Tight Coupling of Character, Word, and Place Recognition for End-to-End Text Recognition in Maps

Archan Ray[1⋆], Aruni Roy Chowdhury[2⋆], Yi Fung[3⋆], Jerod Weinman[4†] Erik Learned-Miller[5⋆]
⋆University of Massachusetts, Amherst, MA 01003, USA
† Grinnell College, Grinnell, Iowa, USA
[1]ray@cs.umass.edu, [2]arunirc@cs.umass.edu, [3]yfung@umass.edu, [4]jerod@acm.org, [5]elm@cs.umass.edu

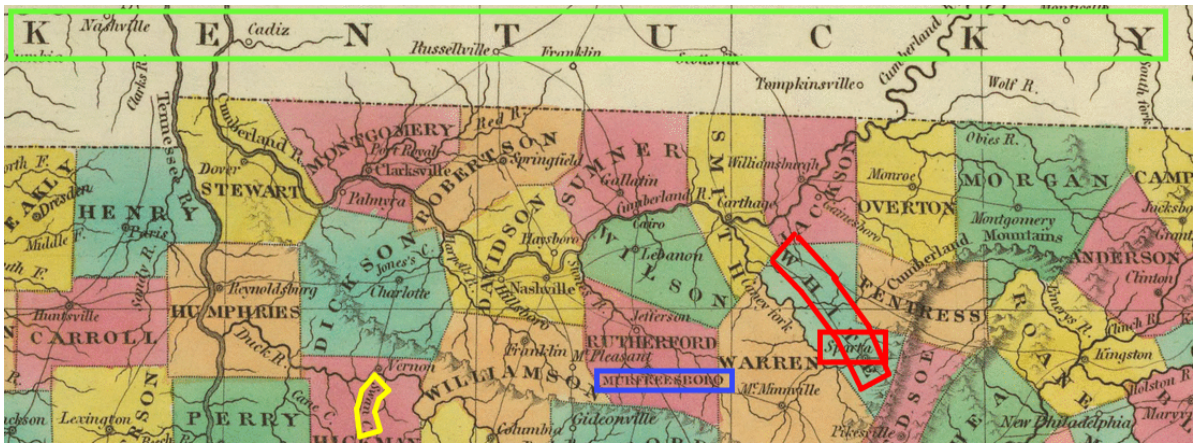

Figure 1: The challenges of text recognition in maps. Text recognition in maps comes with extreme challenges such as wide letter spacing (green box), overlapping words (red boxes), unusual word orientations (yellow box) and dense distractors (blue box).

Abstract

Text recognition in maps is a special case of general text recognition that features some especially difficult challenges, including texts at extreme orientations, wide character spacings, complex text-like distractors, and unusual non-dictionary strings. Off-the-shelf OCR systems, and even sophisticated scene text recognition systems do not work satisfactorily on many map-text recognition problems. While many OCR and scene text systems have produced high quality results by considering detection, recognition, and error-correction as separate components, we believe that map text recognition can benefit immensely from the **tight coupling** *of different components of an overall system. In particular, we present an end-to-end system for recognizing text in maps that uses strong coupling in two different ways. In the first, we train individual* **character** *detectors, and use these detections as inputs in a new* ***word detection CNN architecture*** *to improve word detection. We show dramatic increases in word detection accuracy for a strong baseline detection architecture. In the second contribution, we use a geographically-based lexicon to constrain our interpretations of initial detections. If the lexicon suggests that the word detection is either too short, we "re-prime" the word detector by inserting expected characters locations back into the word detector using a novel input mechanism. We then rerun the word detector using the additional character suggestions, giving a solid improvement in accuracy. We report end-to-end recognition results on a public map-text recognition benchmark.*

1. Introduction

The recognition of text in historical maps is a sub-area in document recognition that is receiving increased attention [4, 6, 57, 58, 64], both because it has important practical applications and because it presents certain technical challenges that require new models and methods. In particular, traditional text recognition methods, including standard optical character recognition (OCR) systems, and more specialized systems such as scene-text recognition systems, simply do not work well in this domain. Text in maps is characterized by certain hallmarks that make it particu-

larly challenging (see Figure 1) – words at extreme orientations, often over an angle span of more than 180 degrees (Fig. 1, yellow box); large and variable spacings between letters Fig. 1, green box); confusing distractors, such as a large numbers of lines and junctions that are easily confused with characters (Fig. 1, blue box); overlapping strings and curved baselines (Fig. 1, red boxes). While traditional text recognition systems may partially deal with some of these issues, the confluence of them can cause traditional methods to break down.

Word recognition systems generally have separate components for detection, segmentation, and recognition of characters, and similar components for detection, segmentation and recognition at the word level. Some systems are strictly "feed-forward", in such a system once a decision has been made about a character, it cannot be changed, even if higher level context suggests that it is wrong. Many systems, such as hidden Markov models [2, 40], hedge their bets on low-level tokens like characters by specifying a probability distribution over them, and use higher level context (such as the probabilities of longer sequences) to resolve ambiguous low-level symbols. Probabilistic models provide elegant systems for managing uncertainty and making maximum likelihood decisions. However, given they have been outperformed by neural network architectures on a wide variety of tasks, this raises the question of how to best incorporate feedback, naturally handled in probabilistic systems, into neural network style models.

There are several possible approaches to unify low-level and high level systems in a complex structured output problem like map text recognition. One method is co-training a system to output intermediate values, such as character labels, even though they are not strictly needed during the final output, in a multiple-task learning (MTL) setup. Another class of methods are based on "attention", where important parts of an input can be revisited for more detailed analysis (see, *e.g.* [52]).

In this paper, we suggest two novel forms of *feedback* in a map text recognition system that dramatically improve its performance. While it is theoretically feasible to train a standard neural network from scratch to develop similar structure, we will argue that the amount of training data required to do this would be prohibitive. Thus, our feedback mechanisms provide more of a mechanism to improve the statistical efficiency of a word recognizer rather than some fundamental new theoretical capability.

We are motivated by the following observations – ***first***, in complex images like maps, detecting letters can be an initial step in finding words, but words can also be used to find letters. *E.g.*, whether or not a word should be "terminated" after a particular character is often judged not by the spacing around the word (as it is in regular optical character recognition) but rather, by whether the recognized string is complete yet. In cases where the spacing of letters is very large, the only clue that the word has not ended yet is that we don't yet have a recognized string (*e.g.* green box in Figure 1). ***Second***, in a highly cluttered map, a reliable way to find words may be to first understand the orientation of a particular character, and then form an imaginary baseline under that character. We then search for a word consistent with that suggested baseline. In this case, it is clear that some characters provide more orientation information than others. *E.g.*, an "Oh" written as a perfect circle may be recognizable, but is not informative of the word's orientation. An "X" has four possible orientations, but the letter "E" just has a single reasonable orientation. ***Third***, in addition to the cues that words give for finding letters, and that letters give for finding words, a good modern CNN-based detector, such as the Faster R-CNN [44] or EAST [68], can also provide a good baseline based on the Gestalt of a word impression. However, as we shall see below, even such advanced modern detectors do not do a great job at detecting words in an error free manner.

Based on these observations, we have designed a system that incorporates tight feedback among character recognizers, word detectors, word recognizers, and lexicons in a CNN based model. We make the following contributions:

- We present a novel neural network architecture in which information about every individual **character** detection in an entire map, including the type of character, its location, its orientation, and its scale, can be provided as input to the network to facilitate the training of a **word** detector.
- We provide a mechanism in which our initial word guesses can be used to hypothesize that the original word detection was truncated; triggering a modification of the original word detection. We show that these "re-detections" improve both detection and recognition performance.
- We provide all of the components of an end-to-end system for map text recognition, including character recognition, word detection, word recognition, and feedback mechanisms discussed above.

The paper is organized as follows – we discuss prior work in map text recognition and related areas like scene text recognition in Sec. 2, our methodology in Sec. 3, the experimental details in Sec. 4 and conclusion in Sec. 5.

2. Related work

Text detection. The deep learning era has shown rapid advancement in computer vision techniques. However, text detection from natural scenes and documents has remained a challenging problem given the presence of extreme variations in forms, shapes and styles of scene text. As convolutional neural networks (CNNs) became more powerful, the hand crafted features from classical methods [25, 39, 60, 62,

63, 39, 7, 28, 53, 56, 11, 22] came to be replaced by end-to-end learnable representations. Pixel level methods (*e.g.* [61, 67, 16]) learn to predict pixel regions corresponding to texts in images. This is followed by a grouping algorithm to construct contiguous regions of text in the image. The instance level methods (*e.g.* [8, 20, 35]) use object detection pipelines to predict text regions. The *region proposal-based methods* (*e.g.* [26, 31, 37, 45, 66, 65]) builds on the R-CNN framework [13, 44, 14] which predicts region proposals followed by classification of these regions to determine texts. *Anchor-based methods* (*e.g.* [18, 35]) are generally derivative of the Single Shot Detector (SSD) [32]. The original image is split into patches and regions of interest for each patch are pre-defined. Prediction and regression are done only for these pre-defined regions. Textboxes [30, 29] adapts SSD [32] by allowing for quadrilaterals instead of bounding boxes. This helps in detecting texts in multiple orientations. SegLink [47] uses a fixed aspect ratio for all anchors. The links between anchor boxes are then predicted by the network. It works especially for long and multi-oriented text. EAST [68] uses the U-Net architecture [46] (a gradual up-sampling procedure) to combine feature maps with varying receptive field sizes.

Wordspotting. The notion of wordspotting—matching word images without explicitly recognizing them—was pioneered by Rath and Manmatha [41], who used the technique to identify additional instances of a given word (through its image) in a corpus of historical documents, i.e., handwritten letters. Alamazàn et al. [1] expanded the utility of this approach by learning projections from different modalities (e.g, image and text) into a shared embedding space. While still useful for query-by-image matching, the technique allows us to find the the best matching string from a lexicon of words by projecting into the embedded space. Wang and Belongie [55] used this alternate notion of wordspotting—recognition from a restricted vocabulary—in the scene text domain.

End-to-end text detection and recognition. The last decade saw an explosion of work focused on robust reading, namely scene text. Most of these works were focused on either detection or recognition, but the field has more recently begun to focus on end-to-end systems and evaluations. Wang *et al.* [54] generalized their wordspotting method to include a detection phase, using a random fern model on HOG features as a generative likelihood for individual characters. Jaderberg *et al.* [24] addressed the poor localization of word bounding boxes with a learned box regression stage. Dey *et al.* [10] explore the impact of improper word segmentation on word recognition in a wordspotting framework, addressing the issue of poor word localization. Tong *et al.* [19] and Liu *et al.* [33] simultaneously learn shared features for both detection and recognition by completely coupling the loss functions for both stages, saving computation and also improving recall. Similarly, Gomez et al. [15] directly combine a CNN for detecting text regions with a regression output that directly produces the PHOC representation of Alamazàn *et al.* [1]. Their combined network enables a reasonably fast search for any image containing a given word, in addition to the usual method of finding the best match from a given lexicon for any detected region. In a similar context, Liu *et al.* in [34] and Busta *et al.* in [3] uses text proposals from EAST [68] and YOLOv2 [43]. Bilinear sampling is used on the detection vectors and CTC-based recognition methods are used. He *et al.* [68] also uses EAST for detection of regions and combine them with character spatial information in their attention-based recognition branch.

Map processing. Much of the work on map understanding is from early document analysis literature [12, 4, 5]; Chiang *et al*. [6] provide a comprehensive review of the field. Despite the plethora of deep-learning based approaches for scene text detection, such approaches have not yet been applied to historical maps in the literature. Tarafdar *et al.* [50] propose a two-stage approach. Following a connected component analysis, initial characters are detected and identified. If these detections represent a partial lexicon word (i.e., with a letter missing from the middle), a more rigorous search for the missing character is made, subject to appropriate geometric constraints. Moreover, the existing exemplars of characters *already detected* are used as models in the search, an important form of feedback. Yu *et al.* [64] correct recognition errors, which may or may not be due to poor bounding boxes, by merging the OCR output from a set of geographically aligned maps. Several works in map reading have used geographical dictionaries or character similarities to aid in recognition. Weinman [57, 58] uses an automatically determined geographical alignment to produce a pixel-specific lexicon for detected words. The work also uses a CNN for robust word recognition.

3. Methodology

Figure 2 shows an overview of our end-to-end map word recognition system. Our system relies heavily on the PHOC representation of Alamazàn et al. [1], and will be discussed in Sec. 3.3. Our system progresses according to the following sequence of events:

1. We run 36 separate character detectors on the input map. Each character detection includes the character detected, its orientation, and its bounding box, which includes scale and location.
2. All of the information about the detected characters is passed as additional inputs, along with the original map, to a word detector. The information about the detection of each separate character (A, B, ..., Z, 0, ..., 9) is passed in 36 separate layers, along with the 3 RGB layers of the original map image, to the word detector

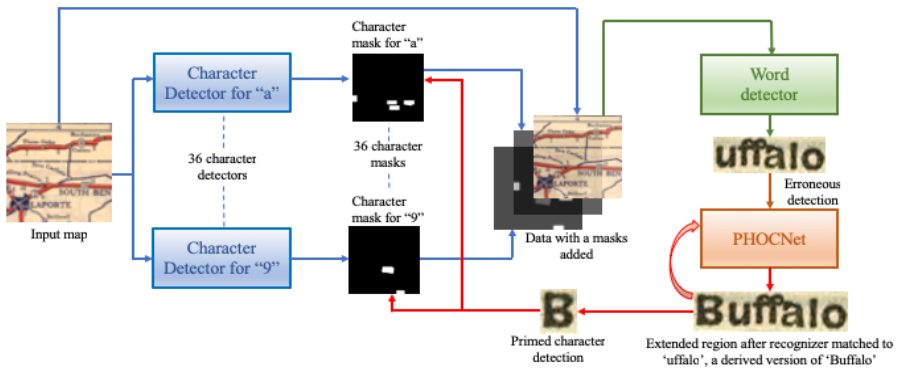

Figure 2: Flowchart of map word detection. The first step is the detection of individual characters at arbitrary positions, scales and orientations within the map. These detections are then encoded in 36 binary layers as inputs, along with the original map image to a word detector. The output from the word detector is input to the pre-trained PHOC Net (word recognizer). The word recognizer predicts *uffalo*, which is recognized as a one character shortening of *Buffalo*. The missing letter 'B' is fed back through the system as though it were recognized, to "prime" the word detector. The red lines denote the feedback path. Finally, the word is re-detected and re-recognized.

CNN.

3. The word detector outputs an initial set of word detections for the map. Of course, it may include true positives, false positives, and partial detections. Non-maximal suppression is done to eliminate duplicate detections.
4. For each word detection, a crop of the word is sent to a word recognizer, whose task is to create the PHOC vector representation of the cropped section of the map.
5. The PHOC vector is compared against a pre-computed database of PHOC vectors that correspond to an "extended lexicon" for each map (see Sec. 3.3.1).
6. The word from the extended lexicon that best matches the initial recognition is determined. If this word is a non-truncated word, the system outputs this word as its final guess. Otherwise, it performs another iteration as follows.
7. When a truncated word is matched, the missing letters from that word are written back into the character detection layers in their expected positions, and the algorithm is started again from step 2.

In the following Sec. 3.1 we describe our character detectors and in Sec. 3.2 we describe the word detector CNNs. This is followed by the word recognition model in Sec. 3.3. In Sec. 3.4 we discuss how we use the recognition model to improve the performance of the detectors.

3.1. Character detection with approximate labels

Our goal is to obtain approximate locations for individual characters in a map image. However, the training annotations consist of ground truth bounding boxes for words and corresponding labels – the precise locations of individual characters in a word are *not* available as ground-truth.

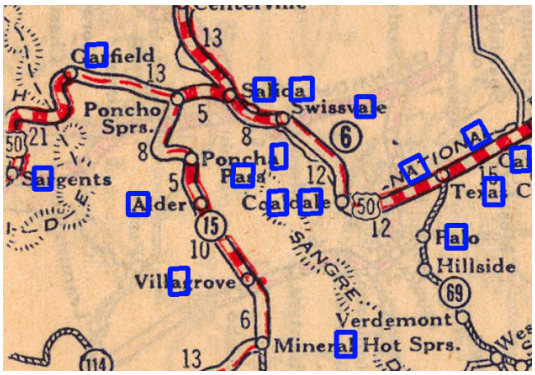

Figure 3: Approximate character location estimates. Locations of "*A*" and "*a*" are marked with blue bounding boxes. Note that certain challenging cases can still be missed, like the rotated "*A*" in SANGRE.

We first describe a pseudo-labeling strategy to accurately estimate character locations given word-level annotations. Individual character detectors are then trained on this set of approximate character locations.

Parsing characters from word annotations. While majority of text detection datasets tend to contain only word level annotations, our the first step requires us to obtain good estimates of character locations within a given word annotation. There are two ways of solving this issue – one, use approximate estimates from ground truth annotations. With conservative estimates it can almost always be ensured that a given character is contained with an estimated region. But this comes with the added disadvantage of containing other characters or distractors and non-centered bounding boxes. Two, we can use character classifiers to get character location estimates within a word. In [51, 21], the authors

have shown that this can be achieved using detectors based off a VGG16 [48] backbone.

One-versus-all character detectors. We train individual character classifiers in a one-versus-all set up on an external dataset of scene text character crops (ICDAR [27] and Chars74K [9] datasets). Given a word annotation (location in an image and the word as text), we have a restricted set of possible characters than can be present in that word-image. We run character classifiers corresponding to each character in the annotated word on all possible crops to obtain an estimate for the location of each character in that word. The exact implementation details such as localizing multiple instances of the same character in a word *etc*. are deferred to the Supplementary Material.

Constructing a multi-class CNN for character detection would lead to a network which may be confused between shapes that closely resemble a certain class of the outputs – *e.g*. a partial crop of "*d*" can be easily confused with a "*c*". Instead, for *each* character class, we retrain a separate detector model to detect it in the image in a one vs. all setup. This ensures a large pool of negative examples, allowing the network to focus on one unambiguous positive category and resulting in robust character classifiers.

Applying all 36 detectors to each map gives us *noisy estimates* of the characters in each map (Figure 3 shows this for detecting upper and lowercase "*a*"). Our goal is to use these character estimates to enhance the performance of a word detector, by giving the detector clues for where to look for words in the map. There are several ways to accomplish this, as described in the next section.

3.2. Word detection with character-level priming

The character detections obtained are very informative of the regions which may contain a word. For example consider two *a*'s detected in the word "`national`". We would be naturally inclined to look for the existence of other characters in that region and if found, we know that these characters are to be clubbed together to construct a word. We employ a similar strategy to cue the word detector to look at regions containing characters detections for possible words. It may so happen that a false positive may exist in the character detection, but we expect that as our detector trains on ground-truth word annotations, it will be able to reason about the plausible associations of characters and learn to reject isolated spurious character detections provides as inputs. We call this approach "priming" and explore two variants – firstly, we input *all* the detected masks corresponding to each character in a *single* extra channel (see Figure 4), resulting in a 4-channel input image (3 RGB channels and one binary mask). This allows the word detector to be aware of all character detections in the mask (some of which may be overlapped). Secondly, to allow the word detector to distinguish among different classes of characters, we put detections for each character class in a *separate* binary mask, resulting in a 39 channel input image (36 character channels of binary masks and 3 RGB channels). An example of this is shown in Figure 5. We then modify the architecture of the word detector's initial layer to accept the modified input images. We specifically did *not* use bounding boxes to create the character masks because their orientations are highly ambiguous, instead using the center-line of the boxes.

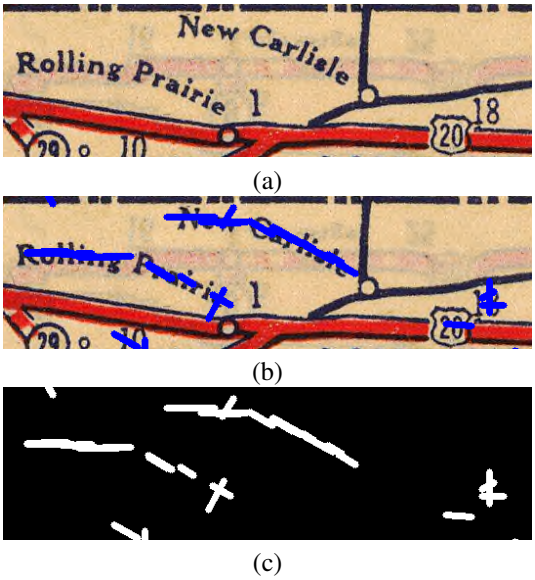

Figure 4: Character detection masks. (a) portion of the original map; (b) center-line of the bounding boxes of detected characters marked in blue; (c) the character detections are represented as masks which are then fed back to the word detector as an extra input channel, priming the network to focus on areas of the image that are more likely to have words.

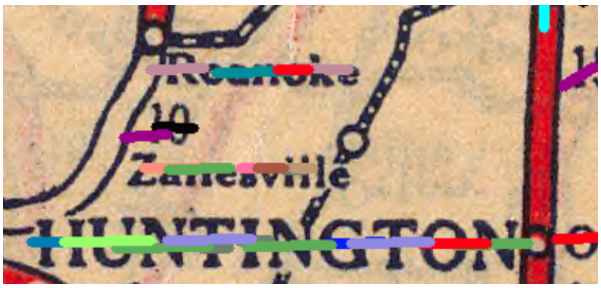

Figure 5: Noisy character detections. Illustrative examples of visualized noisy character estimates, obtained as in Sec. 3.2 (a unique color is used for each character). *E.g*. the letter "*O*" is denoted in red, appearing in `HUNTINGTON` and `Roanoke`; "*T*" is correctly detected in two places in `HUNTINGTON` (marked in purple). Note that the character model can distinguish between the letter "*O*" and numeral "0" in `10`.

This concludes the description of our detection pipeline in the first forward pass of an input image through the sys-

tem. However, later, if we discover that certain letters of a word went undetected, we can add hypothesized detections for missing letters back into these detection map layers and rerun the detector (detailed below).

3.3. Adapting PHOCNet [49] for word spotting

The PHOC representation was introduced in [1], and an architecture to learn this representation from images for word spotting was introduced in [49]. An example image explaining the PHOC representation is shown in Figure 6. We use the same architecture of as PHOCNet[49]. The initial layer and final layers are reshaped to fit our requirements. We re-scale each detection to a fixed size of 135×487 before feeding it into PHOCNet. We augment the PHOC representation with additional bins to account for case sensitivity of letters and to handle special characters present in the detected regions.

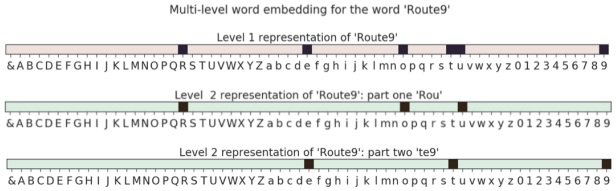

Figure 6: PHOC representation. The above figure shows a two level representation of the word `Route9`. The *bins* for PHOC are defined as 26 lower case characters, 26 upper case characters, 10 bins for numerals 0-9 and one bin for the & symbol. Level 1 puts a 1 at every bin that appears in the word, and a zero elsewhere. In Level 2, the word is split in half – "`Rou`" and "`te9`", resulting in two vectors. After a certain depth of levels, all the sub-vectors are concatenated to form a final PHOC representation.

3.3.1 Extended lexicon

We now describe how a lexicon is created for word spotting by using geo-spatial data.

Incorporating geo-spatial information. Each image in our experimental database is associated with a geo-tagged list of possible names of places in the image (henceforth referred to as GIS). We cross reference the learned PHOC embedding from each detected region with the PHOC embeddings of the words appearing in the GIS. For every map we construct an extended lexicon using the GIS information. Maps can be localized using the prior work of [59] and given their location, list of place names can be obtained from a GIS toponym dictionary. Thus, we can construct a list of geographic terms from an expanded region around a map.

Additional lexicon expansions. The above list is augmented with 5000 most frequent English words, up to 500 Roman numerals, and integers from 0 to 99. In addition, for each word in the extended lexicon we can create three versions from the same word – (i) by converting each word to all upper case; (ii) all lower case; (iii) leading uppercase by all lower case example, *e.g.* "`Rudolph`" will be converted to "`RUDOLPH`", "`rudolph`" and "`Rudolph`". We further augment this list by adding *truncated* versions of all words of length greater than equal to three. The truncations are of the form of dropping the first letter or dropping the last letter from the word, *e.g.* "`Rudolph`" gives rise to the new words "`Rudolp`", "`udolph`" and "`Rudolph`". We call the initial word in the extended lexicon as the *root* word, the subsequent versions of case sensitive words as *altered case* words and the words with dropped first or last character as *decapitated* words.

In the next Sec. 3.4 we discuss the ways we introduce feedback and incorporate them in learning algorithm.

3.4. The feedback loop from recognition

We briefly discussed how feedback could be added from the results of the character detector at the beginning of this section. In this section, we discuss about how the recognizer helps improve detection accuracy.

For each detection, we compare its PHOC representation to the extended lexicon. If the root word is recognized, we stop and report the detection. If any derived word is recognized, we identify if it is a decapitated word. If it is, we use the following methods to improve the recognition accuracy — ***(i)*** we can match the derived word to its root word (*e.g.* after detecting "`udolph`" we can match it to the root word "`Rudolph`"); ***(ii)*** we can go back to the map, extend the detected region in the direction of the missing character and redo word spotting (Sec 3.3). For (ii), if the new region detection matches with the root word we call it an *extended detection*; ***(iii)*** once we have the extended detections, we again go back to the map and create masks for the particular character. Then, we re-run detection followed by word spotting on these modified inputs.

In the next part (Sec. 4), we begin by describing our dataset followed by the evaluation of our method and comparisons with relevant baselines.

4. Experiments

4.1. Dataset description

We perform our experiments on a curated list of 31 historical maps [42] chosen from nine atlases in the David Rumsey Map Collection. [1] The maps are of U.S. regions and are dated between the late 19th and early 20th century, typically of sizes 3000×3000. The images and corresponding annotations for word detection and recognition can be obtained from [42]. The train/val/test splits used to report

[1] https://www.davidrumsey.com

our 5-fold cross-validation results are summarized in Table 1. It should be noted that the annotations are not just rectangular bounding boxes around a word, but can also be oriented polygons. Each rectangle or polygon is associated with a label containing the string corresponding to the word in the image.

Table 1: Dataset description for detection. Here we describe the number of images and total annotations in the maps dataset [42]. Note here the average number of annotations are reported for training, validation and test data as we do five-fold cross-validation for training the models.

	#images	#annotations
Train	18	17,151
Validation	7	9,136
Test	6	7,668

4.2. Baselines and ablation settings

We have experimented with Resnet 50 [17] as our backbone network for the task of text detection. We report results here using the EAST word detector (results using the Faster R-CNN architecture [44] are in the Supplementary Material). For recognition, we use the PHOCNet architecture presented in [49]. We now discuss our implementation details and experimental settings, along with ablations, to show the efficacy of our proposed methods:

Ablation settings for detection.

- **EAST:** we use the EAST detector with a Resnet-50 backbone. We use the same architecture as in the original work. We use our map data to finetune the fourth block of Resnet-50, feature merging branch and the output layer.
- **EAST+1C:** single-channel approximate character localization masks are appended to the input RGB image. This shows the effect of priming the word detector with character level detections. The first convolution layer of the first block along with the fourth block of Resnet-50, feature merging branch and output layer is finetuned for this model.
- **EAST+36C:** the results of character detection are appended as separate channels to the input image, one for each character class. This allows the model to reason separately about the presence of individual characters in a region, their effect on the presence of a word and its effect on word detection. The finetuning of this model is similar to EAST+1C.
- **EAST+36C+feedback:** information from the word spotting module is fed back into the detector as described in Sec. 3.4. The feedback information is provided on-the-fly during test time, without requiring the word detectors to be re-trained.

Ablation settings for word-spotting.

- **PHOC:** as mentioned in Sec. 3.3, the baseline method used for recognition is PHOCNet [49]. We have used the same architecture as in the original work. We retrained the network from scratch to detect words. The baseline is trained to recognize words irrespective of their case. The bounding box annotations in the training images is used as the training data. The lexicon used for word-spotting is created from the GIS data associated with the maps.
- **PHOC+wordvar:** the lexicon from GIS is expanded with word variations as described in Sec. 3.3.1.
- **PHOC+wordvar+ext:** re-doing detection and word-spotting with feedback from the truncated-word-extension stage (PHOC+wordvar), as described in Sec. 3.4 (iii).

Ablation settings for the end-to-end pipeline.

- **EAST+36C $\rightarrow$ PHOC:** the best word-detection model output is fed into the best-performing word-spotting model.
- **EAST+36C+feedback $\rightarrow$ PHOC+ext:** the effect of incorporating feedback from word-spotting into the word detection model, and then re-running word-spotting.

Implementation details. The detection architecture requires a fixed sized input (512×512 for Resnet-50). However, the images in the map dataset are of the order of 3000×3000 pixels. If we shrink the image to the input size of the networks, then we lose a huge amount of information. Moreover, the smaller texts become illegible. Therefore, we use a sliding window approach to feed the image into the detection network. To avoid words getting broken between consecutive windows we keep overlaps of 200 pixels between windows. The output of the detector may contain overlapping bounding-box predictions, which is handled by non-maximal suppression (NMS).

4.3. Results

Detection. The detection results on the map-dataset are shown in Table 2. The *baseline* EAST detector trained on our dataset shows relatively low precision and recall, with an F_1-score of 0.426. When we add the character detection information back into the image, the precision and recall improves, leading to an F_1-score of 0.551 and 0.675 for the 1 channel and 36 channel character masks, respectively. While using a single-channel character location mask does show some improvement, providing this information as separate channels for each character class allows the detector to reason about how individual characters are grouped into words, resulting in more information that helps the task of word detection.

Word-spotting. The *baseline* PHOC [49] is trained with the SynthText dataset [23] and finetuned with the word regions from map annotations. The results are summarized in

Table 2: Comparison of proposed detection algorithms on maps dataset [42].

Method	Precision	Recall	F_1-score
EAST	0.431	0.417	0.426
EAST+1C	0.541	0.560	0.551
EAST+36C	0.673	0.677	0.675
EAST+36C+ feedback	0.682	0.701	0.691
Finetuned MASK-textspotter [36]	0.603	0.764	0.674

Table 3. In section 3.3.1, we discussed the various versions of lexicons we obtained from the GIS. The basic lexicon consisting of concatenated GIS information yields 39.88% accuracy using the output of the EAST+36C detector. When we include word variations the performance increases to 41.08%. This is because the variation of the root word allows from correct spotting of decapitated word detections. When using feedback we get a further improvement. The feedback information helps to break ties for similar derived words with dissimilar root word. *E.g.* consider the root words `Carfield` and `Hatfield`. Consider a partial detection where the detector gets only "`field`". Extending the bounding box to the left would include the missing prefix, allowing for a better match to the correct root word.

Table 3: Comparison of the proposed recognition algorithms on multiple versions of GIS data for maps dataset [42].

Method	**Accuracy** (%)
PHOC	39.88
PHOC+wordvar	41.08
PHOC+wordvar+ext	43.62

End-to-end recognition. We now consider the overall performance of the entire detection and recognition pipeline. There are certain points to consider when evaluating end-to-end results. *E.g.* an extremely high-recall recognition model might get around 99% accuracy, but if the detector in that pipeline has either a very low recall or extremely bad precision, the overall performance will be heavily penalized despite the high-performing recognizer. Similarly, have a weak recognizer coupled with a high-performing detector will also result in sub-optimal end-to-end performance. We compute the Precision and Recall metrics as follows — any true positive in both detection and recognition is considered a true positive for the whole system. Any false positive in detection *and* any true detection subsequently not recognized correctly is considered a false positive. A false negative for the end-to-end system is any bounding box which is not detected by the detector. Having defined all the terms we can now compute the precision and recall for the whole system.

We consider the extended character detections to improve *detection* as well as *recognition* accuracy. The final end-to-end system ("EAST+feedback+PHOC") is compared with the "EAST+PHOCNet" with the extended lexicon. While the latter performs with an F_1-score of 0.4723, our final model's F_1-score is 0.4948. The results are compiled in Table 4.

Table 4: End-to-end comparisons for the complete pipeline of text detection and recognition showing the effect of feedback.

Method	Precision	Recall	F_1-score (%)
EAST to PHOC	0.4084	0.5598	0.4723
EAST+feedback to PHOC	0.4244	0.5933	0.4948

5. Discussion and Conclusions

One of our main contributions is the addition of extra channels to record the results of 36 character detectors which are then used as addition inputs to our word detectors. In addition to detecting words that may otherwise be completely missed, there is another benefit to these extra channels. Object detectors tend to break the word regions into separate parts when used for text detection, leading to poor confidence scores for the sub-regions. To mitigate this in [38], the authors have used multiple RPNs and aggregated the proposed anchors. In this work, we propose a simpler mechanism to overcome this problem. The addition of the extra character channels cues the detectors to look at extended regions. The masks strengthen the response for textual regions increasing the likelihood of correct detection.

Another major contribution is the feedback from the lexicon to the detector, giving us a way to revist the detections and find previously missed letters. Consider the word "`Brattleboro`" and suppose the cropped detection says "`Brattle`". The closest English word to match this would be "`Battle`". Let us assume the recognizer matches with "`Brattle`" and "`Battle`" using our variations of the GIS lexicon. Since "`Brattle`" is a derived word from "`Brattleboro`", we go back to the map and extend the detection boundary towards the right of the original detection. Now the new cropped image has "`Brattleb`" written in it. If we compare the recognized word now, the PHOC-distance between the embeddings of the cropped region and "`Brattleboro`" would decrease and the PHOC-distance between the cropped region and "`Battle`" would increase. This helps in correctly identifying the word or its root word in the GIS lexicon.

We show consistent performance gains on a challenging problem by incorporating feedback and priming mechanisms at various levels of the detection and recognition

pipeline. The improved performance on this specific domain suggests a general approach for designing pipelines that incorporate such feedback at multiple levels in general object detection as future work.

6. Acknowledgement

This work is supported by the National Science Foundation under Grant No.:1526350.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2552–2566, 2014. 3, 6

[2] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966. 2

[3] M. Busta, L. Neumann, and J. Matas. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2204–2212, 2017. 3

[4] R. Cao and C. Tan. Text/graphics separation in maps. In D. Blostein and Y.-B. Kwon, editors, *Graphics Recognition*, volume 2390 of *Lecture Notes in Computer Science*, pages 167–177. 2002. 1, 3

[5] Y.-Y. Chiang and C. A. Knoblock. Classification of line and character pixels on raster maps using discrete cosine transformation coefficients and support vector machine. In *Proc. Intl. Conf. on Pattern Recognition*, volume 2, pages 1034–1037, 2006. 3

[6] Y.-Y. Chiang, S. Leyk, and C. A. Knoblock. A survey of digital map processing techniques. *ACM Comput. Surv.*, 47(1):1:1–1:44, May 2014. 1, 3

[7] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 440–445. IEEE, 2011. 3

[8] Y. Dai, Z. Huang, Y. Gao, Y. Xu, K. Chen, J. Guo, and W. Qiu. Fused text segmentation networks for multi-oriented scene text detection. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3604–3609. IEEE, 2018. 3

[9] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009. 5

[10] S. Dey, A. Nicolaou, J. Lladós, and U. Pal. Evaluation of the effect of improper segmentation on word spotting. *CoRR*, abs/1604.06243, 2016. 3

[11] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2963–2970. IEEE, 2010. 3

[12] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 10(6):910–918, 1988. 3

[13] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 3

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 3

[15] L. Gómez, A. Mafla, M. Rusinol, and D. Karatzas. Single shot scene text retrieval. In *Proc. European Conf. on Computer Vision*, 2018. 3

[16] D. He, X. Yang, C. Liang, Z. Zhou, A. G. Ororbi, D. Kifer, and C. Lee Giles. Multi-scale fcn with cascaded instance aware segmentation for arbitrary oriented word spotting in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3519–3528, 2017. 3

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[18] P. He, W. Huang, T. He, Q. Zhu, Y. Qiao, and X. Li. Single shot text detector with regional attention. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 6, 2017. 3

[19] T. He, Z. Tian, W. Huang, C. Shen, Y. Qiao, and C. Sun. An end-to-end textspotter with explicit alignment and attention. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2018. 3

[20] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu. Deep direct regression for multi-oriented scene text detection. *arXiv preprint arXiv:1703.08289*, 2017. 3

[21] H. Hu, C. Zhang, Y. Luo, Y. Wang, J. Han, and E. Ding. Wordsup: Exploiting word annotations for character based text detection. In *Proc. ICCV*, 2017. 4

[22] W. Huang, Z. Lin, J. Yang, and J. Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1241–1248, 2013. 3

[23] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014. 7

[24] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *Intl. Journal of Computer Vision*, 116(1):1–20, 2016. 3

[25] A. K. Jain and B. Yu. Automatic text location in images and video frames. *Pattern recognition*, 31(12):2055–2076, 1998. 3

[26] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo. R2cnn: rotational region cnn for orientation robust scene text detection. *arXiv preprint arXiv:1706.09579*, 2017. 3

[27] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015. 5

[28] J.-J. Lee, P.-H. Lee, S.-W. Lee, A. Yuille, and C. Koch. Adaboost for text detection in natural scene. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 429–434. IEEE, 2011. 3

[29] M. Liao, B. Shi, and X. Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE Transactions on Image Processing*, 27(8):3676–3690, 2018. 3

[30] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu. Textboxes: A fast text detector with a single deep neural network. In *AAAI*, pages 4161–4167, 2017. 3

[31] M. Liao, Z. Zhu, B. Shi, G.-s. Xia, and X. Bai. Rotation-sensitive regression for oriented scene text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5909–5918, 2018. 3

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 3

[33] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. FOTS: fast oriented text spotting with a unified network. 2018. 3

[34] X. Liu, D. Liang, S. Yan, D. Chen, Y. Qiao, and J. Yan. Fots: Fast oriented text spotting with a unified network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5676–5685, 2018. 3

[35] Y. Liu and L. Jin. Deep matching prior network: Toward tighter multi-oriented text detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3454–3461. IEEE, 2017. 3

[36] P. Lyu, M. Liao, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–83, 2018. 8

[37] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 2018. 3

[38] Y. Nagaoka, T. Miyazaki, Y. Sugaya, and S. Omachi. Text detection by faster r-cnn with multiple region proposal networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 15–20. IEEE, 2017. 8

[39] L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Asian Conference on Computer Vision*, pages 770–783. Springer, 2010. 3

[40] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 2

[41] T. Rath and R. Manmatha. Word spotting for historical documents. *Intl. Journal on Document Analysis and Recognition*, 9(2-4):139–152, 2007. 3

[42] A. Ray, Z. Chen, B. Gafford, N. Gifford, J. J. Kumar, A. Lamsal, L. Niehus-Staab, J. Weinman, and E. Learned-Miller. Historical map annotations for text detection and recognition. Technical report, Grinnell College, Department of Computer Science, Grinnell College, Grinnell, Iowa 50112, Oct. 2018. 6, 7, 8

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 3

[44] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2, 3, 7

[45] L. Rong, E. MengYi, L. JianQiang, and Z. HaiBin. weakly supervised text attention network for generating text proposals in scene images. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, pages 324–330. IEEE, 2017. 3

[46] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[47] B. Shi, X. Bai, and S. Belongie. Detecting oriented text in natural images by linking segments. *arXiv preprint arXiv:1703.06520*, 2017. 3

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[49] S. Sudholt and G. A. Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 277–282. IEEE, 2016. 6, 7

[50] A. Tarafdar, U. Pal, P. P. Roy, N. Ragot, and J.-Y. Ramel. A two-stage approach for word spotting in graphical documents. In *Proc. Intl. Conf. on Document Analysis and Recognition*, pages 319–323, 2013. 3

[51] S. Tian, S. Lu, and C. Li. Wetext: Scene text detection under weak supervision. In *Proc. ICCV*, 2017. 4

[52] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017. 2

[53] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011. 3

[54] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Proc. Intl. Conf. on Computer Vision*, pages 1457–1464, 2011. 3

[55] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. European Conf. on Computer Vision*, September 2010. 3

[56] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012. 3

[57] J. Weinman. Toponym recognition in historical maps by gazetteer alignment. In *Proc. Intl. Conf. on Document Analysis and Recognition*, pages 1044–1048, 2013. 1, 3

[58] J. Weinman. Geographic and style models for historical map alignment and toponym recognition. In *Proc. IAPR International Conference on Document Analysis and Recognition*, Nov. 2017. 1, 3

[59] J. Weinman. Geographic and style models for historical map alignment and toponym recognition. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, pages 957–964. IEEE, 2017. 6

[60] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1083–1090. IEEE, 2012. 3

[61] C. Yao, X. Bai, N. Sang, X. Zhou, S. Zhou, and Z. Cao. Scene text detection via holistic, multi-channel prediction. *arXiv preprint arXiv:1606.09002*, 2016. 3

[62] C. Yi and Y. Tian. Text string detection from natural scenes by structure-based partition and grouping. *IEEE Transactions on Image Processing*, 20(9):2594–2605, 2011. 3

[63] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao. Robust text detection in natural scene images. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):970–983, 2014. 3

[64] R. Yu, Z. Luo, and Y.-Y. Chiang. Recognizing text on historical maps using maps from multiple time periods. In *Proc. Intl. Conf. on Pattern Recognition*, pages 3993–3998, 2016. 1, 3

[65] L. Yuliang, J. Lianwen, Z. Shuaitao, and Z. Sheng. Detecting curve text in the wild: New dataset and new solution. *arXiv preprint arXiv:1712.02170*, 2017. 3

[66] S. Zhang, Y. Liu, L. Jin, and C. Luo. Feature enhancement network: A refined scene text detector. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3

[67] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4159–4167, 2016. 3

[68] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: an efficient and accurate scene text detector. In *Proc. CVPR*, pages 2642–2651, 2017. 2, 3

Supplementary Material: Tight Coupling of Character, Word, and Place Recognition for End-to-End Text Recognition in Maps

Abstract

In the supplemental sections, we report additional results using the standard Faster R-CNN detector instead of the EAST detector reported in the main paper (Sec. 1). In addition, we provide details on how character-level parsing is done from word-level ground-truth annotations (Sec. 2). The results of this character level parsing are used to train our individual character models, as reported in the main paper.

1. Faster R-CNN Results

To show that our approach generalizes beyond a specific detector architecture, we replace the EAST detector [10] with the widely-used Faster R-CNN detector [7] for the word detection task. We demonstrate that we continue to see benefits from the main feedback mechanisms presented in the paper in the context of using another mainstream detection architecture.

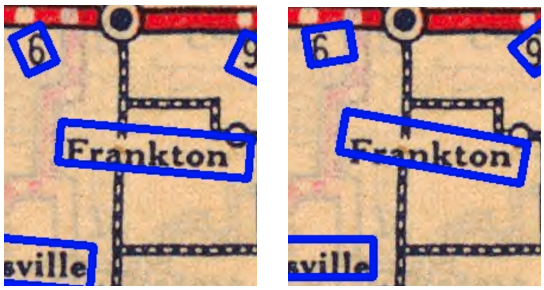

Figure 1: Detections from Faster R-CNN and EAST. A qualitative example showing that EAST (*left*) localizes words better as compared to Faster R-CNN (*right*) – tighter bounding-box, complete words enclosed in the box, *etc*. The lower quality localization affects the downstream task of word recognition, which is why the EAST-based pipeline performs better than Faster R-CNN.

Detection. The detection results on the map-dataset are shown in Table 1. The *baseline* Faster R-CNN detector [7] trained on our dataset shows relatively low precision and recall, with an F_1-score of 0.303. When we add the character detection information back into the image, the precision and recall dramatically improves, leading to an F_1-score of 0.563 and 0.652 for the 1 channel and 36 channel character masks, respectively. While using a single-channel character location mask does show some improvement, providing this information as separate channels for each character class allows the detector to reason about how individual characters are grouped into words, resulting in more information that improves the performance of word detection.

Table 1: Word detection results using Faster R-CNN on the maps dataset [6].

Method	Precision	Recall	F_1-score
F-RCNN	0.322	0.289	0.303
F-RCNN+1C	0.614	0.520	0.563
F-RCNN+36C	0.684	0.623	0.652
F-RCNN+36C +feedback	0.697	0.637	0.665

Word-spotting. For recognition of the words detected in a map by the Faster RCNN, we use a *wordspotting* approach — associating images of words to the best matching string from a given lexicon of words [5, 9, 1]. In particular, we adopt the approach of [8], which performs this matching using the PHOC representation [1] for a word.

The *baseline* PHOC-Net [8] is trained with the Synth-Text dataset [3] and finetuned on the word regions from map annotations. The results are summarized in Table 2. The basic lexicon consisting of concatenated GIS information yields 34.59% accuracy using the output of the FR-CNN+36C detector. When using feedback we get a further improvement.

End-to-end recognition. We consider the extended character detections to improve *detection* as well as *recognition* accuracy. The final end-to-end system ("F-RCNN+feedback+PHOC") is compared with the "F-RCNN+PHOCNet" with the extended lexicon. While the latter performs with an F_1-score of 0.434, our final model's F_1-score is 0.452. The results are compiled in Table 3.

Table 2: Comparison of the word recognition algorithms on multiple versions of GIS data for maps dataset [6].

Method	**Accuracy** (%)
PHOC	34.59
PHOC+wordvar	35.74
PHOC+wordvar+ext	37.29

Table 3: End-to-end comparisons for the complete pipeline of text detection and recognition showing the effect of feedback.

Method	Precision	Recall	F_1-score
F-RCNN to PHOC	0.392	0.487	0.434
F-RCNN+feedback to PHOC	0.408	0.507	0.452

2. Approximate character localization

The map dataset provides us with word bounding boxes and the words present in those boxes. We detail the procedure for obtaining approximate locations of each character in that word bounding box (which is not part of the ground-truth annotations). Given a region (a bounding box) containing some set of characters (annotations/labels), we implement an algorithm that computes the score for each character given any possible sub-regions (by *possible* we mean any sub-region within the word image which can contain the character). Given these scores we can obtain the maximum sum of scores such that all characters in that word can be accommodated in the given region.

The character scores are obtained running a per-character CNN on each word region in a sliding window fashion. The CNNs are trained on ICDAR [4] and Chars74K [2] scene text character crops. There is a substantial domain shift between the external training data we use to train the per-character recognizers and the map text we are applying them on, however these noisy estimates were sufficient for our purpose.

We use Algorithm 1 to obtain the character location estimates. It should be noted that this makes two assumptions: (1) the input region is rectangular, (2) we have a rough estimate of each character's beginning and end position (column). A rough estimate of a character's position would be a crop around $\frac{l}{n}$ in width, where l is the length of the word image and n is the number of characters in the word. An example is shown in Figure 3.

Algorithm 1: Pseudo-code to estimate character regions from an annotated word image

Result: begin_pos, end_pos
Inputs: word_image, word, rough_begin_estimate, rough_end_estimate;
begin_pos, end_pos = [];
for *i in list(word)* **do**
    load detector[word[i]];
    max_score, b_j, b_k = 0;
    for *j,k* $\in$ *[rough_begin_estimate[i], rough_end_estimate[i]]* **do**
        score = detector[i](word_image, j, k);
        if *score* $\geq$ *max_score* **then**
            b_j = j;
            b_k = k;
            max_score = score;
        end
    end
    begin_pos.append(b_j), end_pos.append(b_k);
end

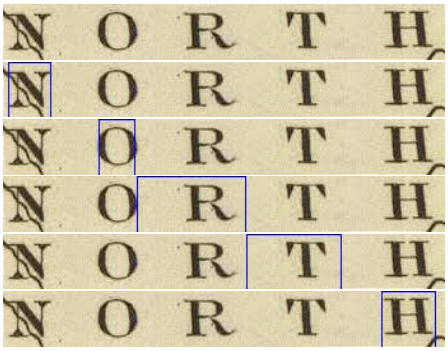

Figure 2: Character parsing from word annotations. Top row: we are provided with the cropped word image, and the word annotation ("NORTH"). Knowing the letters present in the word, we run each character classifier over all sub-regions of this image. The sub-region with the highest response is visualized in the each row, for the letters N, O, R, T and H. This gives us approximate character localization.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(12):2552–2566, 2014. 1

[2] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009. 2

[3] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014. 1

[4] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015. 2

[5] T. Rath and R. Manmatha. Word spotting for historical docu-

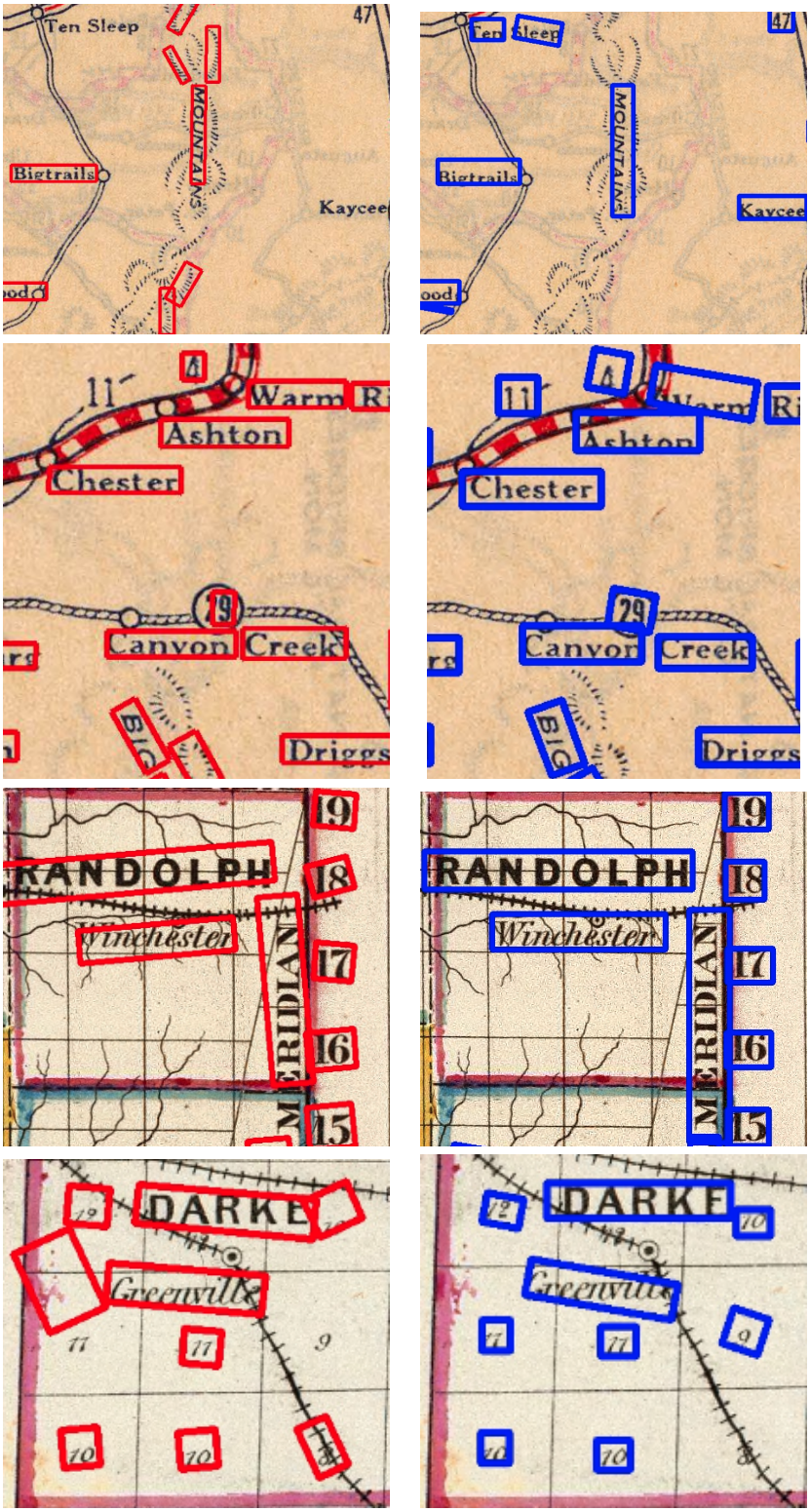

Figure 3: Qualitative results. We show comparative detection results for the baseline EAST detector (*left*) and the final results from our improved pipeline (*right*). There is a clear reduction in false positives (the short parallel lines wrongly considered as text is a prominent example). Further, challenging words are better localized (*e.g.* top row: "MOUNTAINS", 3rd row: "MERIDIAN").

ments. *Intl. Journal on Document Analysis and Recognition*, 9(2-4):139–152, 2007. 1

[6] A. Ray, Z. Chen, B. Gafford, N. Gifford, J. J. Kumar, A. Lamsal, L. Niehus-Staab, J. Weinman, and E. Learned-Miller. Historical map annotations for text detection and recognition. Technical report, Grinnell College, Department of Computer Science, Grinnell College, Grinnell, Iowa 50112, Oct. 2018. 1, 2

[7] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[8] S. Sudholt and G. A. Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 277–282. IEEE, 2016. 1

[9] K. Wang and S. Belongie. Word spotting in the wild. In *Proc. European Conf. on Computer Vision*, September 2010. 1

[10] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: an efficient and accurate scene text detector. In *Proc. CVPR*, pages 2642–2651, 2017. 1