A guard reported that at East Croydon he had seen what was accepted as the same couple sitting close together 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 Victoria and changing at East Croydon. He also believed that they had still been together at South Croydon, and he remembered
Offline Handwriting Recognition

→ Challenges
  o the input is a **variable-sized two-dimensional** image
  o the output is a **variable-sized sequence** of characters
  o the cursive nature of handwriting makes a prior segmentation into characters difficult

→ Methods
  o Over-segmentation and group-of-segments scoring (90s)
  o Sliding window approach with HMMs (2000s) or neural nets (2000-2010s)
  o **MDLSTM** = models handling both the 2D aspect of the input and the sequential aspect of the prediction → **state-of-the-art**

“Look!” he went on earnestly. “You’re not… to write up Pericles in some way or other… you?” “What is there to write up?” said Bawley. “There are forty-three submarines… the Navy - why should I pick on Pericles?” John eyed him steadily. “It’s the Principal business, isn’t it?” he said quietly.
Limitations

➔ Current systems **require segmented text lines**
  o For training = tedious annotation effort or error-prone automatic mapping methods
  o 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

The output predictions are computed with a Maxout network using two filters per unit.

The BiRNN is used to initialize the first state of the decoder.

BiRNN:
Input is 1024 features per frame
Each recurrent layer has 512 hidden units, thus the annotation is 1024-dimensional.

Deep Maxout network reads 11 frames (440 features) and uses 3 hidden layers of 1024 maxout units each using 5 filters.

Encoder RNN:
Computes an annotation for each input frame.

Decoder RNN:
Recursively predicts the next phoneme, input annotations are accessed through a context computed separately for each output.

Context: A score is computed to match the previous hidden state to all input annotations. The context is a weighted combination of the most closely matching annotations.

Input sequence:
Frames of 40 fMLLR features.
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:

\[
\begin{align*}
\text{AAABBCCCC} & \rightarrow \text{ABC} \\
\text{ABBBBBCCCD} & \rightarrow \text{ABC} \\
\vdots \\
\text{AAAABCCCD} & \rightarrow \text{ABC}
\end{align*}
\]

\[
p(c_1 \ldots c_N | x) = \sum_{y_1 \ldots y_T \rightarrow c_1 \ldots c_N} p(y_1 \ldots y_T | x)
\]

\[
= \sum_{y_1 \ldots y_T \rightarrow c_1 \ldots c_N} \prod_t p(y_t | x)
\]

= Net's output at time $t$

→ Computed efficiently with **dynamic programing**
→ **Problem** how to output $\text{ABB}$ ($\text{AAABBBBBBD} \rightarrow \text{AB}$)?
Connectionist Temporal Classification (CTC)

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

\[
\begin{align*}
AAABBCCC & \rightarrow ABC \\
AA-BB--C & \rightarrow A-B-C \rightarrow ABC \\
\ldots \\
-A--B--C & \rightarrow -A-B-C \rightarrow ABC \\
AAABBBBBBB & \rightarrow AB \\
AA-BB--B & \rightarrow A-B-B \rightarrow ABB \\
\ldots \\
-A--B--B & \rightarrow -A-B-B \rightarrow ABB
\end{align*}
\]
The “Collapse” layer

\[ z_j = \sum_i f_{ij} \]

- 2D \( \rightarrow \) 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

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”

"Look!" he went on earnestly. "You're not going to..."
Network’s architecture

➔ Encoder
\[ f_{i,j} = Encoder(I) \]

➔ Attention
\[ \omega_{i,j}^{(t)} = Attention(f, \omega^{(t-1)}, s_{t-1}) \]
\[ z_t = \sum_{i,j} \omega_{i,j}^{(t)} f_{i,j} \]

➔ 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

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

➔ We include a special token \textbf{EOS} at the end of the target sequences (also predicted by the network to \textit{indicate when to stop reading} at test time)

➔ The net has to predict the correct character at each timestep
## Text Lines

<table>
<thead>
<tr>
<th>Model</th>
<th>Inputs</th>
<th>CER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDLSTM + CTC</td>
<td>Full Lines</td>
<td>6.6</td>
</tr>
<tr>
<td>Attention-based</td>
<td>1 word</td>
<td>12.6</td>
</tr>
<tr>
<td></td>
<td>2 words</td>
<td>9.4</td>
</tr>
<tr>
<td></td>
<td>3 words</td>
<td>8.2</td>
</tr>
<tr>
<td></td>
<td>4 words</td>
<td>7.8</td>
</tr>
<tr>
<td></td>
<td>Full Lines</td>
<td>7.0</td>
</tr>
</tbody>
</table>
Learning Line Breaks

<table>
<thead>
<tr>
<th>Two lines of...</th>
<th>CER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 words</td>
<td>11.8</td>
</tr>
<tr>
<td>2 words</td>
<td>11.1</td>
</tr>
<tr>
<td>3 words</td>
<td>10.9</td>
</tr>
<tr>
<td>Full Lines</td>
<td>9.4</td>
</tr>
</tbody>
</table>
Paragraph Recognition
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)

<table>
<thead>
<tr>
<th>Resolution (DPI)</th>
<th>Line segmentation</th>
<th>Attention-based (this work)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ground Truth</td>
<td>Projection</td>
</tr>
<tr>
<td>90</td>
<td>18.8</td>
<td>24.7</td>
</tr>
<tr>
<td>150</td>
<td>10.3</td>
<td>17.2</td>
</tr>
<tr>
<td>300</td>
<td>6.6</td>
<td>13.8</td>
</tr>
</tbody>
</table>
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

\[ \omega_{ij}^{(t)} = \frac{e^{m_{ij}^{(t)}}}{\sum_i e^{m_{i,j}^{(t)}}} \quad z_j^{(t)} = \sum_i \omega_{ij}^{(t)} f_{ij} \]

→ **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 Croydon he had seen what was accepted as the same couple sitting close together 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 Victoria and changing at East Croydon. 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. Étant 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 …

<table>
<thead>
<tr>
<th>Database</th>
<th>Resolution</th>
<th>GroundTruth</th>
<th>Projection</th>
<th>Shredding</th>
<th>Energy</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAM</td>
<td>150 dpi</td>
<td>8.4</td>
<td>15.5</td>
<td>9.3</td>
<td>10.2</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>300 dpi</td>
<td>6.6</td>
<td>13.8</td>
<td>7.5</td>
<td>7.9</td>
<td>4.9</td>
</tr>
<tr>
<td>Rimes</td>
<td>150 dpi</td>
<td>4.8</td>
<td>6.3</td>
<td>5.9</td>
<td>8.2</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>300 dpi</td>
<td>3.6</td>
<td>5.0</td>
<td>4.5</td>
<td>6.6</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Comparison with Explicit Line Segmentation

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

<table>
<thead>
<tr>
<th>Collapse</th>
<th>Decoder</th>
<th>IAM</th>
<th>Rimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Softmax</td>
<td>8.4</td>
<td>4.9</td>
</tr>
<tr>
<td>Standard</td>
<td>BLSTM + Softmax</td>
<td>7.5</td>
<td>4.8</td>
</tr>
<tr>
<td>Attention</td>
<td>BLSTM + Softmax</td>
<td>6.8</td>
<td>2.5</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Method</th>
<th>Processing time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroundTruth</td>
<td>(crop+reco)</td>
</tr>
<tr>
<td>Shredding</td>
<td>(segment+crop+reco)</td>
</tr>
<tr>
<td>Scan, Attend and Read</td>
<td>(reco)</td>
</tr>
<tr>
<td>This Work</td>
<td>(reco)</td>
</tr>
</tbody>
</table>
## Final Results

<table>
<thead>
<tr>
<th></th>
<th>NIPS Paper</th>
<th>Rimes</th>
<th>IAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WER%</td>
<td>CER%</td>
</tr>
<tr>
<td>150 dpi</td>
<td>no language model</td>
<td>13.6</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>with language model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300 dpi</td>
<td>no language model</td>
<td>12.6</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>with language model</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bluche, 2015</td>
<td>11.2</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>Doetsch et al., 2014</td>
<td>12.9</td>
<td>4.3</td>
</tr>
<tr>
<td></td>
<td>Kozielski et al. 2013</td>
<td>13.7</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>Pham et al., 2014</td>
<td>12.3</td>
<td>3.3</td>
</tr>
<tr>
<td></td>
<td>Messina &amp; Kermorvant, 2014</td>
<td>13.3</td>
<td>-</td>
</tr>
</tbody>
</table>

**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)

\[
\begin{pmatrix}
\epsilon_1 \\
m_2 \\
g_1 \\
\epsilon_3 \\
\vdots \\
g_2
\end{pmatrix}
\]

**Training:**

→ Collect a dataset of \((x_t, q_t) = \text{frames with correct HMM state}\)
→ Minimize \(- \log p(q_t | x_t)\)
→ 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:

\[-\log p (c_1, c_2, ... c_N \mid x = x_1, x_2, ... x_T)\]

➔ 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, ...)

MDLSTM (4 directions)

Convolutions

Sum + tanh

Image (269 x 61)

12 features (68 x 8)

12 features (68 x 8)

12 features (68 x 8)

12 features (68 x 8)
What happens in the net? (middle)

Convolutions

MDLSTM (4 directions)

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

Sum + tanh
What happens in the net? (top)

Collapse

MDLSTM (4 directions)

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

Softmax
## Results (Character Error Rate / IAM)

<table>
<thead>
<tr>
<th>Resolution (DPI)</th>
<th>GroundTruth</th>
<th>Line segmentation</th>
<th>Attention-based (this work)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>18.8</td>
<td>24.7</td>
<td>20.8</td>
</tr>
<tr>
<td>150</td>
<td>10.3</td>
<td>17.2</td>
<td>11.8</td>
</tr>
<tr>
<td>300</td>
<td>6.6</td>
<td>13.8</td>
<td>7.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

![Graph showing CER (%) vs. Number of lines]

- CER (%): Character Error Rate
- DPI: Dots Per Inch
- GroundTruth
- Projection
- Shredding
- Energy
- Attention-based (this work)

The graph illustrates the character error rate (%) for different resolutions and line segmentations, with each data point representing a specific number of examples. The data suggests a decrease in error rate as resolution increases, with particular emphasis on the attention-based (this work) method showing significant improvement.
Encoder’s Activations

After 2nd MDLSTM + Conv + Tanh

After top MDLSTM + Linear
Comparison with Explicit Line Segmentation

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

<table>
<thead>
<tr>
<th>Collapse</th>
<th>Decoder</th>
<th>IAM</th>
<th>Rimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Softmax</td>
<td>8.4</td>
<td>4.9</td>
</tr>
<tr>
<td>Standard</td>
<td>BLSTM + Softmax</td>
<td>7.5</td>
<td>4.8</td>
</tr>
<tr>
<td>Attention</td>
<td>BLSTM + Softmax</td>
<td>6.8</td>
<td>2.5</td>
</tr>
</tbody>
</table>
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
- Output = 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