Chargrid-OCR: End-to-end Trainable Optical Character Recognition through Semantic Segmentation and Object Detection

Christian Reisswig* **Anoop R Katti*** **Marco Spinaci***

Johannes Höhne*

SAP SE
Machine Learning R&D, Berlin Germany
{christian.reisswig | anoop.raveendra.katti | marco.spinaci | johannes.hoehne }@sap.com

Abstract

We present an end-to-end trainable approach for optical character recognition (OCR) on printed documents. It is based on predicting a two-dimensional character grid (*chargrid*) representation of a document image as a semantic segmentation task. To identify individual character instances from the chargrid, we regard characters as objects and use object detection techniques from computer vision. We demonstrate experimentally that our method outperforms previous state-of-the-art approaches in accuracy while being easily parallelizable on GPU (therefore being significantly faster), as well as easier to train.

1 Introduction

Optical Character Recognition (OCR) on documents is an age-old problem for which numerous open-source (e.g. [14]) as well as proprietary solutions exist. Especially in the sub-domain of printed documents, it is often regarded as being solved. However, current state-of-the-art document-level OCR solutions (as far as the published research goes) consist of a complicated pipeline of steps, each one either a hand-optimized heuristic or requiring intermediate data and annotations to train.

Deep neural networks have been proven very successful in object detection tasks [8]. In this work, we build on top of these developments and treat OCR as a semantic segmentation and object detection task for detecting and recognizing character instances on a page.[2] We introduce a new end-to-end trainable OCR pipeline for (but not limited to) printed documents that is based on deep fully convolutional neural networks. Our main contribution is to frame the OCR problem as an ultra-dense instance-segmentation task [5] for characters over the full input document image. We do not rely on any pre-processing stages like binarization, deskewing, layout analysis. Instead, our model learns directly from the raw document pixel data. At the core of our method, we predict a *chargrid* representation [6] of the input document - a 1-hot encoded grid of characters. Thus, we call our method *Chargrid-OCR*. Additionally, we introduce two novel post-processing steps, both of which are crucial to performing fast and accurate dense OCR. We show that our method can outperform line-based pipelines like e.g. Tesseract 4 [13] that rely on a combination of deep convolutional and recurrent networks with CTC loss [14, 1] while being significantly simpler to train.

* Equal contribution

[2] A related task of recognizing text in natural images, referred to as Scene Text Recognition (STR), has been faster in adopting techniques from object detection in computer vision [3]. However, compared to STR, document OCR deals with much denser text and very high accuracy requirements [2].

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

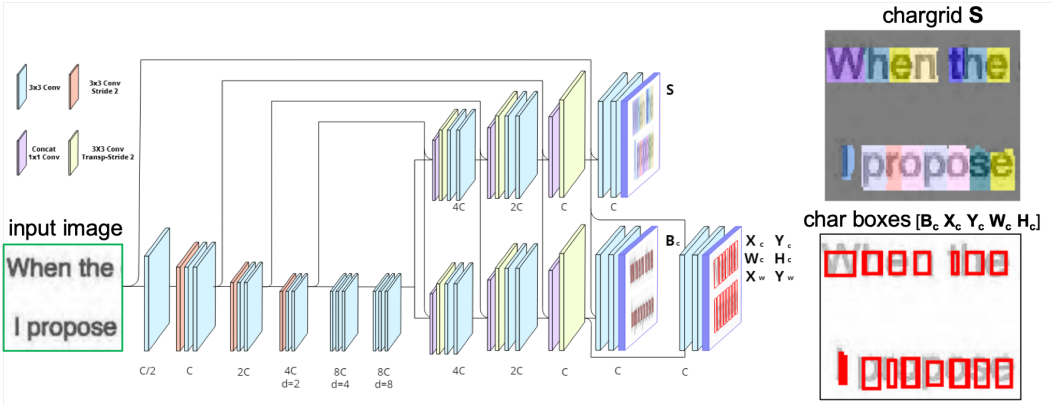

Figure 1: Schematic representation of the Chargrid-OCR network architecture with its input and outputs. Parameters nC and d denote the number of channels and strides per convolution filter. C is referred to as base channels. Colors in S encode the predicted character class for each pixel.

2 Chargrid-OCR: OCR as an ultra-dense object detection task

Chargrid-OCR method is a lexicon-free (only character-based), end-to-end trainable approach for OCR. Given a document image, chargrid-OCR predicts character segmentation mask together with object bounding boxes for characters in one single step (see Fig 1). Both, semantic segmentation and object detection are common tasks in computer vision, e.g. [11, 8, 7]. The character segmentation mask classifies each pixel into a character class and the character bounding box detects a bounding box around each character.

Both, our semantic segmentation and box detection (sub-)networks are fully convolutional and consist of only a single stage (like [8] and unlike [9]). Being single-stage is especially important as there may be thousands of characters (i.e. objects) on a single page which yields an ultra-dense object detection task.

2.1 Chargrid-OCR architecture

The chargrid representation of a document image maps each pixel that is occupied by a single character on the input document to a unique index that corresponds to that character [6].

Given an input document, our model predicts the chargrid representation of the complete document. This is accomplished by using the chargrid as target for a semantic segmentation network. Since the chargrid does not allow one to delineate character instances, we further use class agnostic object detection to predict individual character boxes. We are thus solving an instance segmentation task.

Concretely, the input to our model is an image with text, e.g. a scanned document. The output is a segmentation mask (chargrid) and a set of bounding boxes. The segmentation mask, S, classifies each pixel in the input image into characters (Fig. 1). The bounding boxes are predicted in a similar way as standard object detection methods [8] with (i) a box mask (B_c) whose confidence denotes the presence of a box at that pixel, (ii) box centers (X_c, Y_c), which denote the offset from the location of the predicting pixel to the center of the box and (iii) the box widths and the heights (W_c, H_c). Finally, for grouping characters into words, we also predict offsets to word centers (X_w, Y_w). The architecture of the model is based on a fully-convolutional encoder-decoder structure, with two decoders (one for semantic segmentation, one for bounding box detection) branching out of the common encoder. Fig. 1 illustrates the architecture with an example input and its corresponding outputs. The model is trained using categorical cross-entropy for the segmentation outputs (S, B_c) and using Huber loss for the regression output ($X_c, Y_c, W_c, H_c, X_w, Y_w$) [8].

2.2 Post-processing

The character candidate boxes are those that have confidence surpassing a certain threshold (e.g. 50%) in the box mask, B_c. This gives multiple boxes around the same character. In order to delete

redundantly predicted box proposals of the same character instance, non-maximum suppression (NMS) is applied [8]. However, in our scenario, the number of proposals can be of the order of 10^5. As NMS runs in quadratic time, this can become a computational bottleneck.

To speed up the process, we introduce a preliminary step before NMS, which we call *GraphCore*. Recall that each candidate pixel predicts the offset to the center of its box. We construct a directed graph where each vertex is a candidate pixel and we add a directed edge going from pixel A to pixel B if pixel A predicts pixel B as the center of its bounding box. By taking the k-core, with k=1, of the resulting graph (i.e. only the loops in the graph, which can be done efficiently in linear time) only pixels towards the center of a bounding box (typically, one or two candidate boxes per character) are selected as candidates for NMS.

Another necessary post-processing step is to construct word boxes from character boxes. To do so, we cluster characters together based on their predicted word centers, which even allows us to predict rotated words.

3 Experiments

3.1 Data and metric

For each document input image, we require ground-truth data in the form of character bounding boxes. (**WIKI dataset**) We generated a dataset by synthetically rendering pages in A4 format using English Wikipedia content and applied common data augmentation. We generated 66,486 pages; the page layout and font specifications (type, height and color) were sampled randomly. This enabled to synthesize input images and perfect ground truth labels. (**EDGAR dataset**) We converted a vast set of publicly available scanned financial reports [4] into images and sampled 42,918 pages with a non-repetitive layouts. We processed the images with Tesseract4 and thereby obtained noisy (i.e. including OCR errors) ground truth.

We evaluated the model on both, a held-out dataset from our training data (EDGAR_{77}: 77 pages and 22,521 words ; Wiki_{200}: 200 pages; 76,738 words) as well as benchmark OCR dataset, i.e. **Business letters** (179 pages; 48,639 words; [10]) and **UNLV.** (383 pages; 133,245 words; [12]).

We use Word Recognition Rate to measure the accuracy of OCR which can be computed by the $\frac{N_m}{N_u+N_m}$, with N_m and N_u being the number of matched and unmatched words respectively. A predicted word matches a ground truth if and only if contents agree (i.e. identical string) and they intersect (IoU > 1%). If multiple predictions match the same ground truth, only one prediction is considered a match. The remaining unmatched predictions and unmatched ground truth words are added to obtain N_u.

3.2 Results

We train various versions of our new model on the datasets described in Sec. 3.1 and report results in Table 1. As baseline, we compare against Tesseract, v3 and v4, with v4 [14] (released Oct 2018) being the publicly available state-of-the-art.[3] Tesseract v4 comes with an LSTM-based line recognition engine and achieves much higher accuracy than v3. Unfortunately, re-training Tesseract on our datasets is not possible due to needing intermediate annotations to train Tesseract. We, therefore, use off-the-shelf Tesseract without any retraining or fine tuning. However, we use test data that are domain-independent from training and/or validation data and serve as an indicator for model generalizability.

Our baseline is denoted by "ChargridOCR-32", consists of 32 convolutional base channels ($C = 32$ in Fig. 1), and was trained on the Wiki dataset. This model outperforms Tesseract v3, but not Tesseract v4. The same model trained on Wiki+EDGAR is competitive with, and typically superior to, Tesseract4 on validation and test data. Finally, a model with twice as many convolutional channels "ChargridOCR-64" and thus with higher capacity trained on Wiki+EDGAR outperforms Tesseract on all datasets, however with significant computational overhead (rightmost column in Tab. 1). In Fig. 2, we show some crops exemplifying incorrect predictions from our model.

[3] Commercial solutions were excluded as Trial License Agreements prevent us from analyzing their results.

Model	Training Data	Validation data		Test Data		Time
		Wiki_{200}	EDGAR_{77}	Letters	UNLV	1000 pages
Tesseract3	Unknown	86.3%	83.4%	87.7%	72.4%	5600s
Tesseract4	Unknown	94.6%	91.0%	92.6%	76.8%	14800s
ChargridOCR-32	Wiki	97.3%	86.4%	89.4%	75.3%	241s
ChargridOCR-32	Wiki+EDGAR	97.2%	91.4%	92.3%	80.4%	241s
ChargridOCR-64	Wiki+EDGAR	**98.8%**	**91.6%**	**93.5%**	**81.6%**	550s

Table 1: Results, reported in terms of Word Recognition Rate. Tesseract run-times are obtained using 1 Xeon E5-2698 CPU core and Chargrid-OCR's on 1 V100 GPU.

Figure 2: Three faulty example crops (top, middle, bottom row) from the validation set. From left to right: original image, predicted segmentation mask, predicted character boxes (after postprocessing), and resulting extracted words (blue if they match ground truth, red otherwise).

4 Conclusion

We presented a new end-to-end trainable optical character recognition pipeline that is based on state-of-the-art computer vision approaches using object detection and semantic segmentation. Our pipeline is significantly simpler compared to other sequential and line-based approaches, especially those used for document-level optical character recognition such as Tesseract 4. We empirically show that our model outperforms Tesseract 4 on a number of diverse evaluation datasets by a large margin both in terms of accuracy and run-time.

References

[1] T. M. Breuel. High performance text recognition using a hybrid convolutional-LSTM implementation. In *ICDAR 2017*, pages 11–16, 2017.

[2] T. M. Breuel. Robust, simple page segmentation using hybrid convolutional MDLSTM networks. In *ICDAR 2017*, volume 1, pages 733–740. IEEE, 2017.

[3] M. Busta, L. Neumann, and J. Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *ICCV 2017*, pages 2223–2231, 2017.

[4] EDGAR. Sec. `https://www.sec.gov/Archives/edgar/vprr/index.html`.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, pages 2961–2969, 2017.

[6] A. R. Katti, C. Reisswig, C. Guder, S. Brarda, S. Bickel, J. Höhne, and J. B. Faddoul. Chargrid: Towards understanding 2D documents. In *EMNLP 2018*, pages 4459–4469, 2018.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In *ECCV 2016*, pages 21–37, 2016.

[9] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[10] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fourth annual test of OCR accuracy. Technical report, Technical Report 95, 1995.

[11] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI 2015*, pages 234–241. Springer, 2015.

[12] A. Shahab, F. Shafait, T. Kieninger, and A. Dengel. An open approach towards the benchmarking of table structure recognition systems. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 113–120. ACM, 2010.

[13] R. Smith. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[14] Tesseract. v4. `https://github.com/tesseract-ocr/tesseract/wiki/ReleaseNotes`.