2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)

Robust Handwriting Recognition with Limited and Noisy Data

Hai Pham[†], Amrith Setlur[†], Saket Dingliwal[†], Tzu-Hsiang Lin[†], Barnabás Póczos[†]
Kang Huang[♣], Zhuo Li[♣], Jae Lim[♣], Collin McCormack[♣], Tam Vu[♣]
[†]*Carnegie Mellon University* [♣]*The Boeing Company*

***Abstract*—Despite the advent of deep learning in computer vision, the general handwriting recognition problem is far from solved. Most existing approaches focus on handwriting datasets that have clearly written text and carefully segmented labels. In this paper, we instead focus on learning handwritten characters from maintenance logs, a constrained setting where data is very limited and noisy. We break the problem into two consecutive stages of word segmentation and word recognition respectively, and utilize data augmentation techniques to train both stages. Extensive comparisons with popular baselines for scene-text detection and word recognition show that our system achieves a lower error rate and is more suited to handle noisy and difficult documents.**

***Index Terms*—handwriting recognition, word segmentation, word recognition, character recognition, CTC, object detection**

I. Introduction

Offline handwriting recognition (HWR) is a fundamental problem in computer vision [31]. Unlike online HWR where a stroke direction is a valuable cue [2], [13], in the offline setting, we simply have access to an image of the final handwritten words instead. Nowadays, although data can be easily digitized and stored, there is still a need to recognize and digitize handwritten paper documents [3], [25].

Despite the significant demand, there are few efficient methods able to tackle this problem due to the difficulty of designing a holistic solution suitable across various forms of input. The first challenge is to segment forms (*i.e.* images containing lines) properly to facilitate the recognition process. The most common method is to use a heuristics line-level segmentation [13], [27]. However, this is often impractical since words and characters are not usually handwritten along straight lines. The second challenge is to build a model capable of recognizing and generalizing diverse handwriting styles. Furthermore, in some resource-constrained settings where we have limited access to real data, it is infeasible to manually build large-scale handwriting recognition datasets such as IAM-DB [30], or SD19 [14]. It therefore becomes necessary to find a powerful solution that does not require a large quantity of real data. This problem is ubiquitous in practice, in that we only have access to limited data with inherent noise. Typically in such settings, people rely on commercial systems which are prohibitively expensive, or open APIs such as Google Cloud Vision[1] or Tesseract [36] which typically perform poorly as they are mainly designed for *printed* text and for dealing with many languages with a single model.

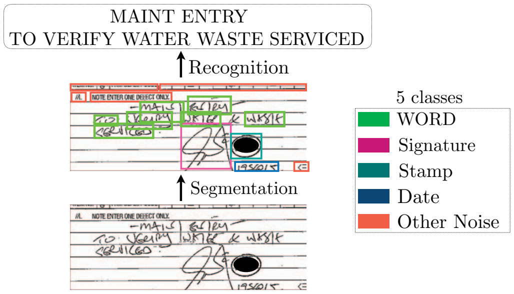

Figure 1: Our model can handle noisy forms by localizing unaligned texts, filter out other types of noise to recognize the sentence(s) in the correct order of words. We hide stamps' contents for security reason.

Furthermore, our dataset is much more difficult than IAM-DB or SD19. First, it has limited and noisy data and annotation. Second, it combines the difficulties of the classical HWR datasets and scene-text detection and recognition ones (Fig. 1 and 2). As a result, we modularize the problem into two stages in order to make it more tractable to train two separate deep models. In the first stage we employ a object detection model, such as R-FCN [6], to detect words from the background with various types of noise. The resulting segments are fed into a recognition model in the second stage which can be a word-based or a character-based model.

We evaluate performance based on multiple metrics and show that our system is able to detect words in challenging settings with high accuracy. In addition, we also demonstrate advantages over several state-of-the-art (SoTA) methods for the related tasks of scene-text detection and recognition. To sum up, our contributions are as follows: we show (i) that in a constrained setting as defined above, a two-phase approach including segmentation and recognition at the form level (instead of line level) is an efficient method, and (ii) extensive experiments and analysis that provide guidelines for similar applications in this setting.

II. Related Work

For offline HWR, there have been many achievements using the classical HMM-based models [2], [5], [11], [19]. Later, with the advent of deep learning, Recurrent Neural Network

Corresponding author: htpham@cs.cmu.edu

[1] https://cloud.google.com/vision/

978-1-7281-9966-5/20/$31.00 ©2020 IEEE
DOI 10.1109/ICFHR2020.2020.00062

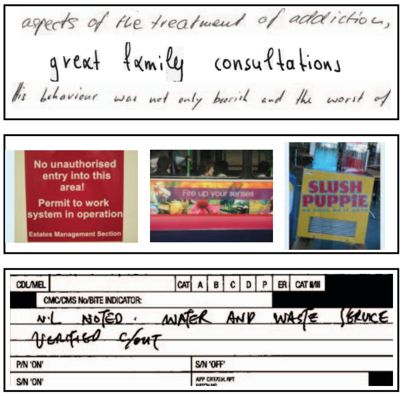
aspects of the treatment of addiction,
great family consultations
his behaviour was not only boorish and the worst of
No unauthorised entry into this area!
Permit to work system in operation
Estates Management Section
SLUSH PUPPIE
CDL/MEL
CAT A B C D P ER CAT
CMC/CMS No/BITE INDICATOR:
N.L NOTED . WATER AND WASTE SERUCE
VERIFIED C/OUT
P/N 'ON'
S/N 'OFF'
S/N 'ON'

Figure 2: Some samples from 3 different problems. *Top*: three lines of IAM [30] dataset which has handwritten text on blank background; most solutions segment them into lines without clarifying segmentation quality. *Middle*: ICDAR [22] dataset for scene-text recognition which has printed text with random background. *Bottom*: our BHD dataset which combines the difficulties of the other two: multi-style, unaligned handwritten text in the whole form (not lines) and noisy background.

based approaches, such as using LSTM [18], gained new successes in this setting [21], [33], [38]. Following this line, there have been also some other solutions that also employ convolutional neural network (CNN) such as in [9], and using CNN plus language-based features [24], [32]. However, in comparison to their settings, our variable-sized forms are more challenging for they include horizontal lines running across the document which contribute to noise since the text doesn't necessarily conform to these lines. Furthermore, the content is mixed with other random noise such as signatures, stamps or other unrecognized marks caused by scanners or inks. Given such difficult inputs, our model can directly process whole forms properly, in contrast to these existing solutions that rely on heuristic methods for line-level segmentation.

A closely related problem to our method is segmentation for which there have been some heuristic [31] or HMM-based [40] methods. Our model instead relies on deep segmentation frameworks which are usually employed for object detection tasks [6], [16], [23], [26], [34]. Unlike those methods that learn to predict a regression bounding box and detect an object at the same time, in our segmentation phase, we reduce the task to a more tractable problem of only predicting a bounding box covering a word, and leave the recognition job to a downstream task. We retain the order of the words while doing this so as to ensure that sentence or document level meaning is retained.

Finally, the most related approaches to our model are those designed for scene-text detection and recognition [1], [4], [8], [28], [39] although their problem is different from HWR. But unlike their solutions which deal mainly with printed text, our setting is intrinsically harder due to the inherent difficulty in recognizing handwritten text of different styles.

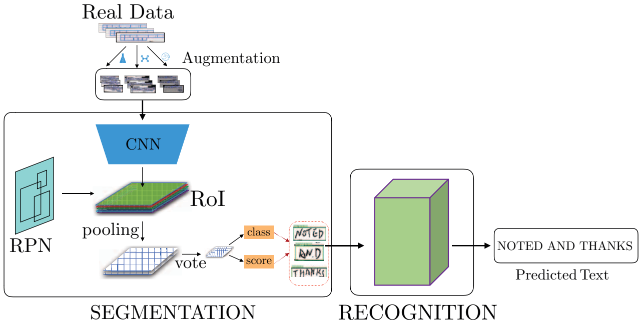

Figure 3: Our model uses data augmentation to train a segmentation module (locating words amidst background noise) and word recognition (word or character-based) models.

III. Model

Our inputs are rectangular images of varying sizes containing handwritten sentences, often in unaligned lines and with lots of noise and other irrelevant content such as stamps, signatures and other types of random noise. Our goal is to recognize those relevant sentences, and output the corresponding texts for further data analysis purposes.

A. *Choice of Two-phase Model*

As mentioned above, we design a two-phase approach (Fig. 3): segment the entire form into words (in the presence of noisy content) while maintaining their original order, and recognize each word individually. There are many reasons for this approach. First, we have very few annotated samples (Table I), thus the generalizability of our model is benefited from the inductive bias of the two stage approach. Second, the difficulties of the forms are unusual. Due to unaligned texts, it is impossible to segment forms into lines without affecting the content as in other HWR methods. Furthermore, like scene-text recognition datasets, our forms have many types of noise (Fig. 1 and 2). Third, this approach is interpretable and easier to train and debug. Finally, it becomes easier to perform parallel training of the two stages across limited resources, allowing for better quality control and modularity in design.

B. *Word Segmentation*

Instead of trying to predict the correct bounding boxes and recognize the words inside simultaneously, the word segmentation phase only focuses on drawing correct bounding boxes at the word level, and leaves the recognition job as a downstream task. We choose this design for the following reasons. First, word-level segmentation is used since separating spaces among words (as opposed to characters) is much more feasible in practice (especially in cursive handwriting). Second, as explained previously, line-level segmentation is not preferred since in our setting words are often not aligned horizontally.

In terms of architecture, since HWR is different from object detection where detection is only a proxy, we explore multiple options like R-FCN [6], Faster R-CNN [12] and YOLO-v3

[34] to identify which kind of architecture is most suited for our HWR pipeline. Although the core components of those detection methods remain unchanged, it is worth noting two important changes in adapting such methods. First, given word segmentation is an intermediate step, we simplify this phase by limiting the number of classes to only 5 (Fig. 1), with the main goal being extracting text out of the forms without having to recognize its content. Second, based on the nature of our dataset, we change the segmentation input to grayscale images with only 1 channel. As a result of these two adjustments, our segmentation phase is much easier and faster to train compared with their original uses.

C. *Word Recognition*

For each form, this module takes the bounding boxes (as images) from the Word Segmentation module as inputs, and outputs a word for each bounding box. Based on the coordinates given by the Word Segmentation module, we are able to reconstruct the entire sentence from individual words. And because the complications of the input forms, we experiment with 3 different models namely Word Model, Character Model and CTCSeq2Seq Model, as detailed below.

1) Word Model: The word model is a CNN-based image classification network which uses an augmented Resnet-18 [17] to predict words from a predefined word vocabulary. Furthermore, due to the low resolution of our input images, we adjust Resnet-18 to only have a stride size of 1 instead of 2 in the residual blocks. This model is simple, but is only capable of predicting words within the predefined vocabulary of 998 words.

2) Character Model: This model shares its architecture with the Word Model, which enables the benifit of initializing weights from a pretrained Word Model, except that it uses a CTC loss [13], [27], [37] instead of cross-entropy loss. For this reason, it predicts a sequence of characters instead of a single word at a time. Furthermore, the last fully-connected layer in Resnet-18 is replaced with a convolutional layer to reshape the output from $H * W * D$ to $1 * W/2 * C$, where C is the cardinality of the character prediction space.

By using CTC, this model has two advantages over Word Model. First, CTC largely reduces the prediction space from 998 words to 35 alpha-numeric characters (our dataset does not have the letter "Z"), making it agnostic to word vocabulary size. Second, it enables the model to predict unseen words.

3) CTCSeq2Seq Model: Our motivation for this model is to learn the embedded latent representation of images that can be decoded into text. As shown in Fig. 4, the model can be broken down into 3 main blocks: Feature Extraction, Encoder and Decoder. The model loss is the weighted sum of CTC loss (Encoder) and softmax cross-entropy loss (Decoder). Except for those 3 main modules, there is an edit-distance based error module which corrects a predicted out-of-vocabulary word within a maximum of 2 wrong characters compared to a known word.

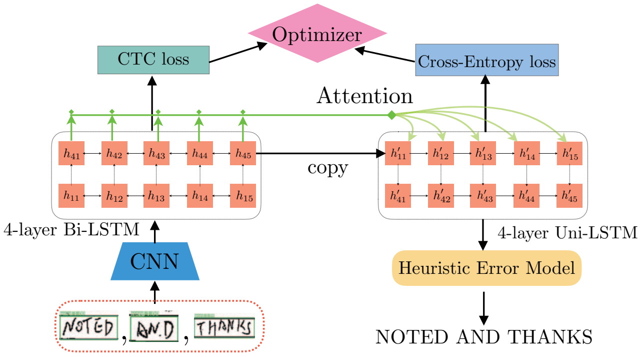
Optimizer
CTC loss
Cross-Entropy loss
Attention
h_{41} h_{42} h_{43} h_{44} h_{45}
h_{11} h_{12} h_{13} h_{14} h_{15}
copy
h'_{11} h'_{12} h'_{13} h'_{14} h'_{15}
h'_{41} h'_{42} h'_{43} h'_{44} h'_{45}
4-layer Bi-LSTM
4-layer Uni-LSTM
CNN
Heuristic Error Model
NOTED , AND , THANKS
NOTED AND THANKS

Figure 4: Our CTCSeq2seq model contains 3 core modules: Feature Extraction which is CNN-based, Encoder and Decoder that combined form a Seq2Seq model. The encoder uses *CTC loss* which helps with alignments of the frames to the outputs.

Feature Extraction: This module accepts variable-sized input images, each of which has a single word. It firstly resizes inputs to the same height but not necessarily to the same width. Next, it slices each one into small patches of equal widths (as illustrated in Fig. 6). Finally, it extracts CNN-based features out of the patches using a custom VGG [35].

Encoder: For our encoder, we use a 4-layer bidirectional LSTM that takes inputs from the Feature Extraction module. Since each input word is segmented into many sequential equal-height patches, the LSTM can model their relations into a hidden representation. Another key feature of this module is to have a CTC loss to enforce reconstruction of the original characters, so that the embedded representation is learned effectively.

Decoder: This module is a 4-layer unidirectional LSTM that consumes the hidden representation from the Encoder and has an attention module [29] which calculates the weighted average of each output with the entire input sequence. This mechanism helps the model learn to focus on more important patches.

In addition, this module uses the softmax cross-entropy loss normalized by the length of input, since we have variable-length sequences of patches. Finally, it also predicts among 35 alpha-numeric characters, same as Character Model (Section III-C2) which also ignores punctuation in the datasets.

IV. Experiments

Dataset	Type	Train	Valid	Test
Segmentation	Real	2,358	-	1,362
	+DA	40,159	-	-
Recognition	Real	6,639	3,400	1,249
	+DA	660,000	-	-
Pipeline	Real	-	-	1,362

Table I: Statistics of BHD dataset. We have 2 types of data (i) Real and (ii) +DA : real images with data augmentation. For each model, we only have a single test set from real forms, and the one used for Pipeline evaluation is shared with Segmentation. Data augmentation is a key preprocessing step to get more samples and styles for training deep models.

A. Dataset

Our in-house BHD dataset, as shown in Table I, comprises of maintenance logbooks in which there are many aerospace terms or abbreviations that do not appear in the normal English vocabulary. Each image is grayscale and may contain from 3 to 50 bounding boxes. Moreover, in addition to the presence of unusual aerospace terms, there are many arbitrary part numbers (*e.g.*, “W308003-12239-22”). As mentioned earlier, our forms contain multiple horizontal lines, with signatures, stamps, dates and other types of noise, making our task even more challenging. Finally, to create word vocabulary, we use `tf-idf` to retrieve the first 1000 words from digitized maintenance logbooks, then remove 2 outliers to finally have 998 words.

Furthermore, our manual inspection of the BHD dataset reveals that in several cases the strokes from adjacent words are connected to each other, while in other cases, the characters in a word are quite far apart, which tempts any object detection model to confuse multiple words with just one. This makes BHD more challenging than ICDAR and other scene-text detection and recognition datasets.

B. Training Data Augmentation

Because we have limited data, and our model contains deep neural networks that are typically data hungry, data augmentation is an important technique to increase the effective size of BHD prior to training, and to improve the generalization capability of our models. In particular, we use two data augmentation techniques for both segmentation and recognition tasks. First, we use several types of noise including pepper, stroke and Gaussian noises. Second, we employ local image transformations that are erosion, dilation and flipping.

C. Evaluation Metrics

Segmentation: We use the canonical MaP metric [10] to evaluate segmentation performance against our annotation in the BHP real-form test dataset.

Recognition: We use word accuracy (WA) and Character Error Rate (CER) to evaluate our recognition models. While WA simply calculates the average number of predicted words that exactly matches with ground truths, CER is calculated as CER $= (\mathrm{D}(w_{gt}, w_{predict}) \times 100) \, / \, |w_{gt}| \, (\%)$, where $\mathrm{D}(w_{gt}, w_{predict})$ is the minimum Damerau-Levenshtein edit distance [7] between the ground-truth word w_{gt} and predicted word $w_{predict}$, and $|w_{gt}|$ is the number of characters in w_{gt}.

Full Pipeline: Our pipeline takes a form as input, and outputs a sequence of predicted words. Therefore we use Word Error Rate (WER) and CER to evaluate performances. For WER, we treat every word as a character. For CER, we concatenate the sequence of words by inserting a space between every two words and treating the concatenated sequence as the predicted string.

D. Baselines

Since different models require different sets of annotations (*e.g.* many HWR models expect noise free input), we cannot fairly compare our full pipeline performances with many SoTA methods for HWR. As a result, the only close HWR pipeline we compare our model with is Convolve-Attend-Spell [21] (after it is fine-tuned on the full-pipeline dataset) which has the capability of accepting the entire form as an input and to some extent is also robust to noise.

However, we can compare each phase of our pipeline with segmentation and recognition baselines developed for scene-text detection. For segmentation, we use EAST [39], PixelLink [8] and CRAFT [1]. In order to have a fair comparison, we fine tune EAST and PixelLink[2] (trained on ICDAR 2015 [22]) and only compare on the *word* class, which is the ultimate goal. For recognition, we use MORAN [28] which is pre-trained on synthetic images [15], [20] and subsequently fine tuned on BHD recognition training data. And last, for full pipeline, we combine PixelLink and MORAN, for which the full training codes are available.

V. Results and Discussion

We compare the performances of our approach to the baselines for the full pipeline, segmentation and recognition. We also perform an ablation study on the impact of segmentation on the full pipeline.

Segmentation	Recognition	WER(↓)	CER(↓)
R-FCN [6]	Word	31.5	22.9
	CTCSeq2Seq	**30.1**	**18.5**
PixelLink [8]	MORAN [28]	80.7	47.4
Convolve-Attend-Spell [21]		38.9	24.1

Table II: Full pipeline performance of our best model compared to the baselines. Our model significantly outperforms all the baselines in both WER and CER metrics.

A. Full Pipeline Results

The full pipeline results are shown in Table II. We observe that that R-FCN [6] in conjunction with CTCSeq2Seq (both of which are trained on the +DA dataset) yields the best performance, and significantly outperforms the baseline models.

Furthermore, Fig. 5 illustrates some qualitative results. The R-FCN is able to filter out several types of noise in each form and pick out the correct bounding boxes with almost 100% confidence for all words. Furthermore, our CTCSeq2Seq is able to detect words and characters of various styles, orientations and intensities. However, the baseline one makes lots of mistakes in word localization, which are compounded in the second phase of recognition.

	EAST	CRAFT	PixelLink	R-FCN	Faster-RCNN	YOLO-v3
AP	38.9	12.8	81.6	89.0	**89.1**	86.0

Table III: AP score comparison on the *word* class (IoU=50%). Our three models significantly outperform the baselines.

B. Segmentation Results

As shown in Table III, our three segmentation models clearly outperforms all baseline methods, especially on EAST

[2] The same cannot be done for CRAFT due to its code’s unavailability.

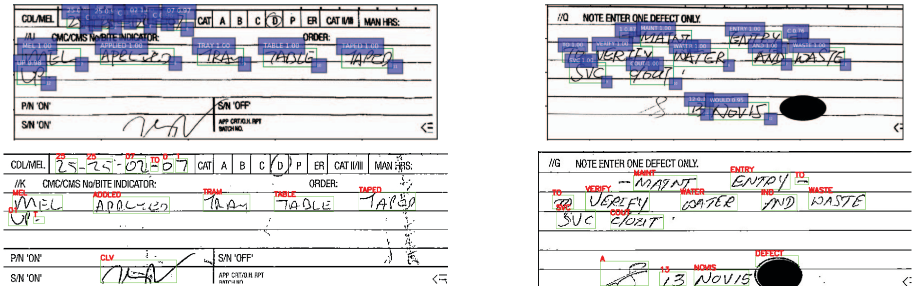

Figure 5: Full pipeline qualitative results of our model R-FCN [6] + CTCSeq2Seq (top) and the baseline PixelLink [8] + MORAN [28] (bottom). Ours performs much better in both locating words and recognizing them.

Class	R-FCN (Real)	R-FCN (+DA)	Faster R-CNN (+DA)	YOLO-v3 (+DA)
Word	85.8	89.0	**89.1**	86.0
Signature	67.9	**78.2**	43.3	40.8
Stamp	86.6	**89.9**	10.7	84.2
Date	70.1	**82.9**	24.7	62.9
Noise	18.2	17.4	**27.3**	15.2
Average	65.7	**71.3**	39.0	57.8

Table IV: AP scores for Segmentation models R-FCN [6], Faster R-CNN [23] and YOLO v3 [34]. R-FCN significantly outperforms others on most classes with augmented training data (IoU=50%).

Model	Dataset	WA (↑)	CER(↓)
Word	Real	76.1	20.4
	+DA	**96.1**	**2.6**
Character	Real	5.0	62.8
	+DA	76.3	9.7
CTCSeq2Seq	Real	87.1	7.8
	+DA	94.9	3.2
MORAN	Real	91.7	3.4
	+DA	**96.4**	**1.5**

Table V: Comparison on recognition models (on Recognition dataset) given ground-truth bounding boxes. Our Word Model and MORAN [28] perform the best compared to others.

and CRAFT. While EAST fails to split large bounding boxes, leading to a low recall (18.4%), CRAFT's pretrained model mistakes printed words for handwritten text and therefore has a low precision (21.4%). Finally, since PixelLink is trained on BHD, it can achieve a decent score of 81.6% AP.

Additionally, considering only our models, Table IV shows that data augmentation leads to improvements on AP for R-FCN (especially for rare categories like *Signature* or *Date*). R-FCN with position-based scores is particularly effective in tackling translation variance [6] for handwriting recognition where the Region-of-Interest (RoI) is fairly small (as seen in Fig. 5).

C. Recognition Results

As demonstrated in Table V, our Word Model achieves the similar performances to the best performer MORAN in both WA and CER given ground-truth bounding boxes. Even being initialized with Word Model's pretrained weights, the Character Model under-performs the other two by a huge margin. We suspect the reason is that CTC is hard to train, and may require more training data or more complex techniques.

Recognition	Segmentation	WER(↓)	CER(↓)
Word	Ground Truth	15.1	9.5
	R-FCN	18.3	13.2
	Faster R-CNN	19.1	21.0
CTCSeq2Seq	Ground Truth	**14.1**	**8.2**
	R-FCN	18.9	12.3
	Faster R-CNN	19.8	19.5
MORAN	Ground Truth	49.2	25.7
	PixelLink	80.7	47.4

Table VI: Impact of different Segmentation methods on the full pipeline (on Pipeline dataset). Our models clearly outperform the baselines, and CER is much higher if we replace R-FCN by Faster R-CNN.

D. Ablation Study

We study how different segmentation models affect pipeline performance on the same recognition model. As shown in Table VI, our models perform much better than the baselines, and CTCSeq2Seq is the best recognition model. As shown in Fig. 6, CTC loss combined with attention module significantly helps with character recognition, making the CTCSeq2Seq the best choice for our full pipeline.

And interestingly, CER increases much more than WER when replacing R-FCN with Faster R-CNN. Our empirical analysis reveals that R-FCN tends to give predictions with higher confidence scores and in difficult cases, it predicts more bounding boxes than Faster R-CNN in the segmentation phase. Finally, given ground-truth bounding boxes, both WER and CER decrease but only to a limited extent. This suggests that the segmentation module is not the bottleneck of whole pipeline system, and we should focus more on the recognition module to increase pipeline performance.

VI. Conclusion

In this paper, we focused on HWR for noisy and challenging maintenance logs, a previously overlooked domain in this field. We presented a two-stage approach that can process the entire forms directly without the need of segmenting them into lines. Our experimental results show that our approach significantly outperforms the HWR and scene-text detection

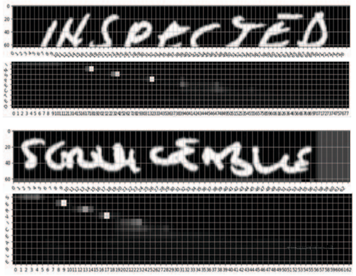

Figure 6: Attention map results of CTCSeq2seq model for 2 words: INSPECTED and SERVICEABLE. The upper image is raw input and the lower one is the corresponding attention map. Brighter squares indicates higher weights (focusing more in decoding). After first several characters are recognized, the model can infer the rest of characters without relying on encoder information.

and recognition baselines on the full pipeline while achieving high accuracies on the individual phases of word segmentation and recognition.

REFERENCES

[1] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *CVPR*, pages 9365–9374, 2019.

[2] Claus Bahlmann and Hans Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE trans on pattern analysis and machine intelligence*, 2004.

[3] Paulo Blikstein and Marcelo Worsley. Multimodal learning analytics and education data mining: using computational technologies to measure complex learning tasks. *Journal of Learning Analytics*, 3(2):220–238, 2016.

[4] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *ACM SIGKDD*, pages 71–79. ACM, 2018.

[5] Horst Bunke and Tamás Varga. Off-line roman cursive handwriting recognition. In *Digital Document Processing*, pages 165–183. Springer, 2007.

[6] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, pages 379–387, 2016.

[7] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[8] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. Pixellink: Detecting scene text via instance segmentation. In *AAAI*, 2018.

[9] Kartik Dutta, Praveen Krishnan, Minesh Mathew, and CV Jawahar. Improving cnn-rnn hybrid networks for handwriting recognition. In *ICFHR*, pages 80–85. IEEE, 2018.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[11] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern Recognition Letters*, 2012.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[13] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recog. *IEEE trans on pattern analysis and machine intelligence*, 2009.

[14] Patrick J Grother. Nist special database 19. *Handprinted forms and characters database, National Institute of Standards and Technology*, 1995.

[15] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, pages 2315–2324, 2016.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recog. In *CVPR*, 2016.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Jianying Hu, Sok Gek Lim, and Michael K Brown. Writer independent on-line handwriting recognition using an hmm approach. *Pattern Recognition*, 2000.

[20] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[21] Lei Kang, J Ignacio Toledo, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusinol. Convolve, attend and spell: An attention-based sequence-to-sequence model for handwritten word recognition. In *German Conference on Pattern Recognition*.

[22] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *ICDAR*, pages 1156–1160. IEEE, 2015.

[23] Gnter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, 2017.

[24] Praveen Krishnan, Kartik Dutta, and CV Jawahar. Deep feature embedding for accurate recognition and retrieval of handwritten text. In *ICFHR*, pages 289–294. IEEE, 2016.

[25] Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. Text line segmentation of historical documents: a survey. *IJDAR*, 9(2-4):123–138, 2007.

[26] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE trans on pattern analysis and machine intelligence*, 2018.

[27] Marcus Liwicki, Alex Graves, Horst Bunke, and Jürgen Schmidhuber. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *ICDAR*, volume 1, pages 367–371, 2007.

[28] Canjie Luo, Lianwen Jin, and Zenghui Sun. Moran: A multi-object rectified attention network for scene text recognition. *Pattern Recognition*, 90:109–118, 2019.

[29] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[30] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *IJDAR*, 2002.

[31] Réjean Plamondon and Sargur Srihari. Online and offline handwriting recognition: a comprehensive survey. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 2000.

[32] Arik Poznanski and Lior Wolf. Cnn-n-gram for handwriting word recognition. In *CVPR*, pages 2305–2314, 2016.

[33] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *ICDAR*, 2017.

[34] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[36] Ray Smith. An overview of the tesseract ocr engine. In *ICDAR*. IEEE, 2007.

[37] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 228–233. IEEE, 2016.

[38] Paul Voigtlaender, Patrick Doetsch, Simon Wiesler, Ralf Schlüter, and Hermann Ney. Sequence-discriminative training of recurrent neural networks. In *ICASSP*. IEEE, 2015.

[39] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *CVPR*, pages 5551–5560, 2017.

[40] Matthias Zimmermann and Horst Bunke. Automatic segmentation of the iam off-line database for handwritten english text. In *Pattern Recognition*, volume 4, pages 35–39. IEEE, 2002.