Dictionary-guided Scene Text Recognition

Nguyen Nguyen[1], Thu Nguyen[1,2,4], Vinh Tran[6],
Minh-Triet Tran[3,4], Thanh Duc Ngo[2,4], Thien Huu Nguyen[1,5], Minh Hoai[1,6]

[1]VinAI Research, Hanoi, Vietnam; [2]University of Information Technology, VNU-HCM, Vietnam;
[3]University of Science, VNU-HCM, Vietnam;
[4]Vietnam National University, Ho Chi Minh City, Vietnam;
[5]University of Oregon, Eugene, OR, USA; [6]Stony Brook University, Stony Brook, NY, USA

{v.nguyennm, v.thunm15, v.thiennh4, v.hoainm}@vinai.io

thanhnd@uit.edu.vn, tmtriet@fit.hcmus.edu.vn, tquangvinh@cs.stonybrook.edu

Abstract

Language prior plays an important role in the way humans detect and recognize text in the wild. Current scene text recognition methods do use lexicons to improve recognition performance, but their naive approach of casting the output into a dictionary word based purely on the edit distance has many limitations. In this paper, we present a novel approach to incorporate a dictionary in both the training and inference stage of a scene text recognition system. We use the dictionary to generate a list of possible outcomes and find the one that is most compatible with the visual appearance of the text. The proposed method leads to a robust scene text recognition model, which is better at handling ambiguous cases encountered in the wild, and improves the overall performance of state-of-the-art scene text spotting frameworks. Our work suggests that incorporating language prior is a potential approach to advance scene text detection and recognition methods. Besides, we contribute VinText, a challenging scene text dataset for Vietnamese, where some characters are equivocal in the visual form due to accent symbols. This dataset will serve as a challenging benchmark for measuring the applicability and robustness of scene text detection and recognition algorithms. Code and dataset are available at https://github.com/VinAIResearch/dict-guided.

1. Introduction

Scene text detection and recognition is an important research problem with a wide range of applications, from mapping and localization to robot navigation and accessibility enhancement for the visually impaired. However, many text instances in the wild are inherently ambiguous due to artistic styles, weather degradation, or adverse illumination conditions. In many cases, the ambiguity cannot be resolved without reasoning about the language of the text.

In fact, one popular approach to improve the performance of a scene text recognition system is to use a dictionary and cast the predicted output as a word from the dictionary. The normal pipeline for processing an input image consists of: (1) detect text instances, (2) for each detected text instance, generate the most probable sequence of characters, based on local appearance of the text instance without a language model, and (3) find the word in the dictionary that has smallest edit distance (also called Levenshtein distance [14]) to the generated sequence of characters and use this word as the final recognition output.

However, the above approach has three major problems. First, many text instances are foreign or made-up words that are not in the dictionary so forcing the output to be a dictionary word will yield wrong outcomes in many cases. Second, there is no feedback loop in the above feed-forward processing pipeline; the language prior is not used in the second step for scoring and generating the most probable sequence of characters. Third, edit distance by itself is indeterminate and ineffective in many cases. It is unclear what to output when multiple dictionary words have the same edit distance to the intermediate output character sequence. Moreover, many languages have special symbols that have different roles than the main characters of the alphabet, so the uniform treatment of the symbols and characters in edit distance is inappropriate.

In this paper, we address the problems of the current scene text recognition pipeline by introducing a novel approach to incorporate a dictionary into the pipeline. Instead of forcing the predicted output to be a dictionary word, we use the dictionary to generate a list of candidates, which will subsequently be fed back into a scoring module to find the output that is most compatible with the appearance feature. One additional benefit of our approach is that we can incorporate the dictionary into the end-to-end training procedure,

training the recognition module with hard examples.

Empirically, we evaluate our method on several benchmark datasets including TotalText [3], ICDAR2013 [10], ICDAR2015 [11] and find that our approach of using a dictionary yield benefits in both training and inference stages. We also demonstrate the benefits of our approach for recognizing non-English text. In particular, we show that our approach works well for Vietnamese, an Austroasiatic language based on Latin alphabet with additional accent symbols (´, `, ̉, ., ˜) and derivative characters (ô, ê, â, ă, ơ, ư). Being the native language of 90 million people in Vietnam and 4.5 million Vietnamese immigrants around the world, Vietnamese texts appear in many scenes, so detecting and recognizing Vietnamese scene text is an important problem on its own. Vietnamese script is also similar to other scripts such as Portuguese, so an effective transfer learning technique for Vietnamese might be applicable to other languages as well. To this end, a contribution of our paper is the introduction of an annotated dataset for Vietnamese scene text, and our experiments on this dataset is a valuable demonstration for the benefits of the proposed language incorporation approach.

In summary, the contributions of our paper are twofold. First, we propose a novel approach for incorporating a language model into scene text recognition. Second, we introduce a dataset for Vietnamese scene text with 2000 fully annotated images and 56K text instances.

2. Related Work

The ultimate task of our work is scene text spotting [4, 15, 17, 19, 24, 29, 31], which requires both detecting and recognizing detected text instances. However, the main technical focus of our work is on the recognition stage. Currently, there are two main approaches in the recognition stage. The first approach is based on character segmentation and recognition [2, 7, 9, 20, 31]; it requires segmenting a text region into individual characters for recognition. One weakness of this approach is that the characters are independently recognized, failing to incorporate a language model in the processing pipeline. The second approach is based on recurrent neural networks [26] with attention [6, 17, 18, 30] or CTC loss [5, 28, 34]. This approach decodes a text instance sequentially from the first to the last character; the most recently recognized character will be fed back to a recurrent neural network for predicting the next character in the text sequence. In theory, with sequential decoding, this approach can implicitly learn and incorporate a language model, similar to probabilistic language models in the natural language domain [12, 25, 27]. However, this approach cannot fully learn a language model due to the limited number of words appearing in the training images. Furthermore, because of the implicitness of the language model, there is no guarantee that the model will not output a nonsensical sequence of characters.

A dictionary is an explicit language model, and the benefits of a dictionary for scene text recognition are well established. In most previous works, a dictionary was used to ensure that the output sequence of characters is a legitimate word from the dictionary, and it improved the accuracy immensely. Furthermore, if one could correctly reduce the size of the dictionary (e.g., only considering words appearing in the dataset), the accuracy would increase further. All of these are the evidence for the importance of the dictionary, and it does matter how the dictionary is used [32]. However, the current utilization of dictionaries based on the smallest edit distance [14] is too elementary. In this paper, we propose a novel method to incorporate a dictionary in both training and testing phases, harnessing the full power of the dictionary.

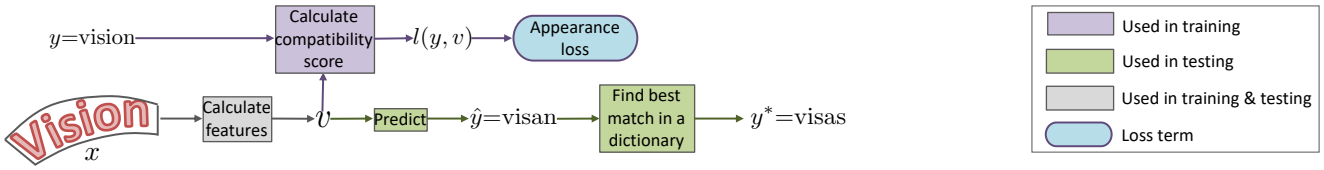
Compared to the number of datasets for other visual recognition tasks such as image classification and object detection, there are few datasets for scene text spotting. Most datasets including ICDAR2015 [11], Total Text [3], and CTW1500 [33] are for English only. Only the ICDAR2017 dataset [21] is multi-lingual with nine languages, which was recently expanded with an additional language to become ICDAR2019 [22]. However, this dataset also does not have Vietnamese. Our newly collected Vietnamese scene text dataset will contribute to the effort of developing robust multi-lingual scene text spotting methods.

3. Language-Aware Scene Text Recognition

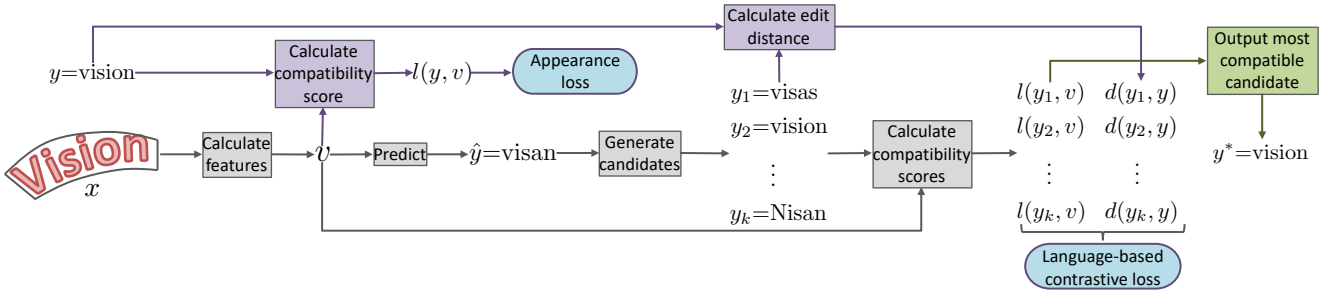
To resolve the inherent ambiguity of scene text in the wild, we propose to incorporate a dictionary into the recognition pipeline. From the initial recognition output, we use the dictionary to generate a list of additional candidates, which will subsequently be evaluated by a scoring module to identify the output that is most compatible with the appearance feature. We also use the dictionary during the training stage to train the recognition module to recognize the correct text instance from a list of hard examples. In this section, we will describe the recognition pipeline and how the candidates are generated in details. We will also describe the architecture of our network and the loss functions for training this network.

3.1. Recognition pipeline

Our scene text spotting system consists of two stages: detection and recognition. Given an input image, the detection stage will detect text instances in the image, which will be then passed to the recognition stage. The main focus of our paper is to improve the recognition stage, regardless of the detection algorithm. Specifically in this paper, we propose to use the state-of-the-art detection modules of ABCNet [19] and MaskTextSpotterV3 [16], but other detection algorithms can also be used. For brevity, we will describe our method together with the ABCNet framework in this



(a) The normal scene text recognition pipeline



(b) The proposed scene text recognition pipeline

Figure 1: **Traditional recognition pipeline (a) and proposed pipeline (b).** In the traditional pipeline, the output is forced to be in the dictionary. The dictionary is only used at the inference time, during a post-processing step. In the proposed approach, the dictionary is used for both inference and training. The dictionary is used to generate a list of candidates, and the candidates are evaluated by a compatibility scoring module. The final output does not have to be a word in the dictionary.

section, but we will demonstrate the empirical benefits of our method with both ABCNet and MaskTextSpotterV3 in the experiment section.

Fig. 1b depicts the processing pipeline of the recognition stage of our method. Given a detected text instance $\mathbf{x}$ (delineated by two Bezier curves [19]), a fixed-size feature map $\mathbf{v}$ will be calculated (using Bezier alignment module [19]). From $\mathbf{v}$, we obtain an initial recognition output $\hat{\mathbf{y}}$. We will then compile a list of candidate words $\mathbf{y}_1, \ldots, \mathbf{y}_k$, which are dictionary words with smallest edit distances to $\mathbf{y}$. We will then calculate the compatibility scores between each candidate word $\mathbf{y}_i$ and the feature map $\mathbf{v}$, and output the word with the highest compatibility score.

During training, we also calculate the compatibility score between the appearance feature map v and the ground truth word $\mathbf{y}$, which is used to calculate the appearance loss. We also minimize a contrastive loss, which is defined based on the compatibility scores between the feature map v and the list of candidate words $\mathbf{y}_1, \ldots, \mathbf{y}_k$.

3.2. Candidate generation

We use a dictionary to generate a list of candidate words in both inference and training phases. During inference, given the initial recognition output $\hat{\mathbf{y}}$, the list of candidates is the k dictionary words with smallest edit distance (Levenshtein distance [14]) to $\hat{\mathbf{y}}$. For example, if $\hat{\mathbf{y}} = visan$ and $k = 10$ then the list of candidates will be: *visas*, *vise*, *vised*, *vises*, *visi*, *vising*, *vision*, *visit*, *visor*, *vista*. During training, we use both the ground truth word $\mathbf{y}$ and the initial recognition output $\hat{\mathbf{y}}$ to generate the list of candidate words, creating a list with a total of k words.

3.3. Training losses

To train our recognition network, we minimize an objective function that is the weighted combination of two losses. The first loss is defined based on the negative log likelihood of the ground truth. The second loss is defined for the list of candidate words to maximize the likelihood of the ones that are close to the ground truth while minimizing the likelihood of the candidates that are further away from the ground truth.

The negative log likelihood for a feature map $\mathbf{v}$ and a word $\mathbf{y}$ is calculated as follows. First, using recurrent neural network with attention [1], we obtain a probability matrix $\mathbf{P}$ of size $s \times m$, where m is the maximum length of a word and s is the size of the alphabet, including special symbols and characters ($m = 25, s = 97$ for English). Let y^j be the index of the j^{th} character of the word $\mathbf{y}$; $y^j \in \{1, \ldots, s\}$. The negative log likelihood for $\mathbf{y}$ is defined as:

$$l(\mathbf{y}, \mathbf{v}) = -\sum_{j=1}^{len(\mathbf{y})} \log(\mathbf{P}[y^j, j]), \tag{1}$$

where $len(\mathbf{y})$ is the length of word $\mathbf{y}$, and $\mathbf{P}[y^j, j]$ denotes the entry at row y^j and column j of matrix $\mathbf{P}$.

The second loss is defined based on the negative log likelihood of candidate words and their edit distances to the

ground truth word. We first convert the list of negative log likelihood values into a probability distribution:

$$L_i = \frac{\exp(-l(\mathbf{y}_i, \mathbf{v}))}{\sum_{j=1}^{k} \exp(-l(\mathbf{y}_j, \mathbf{v}))}. \quad (2)$$

We also first convert the list of edit distances to a probability distribution:

$$D_i = \frac{\exp\left(-\frac{d(\mathbf{y}_i, \mathbf{y})}{T}\right)}{\sum_{j=1}^{k} \exp\left(-\frac{d(\mathbf{y}_j, \mathbf{y})}{T}\right)}, \quad (3)$$

Finally we compute the KL-divergence between two probability distributions D and L:

$$KL(D||L) \propto -\sum_{i=1}^{k} D_i \log(L_i). \quad (4)$$

In Eq. (3), T is a tunable temperature parameter. T is a positive value that should neither be too large nor too small. When T is too small, the target probability distribution D has low entropy, and none of the candidate words, except the ground truth, would matter. When T is too big, the target probability distribution D has high entropy, and there is no contrast between good and bad candidates. In our experiments, T is set to 0.3.

The loss in Eq. (4) is formulated to maximize the likelihood of the candidates that are close to the ground truth, while minimizing the likelihood of the faraway candidates. We call this loss as the contrastive loss because its goal is to contrast the ones that are closer to ground truth with the ones further away.

The total training loss of our recognition network is:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = l(\mathbf{y}, \mathbf{v}) + \lambda KL(D||L), \quad (5)$$

where λ is a hyper parameter that balances these two loss terms. We simply set $\lambda = 1$ in our experiments.

3.4. Network architecture and details

In the detection stage, we use the Bezier detection and alignment module from ABCNet [19]. The output of the detection stage is the input to the recognition stage, and it is a 3D feature tensor of size $n \times 32 \times 256$, with n being the number of detected text instances. Each text instance is represented by a feature map of size 32×256, and we use a sequential decoding network with attention to output a probability matrix $\mathbf{P}$ of size $s \times 25$, where s is the size of the extended alphabet, including letters, numbers, and special characters. In our experiments, $s = 97$ for English and $s = 106$ for Vietnamese. Each column i of $\mathbf{P}$ is a probability distribution of the i^{th} character in the text instance. During inference, we use this matrix to produce the initial recognition output, which is the sequence of characters where each character is the one with highest probability in each respective column of $\mathbf{P}$.

Given a word, either the ground truth word or the initially recognized one, we need to find the list of candidate words that have smallest edit distances to the given word. This can be done based on exact search or approximate nearest neighbor retrieval. The former approach requires exhaustively computing the edit distance between the given word and all dictionary words. It generates a better list of candidates and leads to higher accuracy, but it also takes longer time. The latter approach is more efficient, but it only returns approximate nearest neighbors. We experiment with both approaches in this paper. For the second approach, we use the `dict-trie` library to retrieve all words that have the Levenshtein distance to query word smaller than three. If the number of candidate words is smaller than ten, we fill the missing candidates by `###`. We notice that the query time will increase significantly if we use a larger distance threshold for `dict-trie`. Approximate search can reduce the query time, but it also decreases the final accuracy slightly.

In our experiments, we used Adam optimizer [13] for training. The parameter λ in Eq. (5) was set to 1.0, and the temperature parameter T of Eq. (3) was set to 0.3.

4. VinText: a dataset for Vietnamese scene text

In this section, we will describe our dataset for **Viet**namese scene text, named VinText. This dataset contains 2,000 fully annotated images with 56,084 text instances. Each text instance is delineated by a quadrilateral bounding box and associated with the ground truth sequence of characters. We randomly split the dataset into three subsets for training (1,200 images), validation (300 images), and testing (500 images). This is the largest dataset for Vietnamese scene text.

Although this dataset is specific to Vietnamese, we believe it will greatly contribute to the advancement of research in scene text detection and recognition in general. First, this dataset contains images from a developing country, and it complements the existing datasets of images taken in developed countries. Second, images from our dataset are very challenging, containing busy and chaotic scenes with many shop signs, billboards, and propaganda panels. As such, this dataset will serve as a challenging benchmark for measuring the applicability and robustness of scene text detection and recognition algorithms.

In the rest of this section, we will describe how the images were collected to ensure the dataset covers a diverse set of scene text and backgrounds. We will also describe the annotation and quality control process.

4.1. Image collection

The images from our dataset were either downloaded from the Internet or captured by some data collection workers. Our objective was to compile a collection of images that represent the diverse set of scene texts that are encoun-

Figure 2: **Some representative images from VinText dataset**. This is a challenging data, containing busy and chaotic scenes with scene text instances of various types, appearance, sizes, and orientations. Each text instance is annotated with a quadrilateral bounding box and and world-level transcription. This dataset will be a good benchmark for measuring the applicability and robustness of scene text spotting algorithms.

tered in everyday life in Vietnam. To ensure the diversity of our dataset, we first created a list of scene categories and sub-categories. The list of categories at the first level are: *shop signs*, *notice boards*, *bulletins*, *banners*, *flyers*, *street walls*, *vehicles*, and *miscellaneous items*. These categories were divided into subcategories, and many subcategories were further divided into sub-subcategories. For example, the the first-level category "Miscellaneous Items" contains many subcategories, including *book covers*, *product labels*, *clothes*. Images from these categories and sub-categories were abundant on the Internet, but there were also many more irrelevant and unsuitable images. As a result, for some categories, it was easier to capture the images ourselves rather than wasting time filtering out irrelevant search results. Thus, for a part of our dataset, we hired 20 data collection workers to capture images of the scene texts that they encountered while shopping or walking on the streets, using their own phone or hand-held cameras. To ensure no duplicates, we used p-hashing to find and remove duplicates, and we also visually inspected every image of our dataset. Our final dataset contains 764 images from the Internet and 1236 images captured by the data collection workers.

4.2. Image annotation

We divided the 2000 images into 10 batches, each with 200 images. Each image was annotated by two annotation workers independently. If the correlation between their annotations was smaller than 98%, we would ask them to cross-check each other and resolve the differences. As a final step, we manually chose the annotation from one annotator and visually inspected the annotation to ensure it satisfied our quality requirements.

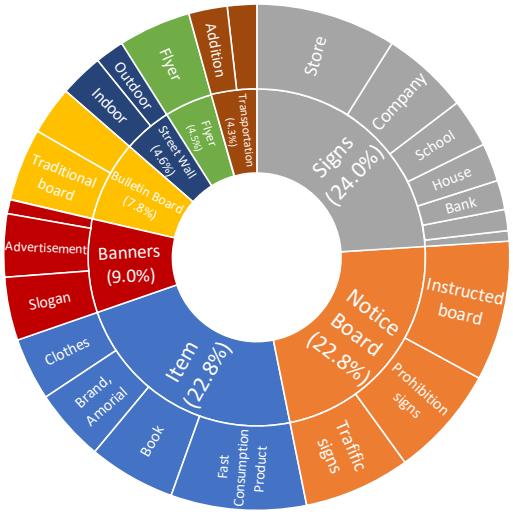


Figure 3: **Scene categories of the VinText dataset**, and the proportion for each category. This is a diverse dataset with many categories and sub-categories.

5. Experiments

In this section, we report the performance of our method on several datasets: TotalText, ICDAR2013, ICDAR2015, and the newly collected VinText dataset. We measure the performance of the entire detection-recognition system. An annotated text instance is considered correctly recognized only if it is detected correctly and the predicted word is the same with the annotated word. A detected output of the system is considered correct only when it corresponds to a correctly detected and recognized text instance. Following pre-

vious works in scene text recognition, we use H-mean score as the performance metric, which is the harmonic mean of the precision and recall values, also known as the F1 score.

The proposed method for using the dictionary during training and inference is general, but our specific implementation in this paper is based either on the ABCNet framework [19] or the MaskTextSpotterV3 [16], so we will refer to our method as either ABCNet+D or MaskTextSpotterV3+D for brevity.

5.1. Experiments on TotalText

TotalText [3] is a comprehensive scene text dataset. It consists of text instances in various orientations, including horizontal, vertical, and curved. The dataset contains 1255 training images and 300 test images with 11459 annotated text instances, covering in-the-wild scenes. All images were annotated with polygons and word-level transcriptions.

Table 1 compares the performance of the proposed approach ABCNet+D with several recently proposed methods. We consider two scenarios, depending on whether a dictionary is used at the inference time or not. This dictionary is the combination of the Oxford VGG dictionary [8] (> 90K words) with all the words in the test set, and this setting corresponds truthfully to the *Full Lexicon* setting reported in the existing scene text recognition literature. ABCNet^{pub} indicates the published result in the ABCNet paper [19]. ABCNet is the released model on github (`https://github.com/aim-uofa/AdelaiDet`), which is also the base of the proposed ABCNet+D method. As can be seen, ABCNet+D yields significant improvement over its direct baseline ABCNet. When the dictionary is not used at the inference stage, the key difference between ABCNet+D and ABCNet is the language-based contrastive loss, so the 1.2 performance gap between the two methods indicates the benefits of this contrastive loss. When the dictionary is also used at the inference time, the performance gap becomes wider, and this demonstrates the benefits of the novel way for using the dictionary at the inference time. These benefits can also be observed when comparing ABCNet+D and Mask TextSpotter v3. ABCNet+D is ranked lower than Mask TextSpotter v3 when the dictionary is not used, but their order is swapped when the dictionary is used.

Table 2 shows how the performance of ABCNet+D varies as the number of candidate words considered during the inference stage increases. As can be seen, the recognition accuracy increases as more candidate words are examined. This demonstrates the usefulness of correlating the candidates back to the visual features for deciding the final output. Using the edit distance alone is not adequate since the correct word might not be the one with the closest edit distance to the intermediate output. In fact, as can be seen from Table 2, the correct word might not even be among the top 70 candidates; increasing from 70 to 300 candidates still provides some accuracy gain. Furthermore,

	Dictionary used?	
Method	No	Yes
TextDragon [4]	48.8	74.8
Boundary TextSpotter [29]	65.0	76.1
CharNet [31]	63.6	-
Mask TextSpotter v2 [15]	65.3	77.4
Mask TextSpotter v3 [16]	**71.2**	78.4
ABCNet^{pub} (reported in [19])	64.2	75.7
ABCNet [19] (github checkpoint)	67.1	76.0
ABCNet+D (proposed)	68.3	**78.5**

Table 1: **Scene text recognition results on Total-Text**. The values in the table are the H-mean scores. We consider both scenarios where a large dictionary (> 90K words) is used or not used during inference. ABCNet+D yields significant improvement over its direct baseline ABCNet. ABCNet+D is not as good as MaskTextSpotterV3 when the dictionary is not used, but it is better when the dictionary is used. This proves the effectiveness of our proposed approach for incorporating the dictionary in the inference phase.

# candidates	1	10	20	30	70	300	600
H-mean	76.1	77.9	78.1	78.2	78.4	78.5	78.5

Table 2: **Recognition accuracy of ABCNet+D on Total-Text as the number of candidates varies**. The accuracy increases as more candidates are evaluated during the inference time, but saturates after 300 candidates have been considered. The second column, when the number of candidate is one, corresponds to the naive way of using the dictionary word with the smallest edit distance.

being able to select the correct word from a large set of 300 candidates means the visual-language compatibility scoring model works reasonably well. The performance saturates after 300 candidates have been considered.

As discussed above, there are strong empirical evidence for the benefits of using the contrastive loss during training. Even without using the dictionary during the inference time, the model trained with the contrastive loss makes fewer recognition mistakes than the model trained without. Fig. 4 displays some qualitative results of the two models.

5.2. Experiments on ICDAR13 and ICDAR15

ICDAR13 [10] is a scene text dataset that focused around the text content of interest. This dataset contains 462 images (229 for training and 233 for testing), together with rectangular bounding boxes and world-level transcription. ICDAR2015 is an incidental scene text dataset, consisting of 1000 training and 500 test images respectively. The images were taken by Google Glasses, and most of them are at

(a) ABCNet: CHIBESE
ABCNet+D: CHEESE

(b) ABCNet: FARN
ABCNet+D: FARM

(c) ABCNet: FAEAD
ABCNet+D: HEAD

(d) ABCNet: HOOME
ABCNEt+D: HOME

(e) ABCNet: KITGHEN
ABCNet+D: KITCHEN

(f) ABCNet: TOSTWORLD
ABCNet+D: LOSTWOLRD

(g) ABCNet: LOUUIE
ABCNet+D: LOUIE

(h) ABCNet: PLAMET
ABCNet+D: PLANET

Figure 4: **Several cases where ABCNet makes mistakes-but ABCNet+D does not.** These are intermediate outputs, when a dictionary has not been used for post processing.

low resolution with blurry and small text. ICDAR2015 focuses on English, and it comes with quadrilateral bounding box annotation and word-level transcriptions.

The results of ABCNet on these datasets are not reported in the ABCNet paper [19], and there is no released model for ICDAR2015. So we train an ABCNet model on these datasets ourselves. The results of ABCNet and ABCNet+D are reported in Table 3. On ICDAR2015, ABCNet performs relatively poor, possibly due to the low quality of Google glasses images with many small and blurry text. In this case, we find that the use of the dictionary boost the performance of the model immensely.

Table 4 compares the performance of MaskTextSpotterV3 with MaskTextSpotterV3+D on the ICDAR15, for different ways of using different types of dictionary during testing. As can be seen, MaskTextSpotterV3+D outperforms MaskTextSpotterV3 for all settings.

		Dictionary used?	
Dataset	Method	No	Yes
ICDAR13	ABCNet [19]	83.5	85.6
	ABCNet+D (proposed)	**84.6**	**87.5**
ICDAR15	ABCNet [19]	36.5	47.6
	ABCNet+D (proposed)	**57.9**	**67.2**

Table 3: **Comparing ABCNet with ABCNet+D on the ICDAR13 and ICDAR15 datasets**. We consider two scenarios when a large dictionary (about 90K words) is used or not during testing. On ICDAR15, ABCNet performs relatively poor, possibly due to the low quality of Google glasses images with many small and blurry text. ABCNet+D outperforms its direct baseline ABCNet by a wide margin.

	Dictionary type		
	Strong	Weak	General
MaskTextspotterV3	83.3	78.1	74.2
MaskTextspotterV3+D (proposed)	**85.2**	**81.9**	**75.9**

Table 4: **H-mean scores on ICDAR15, comparing MaskTextSpotterV3 with MaskTextSpotterV3+D**, the proposed method trained with a general dictionary of ∼90K words. In testing, one can consider different types of dictionary Strong/Weak/General, which corresponds to the standard evaluation protocols for ICDAR15: Strongly/Weakly/Generic Contextualised.

5.3. Experiments on VinText

The Vietnamese script is based on the Latin alphabet like English, but it additionally has seven derivative characters (đ, ô, ê, â, ă, ơ, ư) and five accent symbols (◌́, ◌̀, ◌̉, ◌̣, ◌̃). A derivative character can also be combined with an accent symbol; for example, ế is popular Vietnamese word, combining the letter e with both the circumflex and the acute symbols. It is unclear how to handle these extra symbols, and we consider here two approaches. The first approach is to create a new alphabet symbol for each valid combination of letter and accent symbols. For example, ế would be a character of the alphabet by itself, and also for ề, ệ, ễ, ể, é, è, ẹ, ẽ, e. The second approach is to break a derivative character into two parts: the English character and either the hat ◌̂, the breve ◌̆, or the horn plus one of the accent symbol. Thus, the word ế would be the sequence of three symbols: (e, ◌̂, ◌́). The first approach requires extending the English alphabet of 97 characters to an alphabet with a total of 258 characters, while the second approach only requires only eight additional symbols, leading to a total of 97+9=106

Figure 5: **Detection and recognition results by ABCNet+D**: on TotalText, ICDAR13, ICDAR15, and VinText

characters. Furthermore, one advantage of the second approach is that we can utilize annotated training data more effectively. For example, an annotated instance of ế also gives us one annotated instance of e, while an annotated instance of e would also be an annotated instance for a part of ế. However, the disadvantages of the second approach are: (1) not all symbol combinations are valid, and (2) multiple sequential combinations lead to the same character, and we loose the uniqueness of the sequential order of characters in a word. Given these advantages and disadvantages, it is unclear which of the above two approaches would work best in practice, so we benchmark both of them in this experiment.

We train ABCNet and ABCNet+D on VinText, starting from a English pre-trained ABCNet model. To handle the extra characters of the extended alphabet, we create additional character recognition heads, and replicate the weights of some existing recognition heads to the new ones based on the visual similarity of the characters. For example, we initialize the recognition heads of ế and ê with the recognition head of the character e.

Table 5 shows the recognition accuracy of several methods on the VinText dataset. As can be seen, the second approach of extending the English alphabet works better than the first one. Comparing ABCNet and ABCNet+D, we see clear benefits for using the proposed approach to incorporate the dictionary in both the training and testing stages.

Fig. 5 shows some representative detection and recognition results on VinText and also on TotalText and ICDAR15. As can been, VinText contains more text instances, and is more challenging than the other three datasets. ABCNet+D works well even on these challenging images.

	Dictionary used?	
Method	No	Yes
Alphabet: English + derivative characters		
Mask TextSpotter v3 [16]	53.4	57.4
ABCNet [19]	50.6	55.1
Alphabet: English + accent symbols		
ABCNet	54.2	58.5
ABCNet+D	57.4	63.6
Mask Textspotterv3+D	**68.5**	**70.3**

Table 5: **Recognition accuracy (H-mean) on the VinText dataset.** We consider both scenarios when a large dictionary is used or not during testing. This dictionary is the combination of RDRSegmenter [23] and the words in the test set. We also consider two different approaches of extending the English alphabets on two backbones: ABCNet and Mask Textspotterv3.

6. Conclusions

We have proposed a novel language-aware approach to tackle the visual ambiguity in scene text recognition. Our approach can harness the power of a dictionary in both the training and inference stages. Our approach can resolve ambiguity in many conditions, by considering multiple suggestions from a dictionary given an intermediate recognition output. In addition, we propose VinText, a new dataset for Vietnamese scene text recognition which brings new challenges in discriminating a character from multiple similar ones. Experiment results on TotalText, ICDAR13, ICDAR15, and the newly collected VinText dataset demonstrate the benefits of our dictionary incorporation approach.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015. 3

[2] A. Bissacco, M. Cummins, Yuval Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. *Proceedings of the International Conference on Computer Vision*, 2013. 2

[3] Chee Kheng Chng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 2, 6

[4] Wei Feng, Wenhao He, Fei Yin, Xu-Yao Zhang, and Cheng-Lin Liu. Textdragon: An end-to-end framework for arbitrary shaped text spotting. In *Proceedings of the International Conference on Computer Vision*, 2019. 2, 6

[5] Yunze Gao, Y. Chen, J. Wang, M. Tang, and H. Lu. Reading scene text with fully convolutional sequence modeling. *Neurocomputing*, 2019. 2

[6] Wen-Yang Hu, Xiaocong Cai, J. Hou, Shuai Yi, and Zhiping Lin. Gtc: Guided training of ctc towards efficient and accurate scene text recognition. *ArXiv*, 2020. 2

[7] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *NIPS Workshop on Deep Learning*, 2014. 2

[8] 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. 6

[9] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. In *Proceedings of the European Conference on Computer Vision*, 2014. 2

[10] Dimosthenis Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. I. Bigorda, Sergi Robles Mestre, J. M. Romeu, D. F. Mota, Jon Almazán, and L. D. L. Heras. Icdar 2013 robust reading competition. *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493, 2013. 2, 6

[11] Dimosthenis Karatzas, L. G. I. Bigorda, A. Nicolaou, S. Ghosh, Andrew D. Bagdanov, M. Iwamura, Jiri Matas, Lukas Neumann, V. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and Ernest Valveny. ICDAR2015 competition on robust reading. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2015. 2

[12] Yoon Kim, Yacine Jernite, D. Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2016. 2

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 4

[14] V. I. Levenshtein. Binary codes capable of correcting insertions and reversals. 1966. 1, 2, 3

[15] Minghui Liao, Pengyuan Lyu, Minghang He, Cong Yao, Wenhao Wu, and Xiang Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2, 6

[16] Minghui Liao, Guan Pang, J. Huang, Tal Hassner, and X. Bai. Mask textspotter v3: Segmentation proposal network for robust scene text spotting. *ArXiv*, 2020. 2, 6, 8

[17] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, and Xiang Bai. Mask textspotter v3: Segmentation proposal network for robust scene text spotting. In *Proceedings of the European Conference on Computer Vision*, 2020. 2

[18] Ron Litman, Oron Anschel, Shahar Tsiper, R. Litman, Shai Mazor, and R. Manmatha. Scatter: Selective context attentional scene text recognizer. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[19] Y. Liu, Hao Chen, Chunhua Shen, Tong He, Lian-Wen Jin, and L. Wang. ABCNet: Real-time scene text spotting with adaptive bezier-curve network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 3, 4, 6, 7, 8

[20] Pengyuan Lyu, Minghui Liao, Cong Yao, Wenhao Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *ArXiv*, 2018. 2

[21] N. Nayef, F. Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, U. Pal, Christophe Rigaud, J. Chazalon, Wafa Khlif, M. Luqman, J. Burie, C. Liu, and J. Ogier. ICDAR2017 robust reading challenge on multi-lingual scene text detection and script identification - rrc-mlt. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2017. 2

[22] N. Nayef, Yash Patel, M. Busta, P. Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, U. Pal, J. Burie, Cheng-Lin Liu, and J. Ogier. ICDAR2019 robust reading challenge on multi-lingual scene text detection and recognition — rrc-mlt-2019. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2019. 2

[23] Dat Quoc Nguyen, Dai Quoc Nguyen, Thanh Vu, Mark Dras, and Mark Johnson. A Fast and Accurate Vietnamese Word Segmenter. In *Proceedings of International Conference on Language Resources and Evaluation*, 2018. 8

[24] Liang Qiao, Sanli Tang, Zhanzhan Cheng, Yunlu Xu, Yi Niu, Shiliang Pu, and Fei Wu. Text perceptron: Towards end-to-end arbitrary-shaped text spotting. *arXiv preprint arXiv:2002.06820*, 2020. 2

[25] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 2

[26] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning internal representations by error propagation. 1986. 2

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 2014. 2

[28] Zhaoyi Wan, Fengming Xie, Y. Liu, X. Bai, and Cong Yao. 2d-ctc for scene text recognition. *ArXiv*, 2019. 2

[29] Hao Wang, Pu Lu, Hui Zhang, Mingkun Yang, X. Bai, Yongchao Xu, Mengchao He, Yongpan Wang, and W. Liu. All you need is boundary: Toward arbitrary-shaped text spotting. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2020. 2, 6

[30] T. Wang, Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen, Y. Wu, Qianying Wang, and Mingxiang Cai. Decoupled attention network for text recognition. *ArXiv*, 2020. 2

[31] Linjie Xing, Zhi Tian, Weilin Huang, and Matthew R Scott. Convolutional character networks. In *Proceedings of the International Conference on Computer Vision*, 2019. 2, 6

[32] X. Yang, Kaihua Tang, Hanwang Zhang, and J. Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[33] Liu Yuliang, Jin Lianwen, Zhang Shuaitao, and Zhang Sheng. Detecting curve text in the wild: New dataset and new solution. *arXiv preprint arXiv:1712.02170*, 2017. 2

[34] Ling-Qun Zuo, Hong-Mei Sun, Qi-Chao Mao, Rong Qi, and R. Jia. Natural scene text recognition based on encoder-decoder framework. *IEEE Access*, 2019. 2