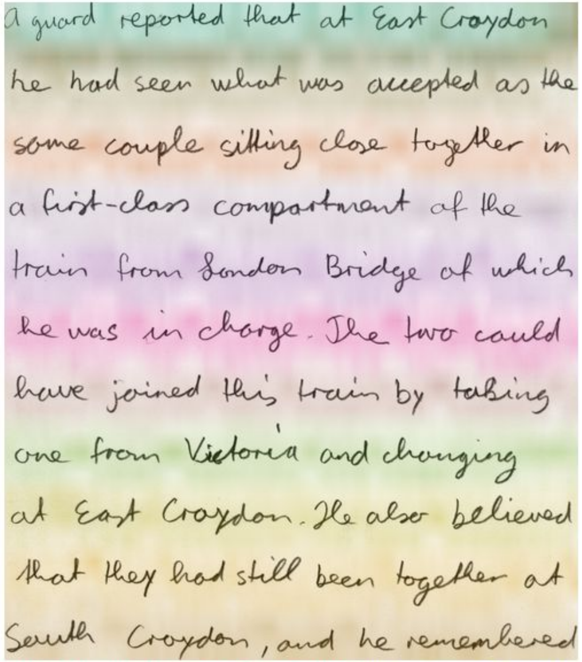
End-to-End Handwritten Paragraph Recognition

Théodore Bluche

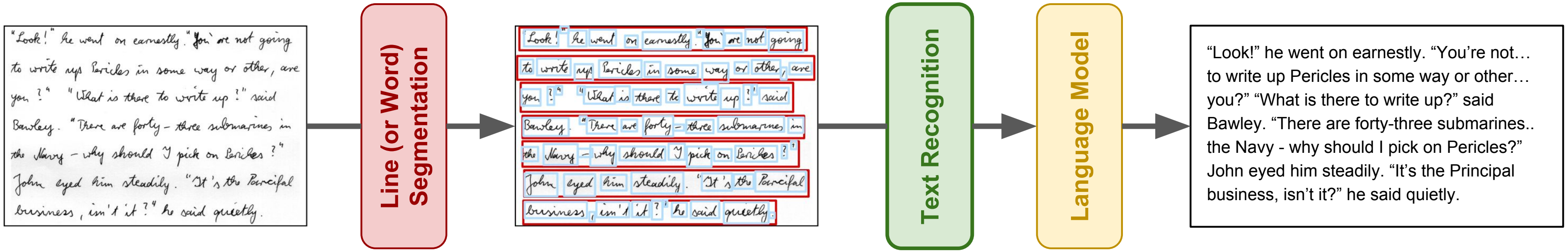
theodore.bluche@gmail.com

Google Zurich - 2 Feb. 2017



A guard reported that at East Craydon he had seen what was ouepted as the some couple sitting close togther in a first-class compartment of the train from London Bridge of which he was in charge . The two could have joined this train by taking one from Vectorin and changing at East Craydon . He also believed that they had still been together at South Croydon , and he remembered

Offline Handwriting Recognition



- ➔ Challenges
  - the input is a **variable-sized two-dimensional** image
  - the output is a **variable-sized sequence** of characters
  - the cursive nature of handwriting makes a prior segmentation into characters difficult
- ➔ Methods
  - Over-segmentation and group-of-segments scoring (90s)
  - Sliding window approach with HMMs (2000s) or neural nets (2000-2010s)
  - **MDLSTM** = models handling both the 2D aspect of the input and the sequential aspect of the prediction → **state-of-the-art**

Limitations

- ➔ Current systems **require segmented text lines**
  - For training = tedious annotation effort or error-prone automatic mapping methods
  - For decoding = need to provide text line images which rarely are the actual input of a production system

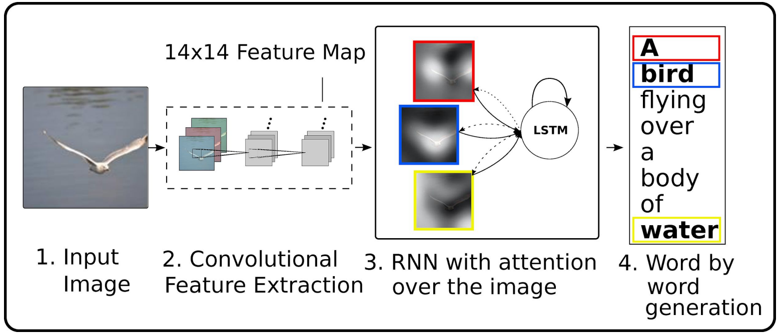
- ➔ Document processing pipelines rely on automatic line segmentation algorithms

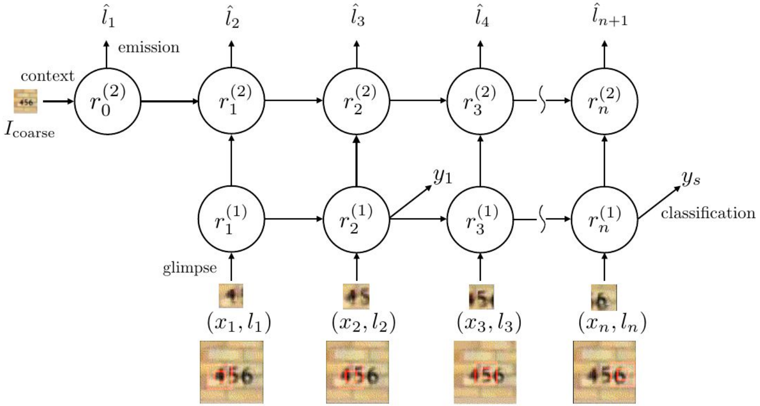
- ➔ ***How to process full pages without requiring an explicit line segmentation?***

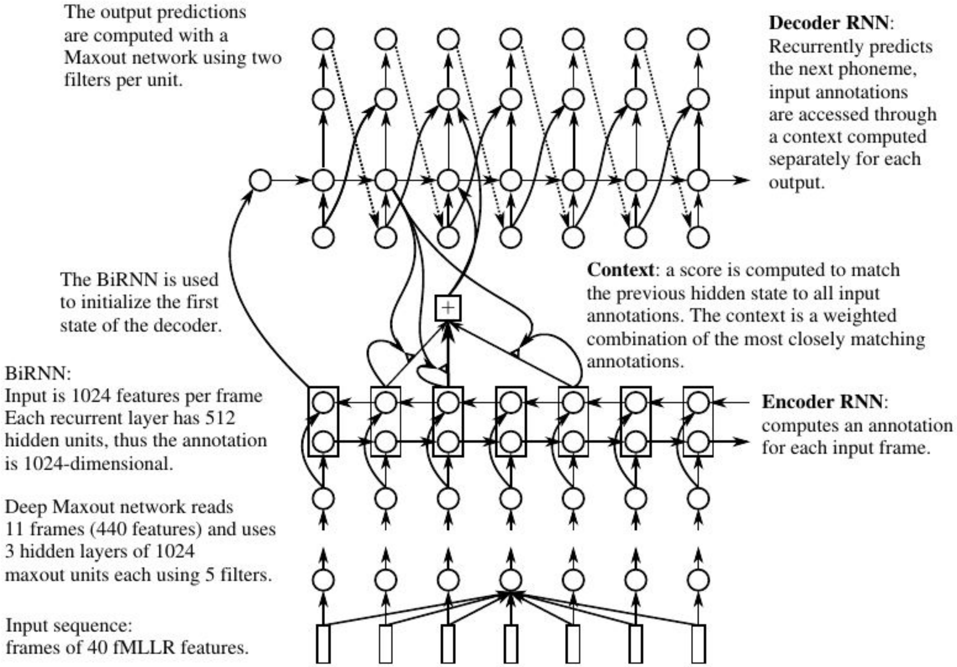
"We believe that the use of selective attention is a correct approach for connected character recognition of cursive handwriting."

—Fukushima et al. 1993

2014-2015 trends





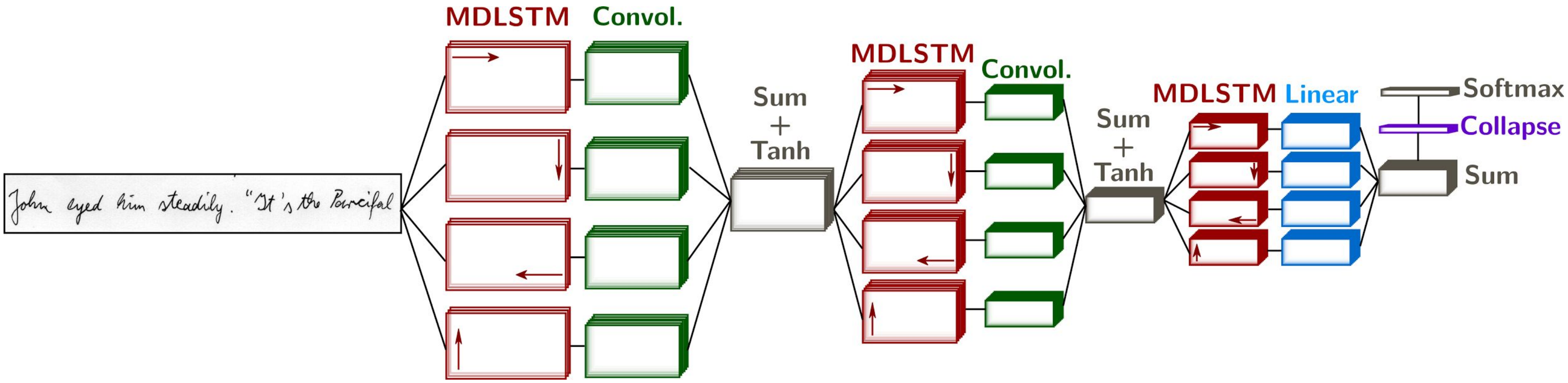


Talk Overview

- ➔ Introduction
- ➔ Handwriting Recognition with Multi-Dimensional LSTM networks
- ➔ Limitations → Motivations of the proposed approach
- ➔ Learning Reading Order - Character-wise Attention
- ➔ Implicit Line Segmentation - Speeding Up Paragraph Recognition
- ➔ Conclusion

Handwriting Recognition with Multi-Dimensional LSTM networks

Handwriting Recognition with MDLSTM

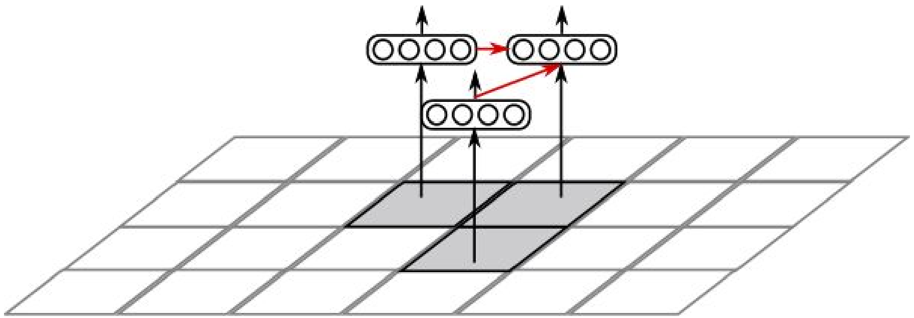


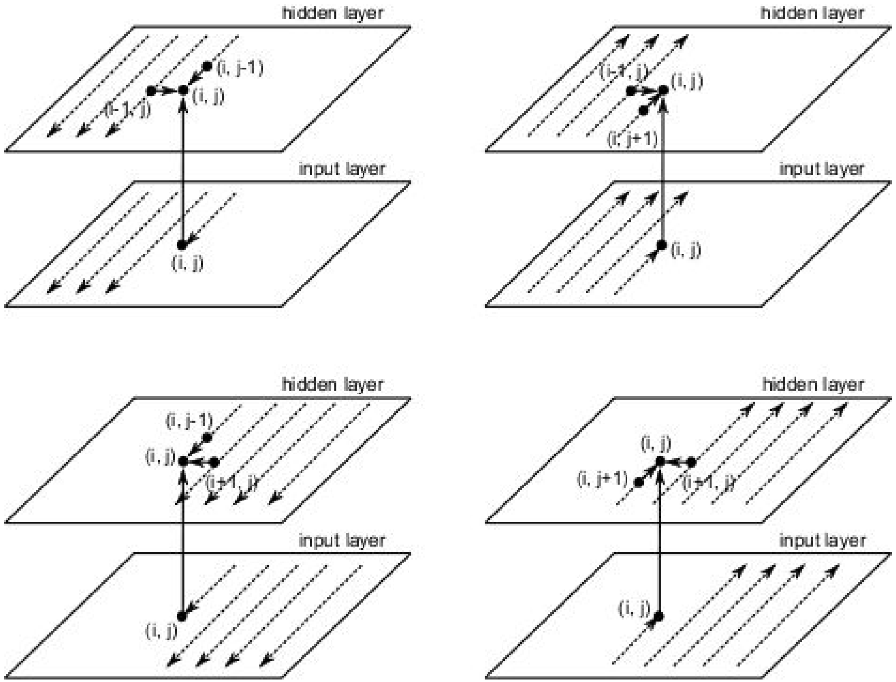
Multi-Dimensional Recurrent Neural Networks

= recurrence in 2D

= 4 possible scanning directions

In MDLSTM, 2 forget gates and
2 recurrent weight matrices





Connectionist Temporal Classification (CTC)

→ The network **outputs characters**

→ **Problem** *T* items in the output sequence, *N* items in the target char sequence

→ Make sure that *T* > *N* and **define a simple mapping** of sequences that removes duplicates:

$$\text{AAABBCCCC} \longrightarrow \text{ABC}$$
$$\text{ABBBBBCCC} \longrightarrow \text{ABC}$$

...

$$\text{AAAABCCCC} \longrightarrow \text{ABC}$$

$$p(c_1 \dots c_N | \mathbf{x}) = \sum\nolimits_{y_1 \dots y_T \rightarrow c_1 \dots c_N} p(y_1 \dots y_T | \mathbf{x})$$
$$= \sum\nolimits_{y_1 \dots y_T \rightarrow c_1 \dots c_N} \prod\nolimits_t p(y_t | \mathbf{x})$$

= Net's output at time *t*

→ Computed efficiently with dynamic programing

→ **Problem** how to output ABB (AAABBBBBB → AB) ?

Connectionist Temporal Classification (CTC)

- → **Problem** how to output ABB (AAABBBBBB → AB) ?
- → The network outputs characters **+ a special NULL** (or blank; non-char) symbol **-**
- → The mapping **removes** duplicates, and ***then NULLs***

AAABBCCCC ⟶ ABC

AA-BB--C- ⟶ A-B-C- ⟶ ABC

...

-A--B--C- ⟶ -A-B-C- ⟶ ABC

AAABBBBBB ⟶ AB

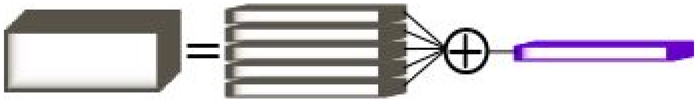
AA-BB--B- ⟶ A-B-B- ⟶ ABB

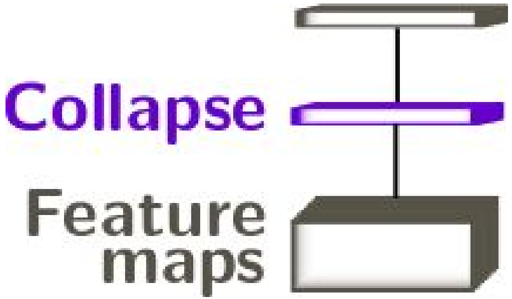
...

-A--B--B- ⟶ -A-B-B- ⟶ ABB

The “Collapse” layer

$$z_j = \sum_i f_{ij}$$



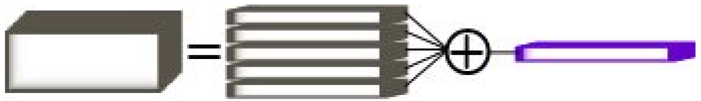


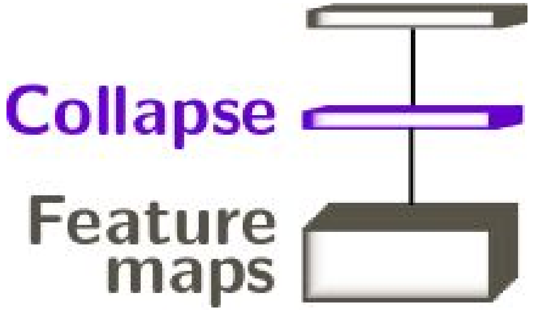
- 2D → 1D conversion
- Simple sum across vertical dimension
- Feature maps of height 1 interpreted as a sequence

Limitations → Motivations of the proposed approach

The “Collapse” layer

$$z_j = \sum_i f_{ij}$$

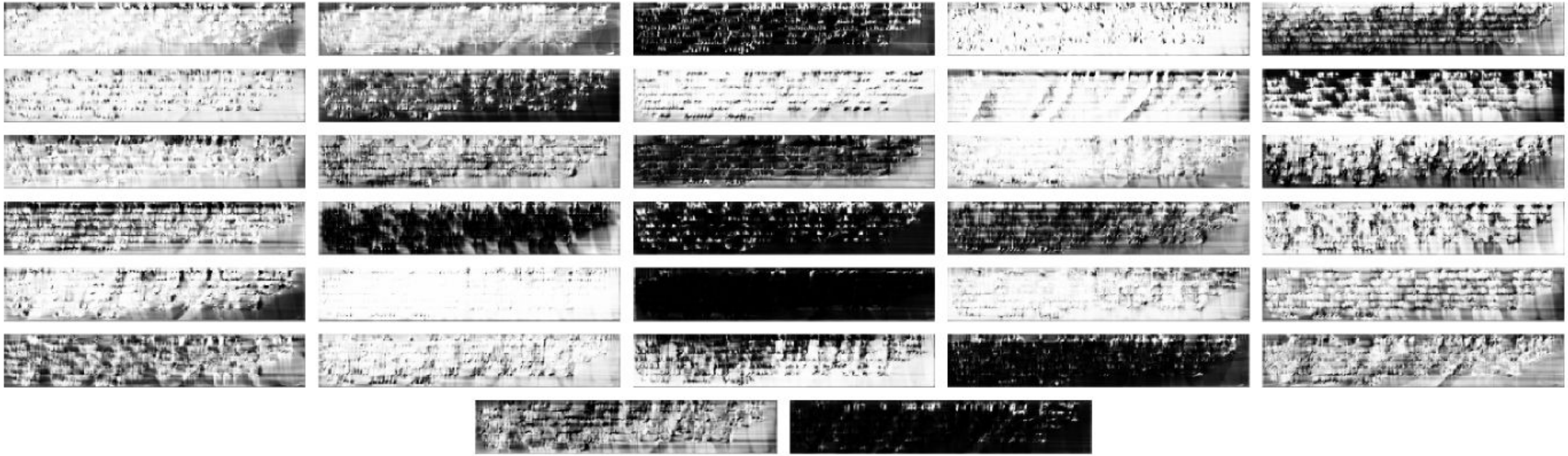




1. all the feature vectors in the same column j are **given the same importance**
2. the **same error is backpropagated** in a given column j

→ **Prevents the recognition of several text lines**

Side effects

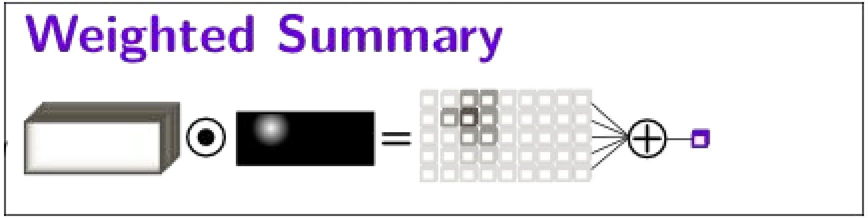


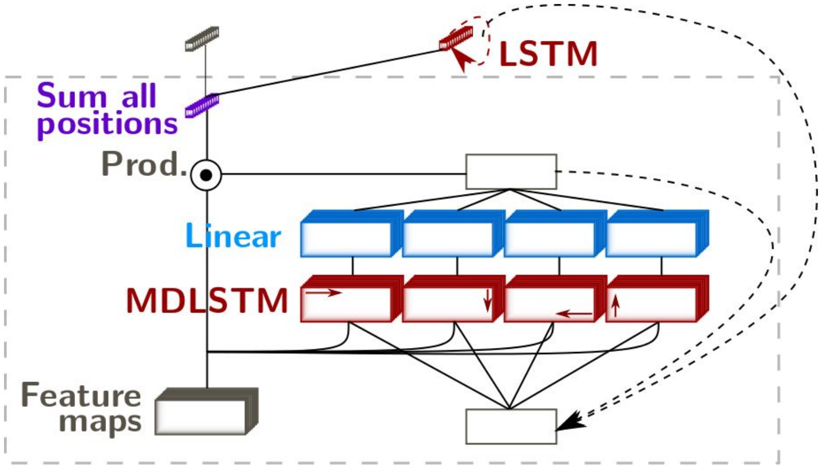
Proposed modification

- Augment the collapse layer with an **"attention" module**, which can learn to focus on specific locations in the feature maps
- Attention on characters or text lines
- Takes the form of a **neural network**, which, applied several times **can sequentially transcribe a whole paragraph**

Weighted Summary:
predict one character at a time

$$z_t = \sum_{i,j} \omega_{ij}^{(t)} f_{ij}$$



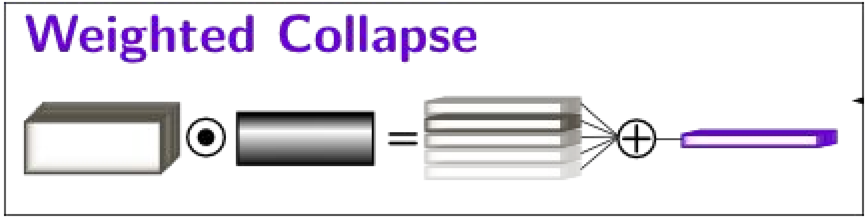


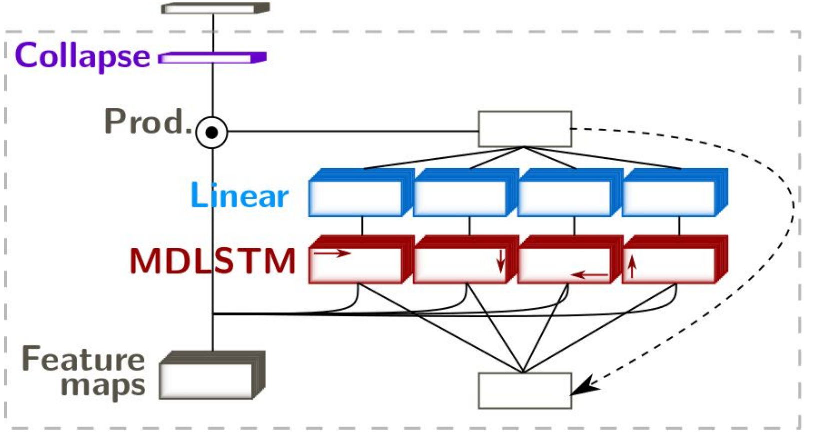
This is the ***"Scan, Attend and Read"*** model.

Weighted Collapse

recognize one line at a time

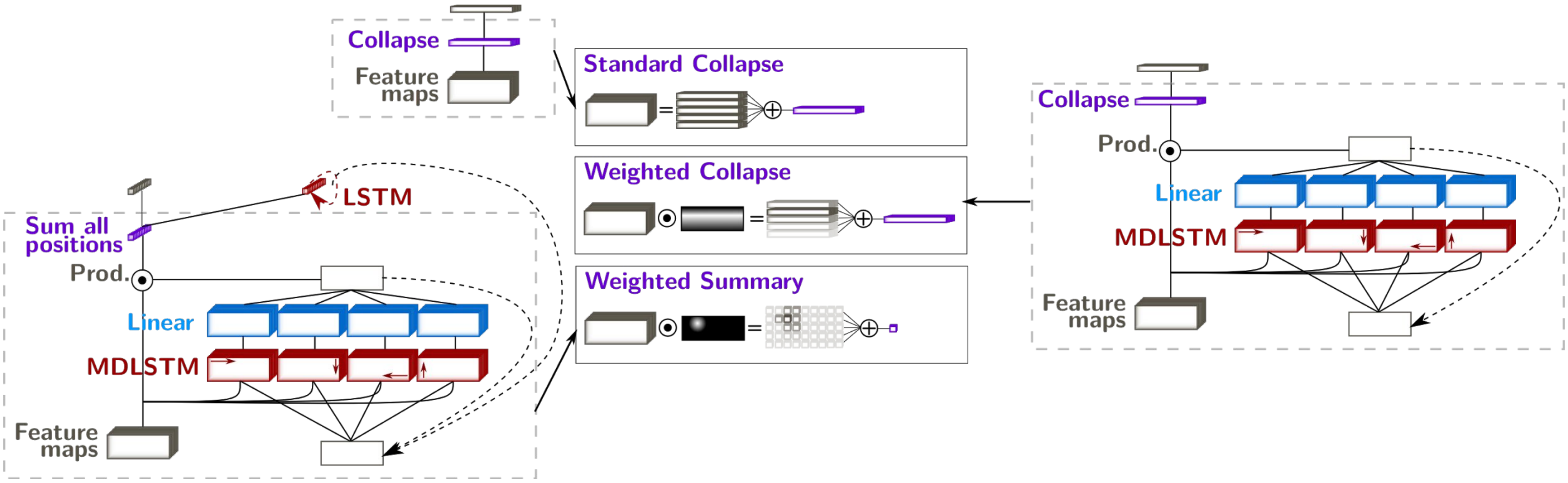
$$z_j^{(t)} = \sum_i \omega_{ij}^{(t)} f_{ij}$$





This is the ***"Joint Line Segmentation and Transcription"*** model.

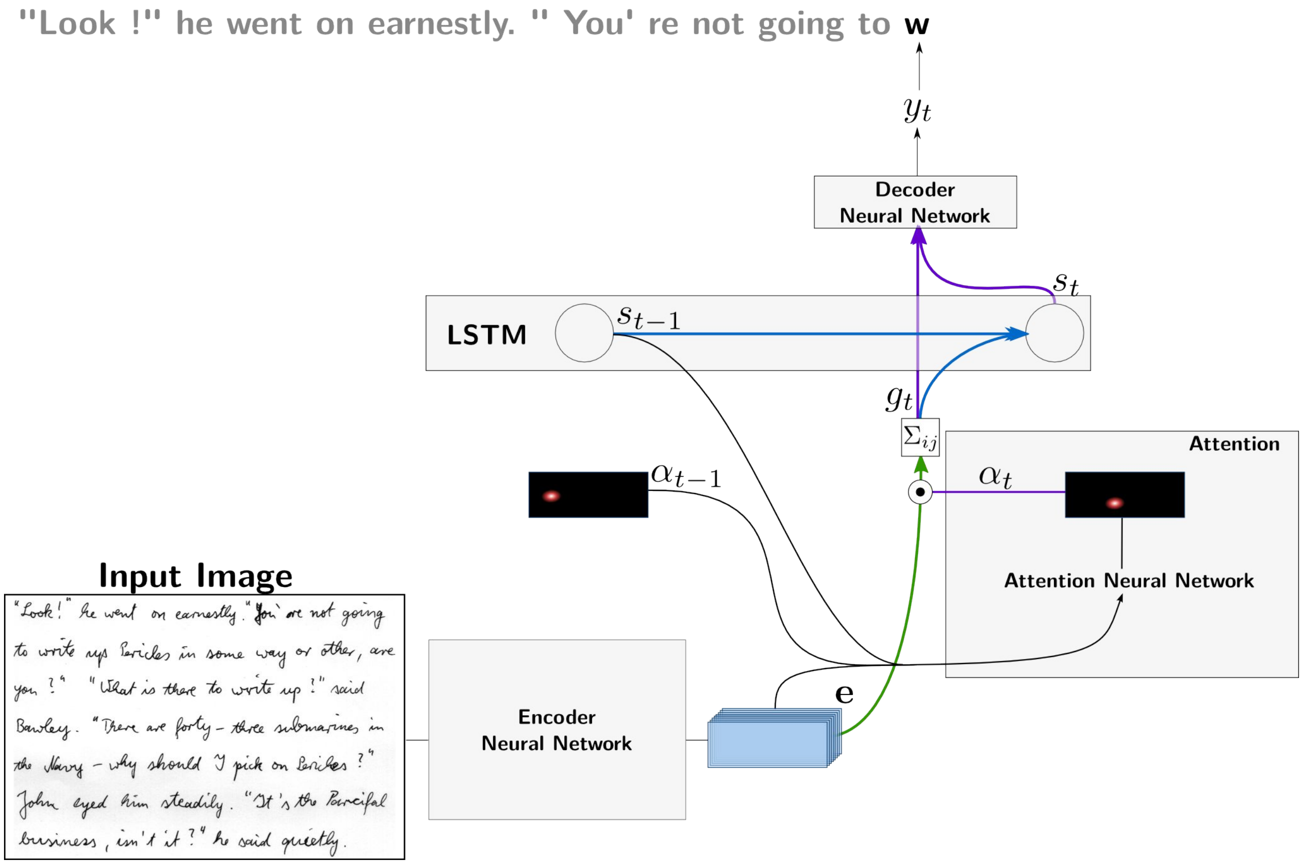
Proposed modifications



Learning Reading Order

Character-wise Attention

“Scan, Attend and Read”



Network's architecture

➔ Encoder

$$f_{i,j} = Encoder(\mathcal{I})$$

➔ Attention

$$\omega_{ij}^{(t)} = Attention(\mathbf{f}, \omega^{(t-1)}, s_{t-1})$$

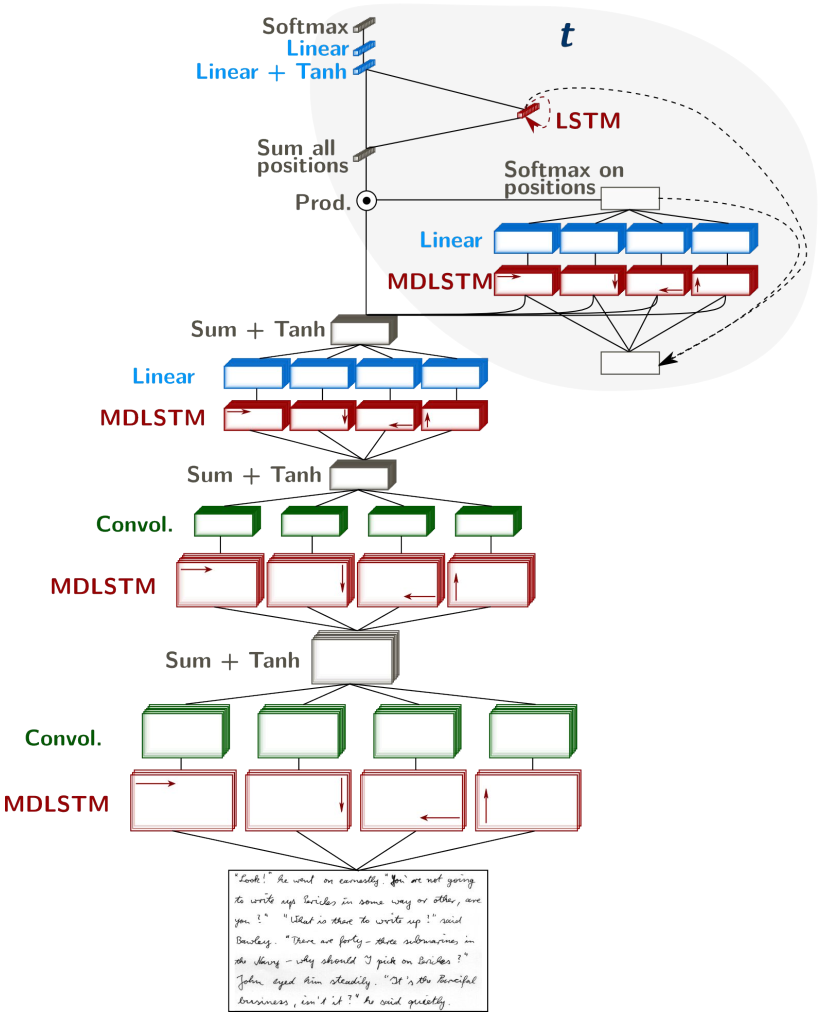
$$z_t = \sum_{i,j} \omega_{ij}^{(t)} f_{i,j}$$

➔ State

$$s_t = LSTM(s_{t-1}, z_t)$$

➔ Decoder

$$y_t = Decoder(s_t, z_t)$$



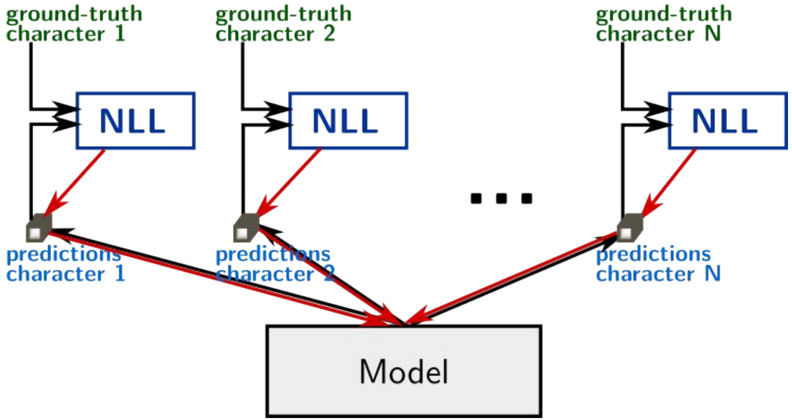
The attention mechanism

- The **attention mechanism provides a summary** of the encoded image **at each timestep**
- The attention network computes a **score for the feature vectors at each position**. The scores are **normalized with a softmax**.

$$\omega_{ij}^{(t)} = \frac{e^{m_{ij}^{(t)}}}{\sum_{i',j'} e^{m_{i'j'}^{(t)}}}$$

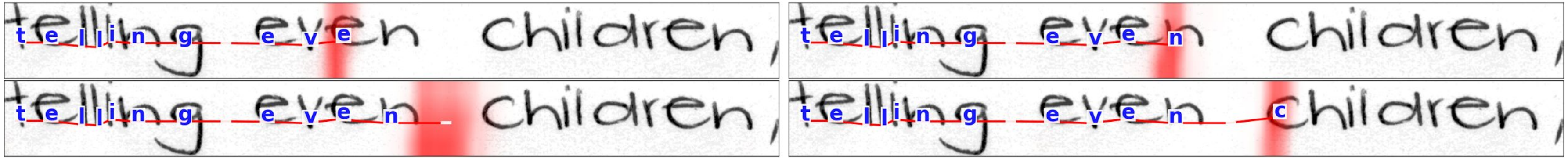
Model Training

$$\mathcal{L}(\mathcal{I}, \mathbf{y}) = -\sum_t \log p(y_t|\mathcal{I})$$



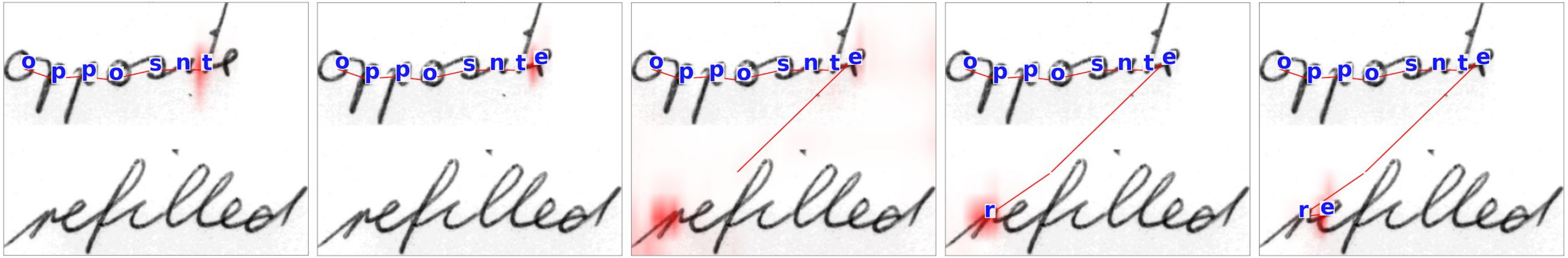
- ➔ We include a special token **EOS** at the end of the target sequences (also predicted by the network to indicate when to stop reading at test time)
- ➔ The net has to predict the correct character at each timestep

Text Lines



Model	Inputs	CER (%)
MDLSTM + CTC	Full Lines	6.6
Attention-based	1 word	12.6
	2 words	9.4
	3 words	8.2
	4 words	7.8
	Full Lines	7.0

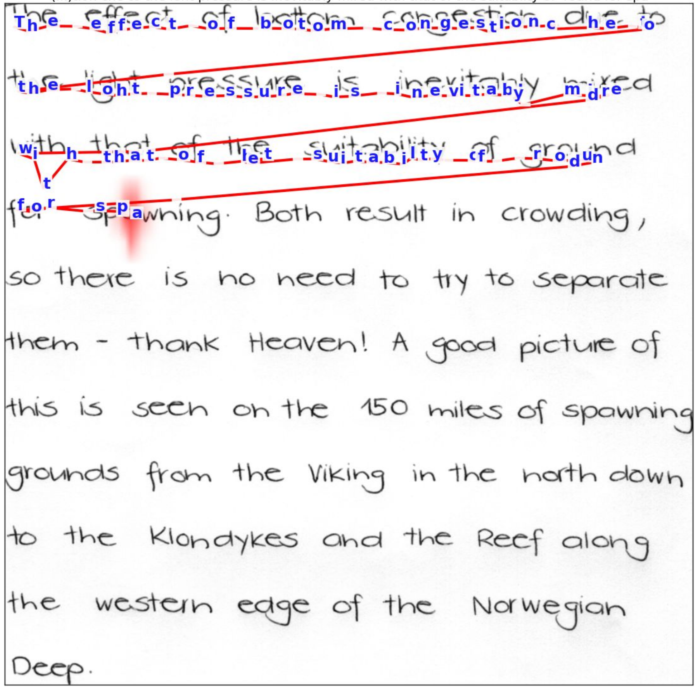
Learning Line Breaks



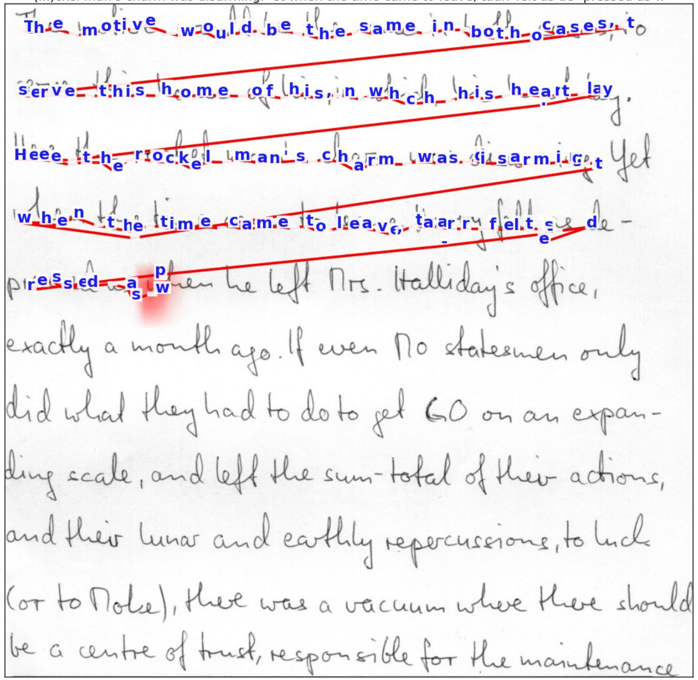
Two lines of...	CER (%)
1 words	11.8
2 words	11.1
3 words	10.9
Full Lines	9.4

Paragraph Recognition

(...)tion che to the loht pressure is inevitably mired with that of the suitabilty of round for spa



(...)ckel man's charm was disarming. et when the time came to leave, taarr felt as de--pressed as w



Training tricks

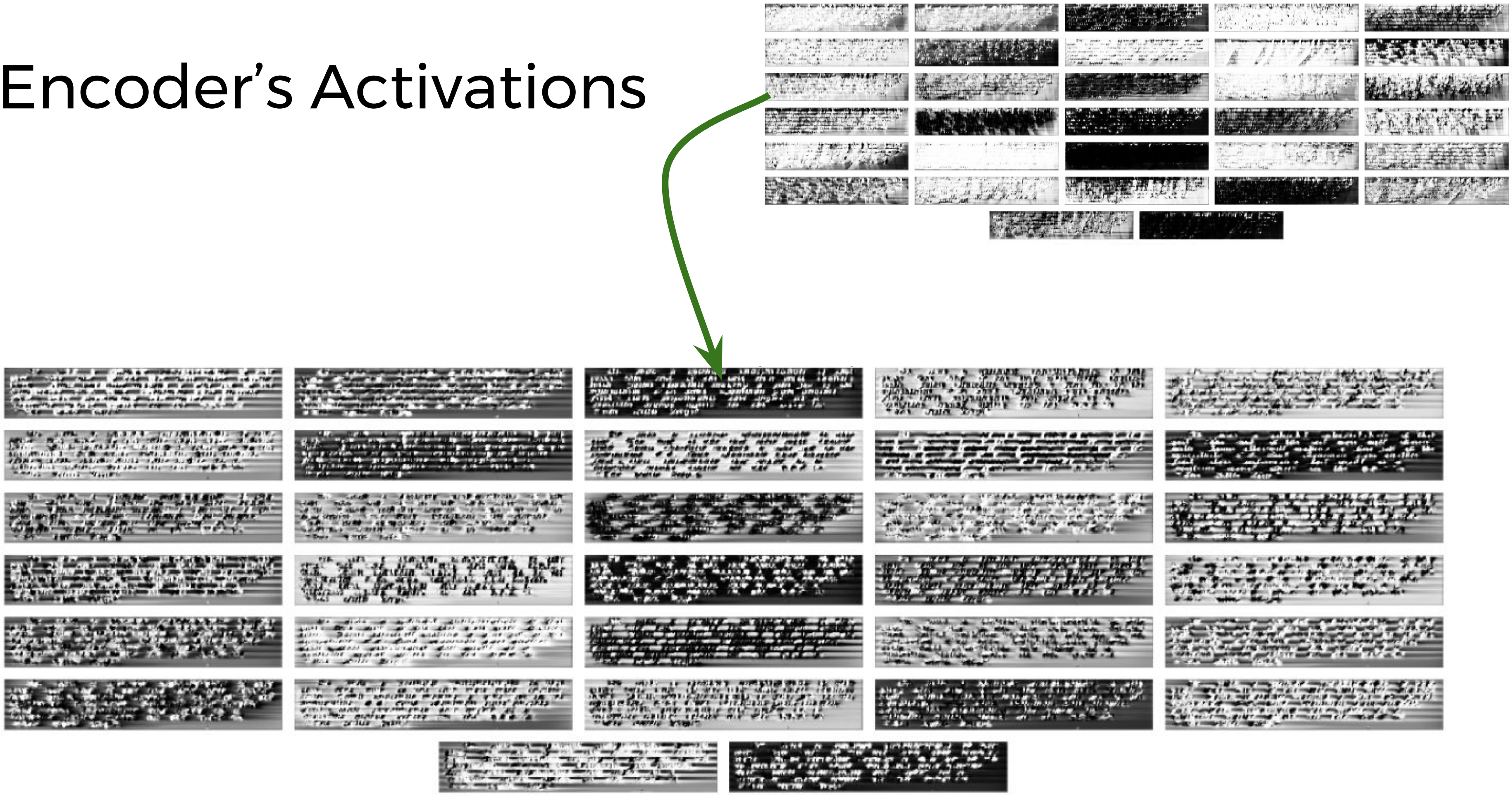
In order to get the model to converge, or to converge faster, a few tricks helped:

- Pretraining use an MDLSTM network (no attention) **trained on single lines with CTC as a pretrained encoder**
- Data augmentation add to the training set **all possible sub-paragraphs** (i.e. one, two, three, ... consecutive lines)
- Curriculum (0/2) training the attention model on **word images or single line images** works quite well, do this as a first step
- Curriculum (1/2) (Louradour et al., 2014) draw **short paragraphs** (1 or 2 lines) samples with higher probability at the beginning of training
- Curriculum (2/2): **incremental learning**. Run the attention model on the paragraph images N times (e.g. 30 times) during the first epoch, and train to output the first N characters (don't add EOS here). Then, in the second epoch, train on the first 2N characters, etc.
- Truncated BPTT to avoid memory issues

Results (Character Error Rate / IAM)

Resolution (DPI)	Line segmentation				Attention-based *(this work)*
	GroundTruth	Projection	Shredding	Energy	
90	18.8	24.7	19.8	20.8	-
150	10.3	17.2	11.1	11.8	16.2
300	6.6	13.8	7.5	7.9	-

Encoder’s Activations



Pros & Cons

- ➔ Can potentially handle any reading order
- ➔ Can output character sequences of any length
- ➔ Can recognize paragraphs (and maybe complete document?)
- ➔ Very slow + Requires a lot of memory during training
- ➔ Not quite close to state-of-the-art performance on paragraphs (for now...)

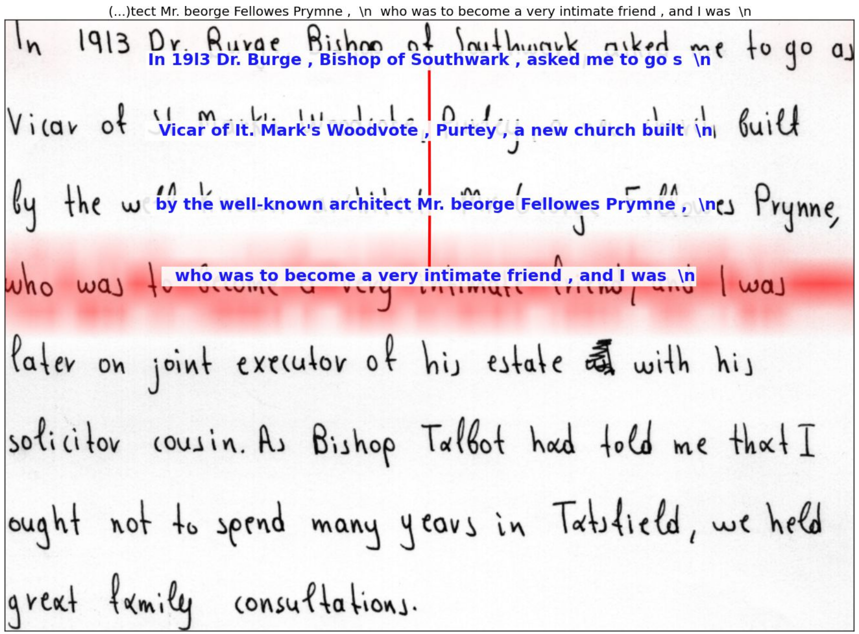
Implicit Line Segmentation

Speeding Up Paragraph Recognition

Joint Line Segmentation and Transcription

- ➔ The previous model is too slow and time consuming
- ➔ Because of **one costly operation for each character**
- ➔ **Idea of this model : one timestep per line**

i.e. put attention on text lines
= reduced from 500+ to ~10 timesteps



Network's architecture

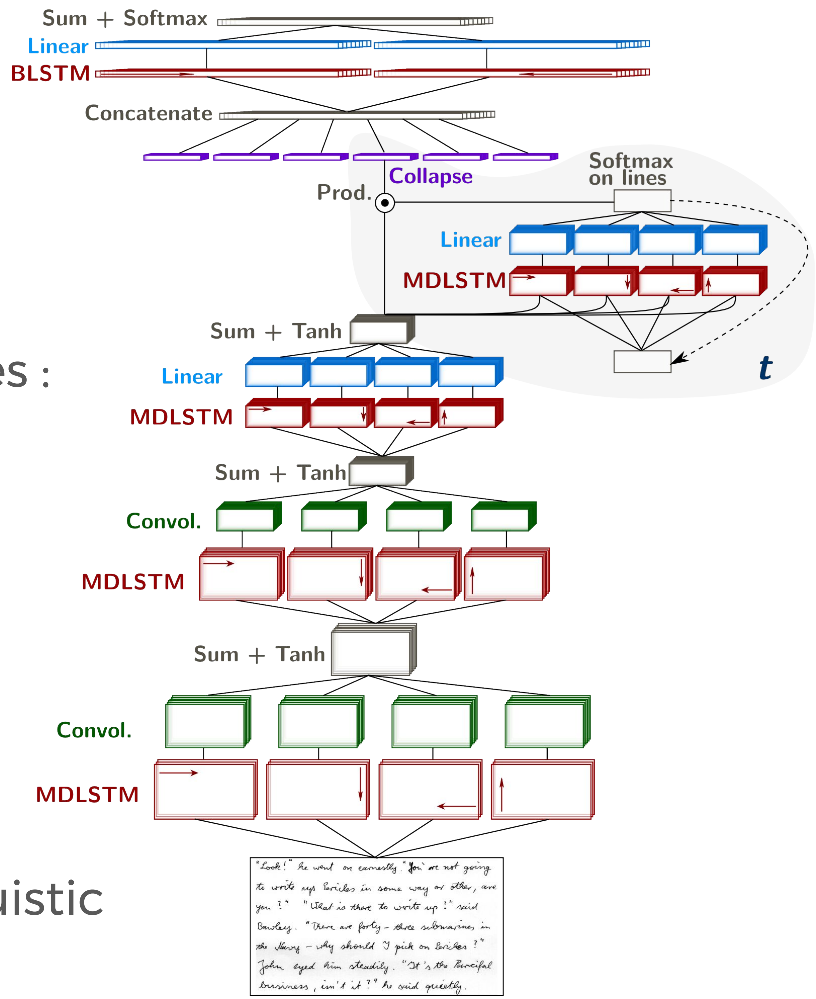
→ Similar Architecture
(encoder, attention, decoder)

→ **Modified attention** to output full lines :
softmax on lines + collapse

$$\omega_{ij}^{(t)} = \frac{e^{m_{ij}^{(t)}}}{\sum_{i'} e^{m_{i'j}^{(t)}}} \qquad z_j^{(t)} = \sum_i \omega_{ij}^{(t)} f_{ij}$$

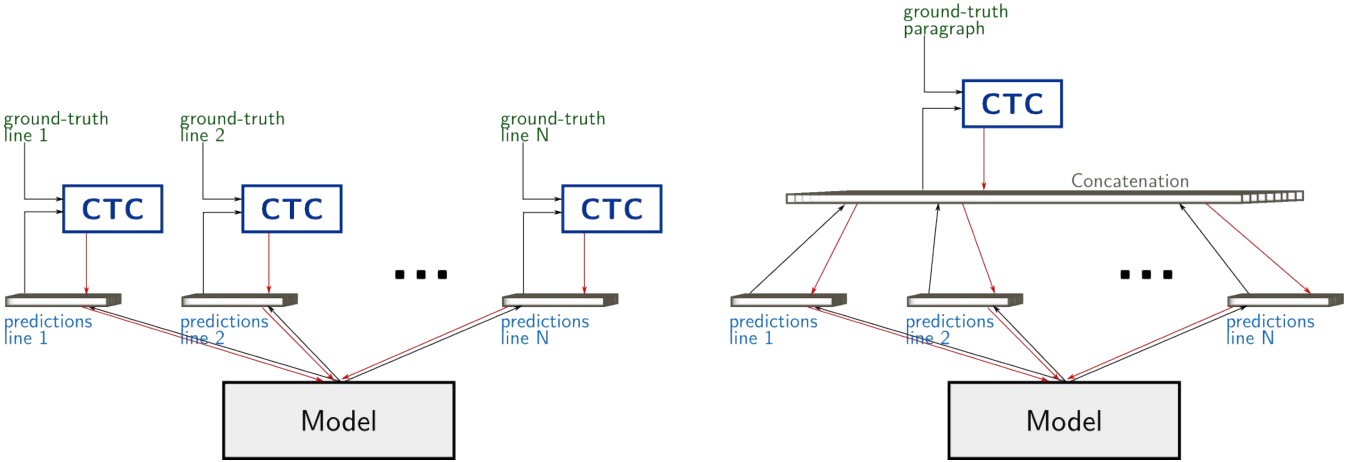
→ No "state"

→ **BLSTM decoder** that can model linguistic dependencies across text lines

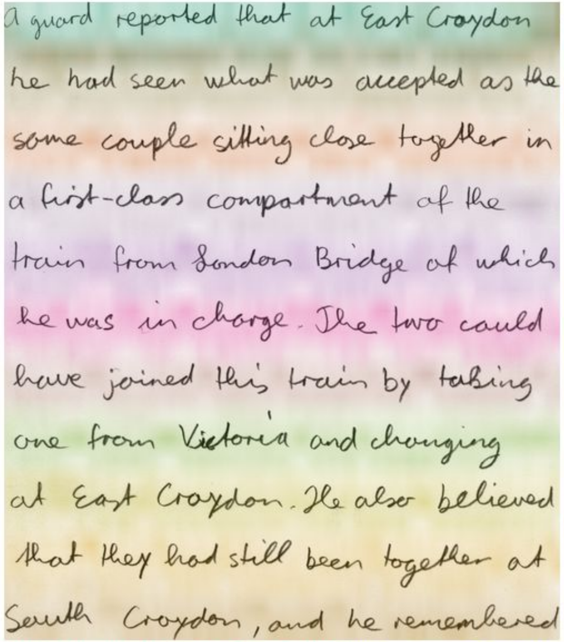


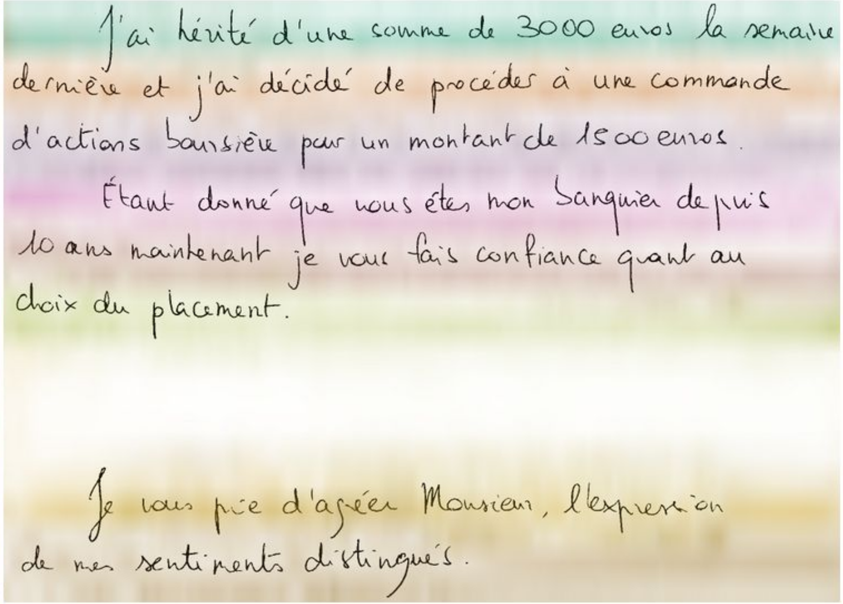
Training

- ➔ In this model we have **more predictions than characters** ⇒ CTC
- ➔ If the line breaks are known → CTC on each segment (attention step)
- ➔ Otherwise → **CTC at the paragraph level**
- ➔ Less tricks required to train
  (only pretraining and 1 epoch on two-line inputs)



Qualitative Results





A guard reported that at East Craydon he had seen what was ouepted as the some couple sitting close togther in a first-class compartment of the train from London Bridge of which he was in charge . The two could have joined this train by taking one from Vectorin and changing at East Craydon . He also believed that they had still been together at South Croydon , and he remembered

J'ai hérité d'une somme de 3000 euros la semaine dernière et j'ai décidé de procéder à une commande d'actions boursière pour un montant de 1500 euros. Etant donné que vous êtes mon banquier depuis 10 ans maintenant je vous fais confiance quant au choix du placement. Je vous prie d'agréer Monsieur, l'expression de mes sentiments distingués.

Comparison with Explicit Line Segmentation

➔ Because of segmentation errors, CERs increase with automatic (explicit) line segmentation

➔ With the proposed model, they are even lower than when using ground-truth positions ...

		Line segmentation				
Database	**Resolution**	**GroundTruth**	**Projection**	**Shredding**	**Energy**	**This work**
IAM	150 dpi	8.4	15.5	9.3	10.2	6.8
	300 dpi	6.6	13.8	7.5	7.9	4.9
Rimes	150 dpi	4.8	6.3	5.9	8.2	2.8
	300 dpi	3.6	5.0	4.5	6.6	2.5

Comparison with Explicit Line Segmentation

➔ … partly because the BLSTM decoder can model dependencies across text lines

BLSTM after collapse but limited to textlines

BLSTM after attention on full paragraphs

Collapse	Decoder	IAM	Rimes
Standard	Softmax	8.4	4.9
Standard	BLSTM + Softmax	7.5	4.8
Attention	BLSTM + Softmax	6.8	2.5

Processing Times

- On average, the first method (Scan, Attend and Read) is
  - **100x slower** than recognition from known text lines
  - **30x slower** than a standard segment+reco pipeline
- The second method is
  - **30-40x faster** than the first one (expected from fewer attention steps)
  - **about the same speed** as a standard segment+reco pipeline

Method		Processing time (s)
GroundTruth	(crop+reco)	0.21 ± 0.07
Shredding	(segment+crop+reco)	0.78 ± 0.26
Scan, Attend and Read	(reco)	21.2 ± 5.6
This Work	(reco)	0.62 ± 0.14

Final Results

NIPS Paper		Rimes WER%	Rimes CER%	IAM WER%	IAM CER%
150 dpi	no language model	13.6	3.2	29.5	10.1
	with language model			16.6	6.5
300 dpi	no language model	12.6	**2.9**	24.6	7.9
	with language model			16.4	5.5
	Bluche, 2015	**11.2**	3.5	**10.9**	**4.4**
	Doetsch et al., 2014	12.9	4.3	12.2	4.7
	Kozielski et al. 2013	13.7	4.6	13.3	5.1
	Pham et al., 2014	12.3	3.3	13.6	5.1
	Messina & Kermorvant, 2014	13.3	-	19.1	-
Latest result		7.9	2.2	10.1	3.3

Pros & Cons

- ➔ Much faster than "Scan, Attend and Read"
- ➔ Easier paragraph training
- ➔ Results are competitive with state-of-the-art models
- ➔ The attention spans the whole image width, so the method is limited to paragraphs (not full, complex, documents)
- ➔ The reading order is not learnt

Conclusions

Conclusions & Challenges

- ➔ Inspired from recent advances in deep learning
- ➔ Attention-based model for end-to-end paragraph recognition
- ➔ A model that can learn reading order (but difficult to train)
- ➔ A faster model that implicitly performs line segmentation
- ➔ Could be trained with limited data (only Rimes or IAM…)

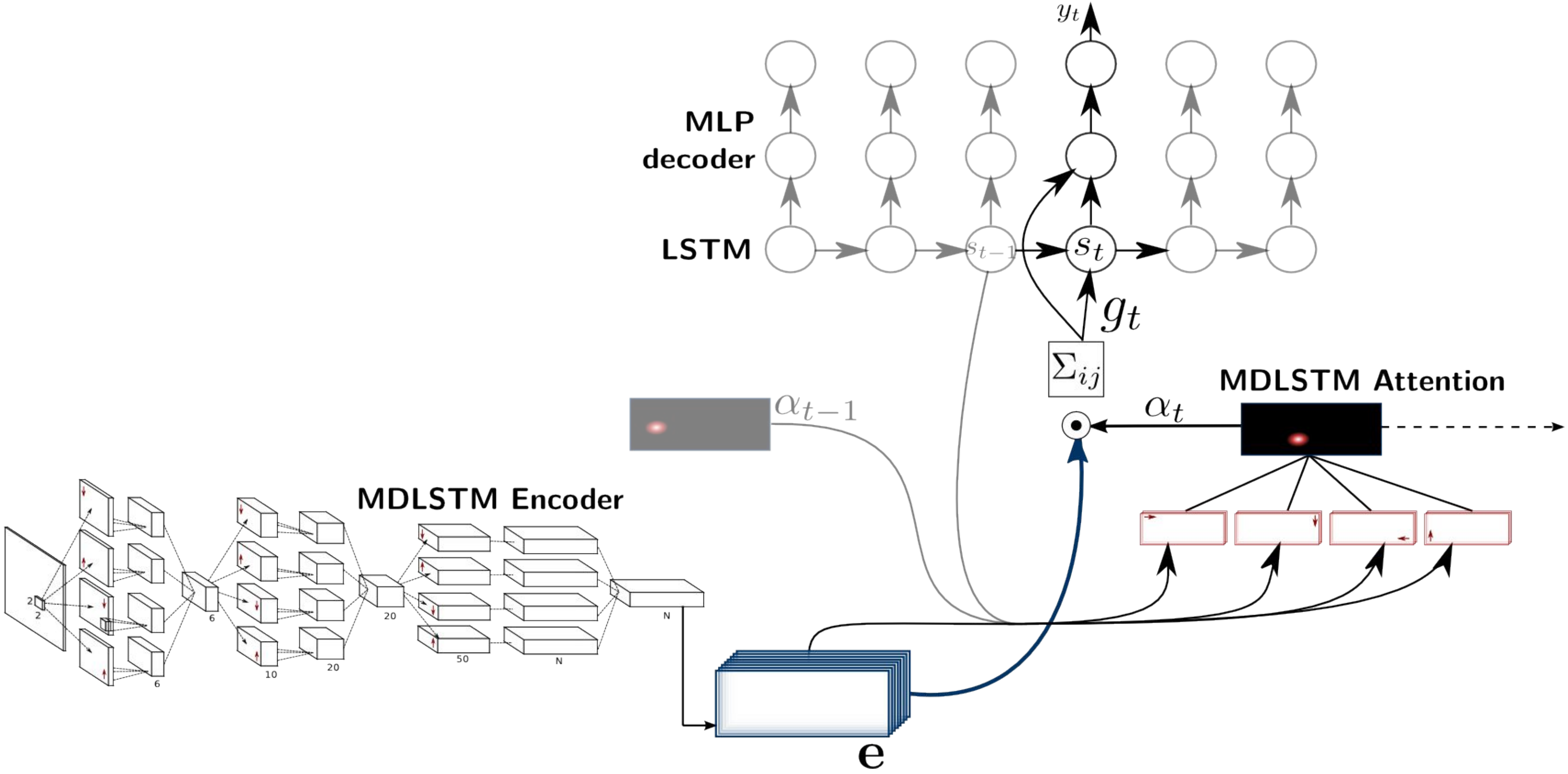
Challenges:

- ➔ How to define attention to smaller blocks to **recognize full, complex documents**?
- ➔ How do we **get training data / evaluation** in that context?
- ➔ How to make the models **faster / more efficient**?

Thanks!
Questions /Discussion

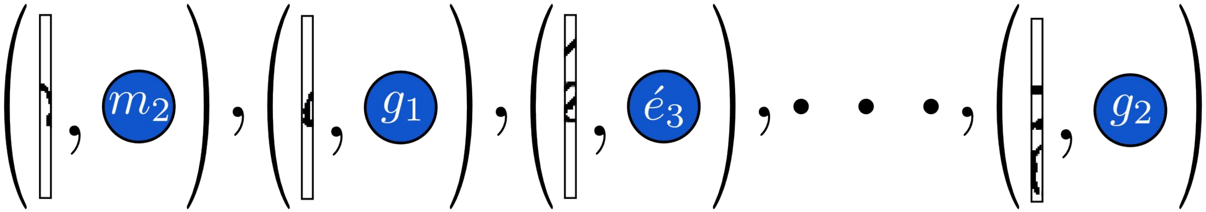
Theodore Bluche

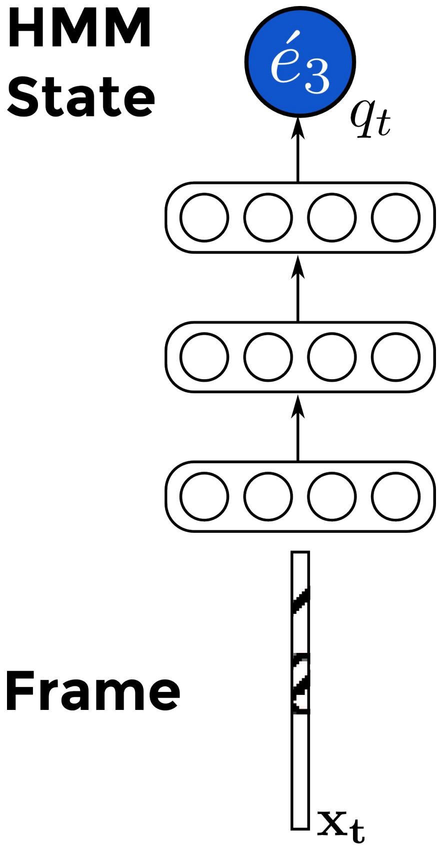
“Scan, Attend and Read”



Frame classification (MLP style)

- ➔ **Input** = one frame = one vector of pixel or feature values
- ➔ **Output** = posterior probabilities over HMM states (or sometimes characters)



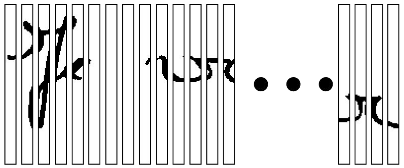


Training :

- ➔ Collect a dataset of (xt, qt) = frames with correct HMM state
- ➔ Minimize *- log p(qt | xt)*
- ➔ Measure the **Frame Error Rate** (% of frames with wrong HMM state prediction)

Sequence classification

- ➔ To train the network **directly with** frame sequences and **character sequences**
- ➔ i.e. no need to label each frame with an HMM state

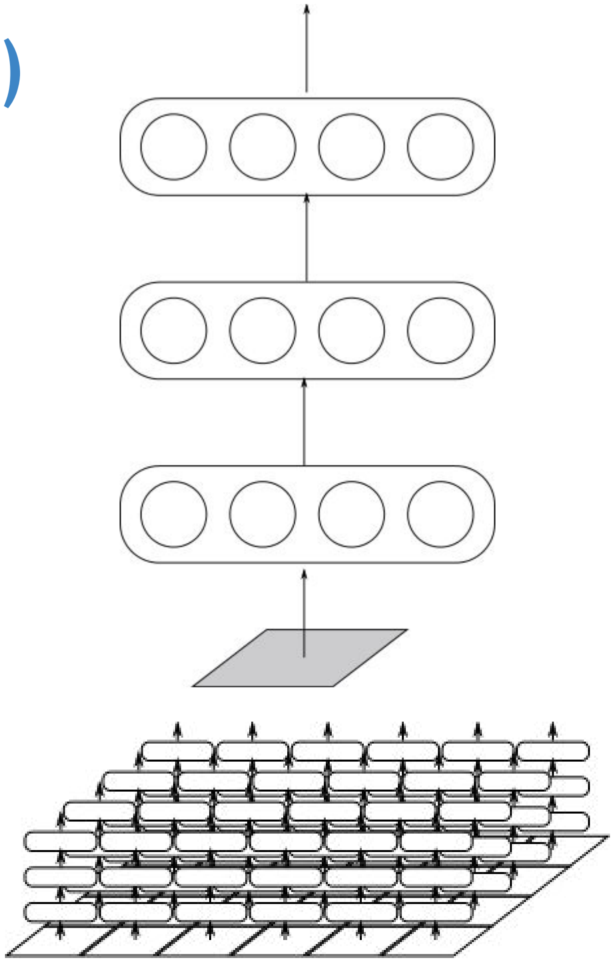
Minimize : ( , **J e _ v o … r**) , . . .

$$-\log p\,(c1, c2, \ldots cN \mid \mathbf{x} = x1, x2, \ldots xT)$$

- ➔ Measure the **Character Error Rate** (% of character substitutions, deletions or insertions)

Sequence sizes are not equal !!!

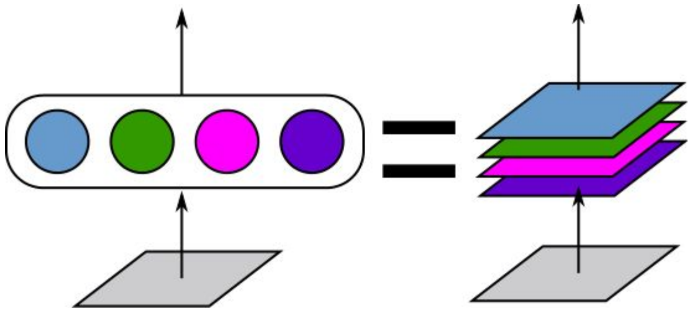
Neural Networks for Images (pixel level)



→ Instead of a feature vector, the **input is only one pixel value** (or a vector of 3 RGB values for color images)

→ The network is ***replicated*** at each position in the image

Feature Maps

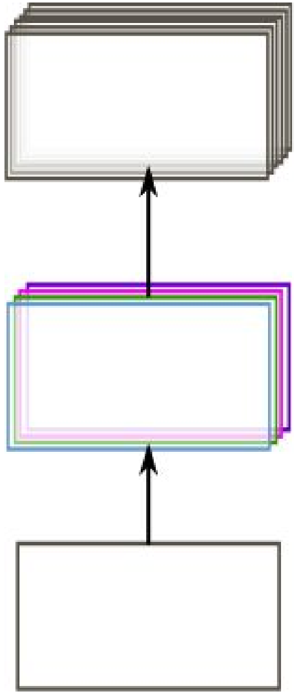


→ The outputs of one hidden layer for a pixel may be viewed as new *"pixel"* values, defining new channels

→ Since the network is replicated, each output have a similar meaning across all pixels (but different values)

→ So a given output across the whole image defines a new (kind of) image : a feature map

in the end, it's just a way of *representing or interpreting* the net...



e.g. Convolutional Neural Network

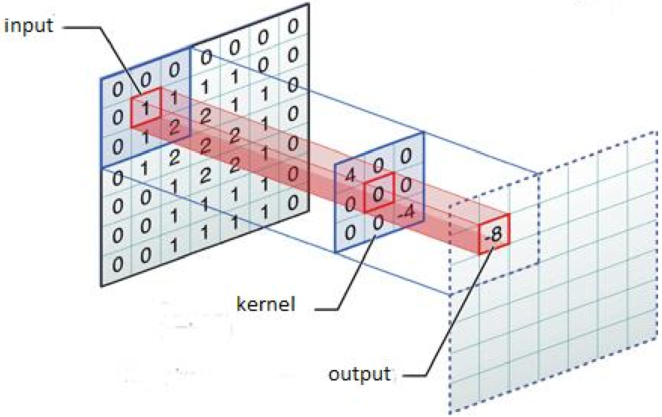
→ We can include spatial (structured) context :

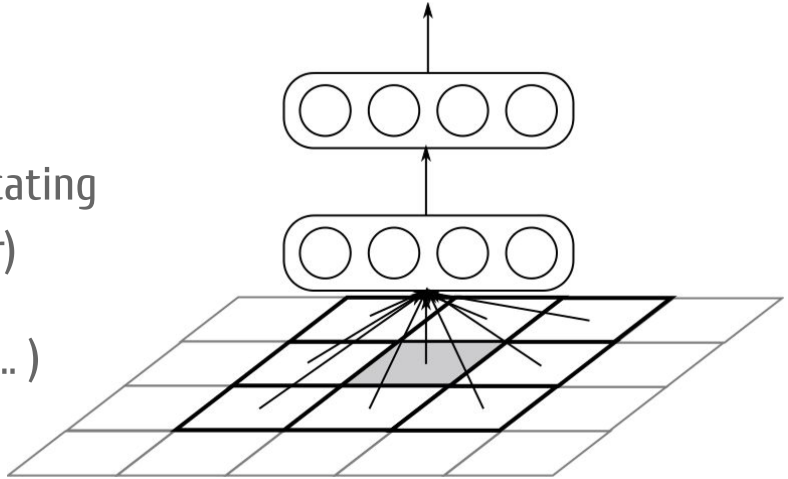
instead of giving 1 pixel value at the current position, we give the values of all pixels in a given neighborhood

→ This is still replicated at all positions = **convolution**, with kernel defined by the weights

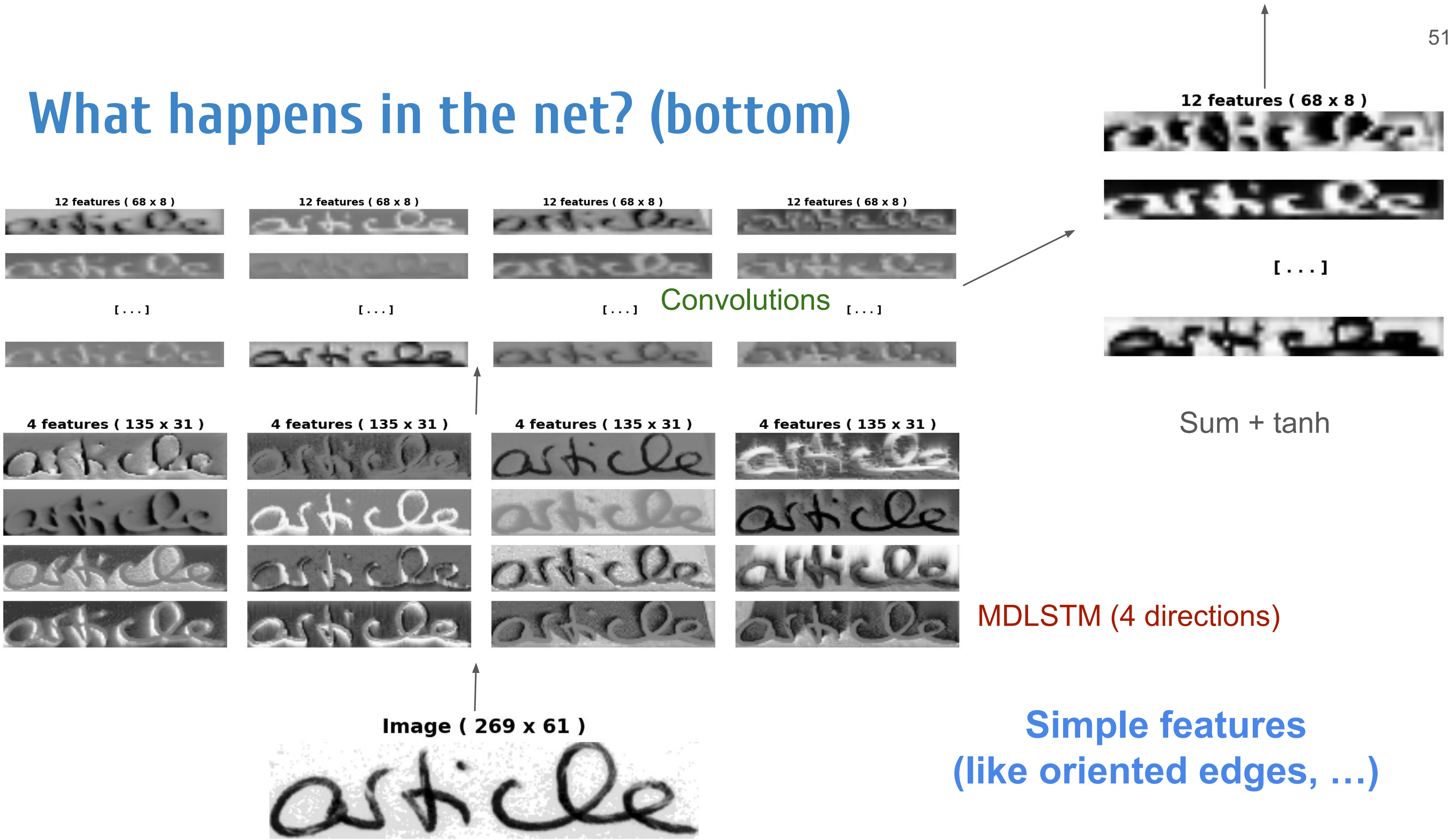
→ You can reduce the size of the feature maps by replicating the net every *N* positions (output will be *N* times smaller)

(nb: also possible to have convolution in sequential nets...)



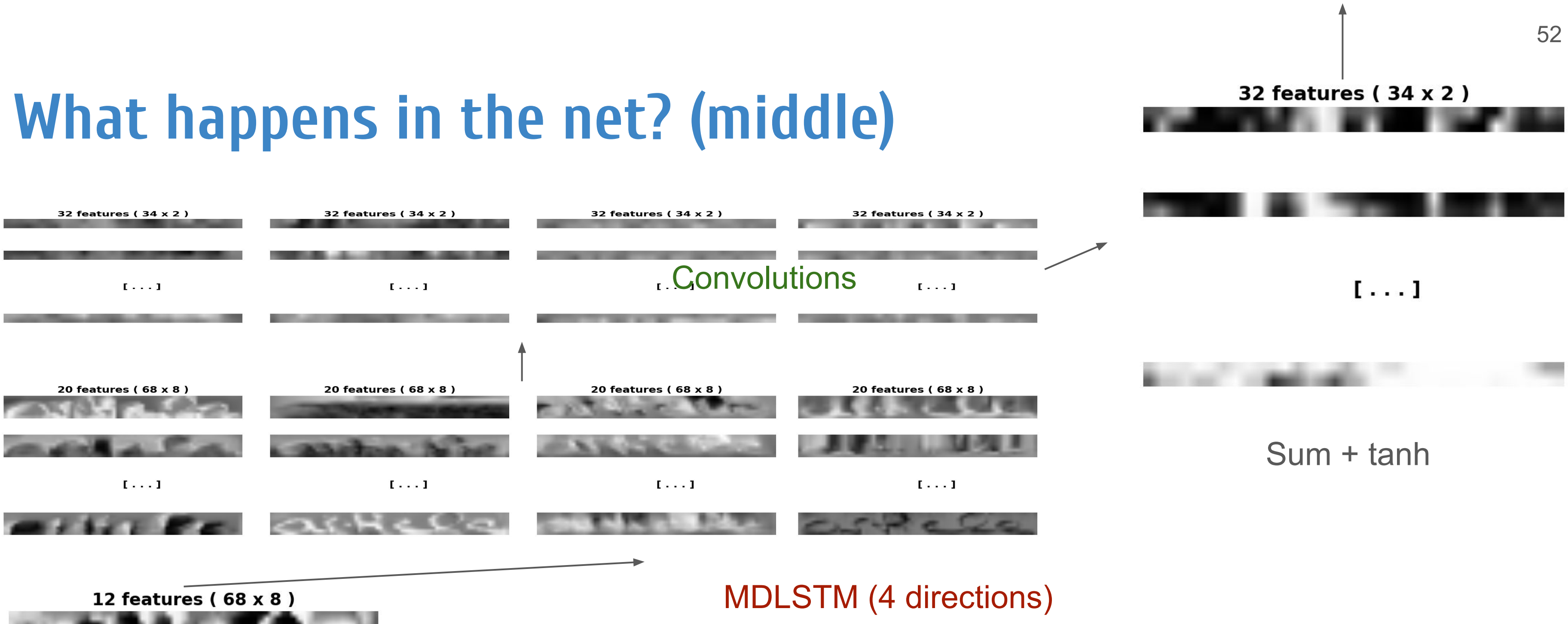


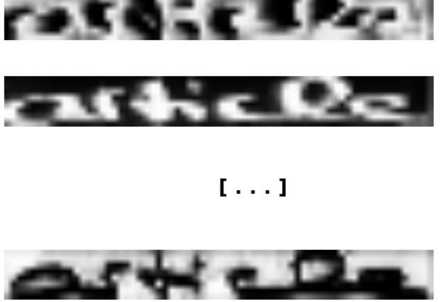
What happens in the net? (bottom)



Simple features
(like oriented edges, …)

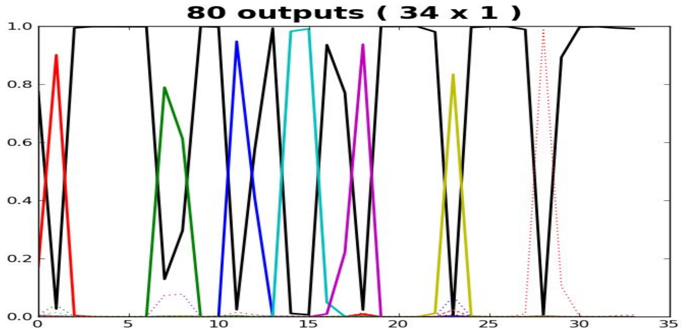
What happens in the net? (middle)





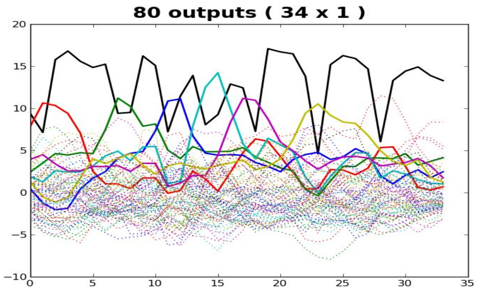
Complex features (like loops, ascenders, vertical strokes, …)

What happens in the net? (top)

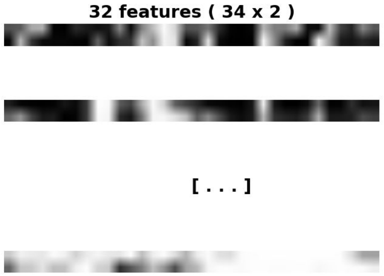


Softmax

Collapse



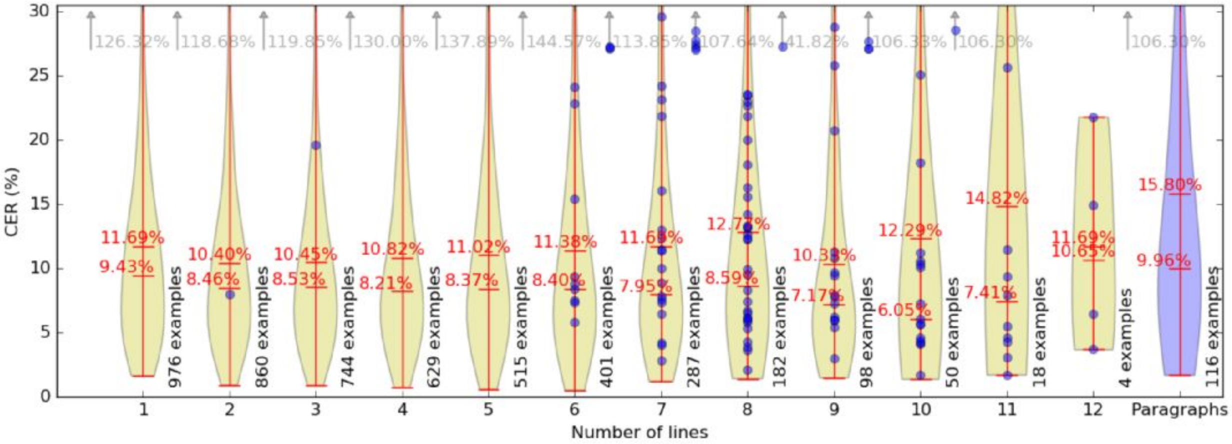
MDLSTM (4 directions)

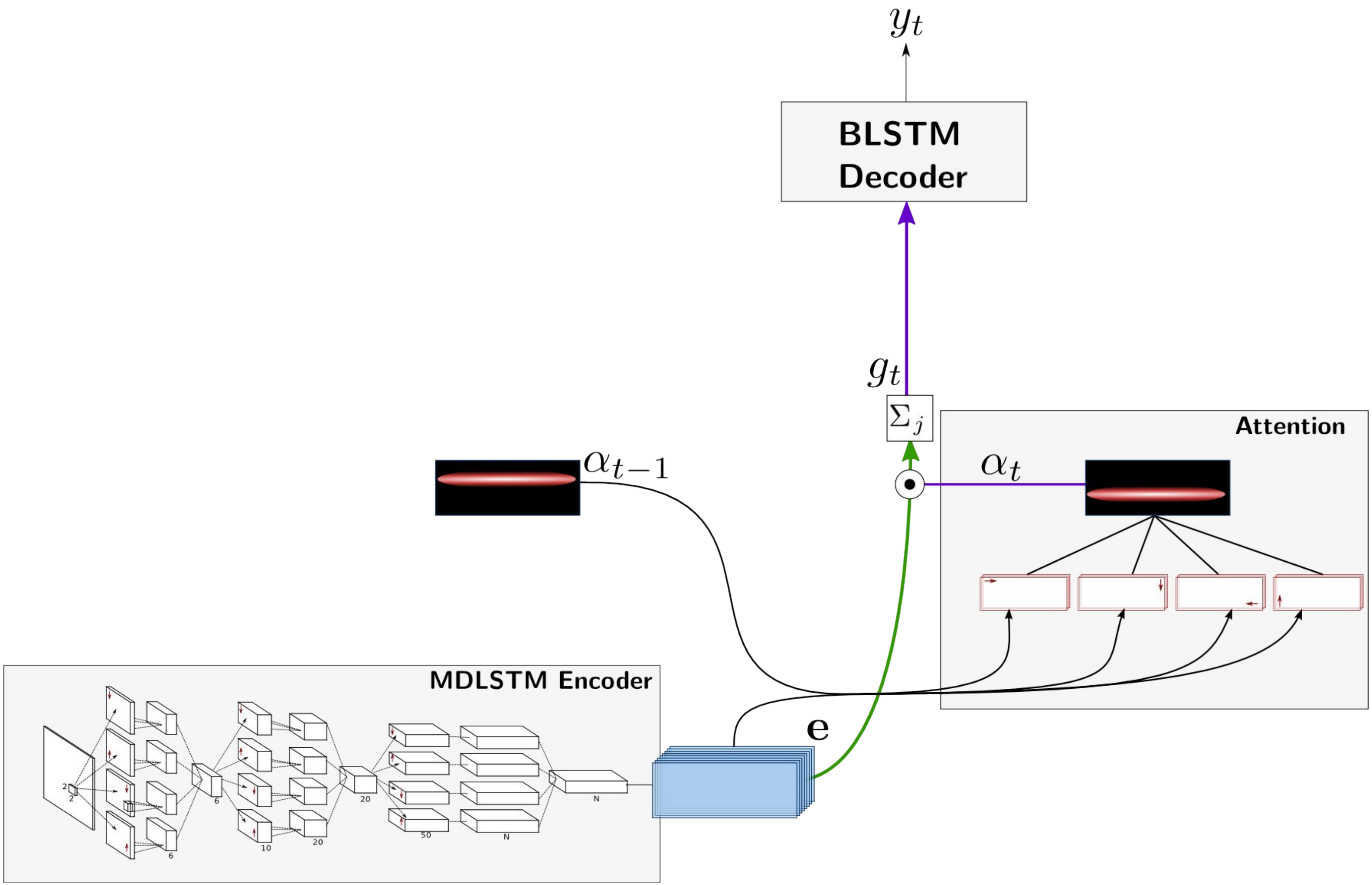


More abstract features (combination of features, closer to character level…)

Results (Character Error Rate / IAM)

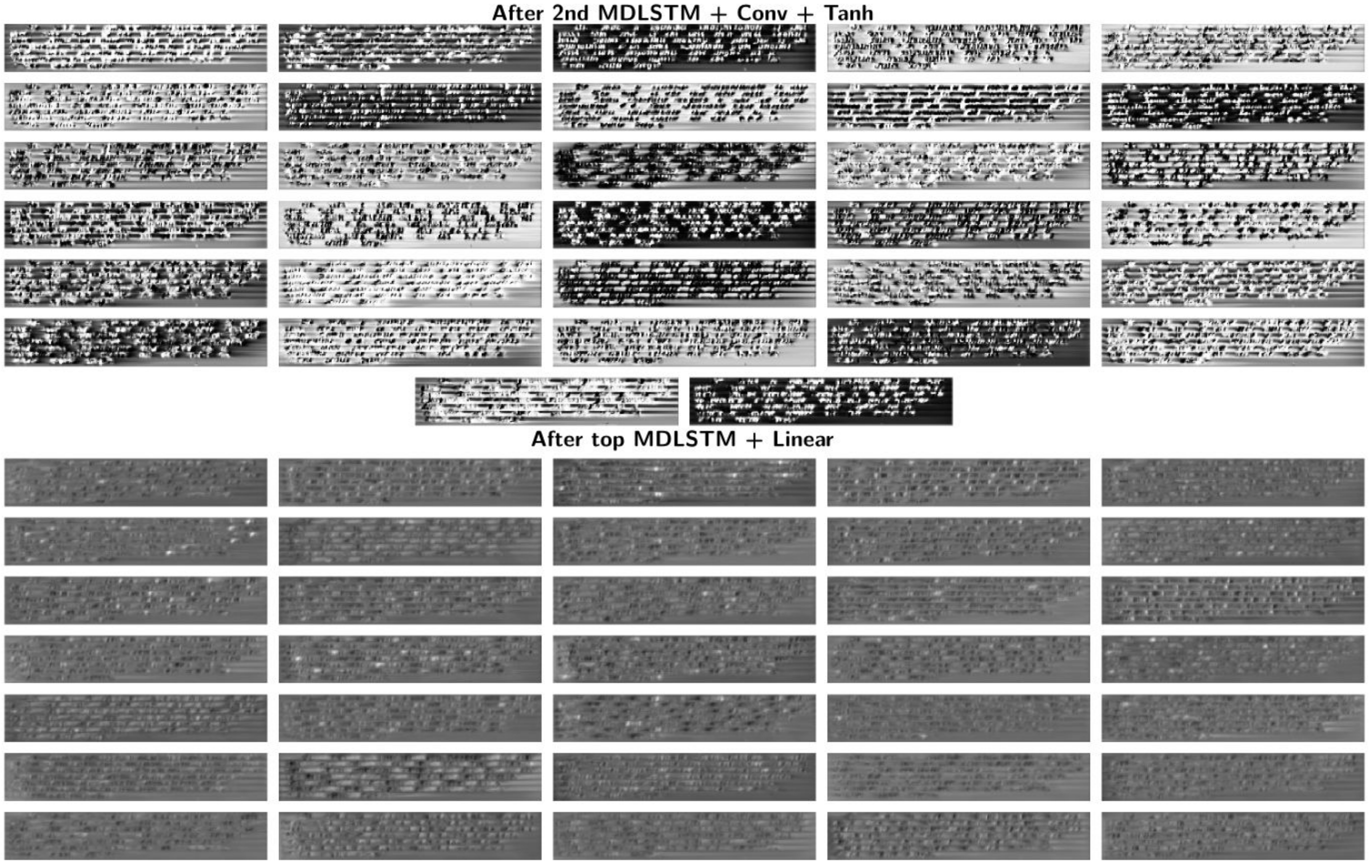
Resolution (DPI)	Line segmentation				Attention-based *(this work)*
	GroundTruth	**Projection**	**Shredding**	**Energy**	
90	18.8	24.7	19.8	20.8	-
150	10.3	17.2	11.1	11.8	16.2
300	6.6	13.8	7.5	7.9	-





y_t
BLSTM
Decoder
g_t
Σ_j
Attention
α_{t-1}
α_t
MDLSTM Encoder
$\mathbf{e}$

Encoder’s Activations



Comparison with Explicit Line Segmentation

➔ … partly because the BLSTM decoder can model dependencies across text lines

BLSTM after collapse but limited to textlines

BLSTM after attention on full paragraphs

Collapse	Decoder	IAM	Rimes
Standard	Softmax	8.4	4.9
Standard	BLSTM + Softmax	7.5	4.8
Attention	BLSTM + Softmax	6.8	2.5

In the literature of …

Computer Vision

[CONV.NET]

Image input

NLP

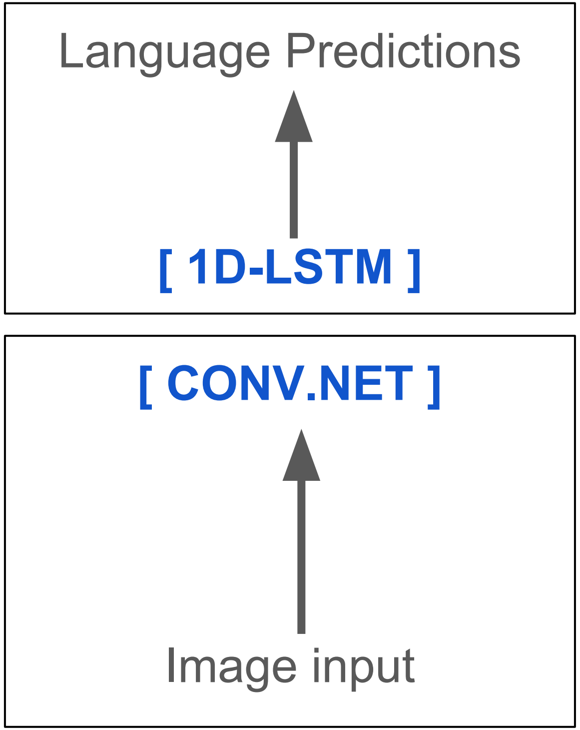
Language Predictions

[1D-LSTM]

Model Compression

- Pruning
  - 20x smaller models
  - But start with huge models >50MB
- Weight Quantization

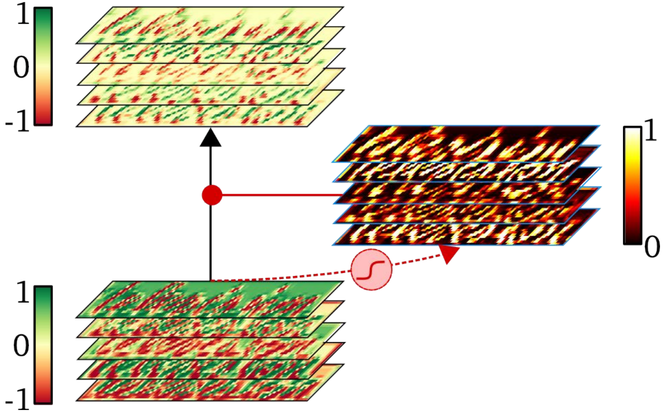
Proposed model

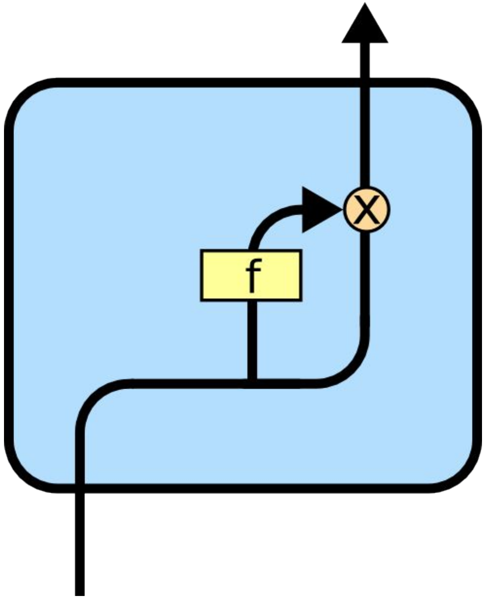


- ➔ Connected with any kind of vertical aggregation (max pooling, collapse, attention, …)
- ➔ We can make the convnet a generic multi-task, multi-language encoder (e.g. use it to predict the language in order to select the appropriate LSTM model, and to provide inputs to this LSTM)

Gates

- Conv 3x3 with appropriate padding and stride 1
- Sigmoid
- Ouput = Result x Input





Gated NN archi.

Many tested, this one works quite well
(at least for HWR…)

- Most (~80%) of the parameters after the max-pooling
- Most (~80%) of the processing time in the convolution