Dropout improves Recurrent Neural Networks for Handwriting Recognition

Vu Pham
Théodore Bluche
Christopher Kermorvant
Jérôme Louradour

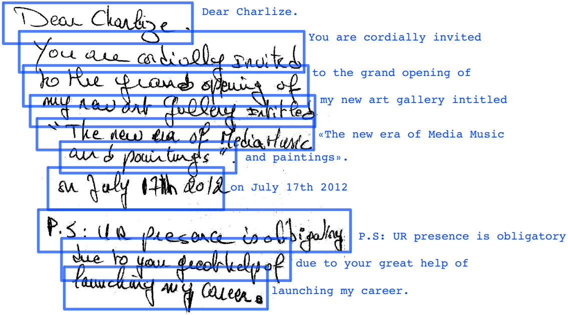
tb@a2ia.com, jl@a2ia.com

Outline

1. **RNN for Handwritten Text Line Recognition**
   - Offline Handwritten Text Recognition
   - Recurrent Neural Networks (RNN)
2. **Dropout for RNN**
3. **Experiments**
   - Improvement of RNN
   - Improvement of the complete recognition system

Outline

Offline Handwritten Text Recognition

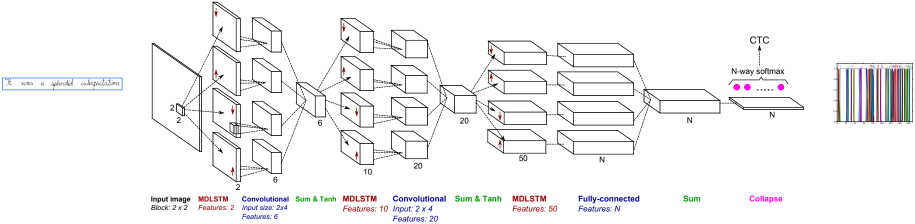


- Line segmentation in the front-end
- “Temporal Classification”:
  Variable-length 1D or 2D input $\mapsto$ 1D target sequence (different length)

Modeling: Recurrent Neural Networks (RNN)

State-of-the-art in Handwritten Text Recognition

Task: Image (2D sequence) $\mapsto$ 1D sequence of characters

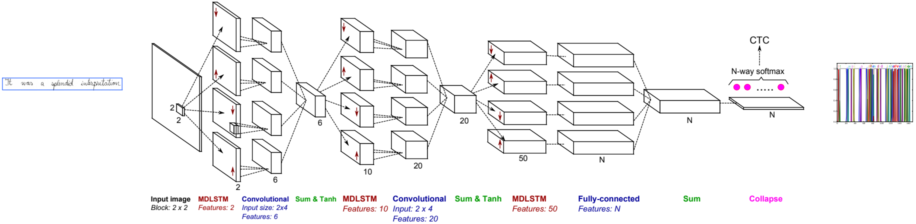


1. RNN Network Architecture (Graves & Schmidhuber, 2008)
   - Multi-Directional layers of LSTM unit
     "Long-Short Term Memory" – 2D recurrence in 4 possible directions
   - Convolutions: parameterized subsampling layers
   - Collapse layer: from 2D to 1D (output $\sim \log P$)

Modeling: Recurrent Neural Networks (RNN)

State-of-the-art in Handwritten Text Recognition

Task: Image (2D sequence) $\mapsto$ 1D sequence of characters

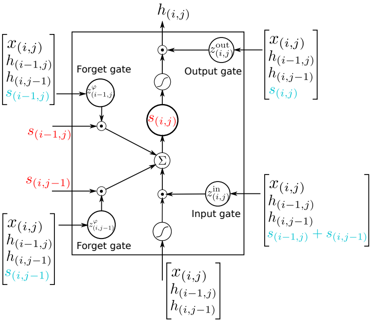


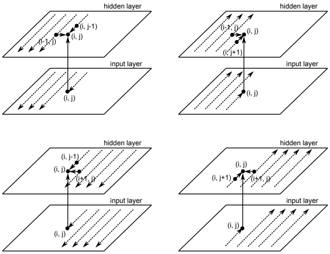
1. RNN Network Architecture (Graves & Schmidhuber, 2008)
   - Multi-Directional layers of LSTM unit
     "Long-Short Term Memory" – 2D recurrence in 4 possible directions
   - Convolutions: parameterized subsampling layers
   - Collapse layer: from 2D to 1D (output $\sim \log P$)
2. CTC Training ("Connectionist Temporal Classification")
   - The network can output all possible symbols and also a *blank* output
   - Minimization of the Negative Log-Likelihood $-\log(P(Y|X))$ (NLL)

Modeling: Recurrent Neural Networks (RNN)

State-of-the-art in Handwritten Text Recognition

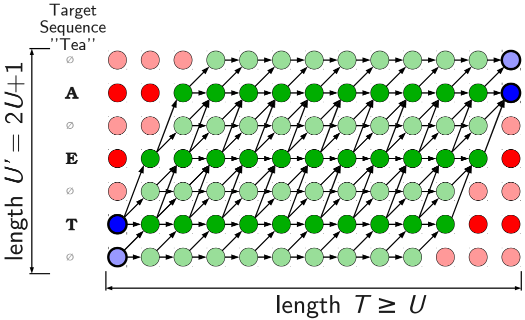
The recurrent neurons are Long Short-Term Memory (LSTM) units





Loss function: Connectionist Temporal Classification (CTC)

Deal with several possible alignments between two 1D sequences



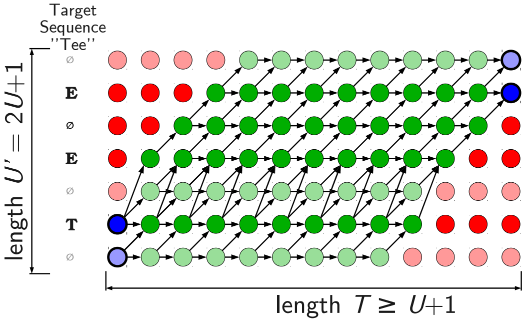
$\rightsquigarrow -\log P(Y|X)$

- $U = 3$: Number of target symbols
- T: Number of RNN outputs $\propto$ image width
- Basic decoding strategy (without lexicon neither language model):

$$[\emptyset\ldots]\texttt{T}\ldots[\emptyset\ldots]\texttt{E}\ldots[\emptyset\ldots]\texttt{A}\ldots[\emptyset\ldots] \quad \mapsto \quad \text{``}\texttt{TEA}\text{''}$$

Loss function: Connectionist Temporal Classification (CTC)

Deal with several possible alignments between two 1D sequences



$\rightsquigarrow -\log P(Y|X)$

- $U = 3$: Number of target symbols
- T: Number of RNN outputs $\propto$ image width
- Basic decoding strategy (without lexicon neither language model):

$$[\emptyset\ldots]\mathtt{T}\ldots[\emptyset\ldots]\mathtt{E}\ldots\ \emptyset\ldots\ \mathtt{E}\ldots[\emptyset\ldots] \quad \mapsto \quad \text{"TEE"}$$

Optimization: Stochastic Gradient Descent

Simple and efficient

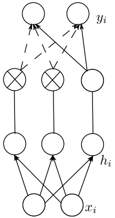
- No mathematical guarantee (no chance to converge to the real global minimum)
- But popular with deep networks: works well in practice! (find "good" local minima)

for (input, target) in Oracle() **do**
```
  output= RNN.Forward( input )
  outGrad= CTC_NLL.Gradient( output, target )
  paramGrad= RNN.BackwardGradient( input, ..., outGrad )
  RNN.Update( paramGrad )
```
end for

Outline

Dropout

General Principle [Krizhevsky & Hinton, 2012]



Training:

- Randomly set to 0 intermediate activities (*) with probability p (typically $p = 0.5$)
  (*) neurons outputs usually in $[-1, 1]$, $[0, 1]$ or $[0, \infty)$
- $\sim$ Sampling from 2^N different architectures that share weights

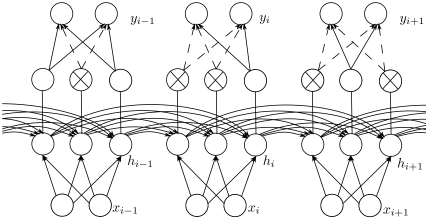
Decoding:

- All intermediate activities are scaled, by $1 - p$
- $\sim$ Geometric mean of the outputs from 2^N models

Featured in award-winning convolutional networks (ImageNet)

Dropout

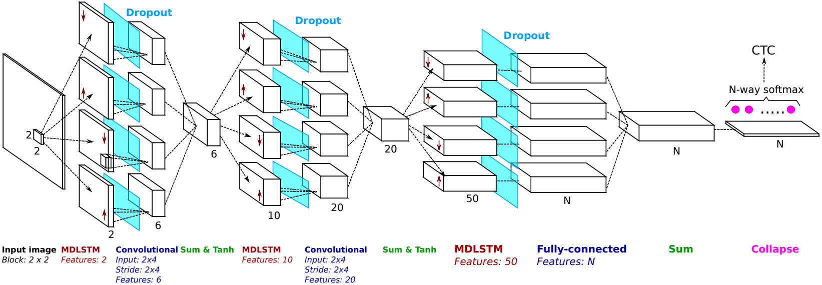
Dropout with recurrent layer



- Recurrent connections are kept untouched
- Dropout can be implemented as separated layer (outputs identical to inputs, except at dropped locations)

Dropout

Overview of the full network



- After recurrent LSTM layers
- Before feed-forward layers (convolutional and linear layers)

Outline

Databases and performance assessment

Database	Language	# different characters	Training subset: # labelled lines	Training subset: # characters (in lines)
IAM	**English**	78	9,462	338,904
Rimes	**French**	114	11,065	429,099
OpenHaRT	**Arabic**	154	91,811	2,267,450

Training:
Minimizing Negative Log-Likelihood (NLL) with CTC alignments.

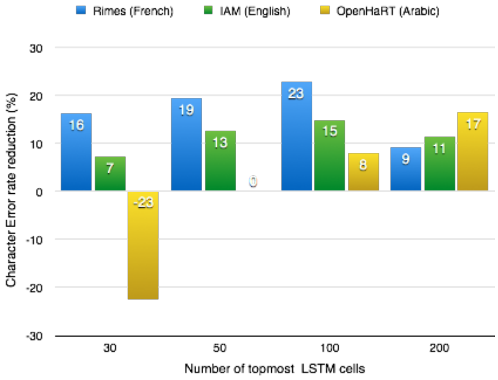
Decoding:
Pick the best label at each timestep, Remove duplicates, then blanks.

Evaluation:
Character Error Rate (%), on a separate dataset.
Reduction w/ and w/o dropout.

Training convergence time is also interesting, but not critical.

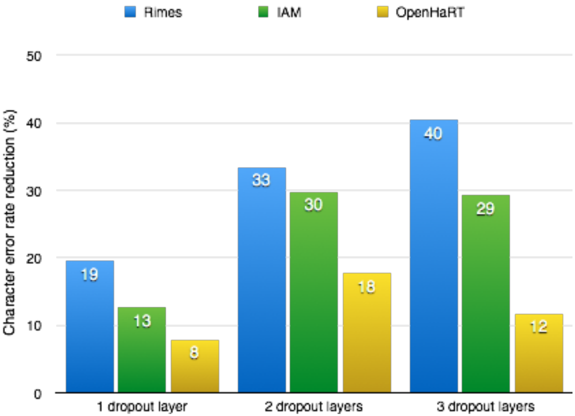
Results: Dropout on the topmost LSTM layer

- ∼ Dropout on high-level features used in Logit Regression
- Error rate reduction when varying the number of hidden units in the topmost layer

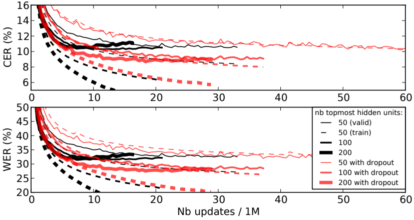


Results: Dropout on all LSTM layers

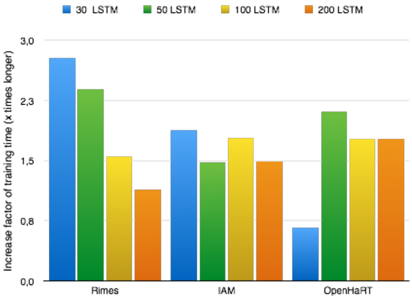
- Use the good recipe whenever possible!
- Number of hidden units tuned (on validation dataset) to reach best performance



Results analysis: Dropout acts as Regularization



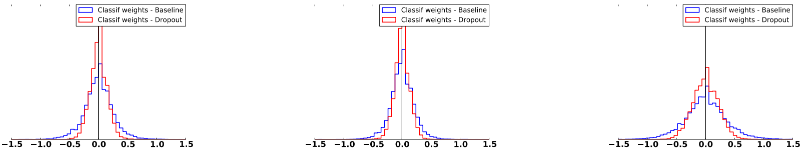
Convergence curves



- **Less overfitting:**
  the gap between training and validation loss is smaller
- **Training with dropout is slower:**
  There is a trade-off between accuracy & training speed.
  (However, decoding speed is the same for a given neural archi.!)

Results analysis: Dropout acts as Regularization

IAM (English) **Rimes** (French) **OpenHaRT** (Arabic)



- Outgoing weights are smaller: L1 and L2 norms are greatly reduced
- Better than L1/L2 Weight Decay (and also simple to implement)
  - Data-driven approach.
  - No need to tune $\lambda \in [0, +\infty($ to control the Bias-Variance Tradeoff. Only one hyper-parameter $p \in [0, 1($ that is less sensitive. NB: $p = 0.5$ works well!
- On the other hand, tanh activations (in [-1,1]) are sharper: More "helpful" features learned by "preventing co-adaptation" (Hinton et al., 2012)

Intergration in a complete recognition system

Performance improves when language constraints (vocabulary, LM) are added.

Decoding in a hybrid RNN/HMM framework ($\frac{p(y|x)}{p(y)} \propto \frac{p(x|y)}{p(x)}$)

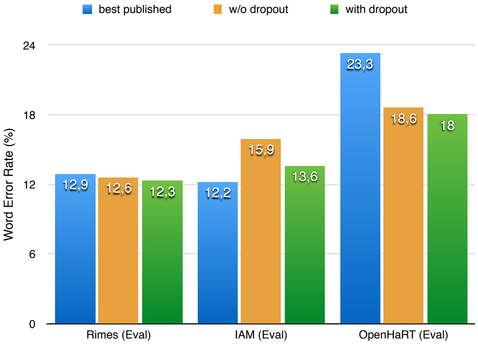
- **HMM:** One state for each label including blank, with self-loop and outgoing transition
- **Lexicon:** Each word is the sequence of character HMMs with optional blanks in between
- **Language Model:** Word *n*-grams

The goal is to find the optimal word sequence $\hat{\mathbf{W}}$

$$\hat{\mathbf{W}} = arg \max_{\mathbf{W}} p(\mathbf{W}|\mathbf{X}) = arg \max_{\mathbf{W}} p(\mathbf{X}|\mathbf{W})p(\mathbf{W}) \tag{1}$$

Results in a complete system:

Word Error Rate of Full Systems (Optical Model + Lexicon/Language Model):



Database	Language	# words	# words in vocabulary	% OOV	LM	Perplexity
Rimes	**French**	5,639	12k	2.6%	4-gram	18
IAM	**English**	25,920	50k	3.7%	3-gram	329
OpenHaRT	**Arabic**	47,837	95k	6.8%	3-gram	1162

Conclusions and future work

- Dropout acts as a regularizer: outgoing weights tend to be lower
- Dropout improves accuracy of Offline Text Recognition with RNN
  about 10-20% improvement in CER and WER
- Training convergence with dropout is longer
  roughly twice slower

Thank you for your attention!

Questions and comments are welcome.

`tb@a2ia.com, jl@a2ia.com`