End-to-End Handwritten Text Detection and Transcription in Full Pages

Manuel Carbonell
omni:us
Berlin, Germany
manuel@omnius.com

Joan Mas
Computer Vision Center
Barcelona, Spain
jmas@cvc.uab.cat

Mauricio Villegas
omni:us
Berlin, Germany
mauricio@omnius.com

Alicia Fornés
Computer Vision Center
Barcelona, Spain
jmas@cvc.uab.cat

Josep Lladós
Computer Vision Center
Barcelona, Spain
josep@cvc.uab.cat

***Abstract*—When transcribing handwritten document images, inaccuracies in the text segmentation step often cause errors in the subsequent transcription step. For this reason, some recent methods propose to perform the recognition at paragraph level. But still, errors in the segmentation of paragraphs can affect the transcription performance. In this work, we propose an end-to-end framework to transcribe full pages. The joint text detection and transcription allows to remove the layout analysis requirement at test time. The experimental results show that our approach can achieve comparable results to models that assume segmented paragraphs, and suggest that joining the two tasks brings an improvement over doing the two tasks separately.**

***Index Terms*—Handwritten Text Recognition; Layout Analysis; Text segmentation; Deep Neural Networks; Multi-task learning**

I. Introduction

In the last years, the performance of handwritten text recognition (HTR) methods has significantly improved with the arrival of new deep convolutional network architectures and attention models [1], [2]. Nevertheless, when transcribing document images, layout analysis (i.e. word, line or paragraph segmentation) is a required previous task that usually supposes a source of error [3], [4]. Traditionally, methods for transcription of handwritten documents rely on the output of some post-processing steps to obtain the different segmented objects, i.e., lines or words depending on the level of recognition that the method works [5], [6]. It is for sure known that the performance of those methods is conditioned by the correctness of the output from the segmentation step. In the other way around to provide a good segmentation it would be beneficial to have the transcription of the word. This dilemma is defined by the well-known Sayre's paradox: a good segmentation is necessary for a good recognition and vice-versa.

Many HTR methods perform a joint segmentation and recognition at line level to cope with the above mentioned paradox. In this way, they can avoid the segmentation at character and word level. However, this is only partially solving the segmentation problem, because lines that are not properly segmented obviously affect the recognition at line level. For this reason, some recent approaches propose to recognize text at paragraph level [2], [7]. But still, an inaccurate segmentation into paragraphs will affect the HTR performance.

If we put the view into another domain such as the detection of text in the wild, where we encounter text in cluttered images, the task is usually divided into first localizing the text, and then transcribing the detected region [8]. Recent work in scene text detection [9] [10] [11] claimed that a single end-to-end model to jointly localize and transcribe words can reduce intermediate step error thereby leading to better detection results, and consequently, better transcriptions. The intuition behind this phenomena would be that giving transcription annotations gives additional valuable information to the detection model.

One may think that this principle applies in handwritten documents as well. In the few last years some works have appeared in the domain of the document transcription following an idea similar to the end-to-end models. However, in [12] this statement is put under doubt, since the best transcription performance is achieved by detecting the start of text line, segmenting it with a line follower and then transcribing it with three separately trained networks. Nevertheless no results are shown regarding the end-to-end trained model approach to come to a definitive conclusion.

In [13] a method to join the two tasks is proposed by predicting the text line beginning and letting the recognition network predict the characters till the end of the line without having an explicit end of line segmentation. The drawback of this method is that it does not attempt to backpropagate the recognition errors to the segmentation which might make it difficult to achieve a high performance on difficult benchmark datasets. In [14] the results of transcribing full paragraphs by implicitly segmenting lines with attention are comparable with the traditional automatic line segmentation methods, but the lack of a comparison using the state of the art neural segmentation separate system followed by the CRNN architecture prevents from concluding whether performing both tasks jointly supposes an advantage or not. In [15] a three-stage model is proposed by joining a two stage detection network with large feature extractor such as ResNet-50 with a recognition CNN.

In this paper we propose a end-to-end model for text detection and recognition at page level. Our method jointly performs text detection and transcription, and thus, the transcription network can benefit from shared features on the detection. In addition, we evaluate our method with an ablation study that suggests that there are benefits in the end-to-end approach over training two models separately.

The rest of the paper is organized as follows, first, in section

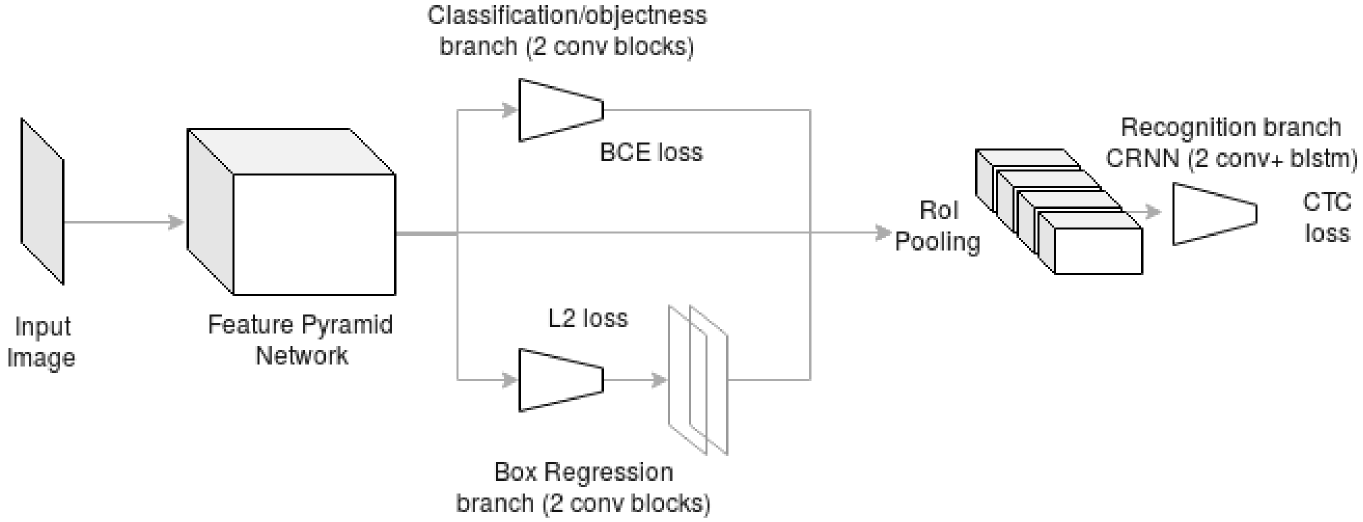

Classification/objectness branch (2 conv blocks)
BCE loss
Recognition branch CRNN (2 conv+ blstm)
RoI Pooling
CTC loss
Input Image
Feature Pyramid Network
L2 loss
Box Regression branch (2 conv blocks)

Fig. 1. Overview of the proposed method. We extract convolutional features using FPN. The classification and regression branches calculate the positive boxes and the recognition branch predicts the transcription of the content of each box. Binary cross entropy, squared-sum and CTC losses are backpropagated through the whole model.

II we describe the proposed model architecture, including the feature extractor, detection and recognition modules. In Section III, we test the proposed method and perform an ablation study to compare our approach with a traditional two-step method. In the last section we draw the conclusions of the work and outline possible continuation lines.

II. Methodology

As explained in the previous section, most of the existing approaches for automatic handwritten text recognition consist of separated models for localizing and transcribing the text. Contrary, in this work we propose a method to exploit the benefits of deep neural architectures for multitasking, and to evaluate its performance compared to traditional approaches. For this purpose, we built a neural model to recognize the text from either a page or paragraph image, by detecting each word and transcribing its content. The architecture consists of four connected deep neural networks, one for extracting page or paragraph features (ResNet18 + feature pyramid network), another for detecting the existence of text in each part of the image (classification/objectness branch), another to regress the bounding box of each one of the words in a image (regression branch), and a recognition branch (conv+blstm). An overview of the whole model can be seen in Figure 1.

A. Feature extractor

The first module of our model is a deep feature extractor, whose weights are shared for the recognition and detection tasks. Taking into account that the localization of text in a scanned document (where we are previously aware of its existence) might be easier than detecting an object in the wild, we have chosen the ResNet18 [16], a light state-of-the-art architecture for object detection and classification. This architecture consists of 5 convolutional residual blocks, i.e. 2 convolutions with rectifier linear unit activation and a residual connection.The detailed list of blocks and their configuration is shown in Table I.

TABLE I
ResNet18 architecture used for feature extraction.

Layer	output shape	kernel size	# kernels
res-conv-block 1	H/2·W/2	3 x 3	64
res-conv-block 2	H/4·W/4	3 x 3	64
res-conv-block 3	H/8·W/8	3 x 3	128
res-conv-block 4	H/16·W/16	3 x 3	256
res-conv-block 5	H/32·W/32	3 x 3	512

We have tried other even lighter configurations, but when reducing the amount of layers, we observed slower convergence and worse final results. This was most probably caused by noisy detections and false positives, confusing text with non-relevant text. For this reason we have chosen an intermediate depth architecture that allowed regressing the characteristics of the text, and skipping the step of separating the regions of interest (i.e. including the layout analysis step in the whole process).

Based on recent work on object detection, we build a feature pyramid network (FPN) [17] which combines the extracted deep features of different levels of abstraction by means of *deconvolutions*. These type of layers consist of bilinear interpolation to apply differentiable upscaling of the high level features, followed by convolutions to reduce the number of channels, allowing to add them to lower level features. A diagram of this module is shown in Figure 2. This approach gave a boost in performance to detect objects at different scales, which is definitely also a beneficial feature for localizing text in documents.

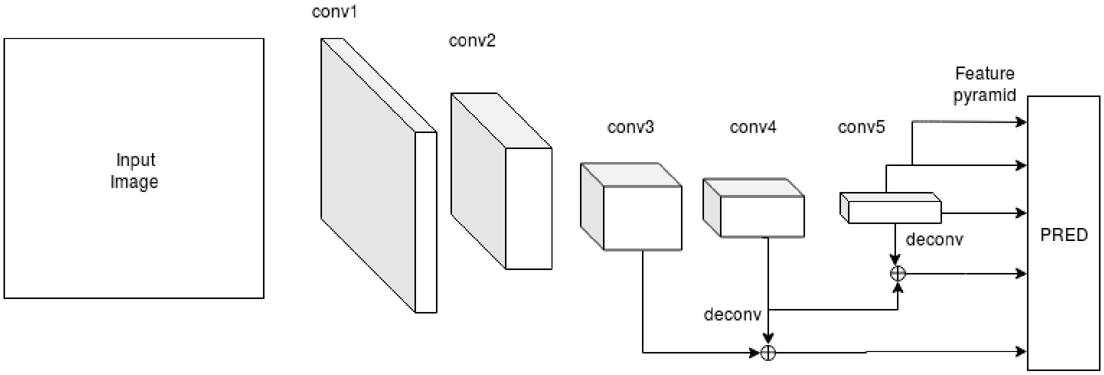


Fig. 2. Feature pyramid network. It consists of 5 ResNet18 convolutional blocks followed by residual connections plus deconvolutions [17]. The input is an image of shape $H \cdot W \cdot 3$, the output is 5 tensors of 256 channels with down sampling factors of 4,8,16,32 and 64 respectively.

B. Classification and Regression branches

For each one of the levels of the pyramid, the extracted features are fed to the classification network, which after four convolutions will predict the probability of the presence of a text object for each point of the image grid.

After predicting the class probabilities vector p_{cl}, the binary cross entropy (CE) loss is calculated and backpropagated through the classification branch and shared feature network. Formally, CE is computed as follows:

$$CE(p_{cl}) = -(y_{cl} \cdot log(p_{cl}) + (1 - y_{cl}) \cdot log(1 - p_{cl})) \quad (1)$$

In this case p_{cl} predicts the probability of the object of being text or not, i.e. it is used as an ”objectness” value but it could also be used to predict which kind of text it is (e.g. handwritten or printed). In a similar way, the regression network receives the whole page image features and after four convolutions, it regresses the box coordinate offsets from the predefined anchors. Formally:

$$\begin{aligned} x &= X + dx \cdot W \\ y &= Y + dy \cdot H \\ w &= e^{dw} \cdot W \\ h &= e^{dh} \cdot H \end{aligned} \quad (2)$$

where (x, y, w, h) are the predicted box coordinates, dx, dy, dw, dh are the predicted offsets and X, Y, W, H are the predefined anchor box values. The anchors are generated as the combination of the ratios $\frac{1}{2}$,1,2 and the scales 1, $2^{\frac{1}{3}}$, $2^{\frac{2}{3}}$ with a base size of 32 (9 anchors) as shown in figure II-B . The offsets are regressed by minimizing the mean square error:

$$MSE(\delta, \delta') = \frac{1}{n} \sum_{i}^{n} (\delta_i - \delta'_i)^2 \quad (3)$$

where δ_i is the vector of target offsets from the anchors for the i-th ground truth box.

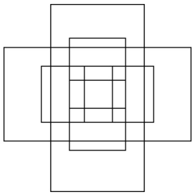


Fig. 3. We used 9 predefined anchors, result of combining three ratios $\frac{1}{2}$,1,2 and scales 1, $2^{\frac{1}{3}}$, $2^{\frac{2}{3}}$.

Once class probabilities and box offsets are predicted, the box sampler computes the bounding box coordinates for those anchors in the image grid whose class probability surpasses a given threshold. Once the box coordinates are calculated, we apply non-maximal suppression based on the maximum class score of each box. This will make that only a variable number of boxes with high confidence are going to be fed to the recognition branch.

C. Recognition branch

This module takes as inputs the features extracted with the FPN network and the corresponding regressed boxes and returns a probability tensor of variable width per alphabet length corresponding to the possible transcriptions of the words. Since the features are computed at page level we first need to apply some strategy that crops these features to be the input of the different successive layers in the module. This crop could be done in different ways, such as Region of Interest (ROI) pooling [18] strategies or affine transformation followed by Bilinear Sampling [19]. Since words in handwritten documents are mostly horizontally oriented and have no significant skew as it would happen in text in the wild data, for each regressed box, we apply RoI pooling to the page image features. Equivalently, for more complex distribution of documents with significant skew, or text orientation variations, affine transformations would allow to predict text skew or vertical orientation. The pooled features are rescaled to a fixed predefined height H' and variable width W', followed by padding to allow later addition of fully connected layers. For the recognition part we use a standard CRNN architecture containing 2 convolutional layers, 2 bidirectional long short-term memory layers [20], and a fully connected layer.

To optimize the model for recognition we calculate the Connectionnist Temporal Classification loss [21], which transforms variable-width feature tensor into a conditional probability distribution over label sequence, using a collapse function $\mathcal{B}$ that joins repeated output characters not separated by blank (non-character). For example $\mathcal{B}$(hheeel-llo)=hello. At each time step we choose the maximum probable character, however, we agree that prefix beam search decoding from recognition branch output matrix $P \in M(T_{steps} \cdot \text{alphabet length})$ probabilities using a language model might improve the performance.

Formally, the CTC loss function can be represented as the maximum likelihood error of the sequence labeling:

$$O^{ML}(\mathcal{D}, Y') = -\sum_{(x,y)\in D} ln(p(y|x)) \tag{4}$$

where $\mathcal{D} = (x, y)_{i=1}^{i=N}$ is the set of training input-target pairs and Y' is the set of network sequence outputs. To get the final full page transcription we concatenate the word predictions in a rule based reading order calculation from the word box coordinates. It consists of a projection of the word boxes to get text line continuation groups.

III. Experiments

This section is devoted to the experimental evaluation. We describe the dataset and the task to be realized, the different metrics and discuss the results.

A. Database and Task description

We tested our method on the IAM [22] database, a multi-writer handwritten document collection with ground-truth at page level. It consists of 1539 pages of scanned text, written by 657 different writers, including a total of 13.353 annotated text lines and 115.320 words. Pages were scanned at a resolution of 300dpi (2479 × 3542) saved as PNG images with 256 gray levels. For our work, due to our limited GPU space we rescale the images to 150dpi (1240 × 1753). In our partition we use 1198 page images, 747 for training and 451 for validation/test.

The objective task works as follows: Given an input sample page, the system localizes the text and transcribes it without any preprocessing steps based on layout analysis techniques, i.e. there is no segmentation into words, lines nor paragraphs.

B. Metrics

Two different metrics have been used in the experimental evaluation of the proposed methodology. One to evaluate the performance of the text detection and another for transcription.

To evaluate the performance in text localization, we used the mean Average Precision (mAP), the standard metric in object detection. Let $p = \frac{TP}{TP+FP}$ be the precision metric, i.e. the number of true positives out of the total positive detections; $r = \frac{TP}{TP+FN}$ be the recall metric, i.e. the number of true positives out of the total ground truth positives, i.e. the true positives plus false negatives. We consider the recall-precision map, $p : [0, 1] \mapsto [0, 1]$ which maps the recall value r to the precision p that we would get if we had the detection threshold to get such a recall. Then, the Average Precision is the value $\int_0^1 p(r)$, i.e. the area under the precision-recall graph.

As a transcription score we choose the character error rate (CER), i.e. the number of insertions, deletions and substitutions to convert the output string into the ground-truthed one, divided by the length of the string. Formally:

$$CER = \frac{i + s + d}{\text{label length}}$$

C. End-to-end vs separate training

The performance of our method in terms of the CER and mAP is shown Table II. As stated in the introduction, we also include an ablation study to assess the model visual capabilities when combining the text location and transcription annotations. Concretely, we are interested in the interaction of the learning processes of text localization and transcription. To evaluate the benefits of the intermediate error reduction, we compare the end-to-end versus the two-step training, i.e. separating localization and transcription. For the end-to-end training, approach we feed the full page image as an input, pass the predicted bounding boxes to the RoI pooling layer and back-propagate the 3 summed losses: CTC, classification and regression. Since our end goal is to get the best possible transcription, we use the validation character error rate as the early stop criterion. A main advantage in using end-to-end with RoI pooling instead from page features instead of training separately and cropping the predicted boxes directly from input image is that in the first approach the features contain contextual information that allows some error in the segmentation while still getting the correct transcription, as seen in figure 5

For the two-step training we first train the model by only back-propagating the regression and classification losses, and use as a early stop criterion the mean average precision of the validation detections (mAP). This means that we train the whole model, ignoring the recognition branch, to get best possible detections in the hypothetical case that we could not do the two tasks jointly. With this approach the model finished the training stage with a mAP of 0.9. When the detection training is finished, we train the whole model, but this time we ignore the classification and regression branches. Here we only backpropagate the CTC loss, using the ground truth word segmentation in the RoI pooling step, i.e. to get the separate best possible transcription using the same architecture as in the end-to-end approach.

Figure 6 presents a comparison of the behaviour (in CER) of both approaches: the separate recognition training and the end-to-end one, where we use the predicted segmentations and backpropagate the classification and regression losses. As expected the curve belonging to the separate is much smoother and decreases fast since there is no noise in the segmentation of the words. Nonetheless, we can see that in general, the end-to-end method is not far away from the separate one, and

TABLE II
COMPARISON OF METHODS FOR FULL PAGE / PARAGRAPH RECOGNITION WITHOUT LANGUAGE MODEL.

	Val CER (%)	Test CER (%)	Detection mAP	Segmentation at test time	Page resolution	End-to-end feed forward
Ours (end-to-end)	13.8	**15.6**	0.89	Full page	150dpi	YES
Ours (separate)	10.5	19.3	0.9	Full page	150dpi	NO
Ours (HTR using test GT boxes)	-	15.5	-	Word	150dpi	NO
Bluche et al. [2]	-	7.9*	-	Paragraph	150dpi	YES
Puigcerver [7]	-	5.8*	-	Paragraph	-	YES

in the end, the difference in CER values is not significant. Looking at test time, the performance of the CER in both methodologies is the reverse, the end-to-end method is able to generalize better than the separate one, and gives a lower CER as shown in Table II.

In addition, we evaluate the separate trained approach using the ground truth test word segmentation, to know which is the best transcription we could get at test time by using our architecture. Surprisingly, the CER in this case is not significantly lower than the end-to-end approach, which does not use segmentation at all.

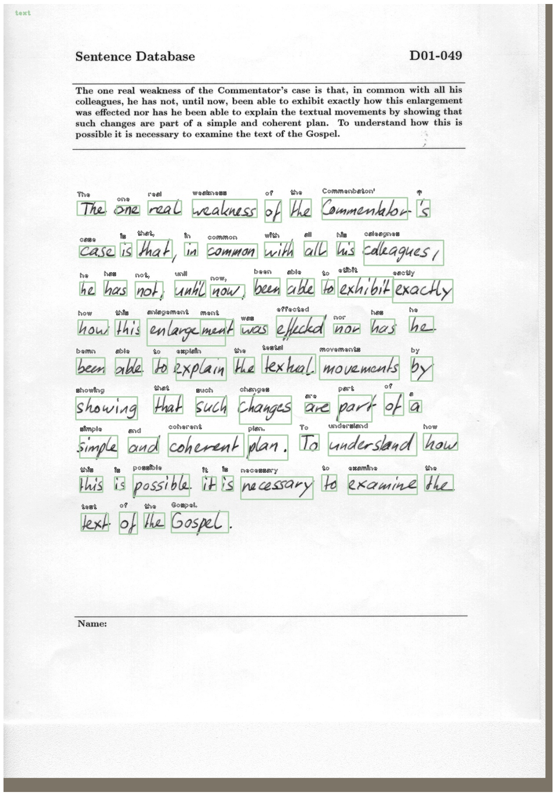
Finally, Table II compares our method to some existing methods in the literature. As far as we know, there is no previous existing work that shows the transcription results for the actual full pages. For this reason, we have chosen two methods that work at paragraph level, as the closest segmentation level to our work. Note that both approaches [2], [7] use the segmented paragraph as input. Moreover, the method described in [7] also uses data augmentation, which also brings a boost in performance. Also, we evaluate the character error rate at full page instead of line level, ignoring special characters and considering all characters lower case. Consequently, these results are not directly comparable to our work. In any case, we can observe that our method is competitive, especially taking into account that it does not require any kind of layout analysis.

Some qualitative results are shown in Figures 4 and 5. As it can be observed, most errors come from the miss-recognition of some characters. Looking in deep at those errors, we realize that humans could make the same mistakes if they only rely on the visual appearance of text. Of course, dictionaries and language models could help to reduce this kind of errors. For example, the last predicted word shown in Fig.5 would be correctly transcribed as ”colleagues” instead of ”caleagues”.

In summary, from the results we can conclude that our end-to-end approach is a promising technique in segmentation-free text recognition scenarios, and it can serve as a baseline for future works.

IV. CONCLUSIONS AND FUTURE WORK

We have proposed an end-to-end method for text detection and recognition. It addresses the potential improvements suggested in previous HTR works by recognizing the text in a full page in a single feed forward end-to-end model. The model successfully allows end-to-end training backpropagating output transcription error to segmentation layers. This



Sentence Database D01-049

The one real weakness of the Commentator's case is that, in common with all his colleagues, he has not, until now, been able to exhibit exactly how this enlargement was effected nor has he been able to explain the textual movements by showing that such changes are part of a simple and coherent plan. To understand how this is possible it is necessary to examine the text of the Gospel.

Name:

text
The one real weakness of the Commenbaton' φ
case is that, in common with all his caleagnes
he has not, unll now, been able to etibit eactly
how this enlagement ment was effected nor has he
bemn able to explain the testal movements by
showing that such changes are part of a
simple and coherent plan. To undersland how
this is possible it is necessary to examine the
test of the Gospel.

Fig. 4. Example of Localized words in a page from the IAM dataset.

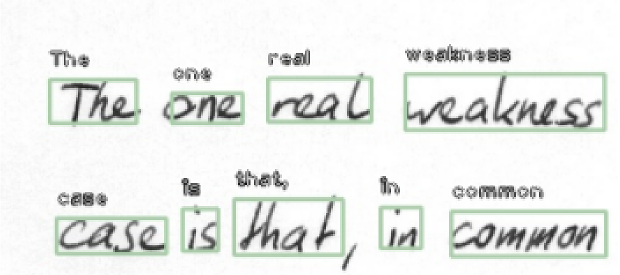


Fig. 5. Example predictions on unseen page. Note that by predicting the text sequence from pooled regions instead of the input image in an end to end fashion, the model is able to handle segmentation errors.

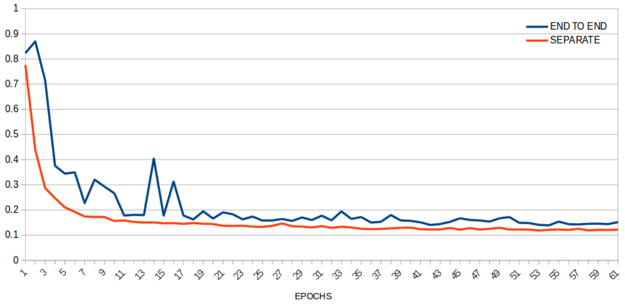


Fig. 6. Character Error Rates for the Validation set. We compare the separate recognition training vs the end-to-end training.

brings a couple of benefits and limitations. First, there is an intermediate step reduction with respect to the separate approach. The improvement could be caused by multiple reasons. One is the intended effect of backpropagating of the transcription (CTC) loss that leads to more adequate segmentation (but not necessarily with a higher detection score) at test time. Another possible cause is the regularization effect of training the recognition with more noisy segmentation, that prevents the model from overfit to the data. A unquestionable improvement is that this approach brings an inference time reduction, so for transcribing a large dataset the total difference might be significant. It would be interesting to perform some experiments in this direction and study in depth the causes of the improvement when the training is end-to-end.

It is also noticeable the memory usage reduction at inference time by sharing the model parameters for localization and transcription, and only seeing the page image once to predict the transcriptions, instead of having to work with two separate models. However, at training time it is necessary a high capacity GPU (at least 10GB) to load the recognition branch which increases the memory requirements significantly. To decode the output of our model we use a rule based reading order extractor from the word boxes. A possible future improvement to handle other arbitrary reading orders would be to add a sorting decoder RNN, inspired by the attention layer in [2].

In the near future we plan to concatenate the detected words in lines before the RoI pooling. This would allow to train with documents where the ground truth is not at segmentation level (i.e. only the transcription is available). Also, in the case of documents with text in different orientations, affine transformation with bilinear sampling as in [19] could be used instead of the RoI pooling.

Acknowledgments

This work has been partially supported by the Spanish project RTI2018-095645-B-C21, the grant 2016-DI-095 from the Secretaria dUniversitats i Recerca del Departament dEconomia i Coneixement de la Generalitat de Catalunya, the Ramon y Cajal Fellowship RYC-2014- 16831, the CERCA Programme /Generalitat de Catalunya.

References

[1] Praveen Krishnan, Kartik Dutta, and C. V. Jawahar. Word spotting and recognition using deep embedding. *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 1–6, 2018.

[2] Théodore Bluche, Jérôme Louradour, and Ronaldo O. Messina. Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 01:1050–1055, 2017.

[3] Tobias Grüning, Gundram Leifert, Tobias Strauß, and Roger Labahn. A two-stage method for text line detection in historical documents. *CoRR*, abs/1802.03345, 2018.

[4] D. Aldavert and M. Rusiñol. Manuscript text line detection and segmentation using second-order derivatives. In *IAPR International Workshop on Document Analysis Systems (DAS)*, pages 293–298, April 2018.

[5] J. Ignacio Toledo, Sounak Dey, Alicia Fornés, and Josep Lladós. Handwriting recognition by attribute embedding and recurrent neural networks. *International Conference on Document Analysis and Recognition (ICDAR)*, 01:1038–1043, 2017.

[6] Lei Kang, J. Ignacio Toledo, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusiñol. Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition. In *German Conference on Pattern Recognition*, pages 459–472. Springer, 2018.

[7] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 67–72. IEEE, 2017.

[8] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *Int. J. Comput. Vision*, 116(1):1–20, January 2016.

[9] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. FOTS: fast oriented text spotting with a unified network. *CoRR*, abs/1801.01671, 2018.

[10] Michal Busta, Lukas Neumann, and Jiri Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. *IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2231, 2017.

[11] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *ECCV*, 2018.

[12] Curtis Wigington, Chris Tensmeyer, Brian Davis, William Barrett, Brian Price, and Scott Cohen. Start, follow, read: End-to-end full-page handwriting recognition. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[13] Bastien Moysset, Christopher Kermorvant, and Christian Wolf. Full-page text recognition: Learning where to start and when to stop. *CoRR*, abs/1704.08628, 2017.

[14] Théodore Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. *CoRR*, abs/1604.08352, 2016.

[15] Wanchen Sui, Qing Zhang, Jun Yang, and Wei Chu. A novel integrated framework for learning both text detection and recognition. pages 2233–2238, 08 2018.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[17] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[18] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[19] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[21] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM.

[22] Urs-Viktor Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 2002.