Connectionist Temporal Classification (CTC) and Hybrid NN/HMMs

Théodore Bluche
tb@a2ia.com

RWTH - i6, Aachen August 14, 2014

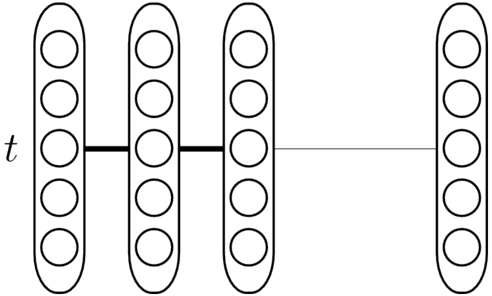
Outline

- Hybrid NN/HMM
  - Forward-Backward Training (Hennebert et al., 1997)
  - RNNs and CTC (Graves et al., 2006)
  - CTC and HMMs
- Experiments around the CTC for Hybrid NN/HMM
  - “HMM” topology
  - Optical model
  - Blank symbol
- Conclusions and Future Work

Neural Networks for Hybrid NN/HMM

Training criteria / Targets

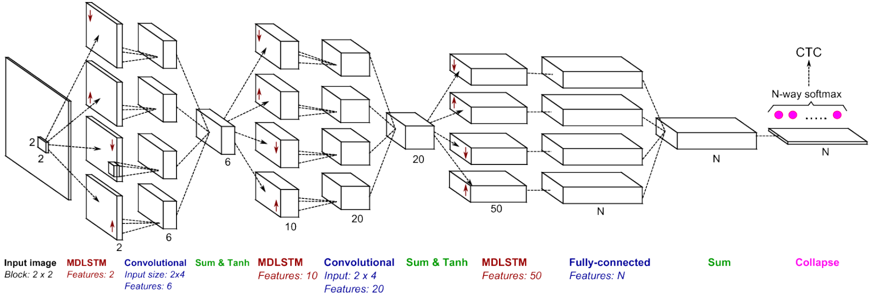
Deep MLP



Framewise		HMM states
Sequence-discriminative (MPE, sMBR)		

$$p\left(q_t | x_{t-\delta_l}^{t+\delta_r}\right)$$

LSTM-RNN



Framewise	HMM states
CTC	Characters (+ blank)

$$p(q_t | x)$$

NN training, Hyb. NN/HMM decoding

Training

$$p(q_t|x_t)$$

The network outputs state probability given input

The training set can be obtained *e.g.* from forced alignments (Viterbi)

Decoding

$$\frac{p(q_t|x_t)}{p(q_t)} \approx \frac{p(x_t|q_t)}{p(x_t)}$$

Outputs are transformed into *pseudo-likelihoods* so that the network can replace the GMM emission model in HMMs

Forward-Backward NN training

Goal - replace a cross-entropy criterion at frame level with one that optimizes the likelihood of the model given the whole input sequence, which does not require prior segmentation

$$p(\lambda|x) \approx p(\lambda) \sum_{q_1,\ldots,q_T} \prod_t p(q_t|x_{t-\delta:t+\delta}, q_{t-1}) \frac{p(q_t|q_{t-1},\lambda)}{p(q_t|q_{t-1})}$$

$$\approx p(\lambda) \sum_{q_1,\ldots,q_T} \prod_t \frac{p(q_t|x_{t-\delta:t+\delta})}{p(q_t)} p(q_t|\lambda)$$

We can use the **forward-backward** algorithm to estimate state posteriors

$$\alpha_t(i) = \frac{p(x_{1:t}, q_t = s_i|\lambda)}{p(x_{1:t})} \qquad \beta_t(i) = \frac{p(x_{t+1:T}|q_t = s_i, \lambda)}{p(x_{t+1:T})}$$

$$\alpha_t(i) = \frac{p(x_t|q_t = s_i)}{p(x_t)} \times \sum_j \alpha_{t-1}(j) p(q_t = s_i|q_{t-1} = s_j) \qquad \beta_t(i) = \sum_j p(q_{t+1} = s_j|q_t = s_i) \frac{p(x_{t+1}|q_{t+1} = s_j)}{p(x_{t+1})} \beta_{t+1}(j)$$

Hennebert, J., Ris, C., Bourlard, H., Renals, S., & Morgan, N. (1997). **Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems.**

Forward-Backward NN training

Goal - replace a cross-entropy criterion at frame level with one that optimizes the likelihood of the model given the whole input sequence, which does not require prior segmentation

$$O = -\log p(\lambda|x) \Rightarrow \frac{\partial O}{\partial u_k^t} = y_k^t - \sum_{i:s_i=k} \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}$$

Assumes the model prior is constant (LM).

In the Viterbi approximation, the posteriors are `1` for states/position in the best alignment, `0` otherwise, and we get the framewise cross-entropy criterion.

$$p(q_t = k|x)$$

Hennebert, J., Ris, C., Bourlard, H., Renals, S., & Morgan, N. (1997). **Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems.**

Connectionist Temporal Classification

Goal - label an unsegmented input sequence of length **T** into a sequence of labels of length **L<T** with a neural network, with no post-processing of the outputs (or a trivial one)

- The possible outputs are **characters**

  → "A B" is a valid labelling. With subunits (like states in HMMs) it would be more difficult to get "A0 A1 A2 B0 B1 B2" and not "A0 B2 A2 …"

- So the only problem to tackle is **L < T** :

  → "AAABB → AB" : doesn't allow to differentiate "AB", "AAB", "ABB", "AABB", …

  → Add a *blank* (or no label) output **#** which should also simplify the problem at the boundaries, where there is no "correct" labeling
  "AAAABBB → AB", "AA##BB# → AB", "A#AA#BB → AAB", …

  → Several output sequences map onto the same labeling

Graves, Alex, et al. "**Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.**" *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

Connectionist Temporal Classification

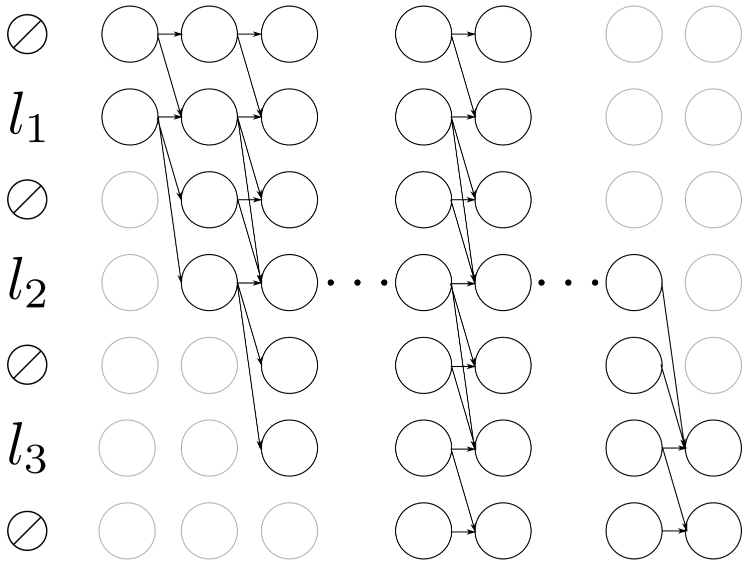
Goal - label an unsegmented input sequence of length **T** into a sequence of labels of length **L<T** with a neural network, with no post-processing of the outputs (or a trivial one)

$$p(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} p(\pi|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} \prod_t y^t_{\pi_t}$$

$$\alpha_t(s) = p(\pi_{1:t} : \mathcal{B}(\pi_{1:t}) = l_{1:s/2}, \pi_t = l'_s|x) = \sum_{\pi:\mathcal{B}(\pi_{1:t})=l_{1:s/2}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}$$

$$\beta_t(s) = p(\pi_{t+1:T} : \mathcal{B}(\pi_{t:T}) = l_{s/2:|l|}, \pi_t = l'_s|x) = \sum_{\pi:\mathcal{B}(\pi_{t:T})=l_{s/2:|l|}} \prod_{t'=t+1}^{T} y^{t'}_{\pi_{t'}}$$

$$O = -\sum_{l,x} \log p(l|x) \Rightarrow \frac{\partial O}{\partial u^t_k} = y^t_k - \sum_{s \in lab(l,k)} \frac{\alpha_t(s)\beta_t(s)}{\sum_{s'} \alpha_t(s')\beta_t(s')}$$



Graves, Alex, et al. "**Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.**" *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

CTC training and HMMs

Decoding

The CTC framework and reduce operation allow to label with the RNN alone ...

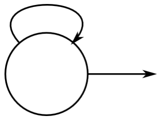
	Rimes	IAM	OpenHaRT
RNN alone	28.5% / 6.8%	35.1% / 10.8%	30.3% / 7.3%
+ LM	12.3% / 3.3%	13.6% / 5.1 %	18.0% / 4.7%

... but the results are much better with a vocabulary and LM

V. Pham, T.Bluche, C. Kermorvant, J. Louradour (2014) ***Dropout improves recurrent neural networks for handwriting recognition.***

We decode the outputs as in standard hybrid NN/HMM, *i.e.* with pseudo, likelihoods, HMM and LM

HMM
(removes duplicates)

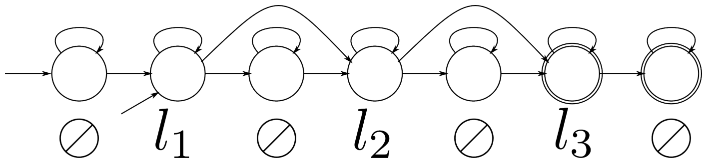


1-state models
for all labels incl.
blank
all trans. prob. = .5

Lexicon
(removes blanks)

word : [#] w [#] o [#] r [#] d [#]

Training



Folded in time, the CTC graph resemble the word transition model described above.

(Hennebert, 1997)

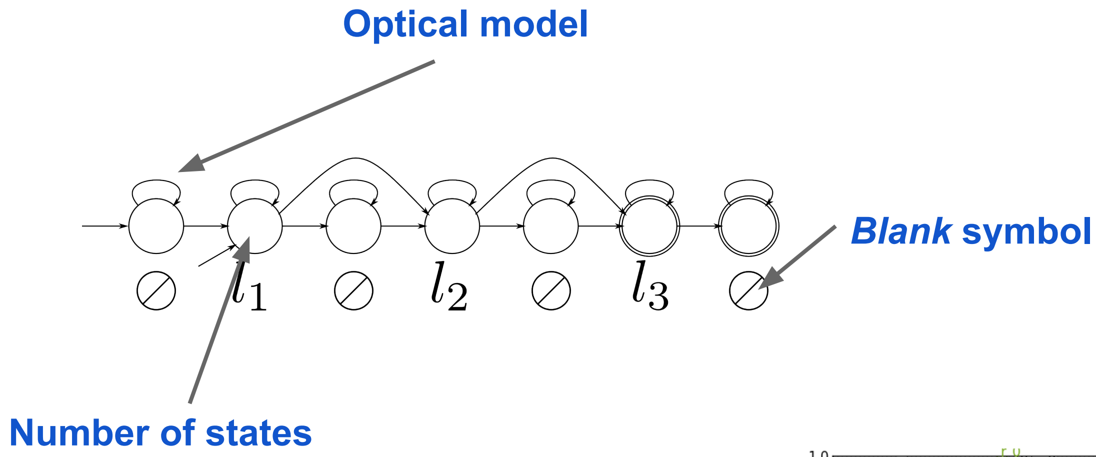
$$\alpha_t(i) = \begin{cases} y^t_{s_i} \sum_{n=0}^{1} \alpha_{t-1}(i-n) \frac{p(q_t=s_i|q_{t-1}=s_{i-n})}{p(s_i)}, & \text{if } s_i = \oslash \text{ or } s_i = s_{i-2}. \\ y^t_{s_i} \sum_{n=0}^{2} \alpha_{t-1}(i-n) \frac{p(q_t=s_i|q_{t-1}=s_{i-n})}{p(s_i)}, & \text{otherwise.} \end{cases}$$

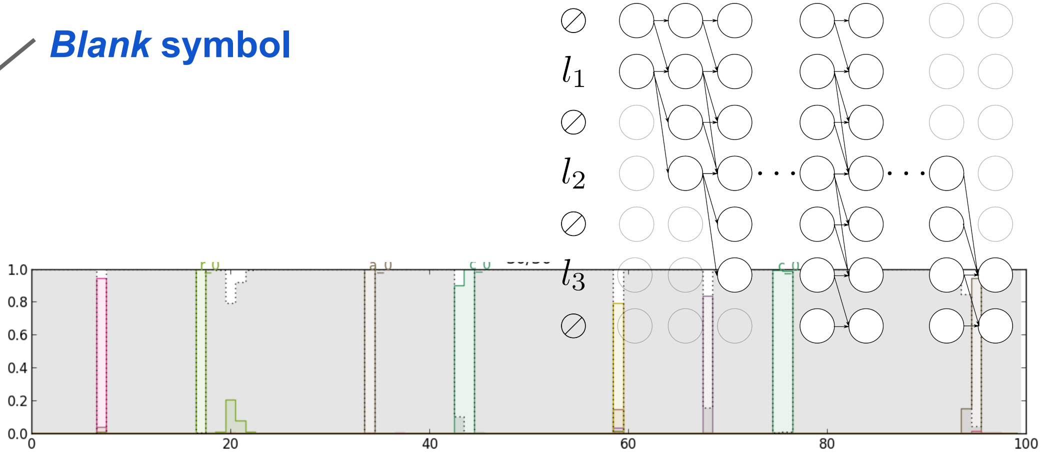
(Graves, 2006)

$$\alpha_t(s) = \begin{cases} y^t_{l'_s} \sum_{n=0}^{1} \alpha_{t-1}(s-n), & \text{if } l'_s = \oslash \text{ or } l'_s = l'_{s-2} \\ y^t_{l'_s} \sum_{n=0}^{2} \alpha_{t-1}(s-n), & \text{otherwise} \end{cases}$$

CTC and HMMs

Since we decode with a HMM and since CTC training is very similar to forward-backward training of hybrids, we can question the aspects of the CTC which are not theoretically required ...





Experimental Setup

- IAM training / validation sets (6,500 lines in training, 976 in validation)
- 3gram LM trained on LOB+Brown+Wellington corpus
- RNN architectures
  - BLSTM-RNNs
  - “smallRNN” : one LSTM layer of 100 units in each direction
  - “bigRNN” : state-of-the-art performing architecture for the baseline
    7 hidden layers with decreasing number of units 200 -> 100
    Subsampling after the first layer
    Training with Dropout and Curriculum training (Pham, 2014), (Louradour, 2014)

Transition model topology

Number of states	With blank	Without blank
1	11.4% / 4.1%	- / -
2	11.8% / 4.0%	16.4% / 6.2%
3	14.2% / 5.2%	14.3% / 5.2%
4	TODO	TODO
5	23.0% / 10.2% (*)	14.9% / 5.7%

WER/CER on IAM dev with "bigRNN" architecture, varying the topology (number of states, blank)

(*) : in the "bigRNN" architecture, there is subsampling, without which the models without blank did not converge to an acceptable result, but with 5 states and blank, it seems to hurt (*see next slide*). 4-state models haven't been trained yet

Topology -- training issues

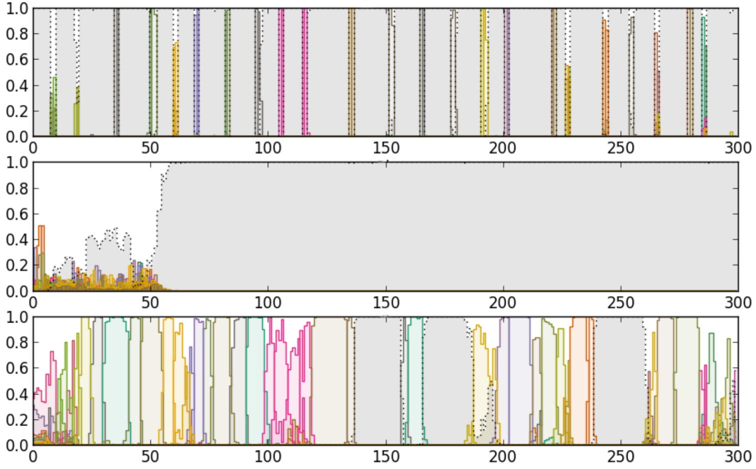
Convergence of the RNN to poor alignments, especially whithout blank

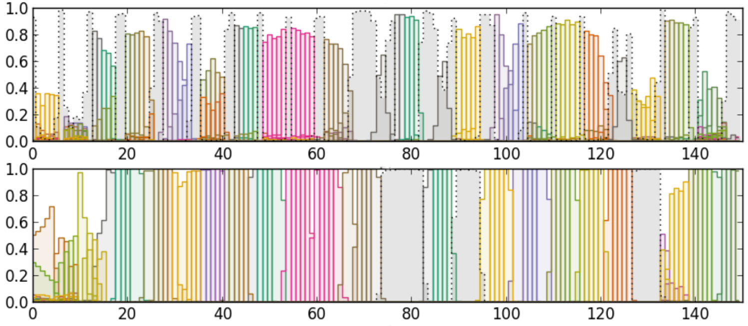
With Blank

Without Blank

Random init.

Framewise pre-training





Sometimes, you don't have enough space for one more state… here 5 states with, …

… and without blank

Optical model and topology

Everything on this slide is trained framewise (no CTC) : GMM with ML criterion and Viterbi realignments, DNN with Xent with GMM alignments first, then realignment, RNN with Xent and DNN alignments

Topology	GMM	DNN	"smallRNN"
1 state + blank	30.1% / 18.0%	19.5% / 9.0%	18.7% / 8.2%
2 states	25.7% / 15.5%	18.0% / 7.6%	17.7% / 7.5%
2 states + blank	23.5% / 12.6%	16.5% / 6.5%	15.9% / 6.1%
3 states	20.8% / 10.7%	14.8% / 5.8%	15.2% / 5.6%
5 states	16.7% / 7.7%	13.4% / 4.8%	14.2% / 5.1%

soon experimented... models with 4 states, missing "with blank" models

Topology and training method

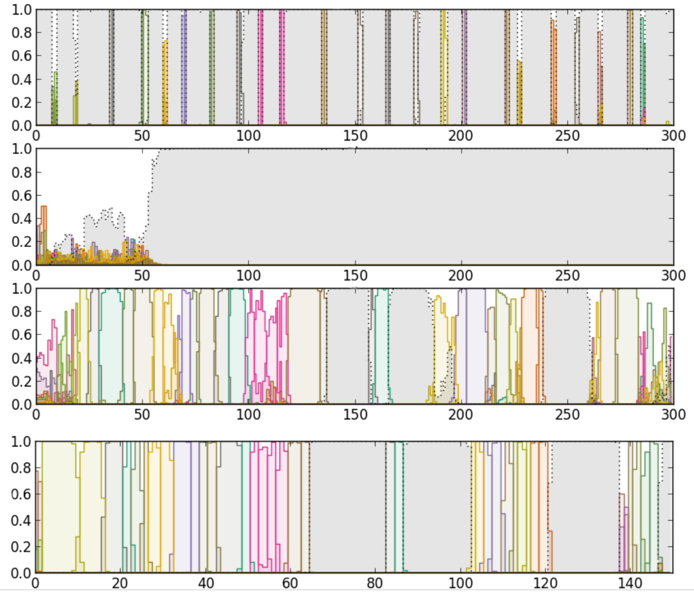
Topology	FRAMEWISE	CTC (Framewise init.)	CTC (Random init.)
1 state + blank	18.7% / 8.2%	13.1% / 4.9%	13.4% / 5.1%
2 states	17.7% / 7.5%	19.3% / 8.0%	*didn't converge*
2 states + blank	15.9% / 6.1%	13.9% / 5.0%	13.7% / 5.2%
3 states	15.2% / 5.6%	16.5% / 6.1%	*didn't converge*
5 states	14.2% / 5.1%	13.7% / 5.0%	*didn't converge*

WER/CER on IAM dev with "smallRNN" architecture, varying the topology and the training criterion

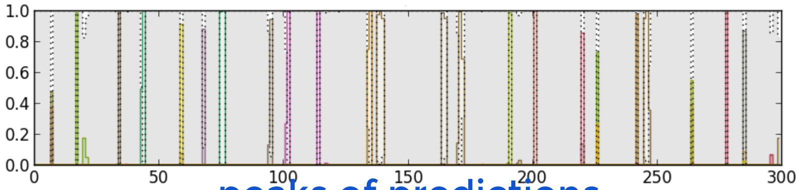
experiments running as I speak... same experiments with DNNs

What is it with the blank symbol?

- It **helps** in most of experiments (different optical model, with CTC and framewise training) … without it, CTC sometimes doesn't converge to something acceptable
- It **sometimes hurts** (e.g. with CTC, subsampling and many states)

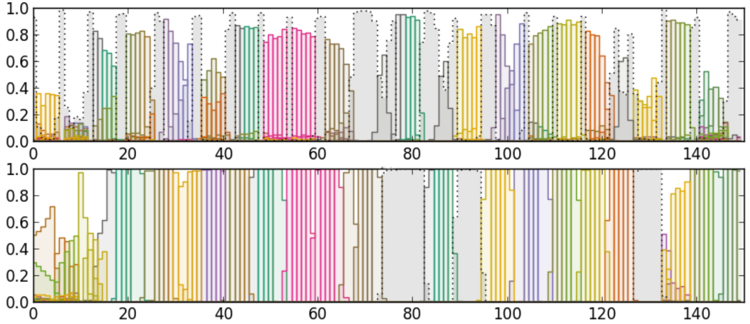


alignment issues

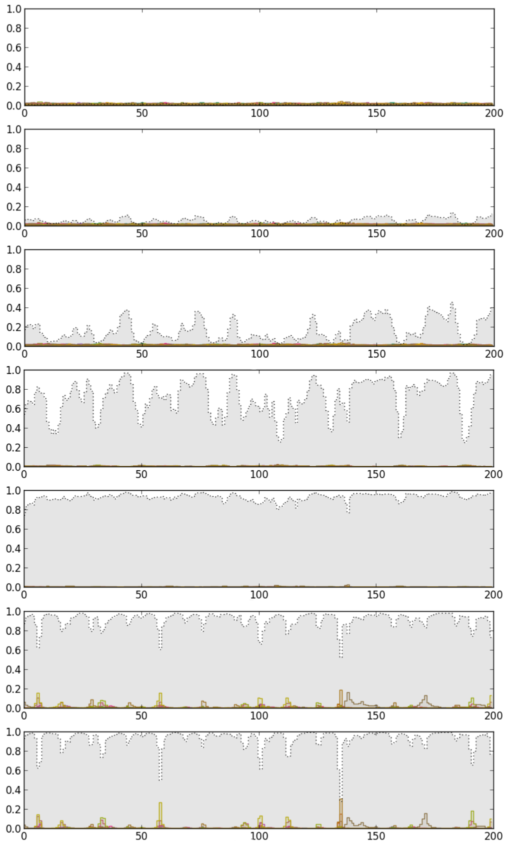


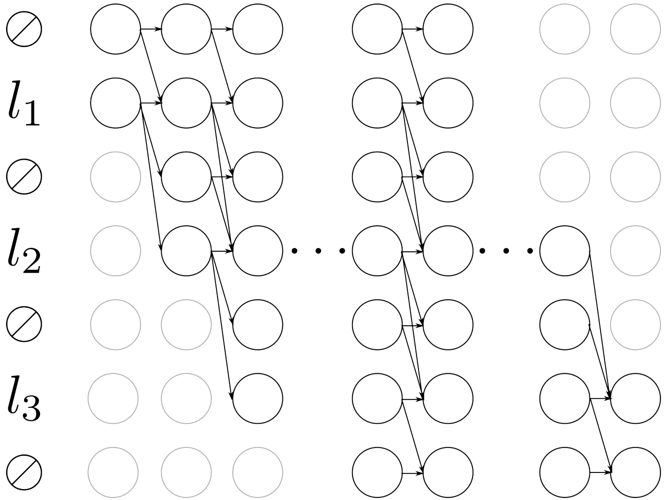
peaks of predictions

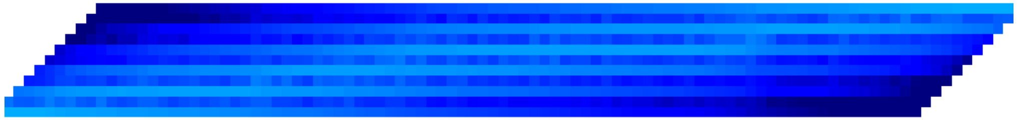
system with blank reaches size limits



Why do we observe peaks?

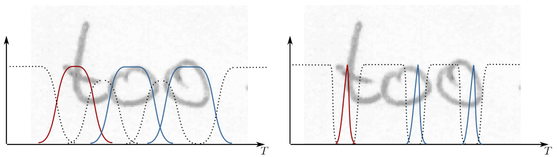


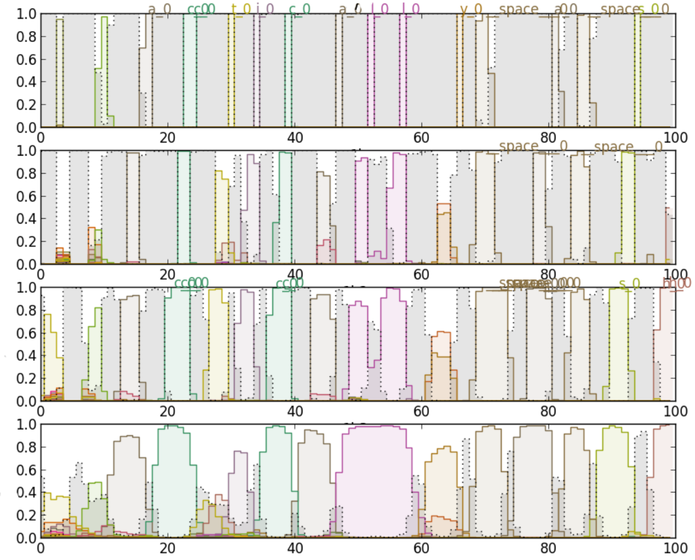




Trying to output longer predictions

- In training we repeat the labels in the CTC graph (like *n* states sharing the same distribution), but we keep one state per label at validation/decoding time
- The goal is to increase the label (not blank) posterior probability at any given time



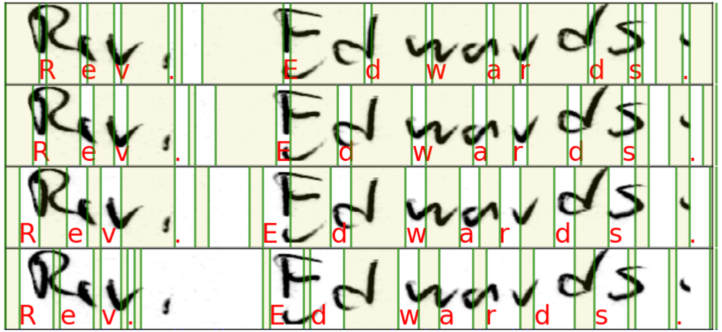


The obtained predictions are indeed longer…

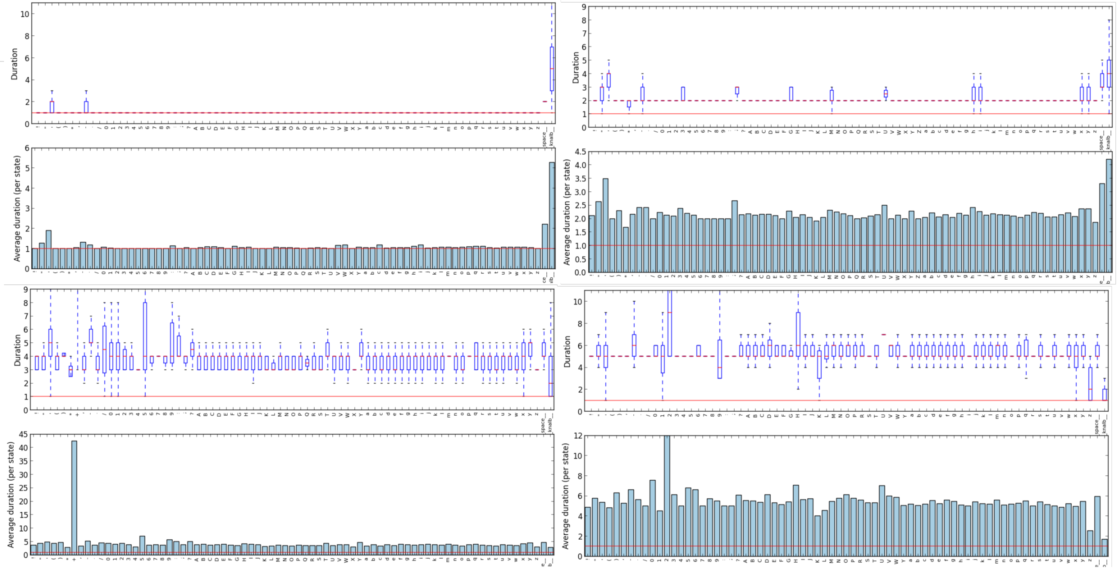
… but the results are not better

Num. Repeats	CTC	CTC - CER	WER / CER
1	0.1512	9.2%	11.4% / 4.1%
2	0.2466	9.5%	11.5% / 4.2%
3	0.3677	10.4%	12.9% / 4.7%
5	0.9429	30.1%	26.6% / 13.5%

Trying to output longer predictions



The predictions are longer but not more helpful for segmentation or localisation…



Even if the blank's length decreases in proportion to label lengths,

it seems like the RNNs have learnt precise durations (remember that there is still only one state in decoding)

Role of the blank symbol (part 1)

The sharp predictions

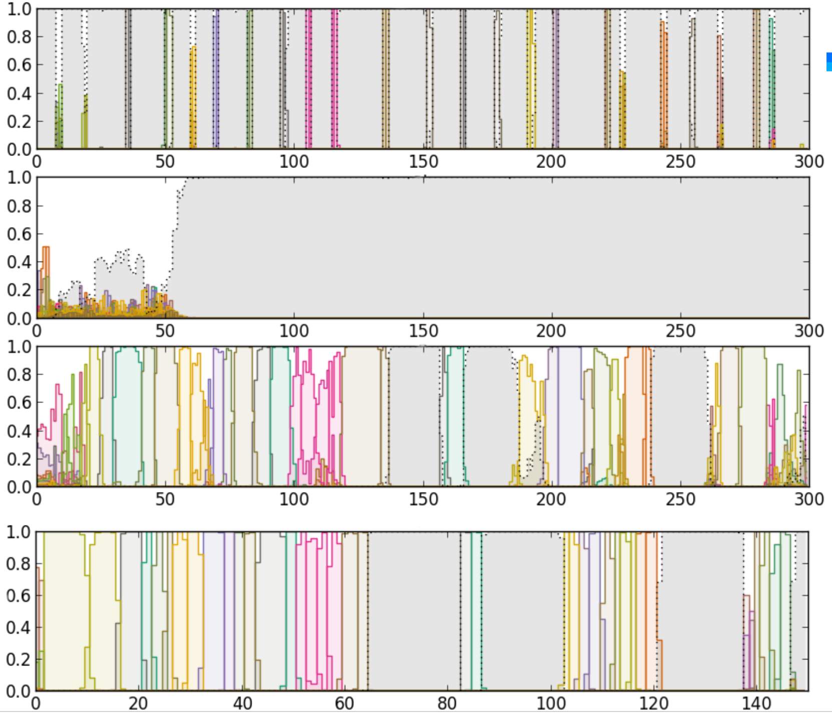
- In **training**, the structure of the CTC gives blank high posterior probabilities (because it is in many valid paths) and it becomes advantageous regarding the training criterion, to output long sequences of highly likely blanks
  → So it is **the CTC trying to have the network predict peaks**, but certainly the properties of RNNs make that learning possible

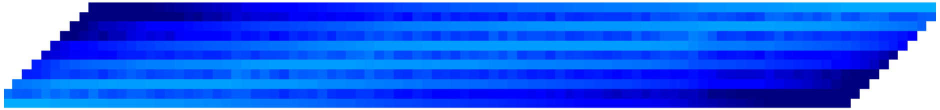
- In **decoding**, the sharper the predictions, the better the results
  Certainly because the **cost to make an edit is limited to one (of many) timesteps** given that all words will be the same in all "blank" segments
  (nb: we use beam search, and it is likely that we keep more alternative in a given beam)

- Note that sharp predictions is **not the original purpose of the blank**, although it's certainly its more important contribution to good results.

By the way… with several states or training with repetitions, when the blank is present in CTC training, the RNN learns to output predictions of exact duration, and the optimal optical scale is always 1/duration *… it's more or less true also for GMM-HMM (average duration 12 → optical scale 1/12)*

Blank and CTC alignment





With blank, I never had problem in CTC

Without blank, it happened that a suboptimal solution is learnt
(often predict whitespace everywhere but start/end)

Some solutions which worked:

Framewise initialization
(here with alignment from DNNs, but probably uniform alignments would do)

Subsampling
(it is more or less like adding states, so be careful, remember the problem described previously)

Role of the blank symbol (part 2)

Alignment during CTC training

- In **training**, the structure of the CTC makes blank provide a kind or “soft uniform segmentation”, which might help the network figure out where to make its predictions

- Without blank, when the input sequences are long and we do not have a prioris about what the segmentation should look like, we often learn suboptimal alignments/segmentations

Its described purpose in (Graves, 2006)

- i.e. to separate two consecutive and identical labels, and to model the input between relevant parts (cores of characters), that is, more or less a garbage label.

Topology	FRAMEWISE	CTC
2 states - no blank	17.7% / 7.5%	19.3% / 8.0%
2 states + blank	15.9% / 6.1%	13.9% / 5.0%

Conclusions and Future Work

Conclusions

- RNNs trained with the CTC structure fit well in the Hybrid/HMM framework
- CTC is actually a simplification of hybrids forward-backward training
- The CTC and blank work especially well together

Topology	FRAMEWISE	CTC
2 states - no blank	17.7% / 7.5%	19.3% / 8.0%
2 states + blank	15.9% / 6.1%	13.9% / 5.0%

- The particular 1 state with blank (baseline), with CTC training has numerous advantages

Future Work

- Fill holes in the result tables, in particular for presence of blanks and training criteria
- Train **DNNs with the CTC** criterion and the different architectures explored in this presentation (in progress)
- Train NNs with **sequence-discriminative criteria** (*e.g.* MMI, MPE). See if the improvements are goods for RNNs / CTC-trained networks, compared to those observed for framewise-trained DNN *(see paper below for DNNs)*

Théodore Bluche, Hermann Ney, Christopher Kermorvant (2014) ***A Comparison of Sequence-Trained Deep Neural Networks and Recurrent Neural Networks Optical Modeling for Handwriting Recognition.*** In International Conference on Statistical Language and Speech Processing (SLSP). *(accepted)*

Thank you!

Théodore Bluche
tb@a2ia.com