Methods for End-to-End Handwritten Paragraph Recognition

Théodore Bluche

tb@a2ia.com

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
  ○ Isolated character classification
  ○ 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

→ neural networks implementing a sort of attention mechanism

→ end-to-end systems that learn to focus on specific parts of their input in order to make predictions
  ○ Machine translation
  ○ Speech Recognition
  ○ Image captioning
  ○ Question Answering
  ○ …

→ We propose to replace line segmentation with this kind of attention model
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 MDLSTM

- Text line images are fed to a Multi-Dimensional LSTM layer
- Feature maps are subsampled by convolutional layers
- At the end, there is one feature map per character
- They are collapsed in the vertical dimension to obtain sequences of character predictions
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 $j$
3. the output sequence will have length $W$, i.e. the width of the feature maps, so at most $W$ characters can be recognized
4. the ordering in the sequence will follow the same (spatial) ordering as the feature maps

$z_j = \sum_i f_{ij}$

→ 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

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

→ the length of the output sequence is independent of the dimensions of the image
→ at each timestep, a map of weights \( \{\omega(t)_{ij}\} \) is computed with a neural network
→ the feature maps are multiplied by these weights, and summed to obtain one vector (summary) \( z_t \)
→ the \( t \)-th character is predicted from vector \( z_t \)

This is the "Scan, Attend and Read" model.
Weighted Collapse
recognize one line at a time

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

➔ intermediate solution between the weighted summary and the standard collapse
➔ amounts to a standard collapse on the weighted sum
➔ the length of the t-th sequence is the width of the feature maps
➔ the weights are recomputed at each time step
➔ the t-th text line is recognized from sequence \( z(t) \)

This is the "Joint Line Segmentation and Transcription" model.
Proposed modifications
"Scan, Attend and Read"

"Look!" he went on earnestly. "You're not going to

\[ Y_t \]

Decoder
Neural Network

\[ s_t \]

LSTM
\[ s_{t-1} \]

\[ g_t \]

\[ \sum_{ij} \]

Attention
Neural Network

Attention

Input Image

"Look!" he went on earnestly. "You're not going to write up bricks in some way or other, are you!" "What is that to write up!" said Bowley. "There are forty-three submarines in the navy - why should I pick on Barker!"

John eyed him steadily. "It's a beautiful business, isn't it?" he said quietly.
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 every positions**. The scores are **normalized with a softmax**.

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

→ **Attention = MDLSTM layer**, → the attention potentially depends on the context of the whole image.

→ the LSTM gating system **allows the network to use the content at one location to predict the attention weight for another location.**(overt and covert attention).
Model Training

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

- 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)
- No "blank/garbage" token as in CTC
- The net has to predict the correct character at each timestep
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
### Text Lines

telling even children, telling even children,
telling even children, telling even children,

telling even children, telling even children,

telling even children, telling even children,

<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

The effect of bottom congestion on the light pressure is inevitably mixed with that of the suitability of ground for spawning. Both result in crowding, so there is no need to try to separate them - thank Heaven! A good picture of this is seen on the 150 miles of spawning grounds from the Viking in the north down to the Klondykes and the Reef along the western edge of the Norwegian Deep.

...jel man's charm was disarming. Yet when the time came to leave, taarr felt as de—pressed as w...
## Results (Character Error Rate / IAM)

<table>
<thead>
<tr>
<th>Resolution (DPI)</th>
<th>GroundTruth</th>
<th>Projection</th>
<th>Shredding</th>
<th>Energy</th>
<th>Attention-based (this work)</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>18.8</td>
<td>24.7</td>
<td>19.8</td>
<td>20.8</td>
<td>-</td>
</tr>
<tr>
<td>150</td>
<td>10.3</td>
<td>17.2</td>
<td>11.1</td>
<td>11.8</td>
<td>16.2</td>
</tr>
<tr>
<td>300</td>
<td>6.6</td>
<td>13.8</td>
<td>7.5</td>
<td>7.9</td>
<td>-</td>
</tr>
</tbody>
</table>
Encoder’s Activations

After 2nd MDLSTM + Conv + Tanh

After top MDLSTM + Linear
Pros & Cons

➔ Can potentially handle any reading order
➔ Can output character sequences of any length
➔ Can recognize paragraphs (and maybe complete document?)

➔ Very slow (one fprop in the attention network and decoder for each character = about 500 times for a complete paragraph) + Requires a lot of memory during training (same reasons)
➔ How to integrate with language models?
➔ Not quite close to state-of-the-art performance on paragraphs (for now...)
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

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

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

→ 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>Ground Truth</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>150 dpi</th>
<th>Rimes</th>
<th>IAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>no language model</td>
<td>WER%: 13.6, CER%: 3.2</td>
<td>WER%: 29.5, CER%: 10.1</td>
</tr>
<tr>
<td>with language model</td>
<td></td>
<td>WER%: 16.6, CER%: 6.5</td>
</tr>
<tr>
<td>300 dpi</td>
<td>Rimes</td>
<td>IAM</td>
</tr>
<tr>
<td>no language model</td>
<td>WER%: 12.6, CER%: 2.9</td>
<td>WER%: 24.6, CER%: 7.9</td>
</tr>
<tr>
<td>with language model</td>
<td></td>
<td>WER%: 16.4, CER%: 5.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>References</th>
<th>Rimes</th>
<th>IAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluche, 2015</td>
<td>WER%: 11.2, CER%: 3.5</td>
<td>WER%: 10.9, CER%: 4.4</td>
</tr>
<tr>
<td>Doetsch et al., 2014</td>
<td>WER%: 12.9, CER%: 4.3</td>
<td>WER%: 12.2, CER%: 4.7</td>
</tr>
<tr>
<td>Kozielski et al. 2013</td>
<td>WER%: 13.7, CER%: 4.6</td>
<td>WER%: 13.3, CER%: 5.1</td>
</tr>
<tr>
<td>Pham et al., 2014</td>
<td>WER%: 12.3, CER%: 3.3</td>
<td>WER%: 13.6, CER%: 5.1</td>
</tr>
<tr>
<td>Messina &amp; Kermorvant, 2014</td>
<td>WER%: 13.3, CER%: -</td>
<td>WER%: 19.1, CER%: -</td>
</tr>
</tbody>
</table>
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 & 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! Gracias!

Questions /Discussion

Theodore Bluche <tb@a2ia.com>
“Scan, Attend and Read”