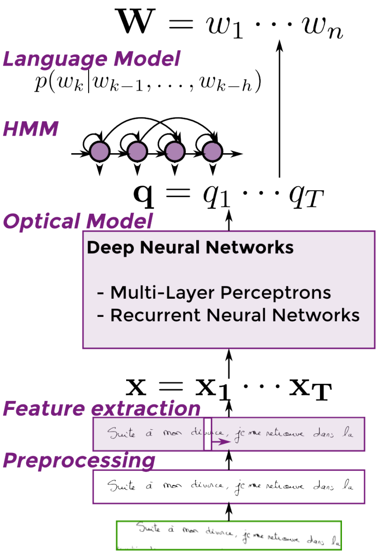
International Conference on Document Analysis and Recognition, Nancy

Framewise and CTC Training of Neural Networks for Handwriting Recognition

Théodore Bluche, Hermann Ney, Jérôme Louradour, Christopher Kermorvant

August 25, 2015

Handwriting Recognition with Hybrid NN/HMM Systems



Recognition

Sequence of observations $\xrightarrow{\text{Optical Model}}$ Sequence of characters $\xrightarrow{\text{Language Model}}$ Sequence of words

$$\mathbf{x} = \mathbf{x_1} \cdots \mathbf{x_T} \qquad \mathbf{C} = c_1 \cdots c_m \qquad \mathbf{W} = w_1 \cdots w_n$$

Recognition

Sequence of observations —Optical Model→ Sequence of characters —Language Model→ Sequence of words

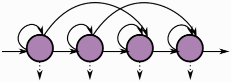
$\mathbf{x} = \mathbf{x}_1 \cdots \mathbf{x}_T$ $\quad \mathbf{C} = c_1 \cdots c_m$ $\quad \mathbf{W} = w_1 \cdots w_n$

GMM → Sequence of states → HMM →

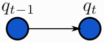
$\mathbf{q} = q_1 \cdots q_T$

Hidden Markov Models (HMMs)

$c_k \equiv$



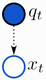
Transition model $P(q_t | q_{t-1})$

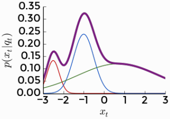


Handles the sequential aspect of the reading task

Emission model $p(x_t | q_t)$

Explains the observations



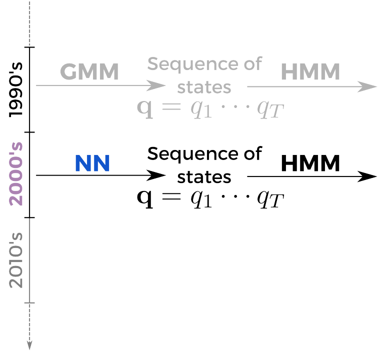


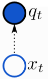
e.g. in (Bianne-Bernard, 2011; Kozielski et al., 2012, 2014)

Recognition

Sequence of observations —Optical Model→ Sequence of characters —Language Model→ Sequence of words

$\mathbf{x} = \mathbf{x_1} \cdots \mathbf{x_T}$ $\quad$ $\mathbf{C} = c_1 \cdots c_m$ $\quad$ $\mathbf{W} = w_1 \cdots w_n$





Neural Networks

$P(q_t|x_t)$

- More classical models in pattern recognition
- Predicts the state from the observation

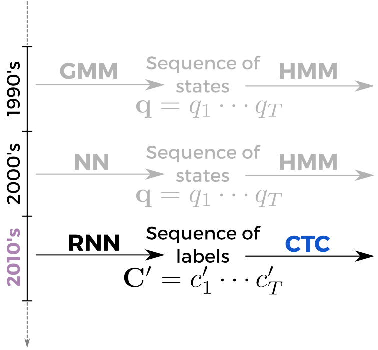
Hybrid NN/HMMs (Bourlard & Morgan, 1994)

e.g. in (Dreuw et al., 2011; Espana-Boquera et al., 2011; Doetsch et al., 2014), with 1-2 hidden layer NNs

Recognition

Sequence of observations —Optical Model→ Sequence of characters —Language Model→ Sequence of words

$\mathbf{x} = \mathbf{x}_1 \cdots \mathbf{x}_T$ $\quad$ $\mathbf{C} = c_1 \cdots c_m$ $\quad$ $\mathbf{W} = w_1 \cdots w_n$



Neural Networks
handling the
sequential aspect

The neural network:

- looks at the whole observation sequence (length T)
- predicts the whole character sequence (length $m \leq T$)

Connectionist Temporal Classification (CTC; Graves et al. (2006))

e.g. in (Strauß et al., 2014; Moysset et al., 2014; Pham et al., 2014)

Observations

- We are interested in the hybrid (R)NN/HMM framework where the network predicts HMM states
- Some works in the '90s for integrated NN/HMM training
- Today, mostly framewise training from forced alignments, or CTC with RNNs

Questions

$\longrightarrow$ What are the **differences bewteen framewise, CTC, and integrated training** ?

$\longrightarrow$ Can we apply **CTC to other neural nets** than RNNs ?

$\longrightarrow$ Can we use the **CTC paradigm with other targets than characters** and blank (e.g. HMM states) ?

$\longrightarrow$ What is the **role of the blank symbol** in CTC? How and when does it help ?

Overview

Neural Network Training for Hybrid NN/HMM

Integrated NN/HMM training

- In the hybrid NN/HMM approach, the Gaussian likelihoods $p_{gmm}(x_t|q_t)$ are replaced by the scaled NN posteriors: $\frac{p_{nn}(q_t|x_t)}{p(q_t)}$.
- An HMM is a segmentation-free approach, and training methods like Baum-Welch may be applied to consider all possible segmentations :

$$\alpha_t(s) = \frac{p(q_t=s|x_t)}{p(s)} \times \sum_r \alpha_{t-1}(r) p(q_t = s | q_{t-1} = r)$$
$$\beta_t(s) = \sum_r \frac{p(q_{t+1}=r|x_{t+1})}{p(r)} p(q_{t+1} = r | q_t = s) \beta_{t+1}(r)$$
$$p(q_t = s | \mathbf{x}, \lambda) = \frac{\alpha_t(s)\beta_t(s)}{\sum_r \alpha_t(r)\beta_t(r)}$$

- Senior & Robinson (1996); Yan et al. (1997): first with hard alignments, then with forward-backward-computed soft estimates
- Konig et al. (1996); Hennebert et al. (1997): similar formulation.

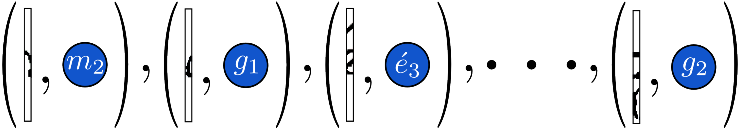
Integrated training cost function

$$E_{hmm} = -\log \sum_{\mathbf{q}} \prod_t \frac{P(q_t|x_t)}{P(q_t)} P(q_t|q_{t-1})$$

- Bengio et al. (1992); Haffner (1993): MMI loss to train the whole system.

Framewise Cross-Entropy Training

1. Compute the forced alignments of the frame sequence with the HMM of the correct word sequence
   $\longrightarrow$ labeled dataset of frames $\mathcal{S} = \{(x_t, q_t)\}$



2. Train the network to classify each frame individually

Cross-entropy cost function:

$$E_{xent} = -\sum_{(x_t,q_t)\in\mathcal{S}} \log P(q_t|x_t)$$

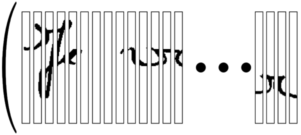
Evaluation

Frame Error Rate (FER%)

$$\frac{\text{\# incorrectly classified frames}}{\text{\# of frames}}$$

Connectionnist Temporal Classification Training (CTC)

1. Use the dataset of frame sequence, with character sequence targets $\mathcal{S} = \{(\mathbf{x}, \mathbf{c})\}$



J e _ v o ... r), ...

2. Train to predict the character sequence $\mathbf{c}$ directly
   - NN outputs = characters + $\oslash$
   - Mapping $\mathcal{B} : a\ a \oslash \oslash b\ b \oslash\ b\ a \mapsto abba$

CTC cost function:

$$E_{ctc} = -\sum_{(\mathbf{x},\mathbf{c})\in\mathcal{S}} \log P(\mathbf{c}|\mathbf{x})$$

with

$$P(\mathbf{c}|\mathbf{x}) = \sum_{\mathbf{q}\in\mathcal{B}^{-1}(\mathbf{c})} P(\mathbf{q}|\mathbf{x}) = \sum_{\mathbf{q}\in\mathcal{B}^{-1}(\mathbf{c})} \prod_t P(q_t|\mathbf{x})$$

Evaluation

NN - Character Error Rate (NN-CER%)

$$\frac{\text{edit distance between reference and recognition}}{\text{\# of reference characters}}$$

(Graves et al., 2006)

Summary of NN training strategies

	Training cost	Outputs
Framewise cross-entropy (MLPs)	$-\log \prod_t P(q_t \vert x_t)$	HMM states (5-6 / character)
CTC (RNNs) (Graves et al., 2006)	$-\log \sum_{\mathbf{q}} \prod_t P(q_t \vert \mathbf{x})$	Characters and blank label $\oslash$

Summary of NN training strategies

	Training cost	Outputs
Framewise cross-entropy (MLPs)	$-\log \prod_t P(q_t \mid x_t)$	HMM states (5-6 / character)
CTC (RNNs) (Graves et al., 2006)	$-\log \sum_{\mathbf{q}} \prod_t P(q_t \mid \mathbf{x})$	Characters and blank label $\oslash$
HMM training (NN/HMM) (Hennebert et al., 1997)	$-\log \sum_{\mathbf{q}} \prod_t \frac{P(q_t \mid x_t)}{P(q_t)} P(q_t \mid q_{t-1})$	HMM states (5-6 / character)

Comparison of Framewise and CTC training of MLPs and RNNs

Training strategies of Hybrid NN/HMM

CTC = HMM training, without transition/prior probabilities (zeroth-order model), and with specific outputs for standalone NN recognition ($\approx$ one HMM state / char. $+$ blank)
$\Longrightarrow$ CTC could be applied with different HMM topologies, to other kinds of NN than RNN (e.g. MLPs)

Training strategies of Hybrid NN/HMM

CTC = HMM training, without transition/prior probabilities (zeroth-order model), and with specific outputs for standalone NN recognition ($\approx$ one HMM state / char. $+$ blank)
$\Longrightarrow$ CTC could be applied with different HMM topologies, to other kinds of NN than RNN (e.g. MLPs)

CTC = Cross-entropy training + forward-backward to consider all possible segmentations
$\Longrightarrow$ we can compare the training strategies, see the effect of forward-backward, with different HMM topologies (number of HMM states / char.)

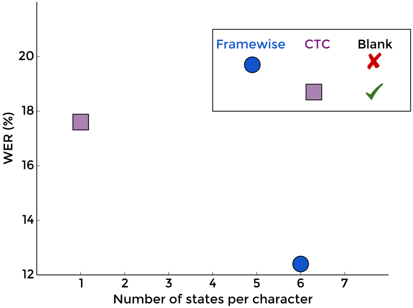
Experimental Setup

Goal: study the differences and dynamics of training methods

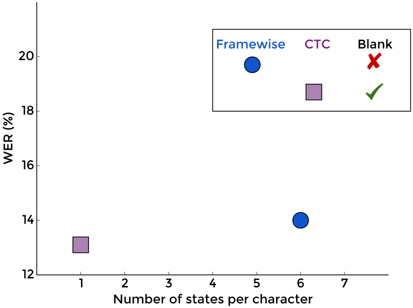
- **IAM database** : 6.5k lines for training, results reported on validation set of 976 lines
- **Image pre-processing**
  - skew and slant correction (Buse et al., 1997)
  - contrast enhancement
  - height normalization
- **Feature extraction**
  - Sliding window of 3px scanned left-to-right with no overlap
  - Extraction of 56 geometrical and statistical features (Bianne-Bernard, 2011)
- **Hybrid NN/HMM systems** using scaled NN posteriors $p(q_t|x_t)/p(q_t)$
- **Trigram language model** with vocabulary of 50k words (trained on LOB, Brown and Wellington corpora), ppl 298, OOV 4.3%
- **FST-based decoder** of the KALDI toolkit
- **Small Neural Nets** for fast experiments
  - MLP with 2 sigmoid hidden layers of 1,024 units
  - BLSTM-RNNs with one hidden layer of 100 LSTM units in each direction

Framewise and CTC

MLPs



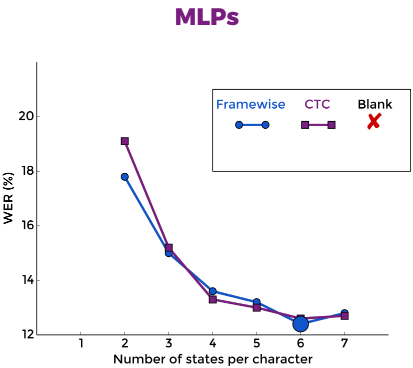
RNNs

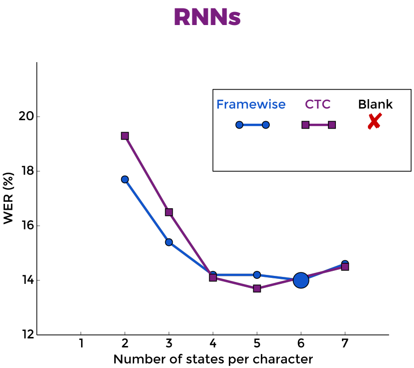


$\longrightarrow$ CTC works well with RNNs, not so much with MLPs

(MLP: 2x1024, ± 5 frames - RNN: 1x100)

Framewise and CTC



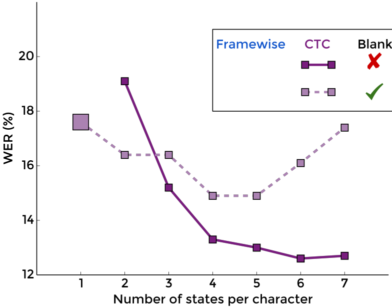


$\longrightarrow$ Forward-backward aspect does not improve the results, and is worse with too few states

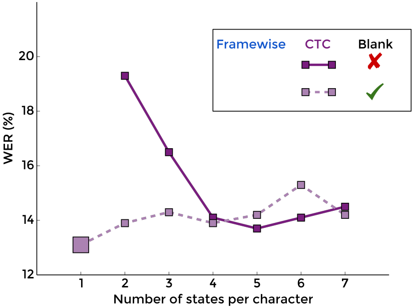
(MLP: 2x1024, ± 5 frames - RNN: 1x100)

Framewise and CTC

MLPs



RNNs

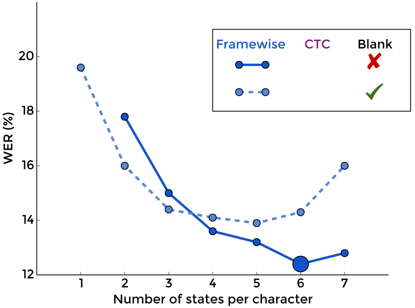


$\longrightarrow$ The blank symbol only helps with a few states for CTC training, ...

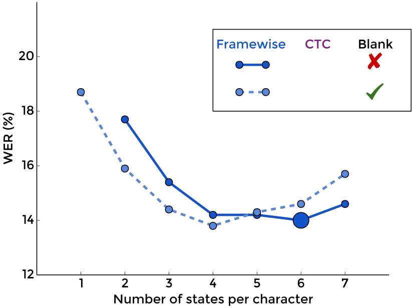
(MLP: 2x1024, ± 5 frames - RNN: 1x100)

Framewise and CTC

MLPs



RNNs

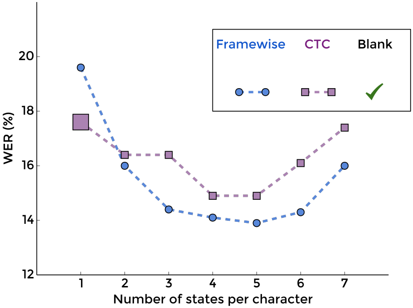


$\longrightarrow$ ... and for framwise training too, although not as much as adding a state to the character models

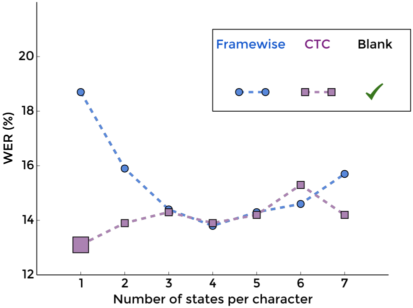
(MLP: 2x1024, ± 5 frames - RNN: 1x100)

Framewise and CTC

MLPs



RNNs

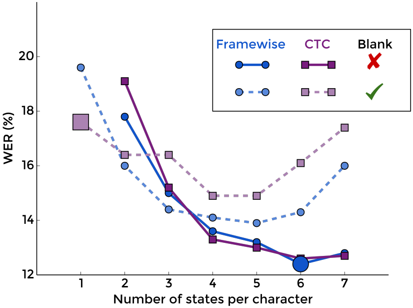


$\longrightarrow$ Forward-backward with blank does not improve so much the results except with only a few states

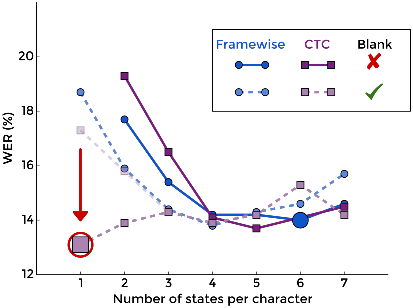
(MLP: 2x1024, ± 5 frames - RNN: 1x100)

Framewise and CTC

MLPs



RNNs

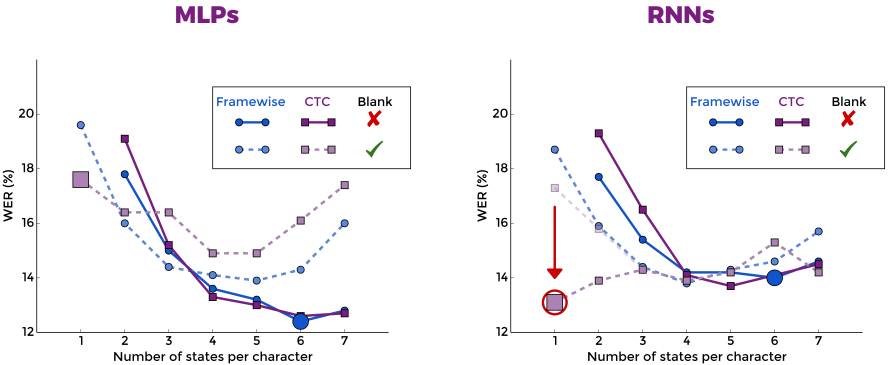


$\longrightarrow$ CTC+blank, with one-state models, is especially suited to RNNs

(MLP: 2x1024, ± 5 frames - RNN: 1x100)

Framewise, CTC and Blank: Summary

Complete systems (with LM; WER%)



⟶ CTC works well with RNNs, not so much with MLPs

⟶ Summation aspect does not improve the results, except for RNN+blank

⟶ The blank symbol only helps with a few states

⟶ CTC+blank, with one-state models, is especially suited to RNNs

(MLP: 2x1024, ± 5 frames - RNN: 1x100)

The Intriguing Blank Symbol in CTC

The Intriguing Blank Symbol in CTC

The blank symbol...

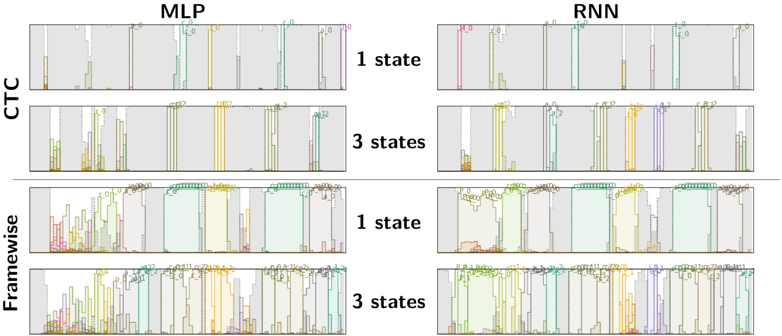
- is necessary for good results with a few states
- from (Graves et al., 2006): required for the mapping of the output to transcription + modelling inter-characters
- ... but otherwise uninformative,
- not helpful with more states
- and has a big impact on what the net's outputs look like! (next slides)

$\longrightarrow$ so what with this symbol? What is the behaviour of the systems with it, and how can it help?

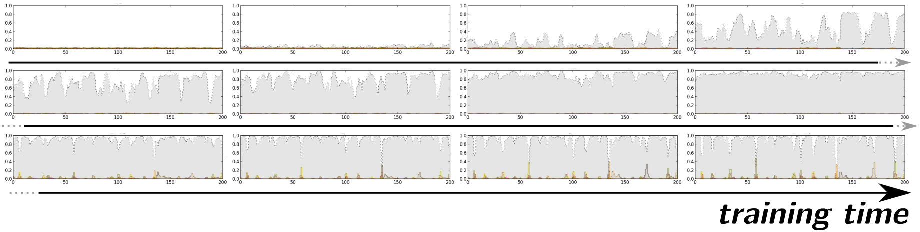
nb. this kind of not-a-character/junk/garbage symbol can be found in other works, e.g. (Tay et al., 2001; Rashid et al., 2012; Elagouni et al., 2012)

Framewise vs. CTC Outputs with Blank

- x-axis = time
- y-axis = predicted probability
- gray = blank symbol
- color = characters



CTC - Evolution of Outputs during Training



- x-axis = time
- y-axis = predicted probability
- gray = blank symbol
- color = characters

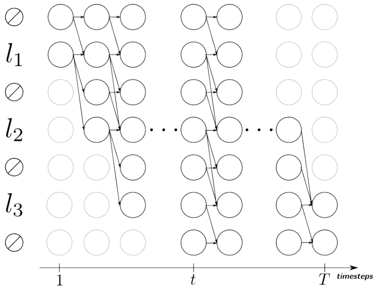
CTC Training

$$p(\mathbf{L}|\mathbf{x}) = \sum_{\mathbf{q}} \prod_{t=1}^{T} p_{nn}(q_t|\mathbf{x})$$

$$\alpha_t(l'_s) = p_{nn}(q_t = l'_s|\mathbf{x}) \sum_{n=0}^{k} \alpha_{t-1}(l'_{s-n})$$

$$\beta_t(l'_s) = \sum_{n=0}^{k} p_{nn}(q_{t+1} = l'_{s+n}|\mathbf{x})\beta_{t+1}(l'_{s+n})$$

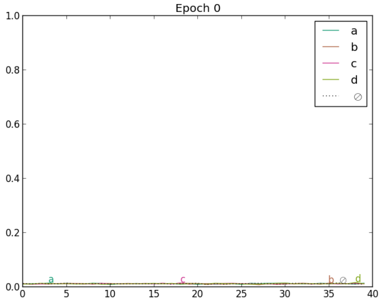
($k = 1$ if $l'_s = \oslash$ or $l'_s = l'_{s-2}$ and 2 otherwise)



$$\frac{\partial E}{\partial a_k^t} = p_{nn}(q_t = k|x_t) - \sum_{s:\mu(s)=k} \frac{\alpha_t(s)\beta_t(s)}{\sum_r \alpha_t(r)\beta_t(r)}$$

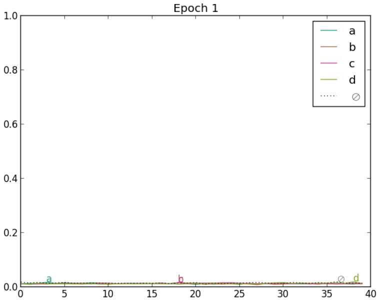
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



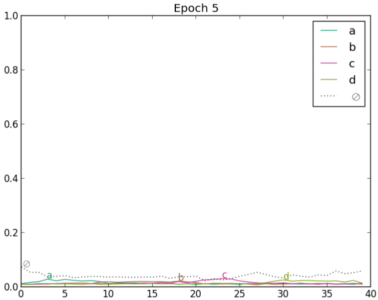
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



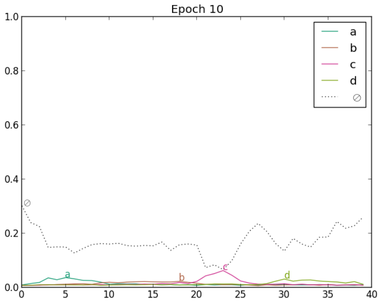
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



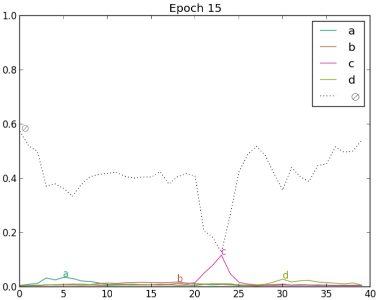
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



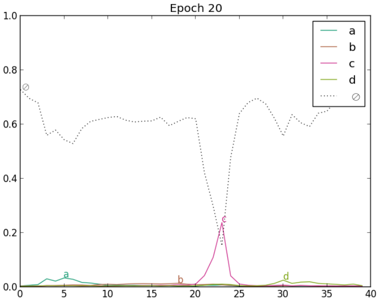
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



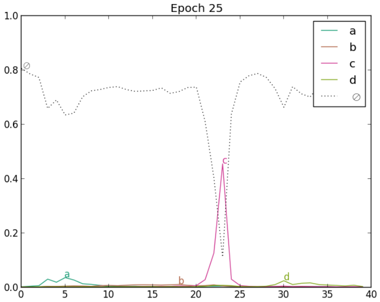
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



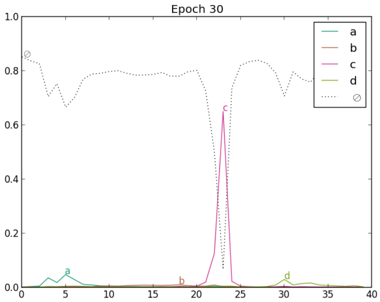
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



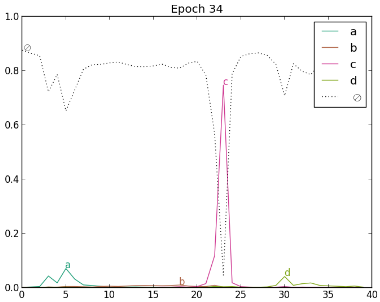
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



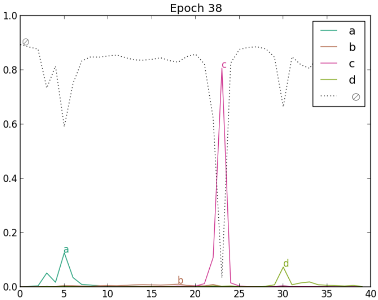
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



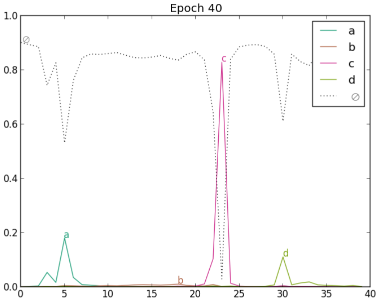
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



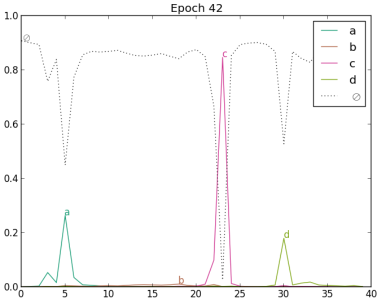
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



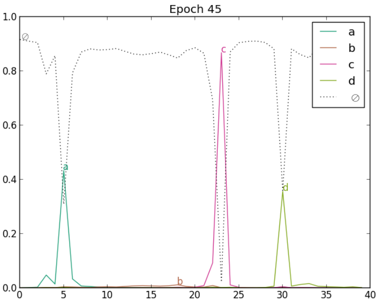
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



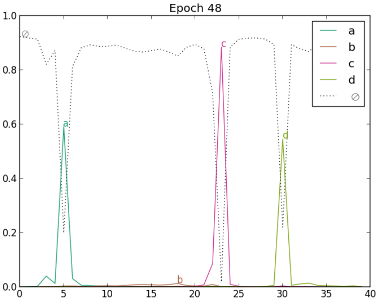
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



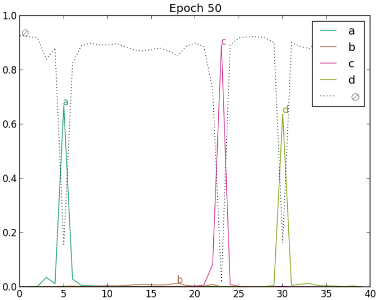
CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)

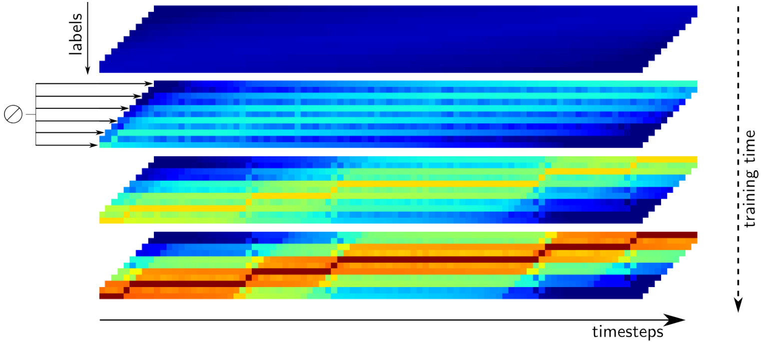


CTC Training, toy example

This behaviour **is due to CTC, not RNNs**.
Toy example: gradient descent on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)

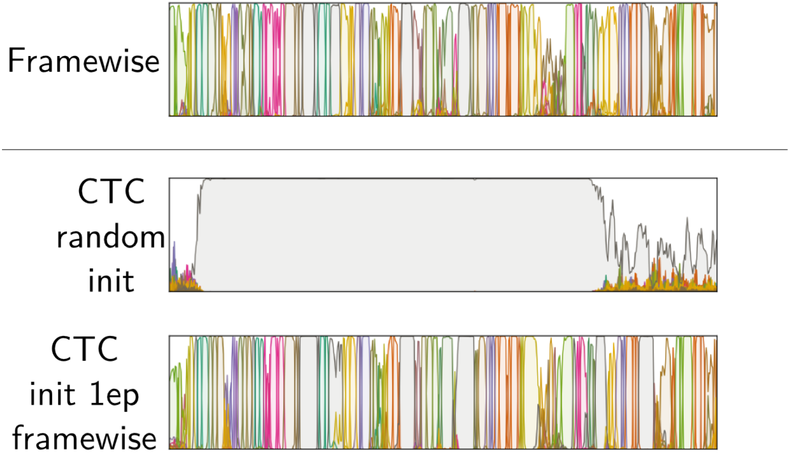


CTC - Why Training Results in Peaks of Character Predictions



CTC - What Happens without Blank

(gray is now the whitespace label)



The Possible Advantage of the Blank Label

For training

By unbalancing the output distribution towards blanks, it prevents from alignment issues in early stages of training - especially for whitespaces - when the net's outputs are not informative

For decoding

The blanks are uninformative and shared by all word models. Peaked and localized character predictions make corrections less costly (only one frame has to be changed to correct a substitution/deletion/insertion)

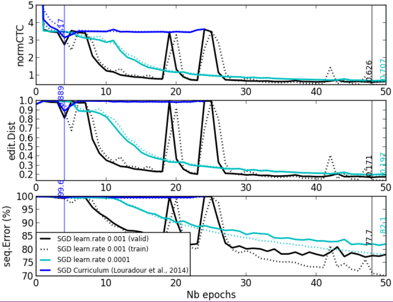
= faster / keeps more hypotheses for fixed beam

(Optimal optical scale in decoding was 1 with CTC, and $1/(avg.char.length)$ for framewise or no-blank training)

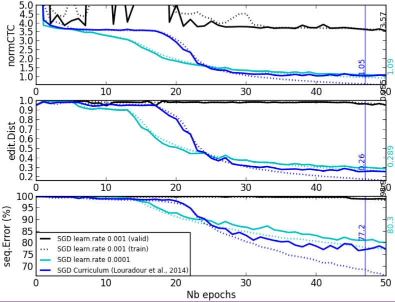
Knowing why blank quickly overwhelms the prediction sequence, what could we do?

- We "waste" some time at the beginning learning only uninformative blanks
- Sometimes we observe a plateau, and it takes some time before the network start predicting actual characters
- Curriculum learning (Louradour & Kermorvant, 2014) or smaller learning rates help sometimes but the problem does not disappear

MAURDOR HWR-FR



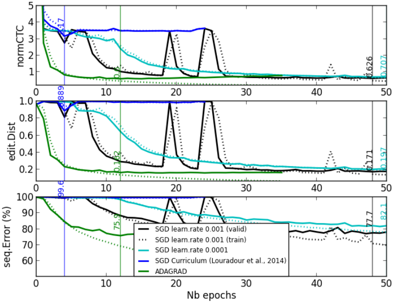
MAURDOR HWR-EN



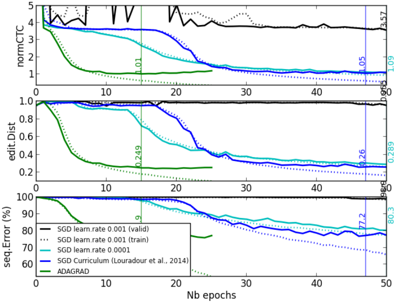
Knowing why blank quickly overwhelms the prediction sequence, what could we do?

- Adaptive per-parameter learning rates (ADAGRAD; Duchi et al. (2011)) to progressively give less importance to the error signal coming from the blank label.

MAURDOR HWR-FR

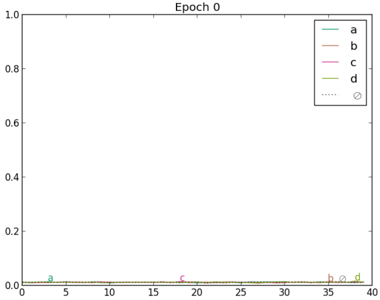


MAURDOR HWR-EN



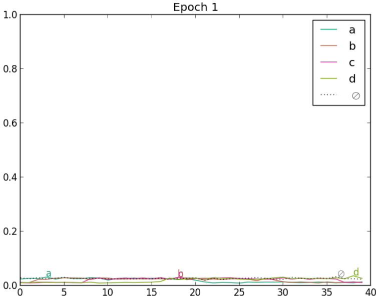
CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



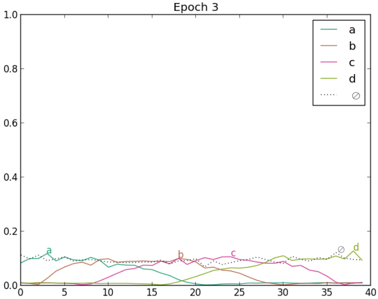
CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



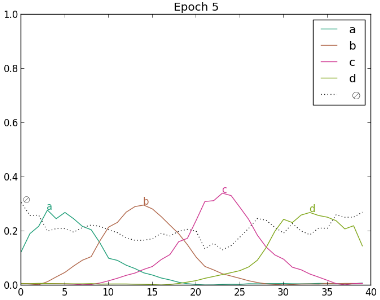
CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



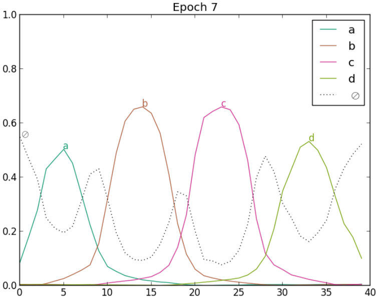
CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



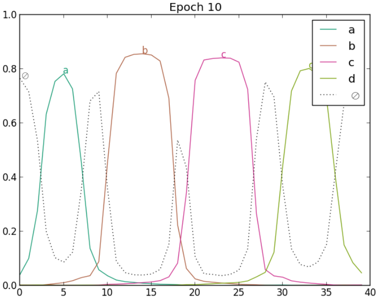
CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



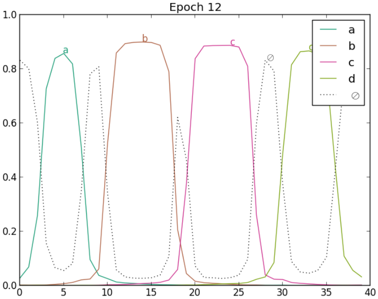
CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



CTC Training, toy example (2)

Toy example: gradient descent **with ADAGRAD** on the outputs of the network with CTC cost function (i.e. assume that the net is able to learn perfectly)



Conclusion

Conclusion

We studied the **CTC training** algorithm, its relation to framewise and HMM training, and **the role of the blank symbol** in the CTC.

- CTC training is similar to:
  - framewise training but sums over all possible alignments
  - NN/HMM training but without transition probabilities and state priors
- It is not limited to one state per character + blank in the hybrid NN/HMM framework, but it is only interesting in that setup
- CTC+Blank **works especially well with RNNs** (because it asks the net to produce peaked and localized character predictions)
- It seems to give advantages for finding alignments during training and for efficient decoding

Future work: add transition probabilities and state priors to CTC training

Thank you for your attention!

tb@a2ia.com

References

Bengio, Y., De Mori, R., Flammia, G., & Kompe, R. (1992). Global optimization of a neural network-hidden Markov model hybrid. Neural Networks, IEEE Transactions on, 3(2), 252--259.

Bianne-Bernard, A.-L. (2011). Reconnaissance de mots manuscrits cursifs par modèles de Markov cachés en contexte. Ph.D. thesis, Telecom ParisTech.

Bourlard, H., & Morgan, N. (1994). Connectionist speech recognition: a hybrid approach Chapter 7, vol. 247 of The Kluwer international series in engineering and computer science: VLSI, computer architecture, and digital signal processing. Kluwer Academic Publishers.

Buse, R., Liu, Z. Q., & Caelli, T. (1997). A structural and relational approach to handwritten word recognition. IEEE Transactions on Systems, Man and Cybernetics, 27(5), 847--61.

Doetsch, P., Kozielski, M., & Ney, H. (2014). Fast and robust training of recurrent neural networks for offline handwriting recognition. (pp. --).

Dreuw, P., Doetsch, P., Plahl, C., & Ney, H. (2011). Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained gaussian HMM: a comparison for offline handwriting recognition. In Image Processing (ICIP), 2011 18th IEEE International Conference on, (pp. 3541--3544). IEEE.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12, 2121--2159.

Elagouni, K., Garcia, C., Mamalet, F., & Sébillot, P. (2012). Combining multi-scale character recognition and linguistic knowledge for natural scene text ocr. In Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on, (pp. 120--124). IEEE.

Espana-Boquera, S., Castro-Bleda, M. J., Gorbe-Moya, J., & Zamora-Martinez, F. (2011). Improving offline handwritten text recognition with hybrid HMM/ANN models. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(4), 767--779.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In International Conference on Machine learning, (pp. 369--376).

Haffner, P. (1993). Connectionist speech recognition with a global MMI algorithm. In EUROSPEECH.

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

Konig, Y., Bourlard, H., & Morgan, N. (1996). Remap: Recursive estimation and maximization of a posteriori probabilities-application to transition-based connectionist speech recognition. Advances in Neural Information Processing Systems, (pp. 388--394).

Kozielski, M., Doetsch, P., Hamdani, M., & Ney, H. (2014). Multilingual Off-line Handwriting Recognition in Real-world Images. (pp. 1--1).

Framewise vs. CTC - Nets alone

Framewise -- Label Classification Error (frame level)								
	States	**1**	**2**	**3**	**4**	**5**	**6**	**7**
MLP	No blank		23.8	24.7	25.8	26.2	28.2	29.3
	Blank	17.1	18.8	20.8	22.0	23.2	25.4	28.5
RNN	No blank		14.4	15.4	16.3	17.2	19.6	20.7
	Blank	11.3	12.8	14.2	15.0	16.0	19.0	22.2
CTC -- Label Edit Distance (sequence level)								
	States	**1**	**2**	**3**	**4**	**5**	**6**	**7**
MLP	No blank		77.0	53.8	44.4	39.6	34.8	32.6
	Blank	18.5	18.9	21.8	26.1	23.9	22.9	24.0
RNN	No blank		23.6	19.0	17.7	16.6	15.6	15.8
	Blank	9.2	10.7	11.5	11.6	12.2	13.0	13.0

Framewise vs. CTC - Net+LM

	States	Without blank: Framewise	Without blank: CTC	With blank: Framewise	With blank: CTC
MLP	1	-	-	19.6 / 9.0	17.6 / 7.4
	2	17.8 / 8.2	19.1 / 8.5	16.0 / 6.3	16.4 / 6.7
	3	15.0 / 6.1	15.2 / 6.1	14.4 / 5.5	16.4 / 6.5
	4	13.6 / 5.3	13.3 / 4.9	14.1 / 5.2	14.9 / 5.6
	5	13.2 / 4.8	13.0 / 4.5	13.9 / 5.2	14.9 / 5.6
	6	**12.4** / 4.6	12.6 / **4.3**	14.3 / 5.9	16.1 / 6.4
	7	12.8 / 4.8	12.7 / 4.3	16.0 / 6.7	17.4 / 7.0
RNN	1	-	-	18.7 / 8.2	**13.1 / 4.9**
	2	17.7 / 7.5	19.3 / 8.0	15.9 / 6.1	13.9 / 5.0
	3	15.4 / 5.6	16.5 / 6.1	14.4 / 5.8	14.3 / 5.2
	4	14.2 / 5.4	14.1 / 5.3	13.8 / 5.3	13.9 / 5.1
	5	14.2 / 5.1	13.7 / 5.0	14.3 / 5.2	14.2 / 5.1
	6	14.0 / 5.1	14.1 / 4.9	14.6 / 5.8	15.3 / 5.8
	7	14.6 / 5.2	14.5 / 5.1	15.7 / 6.4	14.2 / 5.4