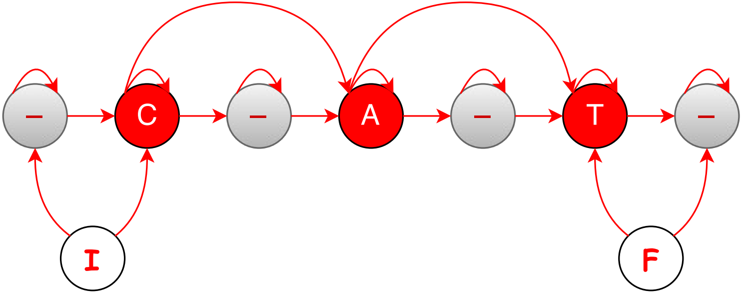
$$W(t,u) = \left\{\pi \in L'^{T-t} : \mathcal{B}(\hat{\pi} + \pi) = \mathbf{l} \; \forall \hat{\pi} \in V(t,u)\right\}$$

- Forward and backward variables at time t can be computed recursively using values at time $t-1 \Rightarrow$ forward-backward algorithm of HMM's

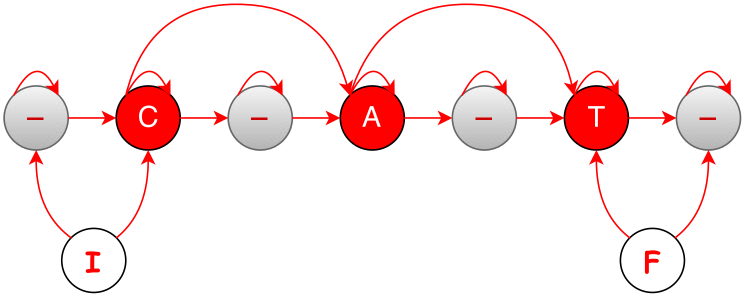
Forward-backward Algorithm



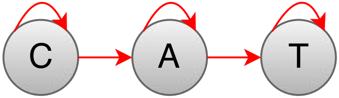
- **CTC Training**



- **CTC Training**



- **DNN/HMM Training**



Forward-backward Algorithm

- By simple calculus, it can be shown that:

$$\alpha(t,u)\beta(t,u) = \sum_{\pi \in X(t,u)} \prod_{t=1}^{T} y^t_{\pi_t} = \sum_{\pi \in X(t,u)} p(\pi|\mathbf{x})$$

where $X(t,u) = \{\pi \in L'^T : \mathcal{B}(\pi) = \mathbf{l}, \pi_t = l'_u\}$

- Therefore, for any t:

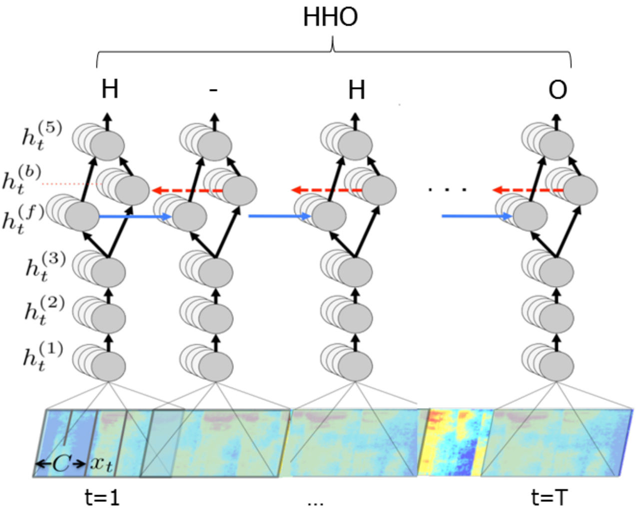
$$p(\mathbf{l}|\mathbf{x}) = \sum_{u=1}^{|\mathbf{l'}|} \alpha(t,u)\beta(t,u)$$

- Back-propagated gradient:

$$\frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y^t_k} = \frac{1}{y^t_k} \sum_{u \in C(\mathbf{l},k)} \alpha(t,u)\beta(t,u), \quad \text{where } C(\mathbf{l},k) = \{u : l'_u = k\}$$

Best-Path Decoding

- Most likely path corresponds to most likely label



Experimental Setup

- "Deep Speech" Motivation: CTC Training with huge amounts of training data

Dataset	Type	Hours
WSJ	read	80
Switchboard	conversational	300
Fisher	conversational	2000
Baidu	read	5000
		7380

Experimental Setup

- "Deep Speech" Motivation: CTC Training with huge amounts of training data

Dataset	Type	Hours
WSJ	read	80
Switchboard	conversational	300
Fisher	conversational	2000
Baidu	read	5000
		7380

- "Deep Speech" Results: Testing Set of 100 noisy and 100 noise-free utterances (SNR between 2 and 6 dB in noisy samples)

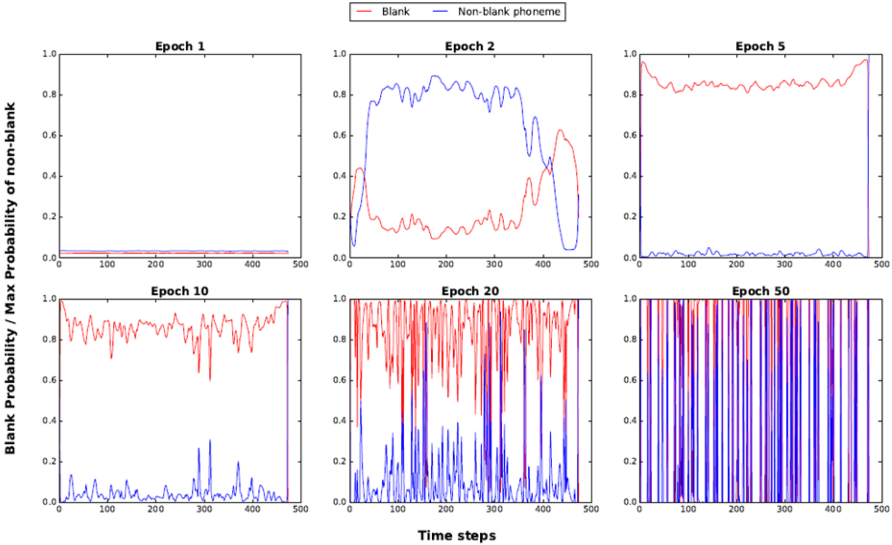
System	Clean	Noisy
Google API	6.64	30.47
Deep Speech	6.56	19.06

Experimental Setup

- Use MFCC features (dimensionality = 13)
- Use TIMIT dataset ($\approx$ 5 hours of speech data)
- Phoneme-level transcriptions (48 phonemes)
- 3696 utterances for training set, 400 utterances for development set, 192 utterances for testing set

- Use Kaldi for DNN/HMM baseline: Fully connected network with 6 hidden layers, 512 neurons each, output layer of size 48
- Use Lasagne for CTC training: 2 B-LSTM layers with 832 cells per layer, 2 fully connected layers with 512 neurons each, output layer of size 49
- Phoneme-error rate as target comparison criterion

CTC Training



Experimental Results

- PER Result over clean TIMIT:

	DNN/HMM	**CTC**
Clean Data	24.9%	26.9%

Robustness

- Got various noise samples (10 seconds long) from Aurora corpus (sounds recorded in car, airport, restaurant, ...)
- Need more noise samples (at least proportional to training data size) in order not the network to learn the noise
- Generate new noise samples from existing ones by mixing
- Same experiments as before for the new "noisy" TIMIT
- Results generated for different SNR values: -10dB, -3dB, 0dB, 3dB, 10dB, 100dB

Experimental Results - 2

- PER Results over noisy TIMIT:

SNR (dB)	DNN/HMM	CTC
-10	52.1%	54.83%
-3	44.2%	47.04%
0	40.3%	43.22%
3	38.1%	41.33%
10	31.1%	34.84%
100	26.5%	32.3%

Benefits of CTC Training

- Context-independent label sequences

Benefits of CTC Training

- Context-independent label sequences
- Single state per target label, plus an optional blank state

Benefits of CTC Training

- Context-independent label sequences
- Single state per target label, plus an optional blank state

  $\Rightarrow$ reduces number of states from thousands of senones to tens of labels

Benefits of CTC Training

- Context-independent label sequences
- Single state per target label, plus an optional blank state

$\Rightarrow$ reduces number of states from thousands of senones to tens of labels
$\Rightarrow$ smaller decoding graph
$\Rightarrow$ faster decoding

Benefits of CTC Training

- Context-independent label sequences
- Single state per target label, plus an optional blank state

  $\Rightarrow$ reduces number of states from thousands of senones to tens of labels
  $\Rightarrow$ smaller decoding graph
  $\Rightarrow$ faster decoding

  $\Rightarrow$ can keep more hypotheses during beam search decoding

Benefits of CTC Training

- Context-independent label sequences
- Single state per target label, plus an optional blank state

  ⇒ reduces number of states from thousands of senones to tens of labels
  ⇒ smaller decoding graph
  ⇒ faster decoding

  ⇒ can keep more hypotheses during beam search decoding

- Same order of decoding complexity as monophone training of GMM/HMMs

Experimental Results - 3

- PER Results over noisy TIMIT:

SNR (dB)	Monophone	CTC
-10	58.2%	54.83%
-3	54.7%	47.04%
0	50.5%	43.22%
3	48.0%	41.33%
10	41.6%	34.84%
100	35.6%	32.3%

Influence of Training Set Size

- Only kind of information given to the network is the *sequence* of target labels and *sequence* of feature vectors
- A lot of these training examples should be given to the network
- Perform same experiments on a subset of TIMIT (namely, 10% of the training examples: 370 utterances)

Experimental Results - 4

- PER Results over a smaller subset of TIMIT:

SNR (dB)	Monophone	DNN/HMM	CTC
-10	64.6%	65.1%	72.42%
-3	59.7%	61.9%	66.18%
0	56.8%	58.3%	64.25%
3	53.2%	54.4%	62.27%
10	47.3%	48.3%	57.73%
100	42.6%	44.1%	53.31%
∞ (Clean)	39.4%	42.7%	49.98%

CTC Robustness

- Main obstacle in speech recognition is the variability of speech features with respect to target labels pronounced (inter- and intra-speaker variations)
- For noisy speech, the variability is higher
- Forced alignments "confuse" the network: some noisy frames are not representative of any target label
- CTC has the option to map "unclear" frames to the blank symbol $\Rightarrow$ less confusion
- Large training set still needed

Conclusions

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- CTC "shines" when large training sets are available, especially for noisy data:

  $\Rightarrow$ As less amount of information is given to the network (no input-output segmentation), more training data should be given for the network to *learn* the alignment.

- Inherent benefits: faster and more scalable decoding

Thank You!