What If We Only Use Real Datasets for Scene Text Recognition? Toward Scene Text Recognition With Fewer Labels – Supplementary Material –

Jeonghun Baek Yusuke Matsui Kiyoharu Aizawa
The University of Tokyo
{baek, matsui, aizawa}@hal.t.u-tokyo.ac.jp

A. Contents

Supplement B : **Needs for STR with Fewer Real Labels**

- We complement the needs for STR with fewer real labels by illustrating the detailed examples.

Supplement C : **STR Datasets - Details and More Examples**

- We describe the detail of preprocessing in §3 and show more examples of public real data.

Supplement D : **STR With Fewer Labels - Details**

- We describe the details of STR models and semi- and self-supervised methods in §4.

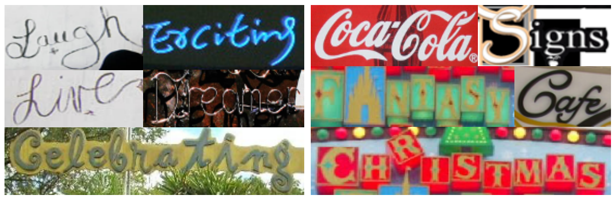
Supplement E : **Experiment and Analysis - Details**

- We provide the details and comprehensive results of our experiments in §5.

B. Needs for STR with Fewer Real Labels

In this section, we complement the necessity of training STR models only on fewer real labels (STR with fewer labels). The study of STR with fewer labels generally *aims to exploit a few real labels efficiently*. We describe the detailed examples of two needs as mentioned in §1, and introduce an additional need based on our experimental results.

Need #1: To recognize handwritten or artistic data in public real datasets Figure 1 shows the handwritten or artistic data in public real datasets. It is difficult to generate them with the current synthetic engine [9, 6]. If we have appropriate handwritten fonts, we can generate similar texts with handwritten texts synthetically and cover some of the handwritten texts. However, because the number of fonts (about thousands) is quite lower than the number of people, there can still be uncovered handwritten texts by handwritten fonts. Furthermore, generating artistic texts with handwritten fonts is difficult: artistic texts such as text logo and calligraphy as shown in Figure 1.


(a) Examples from benchmark datasets for evaluation.

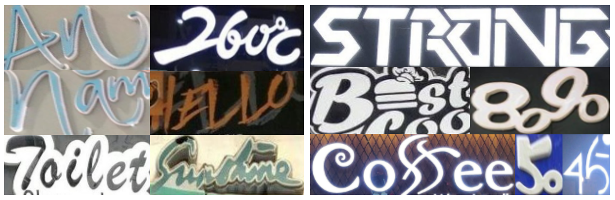

(b) Examples from real datasets other than benchmark datasets.

Figure 1: Handwritten or artistic texts in public real data.

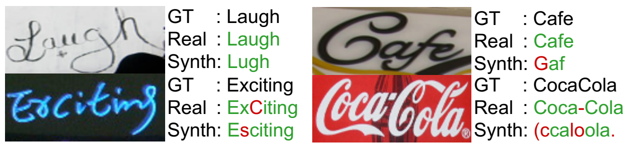

GT : Laugh
Real : Laugh
Synth: Lugh
GT : Exciting
Real : ExCiting
Synth: Esciting
GT : Cafe
Real : Cafe
Synth: Gaf
GT : CocaCola
Real : Coca-Cola
Synth: (ccaloola.

Figure 2: Predictions on handwritten or artistic texts. GT, Real, and Synth denote the ground truth, the prediction of TRBA-Baseline-real, and TRBA-Baseline-synth, respectively.

In this case, exploiting the few real data of them can be more efficient rather than generating synthetic data close to them. Namely, we need STR with fewer labels in this case. Figure 2 shows the predictions by trained models (TRBA-Baseline-real and -synth) in our experiments. The model trained with real data (Real) has better performance than that with synthetic data (Synth). These results show that exploiting real data can be useful for these types of texts.

Need #2: To recognize language other than English (LOTE) without synthetic data In the other case, when we have to recognize LOTE, there are not always synthetic

Method	Model: CRNN COCO	RCTW	Uber	ArT	LSVT	MLT19	ReCTS	Avg.	Model: TRBA COCO	RCTW	Uber	ArT	LSVT	MLT19	ReCTS	Avg.
Baseline-synth	37.4	50.3	32.1	48.1	50.3	74.6	73.6	52.3	50.2	59.1	36.7	57.6	58.0	80.3	80.6	60.4
Baseline-real	46.3	54.3	45.8	48.1	58.6	78.2	74.7	58.0	62.7	67.7	52.7	63.2	68.7	85.8	83.4	69.2
PR	56.6	61.1	44.8	58.7	62.2	82.5	80.9	63.8	66.9	71.5	54.2	66.7	73.5	87.8	85.6	72.3

Table 1: Accuracy of seven evaluation datasets: COCO, RCTW, Uber, ArT, LSVT, MLT19, and ReCTS. Avg. denotes the averaged accuracies of seven datasets. The number of evaluation sets of each datasets is described in Table 2.

data for LOTE. For such a case, we should recognize LOTE without synthetic data or generate synthetic data.

However, generating appropriate synthetic data for LOTE is difficult for those who do not know target languages. When we generate synthetic data, we should prepare at least three elements: 1) Word vocabulary (word corpus or lexicon). 2) Characters that compose words. 3) Font to render words. However, for people who do not know LOTE, preparing three elements for LOTE is difficult; difficult to decide appropriate words, characters, and fonts. In addition, some languages have specific rules to generate their languages. For example, Arabic texts have two different features from English: 1) They are written from right to left. 2) Some of the Arabic characters change their shape depending on their surrounding characters. These factors make generating synthetic data of Arabic difficult for those who do not know Arabic.

In this case, when we have to recognize LOTE but generating synthetic data for LOTE is difficult, we need to achieve competitive performance with a few real labels.

Need #3: Current public synthetic data can be inappropriate to datasets other than STR benchmark datasets In addition to evaluation on benchmark datasets in §5.2, we also evaluate the other seven datasets: COCO, RCTW, Uber, ArT, LSVT, MLT19, and ReCTS. Table 1 shows the results. Baseline-real has higher accuracy than Baseline-synth (Avg. 58.0% vs. 52.3% for CRNN and 69.2% vs. 60.4% for TRBA). Our best setting PR successfully improve Baseline-real (Avg. +5.8% for CRNN and +3.1% for TRBA). These results indicate that *fewer real data (Real-L) can be more appropriate to evaluation sets of these seven datasets, rather than current synthetic data (MJ+ST)*. We presume that because some of the fewer real data (Real-L) derive from the same domain with evaluation sets of seven datasets, Baseline-real has higher accuracy. Namely, using only fewer real data collected from the target domain can be more significant than using large synthetic data from the other domain. This indicates that studies on STR with fewer labels are necessary, fully exploiting few real data derived from the target domain.

Detailed survey of the literatures of STR with fewer real labels As mentioned in §1, after emerging large synthetic data [9](2014), the study of STR with fewer labels has decreased. Thus, there are only a few studies on STR with fewer labels for five years. Searching for studies on STR with fewer labels is difficult. In this work, we struggle to search for studies on STR with fewer labels via the widely-used search engine: Arxiv Sanity[1].

Specifically, we search studies with query "text recognition" from 2016 to 2020. The number of searched papers is 131 papers. We manually check the contents of all of them, and find four papers related to STR with fewer labels; they only use real data rather than synthetic data. Three of them are published three or four years ago, and do not use deep learning based methods; Mishra *et al.* [23](2016) uses the conditional random field model and support vector machine. Lou *et al.* [22](2016) uses the generative shape model and support vector machine. Bhunia *et al.* [2](2017) uses the hidden Markov model and support vector machine. However, it is difficult to compare them with recent state-of-the-art methods. Because their evaluation data are different from the recent STR benchmark: they all do not evaluate on IC15, SP, and CT. In addition, they use lexicons for calculating accuracy. In general, the accuracy calculated with the lexicon has higher accuracy than the one calculated without the lexicon. Thus, comparing them is unfair. Recent state-of-the-art methods present both accuracies calculated with lexicons and calculated without lexicons [29, 31, 42, 14, 39, 35] or do not present the accuracy calculated with lexicons [1, 36, 28, 40, 24]. Therefore, the former can be compared with the three methods [23, 22, 2] but the latter cannot. The process of using lexicon to calculate accuracy is as follows: 1) IIIT, SVT, and SP have predefined lexicons which contain a target word and some candidate words. 2) Transform the output word from the model into the closet word in predefined lexicons. 3) Calculate accuracy between transformed one and the target word.

Another paper uses deep learning based methods [43](2019). The paper addresses recognizing English and Chinese texts. However, strictly speaking, the paper describes scene text spotting (STS), which is the combination of scene text detection (STD) and STR, rather than sole STR. Because they do not describe the result of STR in their paper, it is difficult to compare their method with other STR methods. From the results of STS, we can

[1]http://www.arxiv-sanity.com/

presume that their detector has great performance but their recognizer has a less competitive performance to other English STR methods. Because they won 1st for STS in the ArT competition (English only), but they recorded 15th of 23 teams for STR in the ArT competition (English only)[2]. However, this is not a concrete comparison but only our presumption. We need the results on STR benchmark datasets rather than the results of STS to compare their method with other STR methods.

It is currently difficult to search for studies on STR with fewer labels and compare those studies with state-of-the-art methods. We hope that our study becomes the appropriate study on STR with fewer labels that can be easily searched and compared.

C. STR Datasets - Details and More Examples

In this section, we describe the details of our preprocessing process in §3. Furthermore, we illustrate more examples of real data.

C.1. Preprocessing Real Datasets

Consolidating real datasets We collect SVT and IIIT from their web page[3,4]. However, the labels of training set of SVT and IIIT are case-insensitive; they are all uppercase alphabet. We use case sensitive data of SVT and IIIT corrected by [21][5]. We collect IC13, IC15, COCO, ArT, LSVT, MLT19, and ReCTS from the web page of ICDAR competition[6]. We download RCTW and Uber from their web page[7,8].

Excluding duplication between datasets If we do not pay attention the duplication between datasets, we might use the training set that includes some of the evaluation set. To avoid this duplication, we investigate the duplication between all training set and all evaluation set. Specifically, we investigate whether the scene images of training sets match the scene images of evaluation sets. As a result, we find that the training set of Art and the evaluation set of CT have 27 duplicated scene images and 122 duplicated word boxes, as shown in Figure 3. We exclude them for a fair comparison. The journal version paper of Total-Text [4] indicates that some of the word boxes in Total-Text are duplicated in CT. ArT includes Total-Text, and therefore some of the word boxes in ArT can be also duplicated in CT. However, we do not know how many Total-Text images are in the training

[2]https://rrc.cvc.uab.es/?ch=14&com=evaluation&task=2

[3]http://vision.ucsd.edu/ kai/svt/

[4]https://cvit.iiit.ac.in/research/projects/cvit-projects/the-iiit-5k-word-dataset

[5]https://github.com/Jyouhou/Case-Sensitive-Scene-Text-Recognition-Datasets

[6]https://rrc.cvc.uab.es/

[7]https://rctw.vlrlab.net/dataset

[8]https://s3-us-west-2.amazonaws.com/uber-common-public/ubertext/index.html

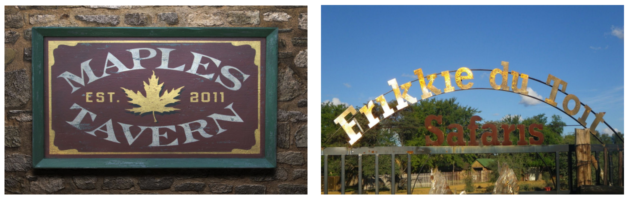
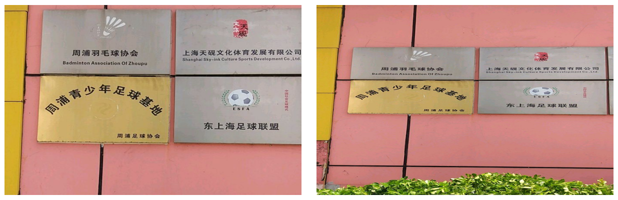


Figure 3: Duplicated scene images. These images are found in both the training set of ArT and the evaluation set of CT.



(a) Image from ArT (b) Image from LSVT

Figure 4: Similar scene images. These images are found in both training sets of ArT and LSVT. They are different images but contain the same texts; the text regions of ArT are usually more enlarged than LSVT.

set of ArT. Thus, we investigate the duplicated word boxes by comparing scene images of ArT and CT. For another example, according to [1], 215 word boxes are duplicated in the training set of IC03 and the evaluation set of IC13. This is one of the reasons why we exclude IC03; the other reason is that IC13 inherits most of IC03 data.

According to [3], ArT includes the subset of LSVT. We cannot find duplicated images by matching scene images because they are slightly different, as shown in Figure 4. In this case, we investigate duplication by matching labels of scene images. As a result, we find that the training set of Art and the training set of LSVT have 814 similar scene images (433 scene images for English) and 4,578 similar word boxes (861 word boxes for English).

However, since labels of scene images for the evaluation set of ArT and LSVT are not provided, we cannot find duplication by matching labels of scene images. Thus, it is difficult to exclude the duplication between the evaluation set of ArT and the training set of LSVT. If there are duplicated images between them, comparing the accuracy of ArT between a method that uses the training set of LSVT and another method that does not use the training set of LSVT can be unfair.

In the case of evaluation on LSVT, in our experiments, we split the training set of LSVT into training, validation, and evaluation set. We exclude the duplicated word boxes between the training set of ArT and the evaluation set (after splitting) of LSVT; exclude 74 word boxes in the evaluation set of LSVT. In our experiments, we would like to use more real data for training and validation; thus, we exclude the duplicated word boxes in the evaluation set of LSVT rather than exclude training and validation sets of ArT.

Collecting only English words In this study, we take only English and symbols. Specifically, we exclude Chinese characters in RCTW, ArT, LSVT, and ReCTS. For MLT, all word labels have "a script label" representing the language of each word label. We exclude the words whose script label is Arabic, Chinese, Japanese, Korean, Bangla, or Hindi.

Excluding don't care symbol Don't care symbol "*" or "#" is usually included in recent public real data: RCTW, Uber, ArT, LSVT, MLT19, and ReCTS. We do not conduct this filtering on SVT, IIIT, IC13, IC15, and COCO.

We do not exclude the texts contains "#" symbol. Instead, we only exclude the texts is "#", "##", "###", or "####". Therefore, we can train "#" symbol with other characters. In the case of MLT19, we only exclude the texts is "###" or "####".

In contrast to "#" symbol, we exclude the image whose label contains "*" symbol in Uber. About half of Uber images (149K of 285K training images) contain "*" symbol. Although some of them are sufficiently readable, they contain "*" symbol. Thus, we exclude them because they can be the noise of data.

Excluding vertical or $\pm$ 90 degree rotated texts As described in §3.3, we mainly focus on horizontal texts and thus exclude vertical texts. For the characters such as "1, i, j, l, t" and the words such as "it", their height usually are greater than their width. To avoid excluding these characters and words, we only exclude images *whose texts have more than two characters* and whose height is greater than the width.

Splitting each real dataset into training, validation, and evaluation sets Table 2 shows whether each dataset originally has training, validation, and evaluation sets. Some datasets do not have them, and thus we split the training set of the dataset. For example, RCTW, LSVT, and MLT19 only have a training set. Thus we split the training set of each dataset into training, validation, and evaluation sets with ratios of 80%, 10%, and 10%, respectively. The evaluation sets of RCTW, LSVT, and MLT19 have been released, but the evaluation sets only contain images and do not contain labels. Therefore, we cannot evaluate their evaluation sets, and thus RCTW, LSVT, and MLT19 are considered not to have the evaluation set.

In the other case, SVT, IIIT, IC13, IC15, ArT, and ReCTS do not have a validation set, and thus we split the training set of each dataset into training and validation sets with ratios of 90% and 10%, respectively.

COCO and Uber originally have training, validation, and evaluation sets. Thus we use their original training, validation, and evaluation sets. We do not split their training sets.

Table 2 shows the number of word boxes used in our experiments. The number is calculated as follows: 1) Conduct preprocessing on the training set of each dataset as described in §3.3 and §C.1. 2) Following the base code [1], exclude the images whose texts have more than 25 characters in our experiments.

	Check for inclusion			# of word boxes		
Dataset	Train.	Valid.	Eval.	Train.	Valid.	Eval.
Real labeled datasets (Real-L)						
SVT	✓	–	✓	231	25	647
IIIT	✓	–	✓	1,794	199	3,000
IC13	✓	–	✓	763	84	1,015
IC15	✓	–	✓	3,710	412	2,077
COCO	✓	✓	✓	39K	9,092	9,823
RCTW	✓	–	–	8,186	1,026	1,029
Uber	✓	✓	✓	92K	36K	80K
ArT	✓	–	✓	29K	3,202	35K
LSVT	✓	–	–	34K	4,184	4,133
MLT19	✓	–	–	46K	5,689	5,686
ReCTS	✓	–	✓	23K	2,531	2,592
Total	–	–	–	276K	63K	146K

Table 2: Number of word boxes used in our experiments, after splitting each real dataset into training, validation, and evaluation sets.

Detector for cropping texts in unlabeled scene images Unlabeled datasets described in §3.2 contain scene images. The scene images do not have labels indicating text region. Therefore, we use pretrained scene text detection (STD) model for cropping texts in the scene images. In this paper, we assume the case when we have to train STR models without synthetic data. In this case, we cannot train STD model with synthetic data, and thus we use STD model trained only on real data, called BDN [19]. BDN has two versions; 1) BDN published in IJCAI [20] use synthetic data (ST [6]) for training. 2) BDN used for ReCTS competition [19] do not use synthetic data. We use the second one for cropping texts in unlabeled scene images.

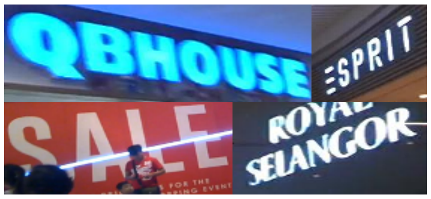
C.2. More Examples of Public Real Data

Recently, various types of texts have been accumulated. They can improve the robustness of STR models. We present more examples of public real data. Figure 5 shows the examples of benchmark datasets for evaluation, as described in §2.2. Figure 6, 7, and 8 are the extended versions of Figure 3 in §3. Figure 6 shows examples of accumulated real labeled data for Year 2011, 2013, and 2015. Figure 7, and 8 show examples of that of Year 2017 and 2019, respectively. Figure 9 shows examples of unlabeled scene images and word boxes after cropping.

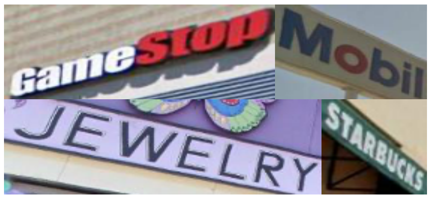
(a) SVT

(b) IIIT

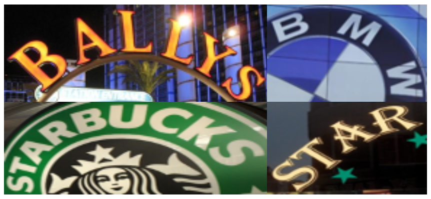
(c) IC13



(d) IC15

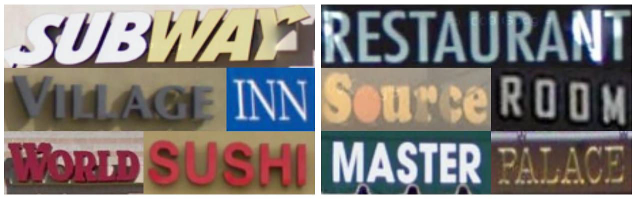


(e) SP



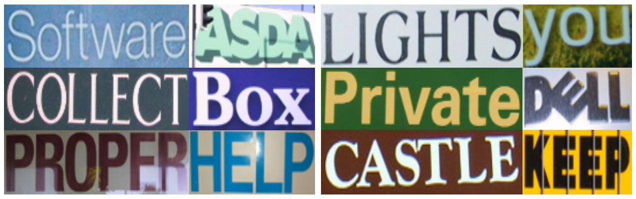
(f) CT

Figure 5: Examples of benchmark datasets for evaluation. SVT, IIIT, and IC13 are regarded as regular datasets. They contain many horizontal texts. IC15, SP, and CT are regarded as irregular dataets. They contain many perspective or curved texts.

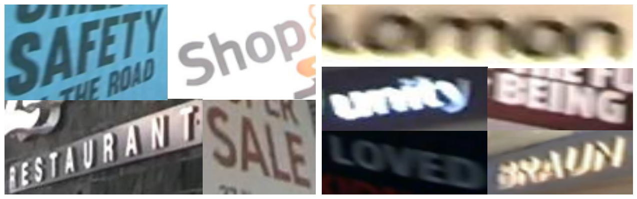


(a) Year 2011: SVT

(b) Year 2013: IIIT

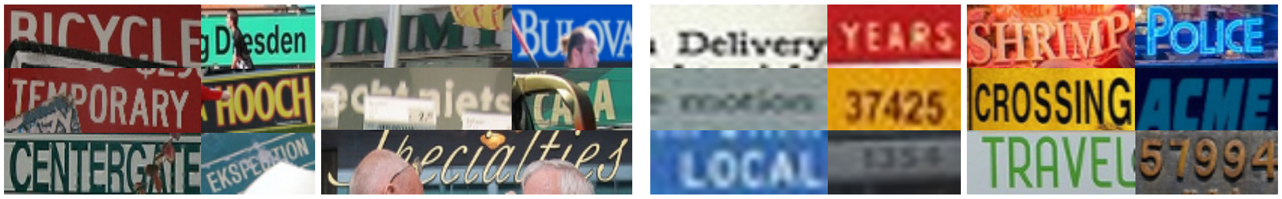


(c) Year 2013: IC13

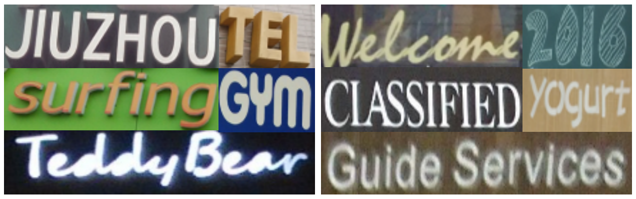


(d) Year 2015: IC15

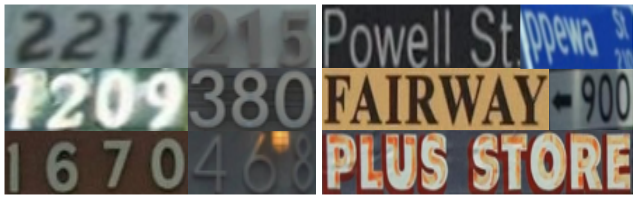
Figure 6: Examples of public real datasets for Year 2011 (SVT), Year 2013 (IIIT and IC13), and Year 2015 (IC15). SVT, IIIT, and IC13 contains many horizontal texts. IC15 contains many perspective or blurry texts.



(a) COCO contains many occluded or low-resolution texts.



(b) RCTW: Most images are collected in China.

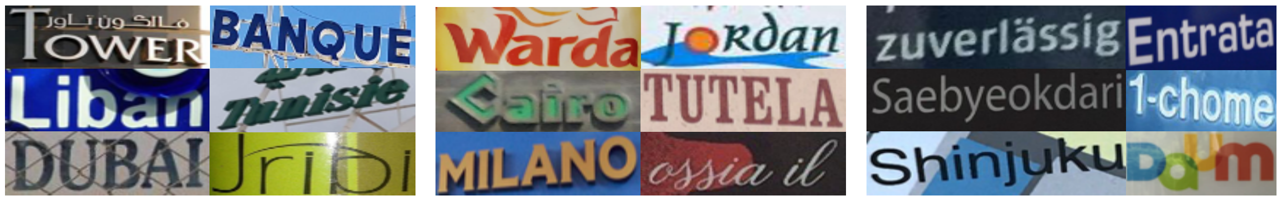


(c) Uber contains many house number or signboard texts.

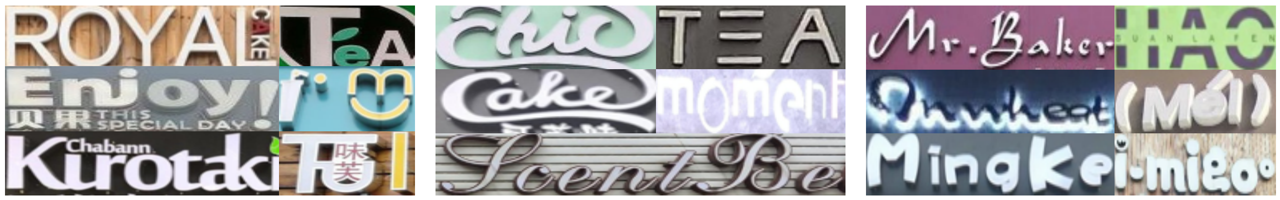
Figure 7: Examples of public real datasets for Year 2017: COCO, RCTW, and Uber.

(a) ArT contains many arbitrary-shaped texts: rotated, perspective or curved texts.

(b) LSVT is collected from streets in China.



(c) MLT19 is collected to recognize multi-lingual texts.



(d) ReCTS contains texts arranged in various layouts or texts written in difficult fonts.

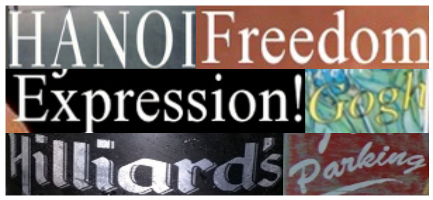
Figure 8: Examples of public real datasets for Year 2019: ArT, LSVT, MLT19, and ReCTS.

(a) Book32: scene images.

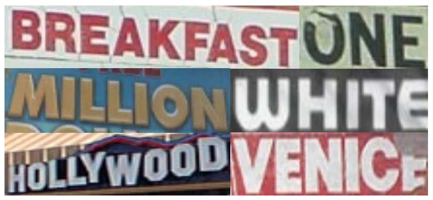
(b) TextVQA: scene images.

(c) ST-VQA: scene images.

(d) Book32: word boxes after cropping.



(e) TextVQA: word boxes after cropping.



(f) ST-VQA: word boxes after cropping.

Figure 9: Examples of unlabeled datasets: Book32, TextVQA, and ST-VQA. We show their scene images and word boxes after cropping.

D. STR With Fewer Labels - Details

We describe more details of STR models and semi- and self-supervised learning in §4.

D.1. Description of STR Models

We select two models in order to represent that our experimental results are not limited to a specific STR model. We adopt two widely-used STR models in STR benchmark repository [1][9]: CRNN [29] and TRBA [1].

CRNN, the abbreviation of Convolutional-Recurrent Neural Network, is the first model that combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for STR. CRNN is regarded as a basic and lightweight STR model. Thus, CRNN is usually selected as one of the baseline models. Also, CRNN is sometimes adopted to the STR part of scene text spotting that is the combination of scene text detection and recognition [16, 15, 18]. CRNN consists of 7 layers of VGG-like CNNs, 2 layers of BiLSTM, and CTC decoder.

TRBA, the abbreviation of TPS-ResNet-BiLSTM-Attention, has the best performance in STR benchmark repository [1]. TRBA is also usually selected as one of the baseline models [17, 37, 10, 27, 38]. TRBA consists of 4 layers of CNNs for TPS, 29 layers of ResNet, 2 layers of BiLSTM, and an attention decoder. According to [1], TRBA is created by composing existing modules used in prior works. Thus, TRBA can be regarded as a variant of RARE [30]. RARE is a well-known architecture that introduces an image transformation module into STR models. The main differences between TRBA and RARE are as follows: TRBA uses 1) ResNet instead of VGG-like CNNs. 2) BiLSTM instead of GRU. TRBA can be also regarded as a variant of another widely-used baseline model called ASTER [31] because ASTER is the successor of RARE. The main differences between TRBA and ASTER are as follows: 1) Details of TPS. 2) ASTER uses a bidirectional attention decoder.

D.2. Objective Function

We use labeled training set $\mathcal{D}_l = \{(x_1, y_1), ..., (x_n, y_n)\}$ and unlabeled training set $\mathcal{D}_u = \{u_1, ..., u_m\}$ to calculate object function, where x is the labeled training image, y is the word label, and u is the unlabeled training image.

Objective function for STR STR models are trained by minimizing the objective function as follows,

$$\mathcal{L}_{\text{recog}} = -\frac{1}{|\mathcal{D}_l|} \sum_{(x,y)\in\mathcal{D}_l} \log p(y|x) \tag{1}$$

where $p(y|x)$ is the conditional probability of word label.

[9]https://github.com/clovaai/deep-text-recognition-benchmark

Objective function for semi-supervised learning Pseudo-Label (PL) [12] uses pseudolabeled training set $\mathcal{D}_s = \{(u_1, s_1), ..., (u_m, s_m)\}$ for training, where s is the pseudolabel of u. The object function of PL calculated as follows,

$$\mathcal{L} = \mathcal{L}_{\text{recog}} - \frac{1}{|\mathcal{D}_s|} \sum_{(u,s)\in\mathcal{D}_s} \log p(s|u) \tag{2}$$

where $p(s|u)$ is the conditional probability of the pseudolabel.

Mean Teacher (MT) [34] uses the object function as follows,

$$\mathcal{L} = \mathcal{L}_{\text{recog}} + \alpha \frac{1}{|\mathcal{D}_l| + |\mathcal{D}_u|} \sum_{[x;u]} \text{MSE}(f_\theta([x;u]^\eta), f_{\theta'}([x;u]^{\eta'})) \tag{3}$$

where $[x;u]$ denotes the concatenation of x and u. η and η' denote two random augmentations. $[x;u]^\eta$ and $[x;u]^{\eta'}$ are the images augmented by η and η', respectively. MSE denotes the mean squared error, f_θ is the student model, $f_{\theta'}$ is the teacher model, and α is coefficient of MSE loss.

Objective function for self-supervised learning The pretext task of RotNet [5] is conducted by minimizing the objective function as follows,

$$\mathcal{L}_{\text{Rot}} = -\frac{1}{|R|}\frac{1}{|\mathcal{D}_u|} \sum_{r\in R} \sum_{u\in\mathcal{D}_u} \log p(r|u^r) \tag{4}$$

where R is the set of four rotation degrees $\{0°, 90°, 180°, 270°\}$, $p(r|u^r)$ is the conditional probability of rotation degree, and u^r is the unlabeled image u rotated by r. This objective function conducts 4-class classification.

Following the authors of MoCo [7], we calculate infoNCE [25]. For each mini-batch, we obtain queries $\boldsymbol{q}$ and keys $\boldsymbol{k}$ as described in §4.3. For each pair of a query q and a key k, we consider that the key is positive if they derive from the same image, otherwise negative. MoCo uses a dictionary that contains negative keys. The dictionary is a queue that enqueues keys from the current mini-batch and dequeues keys from the oldest mini-batch. The dictionary size K is generally much larger than the batch size. For each query q, we calculate the object function as follows,

$$\mathcal{L}_q = -\log \frac{\exp(q^\top k_{pos}/\tau)}{\sum_{i=0}^{K} \exp(q^\top k_i/\tau)} \tag{5}$$

where τ is the temperature value, k_{pos} is the positive key. This object function conducts (K+1)-way classification with the softmax function on K negative keys and 1 positive key.

E. Experiment and Analysis - Details

We show more details of our implementation and the extended version of our experiments.

Setting	Description
Baseline-synth	Model trained on 2 synthetic datasets (MJ+ST)
Baseline-real	Model trained on 11 real datasets (Real-L)
Aug.	Best augmentation setting in our experiments
PR	Combination of *Aug.*, PL and RotNet

Table 3: Description of our experimental settings.

E.1. Implementation Detail

In this section, we describe common factors in our experiments. Specific factors of each experiment are described at the beginning of each experiment.

Description of our settings Table 3 shows the description of our main experimental settings.

Training strategy Input images are resized into 32×100. We use He's initialization method [8] and gradient clipping at magnitude 5. The maximum word length is 25.

We use 94 characters for prediction, same with [31]: 26 upper case alphabets, 26 lower case alphabets, 10 digits, and 32 ASCII punctuation marks. In addition, 3 special tokens are added: "[PAD]" for padding, "[UNK]" for unknown character, and " " for space between characters. For CTC decoder, "[CTCblank]" token is also added. For attention mechanism, "[SOS]" and "[EOS]", which denote the start and end of sequence, are added.

As shown in Table 2, the number of training set is imbalance. To handle data imbalance, we sample the same number of data from each dataset to make a mini-batch. For example, when we use 11 datasets for training, we sample 12 images (= round(128/11)) per dataset to make a mini-batch. As a result, the batch size slightly changes depending on the number of datasets for training. However, the difference is marginal in our experiments and thus we use the balanced mini-batch. We also use the balanced mini-batch for three unlabeled datasets: 43 images (= round(128/3)) per dataset to make a mini-batch.

Evaluation metric As described in §5.1, the word-level accuracy is calculated only on the alphabet and digits. Following common practice [31], the accuracy is calculated only on alphabet and digits, after removing non-alphanumeric characters and normalizing alphabet to lower case.

Differences from the base code [1] We use the code of the STR benchmark repository as our base code. The code used in the STR benchmark is different from our settings: We use 1) All text in ST (7M) while the base code uses only alphanumeric texts in ST (5.5M), 2) Adam [11] instead of Adadelta [41], 3) An one-cycle learning rate scheduling [32], 4) Batch size reduced from 192 to 128, 5) Upper/lower case alphabets, digits, and some symbols (94 characters in total) for training while the base code uses lower case alphabets and digits (36 characters in total) for training, 6) The balanced mini-batch. This is not used for STR benchmark paper. 7) The CTCLoss from the PyTorch library instead of the CTCLoss from Baidu[10]. According to the STR benchmark repository[11], the CTCLoss from Baidu has higher accuracy by about 1% for CRNN than the CTCLoss from PyTorch library. In addition, 8) we construct and use our own validation set. 9) We calculate total accuracy on union of SVT, IIIT, IC13, IC15, SP, and CT, except for IC03. IC03 usually has higher accuracy than other datasets (over 90% accuracy), and thus excluding IC03 results in lower total accuracy than that of including IC03.

Environment : All experiments are conducted using Pytorch [26] on a Tesla V100 GPU with 16GB memory.

E.2. Comparison to State-of-the-Art Methods

We present the extended version of Table 2 in §5.2. Table 2 in §5.2 lists the methods that use only MJ and ST for training, and evaluate six benchmarks: IIIT, SVT, IC13-1015, IC15-2077, SP, and CT. In addition to Table 2 in §5.2, Table 4 also lists the methods that use SynthAdd (SA) [13] or both synthetic and real data,

SA is a synthetic dataset generated by Li *et al.* [13]. To compensate for the lack of special characters, Li *et al.* generated SA using the same synthetic engine with ST [6]. SA is used in some STR methods [13, 17].

Some methods use both synthetic and real data for training [13, 35, 40]. Their models trained on both synthetic and real data show better performance than their models trained only on synthetic data. However, fairly comparing the three methods [13, 35, 40] is difficult since they use the different numbers of real data (50K [13], 16K [35], and 7K [40]). Although they use different real data, we list them in Table 4.

Some methods use character-level labels [14, 39, 35]. ST has character-level labels, and the researchers use them. The methods using character-level information tend to obtain higher accuracy on irregular texts.

TRBA with our best setting (TRBA-PR) trained on both synthetic (MJ+ST) and real data has a competitive performance of 90.0% to state-of-the-art methods. Adding synthetic data SA for training further improves by 0.3% (from 90.0% to 90.3%).

E.3. Training Only on Real Labeled Data

Accuracy depending on dataset increment: Extended version Figure 10 shows the accuracy vs. the number of accumulated real labeled data per dataset increment. This is the extended version of Figure 1 in §1. Table 5 shows the accuracy along with the increment of real datasets.

Improvement by simple data augmentations: Varying degree of each augmentation As we mentioned in §5.3, the intensity of each augmentation affects the performance.

[10]https://github.com/baidu-research/warp-ctc
[11]https://github.com/clovaai/deep-text-recognition-benchmark/pull/209

			Type of training data				Dataset name and # of data						
			Synthetic data		Real data								
	Method	Year	MJ+ST	SA	labeled	unlabeled	IIIT 3000	SVT 647	IC13 1015	IC15 2077	SP 645	CT 288	Total 7672
Reported results	ASTER [31]	2018	✓				93.4	89.5	91.8	76.1	78.5	79.5	86.4
	SAR [13]	2019	✓	✓			91.5	84.5	91.0	69.2	76.4	83.3	83.2
	SAR [13]	2019	✓	✓	✓		95.0	91.2	94.0	78.8	86.4	89.6	89.2
	ESIR [42]	2019	✓				93.3	90.2	91.3	76.9	79.6	83.3	86.8
	MaskTextSpotter [14]*	2019	✓				95.3	91.8	**95.3**	78.2	83.6	88.5	89.1
	ScRN [39]*	2019	✓				94.4	88.9	93.9	78.7	80.8	87.5	88.2
	DAN [36]	2020	✓				94.3	89.2	93.9	74.5	80.0	84.4	86.9
	TextScanner [35]*	2020	✓				93.9	90.1	92.9	79.4	84.3	83.3	88.3
	TextScanner [35]*	2020	✓		✓		**95.7**	**92.7**	94.9	**83.5**	84.8	91.6	**91.0**
	SE-ASTER [28]	2020	✓				93.8	89.6	92.8	80.0	81.4	83.6	88.2
	SCATTER [17]	2020	✓	✓			93.7	**92.7**	93.9	82.2	**86.9**	87.5	89.7
	PGT [10]	2020	✓			✓	93.5	90.7	94.0	74.6	80.1	77.8	86.5
	RobustScanner [40]	2020	✓				95.3	88.1	94.8	77.1	79.5	90.3	88.2
	RobustScanner [40]	2020	✓		✓		95.4	89.3	94.1	79.2	82.9	**92.4**	89.2
	PlugNet [24]	2020	✓				94.4	92.3	95.0	82.2	84.3	85.0	89.8
Our experiment	TRBA												
	Original [1]	2019	✓				87.9	87.5	92.3	71.8	79.2	74.0	82.8
	Baseline-synth		✓				92.1	88.9	93.1	74.7	79.5	78.2	85.7
	+ SA		✓	✓			93.6	88.8	92.9	76.4	81.1	84.7	87.0
	Baseline-real				✓		93.5	87.5	92.6	76.0	78.7	86.1	86.6
	+ MJ + ST		✓		✓		95.1	91.0	94.9	79.0	82.4	89.1	89.1
	+ MJ + ST + SA		✓	✓	✓		95.4	91.3	95.2	80.2	83.8	92.1	89.8
	PR				✓	✓	94.8	91.3	94.0	80.6	82.7	88.1	89.3
	+ MJ + ST		✓		✓	✓	95.2	92.0	94.7	81.2	84.6	88.7	90.0
	+ MJ + ST + SA		✓	✓	✓	✓	95.4	92.4	95.0	81.9	84.8	89.5	90.3

Table 4: Extended version of Table 2 in §5.2. We show the results reported in original papers. MJ+ST, SA, and "*" denote union of MJSynth and SynthText, SynthAdd, and the model that uses character-level labeled data, respectively. Real labeled data has various variants as described in §E.2. In each column, top accuracy is shown in **bold**.

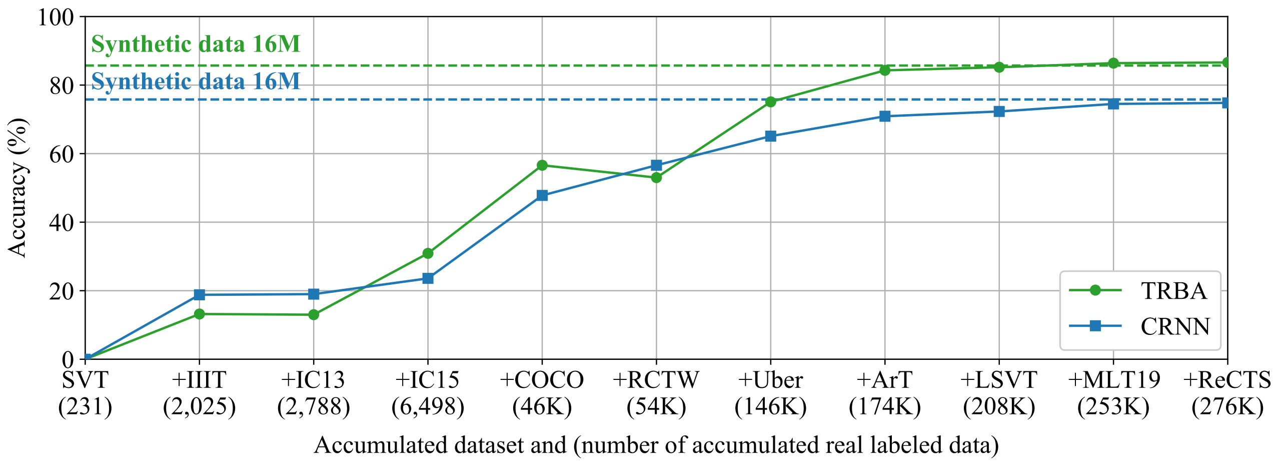


Figure 10: Accuracy vs. number of accumulated real labeled data, extended version of Figure 1 in §1. Along with the increment of real data, the accuracy obtained using real data approaches the accuracy obtained using synthetic data.

Dataset	CRNN	TRBA
SVT[(a)]	0.0	0.2
+ IIIT	18.8	13.2
+ IC13[(b)]	19.0	13.0
+ IC15[(c)]	23.6	30.9
+ COCO	47.8	56.6
+ RCTW	56.6	53.0
+ Uber[(d)]	65.1	75.1
+ ArT	70.9	84.3
+ LSVT	72.3	85.2
+ MLT19	74.5	86.4
+ ReCTS[(e)]	74.8	86.6

Table 5: Accuracy vs. dataset increment. (a), (b), (c), (d), and (e) denote the results of Year 2011, Year 2013, Year 2015, Year 2017, and Year 2019 in Figure 1 in §1, respectively.

We find the best intensities for each augmentation. Table 6 shows the results of them. We adjust the maximum radius r of Gaussian blur (Blur), the minimum percentage for cropping (Crop), and the maximum degree $^\circ$ of rotation (Rot).

We apply the augmentations with an intensity of 1 to confirm whether applying the weak augmentation will be effective. In our experiment, the weak augmentations are effective. $1r$ of Blur improves the accuracy by 0.8% for CRNN and 0.1% for TRBA. 99% of Crop improves the accuracy by 3.0% for CRNN and 0.5% for TRBA. TRBA performs the best at Crop 99%. 1° of Rot improves the accuracy by 0.3% for CRNN but decreases the accuracy by 0.8% for TRBA.

CRNN shows the best performance $5r$ of Blur, 90% of Crop, and 15° of Rot, respectively. They improve the accuracy by 0.9%, 4.0%, and 4.7% than that of no augmentation, respectively. TRBA shows the best performance $5r$ of Blur and 99% of Crop, respectively. They improve the accuracy by 0.2% and 0.5% than that of no augmentation, respectively.

When Rot is applied to TRBA, the accuracy of TRBA decreases. We presume that Rot can hinder the training of TPS in TRBA, which normalizes rotated texts into horizontal texts. It can result in the decrease in accuracy. We conduct additional experiments with the model called RBA, which is TRBA without TPS. When Rot is applied to RBA, the accuracy of RBA increases. 30° of Rot improves accuracy by 3.1% for RBA. This shows that Rot can improve the accuracy of STR models but might not be compatible with TPS.

In §5.3, we use 15° of Rot for TRBA to investigate the effect of combination of augmentations.

E.4. Semi- and self-supervised framework

Details of semi-supervised learning For Mean Teacher (MT), we basically follow the code from authors[12]. We do not use dropout [33] because most STR models do not use dropout [29, 31, 1]. In our experiments, we use 1.0 for the coefficient α of MT loss.

Details of self-supervised learning We follow the code from authors[13,14]. As a default, we use same hyperparameters with their code except for the number of iterations; we use 200K iterations. We use the same settings for the pretext task of RotNet: 1) Batch size 512; 128 $\times$ 4 rotations $(0^\circ, 90^\circ, 180^\circ, 270^\circ)$. 2) SGD optimizer with same setting. We use the same settings for the pretext task of MoCo: 1) Batch size 256. 2) SGD optimizer with same setting. 3) Same augmentation policy: resized crop, gray scale, color jitter, and horizontal flip.

E.5. Varying Amount of Real Labeled Data

To investigate the effect of the amount of real labeled data, we vary the ratio of each dataset while maintaining the diversity of datasets (keep using 11 datasets). Table 7 shows the number of word boxes used in the experiments varying amount of real labeled data. Table 8 shows the accuracy with varying amount of real labeled data.

References

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *ICCV*, 2019. 2, 3, 4, 7, 8, 9, 10

[2] Ayan Kumar Bhunia, Gautam Kumar, Partha Pratim Roy, R. Balasubramanian, and Umapada Pal. Text recognition in scene image and video frame using color channel selection. *Multimedia Tools and Applications*, 2017. 2

[3] Chee Kheng Chng, Yuliang Liu, Yipeng Sun, Chun Chet Ng, Canjie Luo, Zihan Ni, ChuanMing Fang, Shuaitao Zhang, Junyu Han, Errui Ding, et al. Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art. In *ICDAR*, 2019. 3

[4] Chee Kheng Ch'ng, Chee Seng Chan, and Chenglin Liu. Total-text: Towards orientation robustness in scene text detection. *IJDAR*, 2020. 3

[5] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 7

[6] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *CVPR*, 2016. 1, 4, 8

[7] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 7

[12]https://github.com/CuriousAI/mean-teacher
[13]https://github.com/gidariss/FeatureLearningRotNet
[14]https://github.com/facebookresearch/moco

Radius r	CRNN	TRBA
0	74.8	86.6
1	75.6	86.7
5	**75.7**	**86.8**
10	75.4	86.7
15	75.0	86.5

(a) Blur

Percentage %	CRNN	TRBA
100	74.8	86.6
99	77.8	**87.1**
95	78.6	87.0
90	**78.8**	86.0
85	78.3	85.7

(b) Crop

Degree °	CRNN	TRBA	RBA
0	74.8	**86.6**	83.8
1	75.1	85.8	83.9
15	**79.5**	86.2	86.7
30	79.4	86.0	**86.9**
45	79.3	85.6	86.5

(c) Rot

Table 6: Accuracy with varying degrees of data augmentations.

	# of word boxes per ratio				
Dataset	20%	40%	60%	80%	100%
Real labeled datasets (Real-L)					
SVT	46	92	138	184	231
IIIT	358	717	1,076	1,435	1,794
IC13	152	305	457	610	763
IC15	742	1,484	2,226	2,968	3,710
COCO	7,868	16K	24K	31K	39K
RCTW	1,637	3,274	4,911	6,548	8,186
Uber	18K	37K	55K	73K	92K
ArT	5,771	12K	17K	23K	29K
LSVT	6,702	13K	20K	27K	34K
MLT19	9,102	18K	27K	36K	46K
ReCTS	4,558	9,116	14K	18K	23K
Total	55K	110K	166K	221K	276K

Table 7: Number of word boxes used in the experiments with varying amount of real labeled data.

	Accuracy per ratio				
	20%	40%	60%	80%	100%
Dataset	55K	110K	166K	221K	276K
TRBA					
Baseline-real	51.2	78.2	83.6	85.4	86.6
+ *Aug.*	71.2	81.9	84.7	86.3	87.5
+ PR	80.2	85.1	87.0	88.5	89.3
CRNN					
Baseline-real	53.4	66.6	70.5	73.0	74.8
+ *Aug.*	66.4	73.6	76.3	78.6	80.0
+ PR	76.8	80.4	81.9	82.6	83.4

Table 8: Accuracy vs. amount of real labeled data. These values are plotted in Figure 6 in §5.5.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 8

[9] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Workshop on Deep Learning, NeurIPS*, 2014. 1, 2

[10] Klára Janoušková, Jiri Matas, Lluis Gomez, and Dimosthenis Karatzas. Text recognition–real world data and where to find them. *arXiv:2007.03098*, 2020. 7, 9

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 8

[12] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 7

[13] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. In *AAAI*, 2019. 8, 9

[14] 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. *TPAMI*, 2019. 2, 8, 9

[15] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *TIP*, 2018. 7

[16] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. In *AAAI*, 2017. 7

[17] Ron Litman, Oron Anschel, Shahar Tsiper, Roee Litman, Shai Mazor, and R Manmatha. Scatter: selective context attentional scene text recognizer. In *CVPR*, 2020. 7, 8, 9

[18] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang. Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In *CVPR*, 2020. 7

[19] Yuliang Liu, Tong He, Hao Chen, Xinyu Wang, Canjie Luo, Shuaitao Zhang, Chunhua Shen, and Lianwen Jin. Exploring the capacity of sequential-free box discretization network for omnidirectional scene text detection. *arXiv:1912.09629*, 2019. 4

[20] Yuliang Liu, Sheng Zhang, Lianwen Jin, Lele Xie, Yaqiang Wu, and Zhepeng Wang. Omnidirectional scene text detection with sequential-free box discretization. In *IJCAI*, 2019. 4

[21] Shangbang Long and Cong Yao. Unrealtext: Synthesizing realistic scene text images from the unreal world. *arXiv:2003.10608*, 2020. 3

[22] Xinghua Lou, Ken Kansky, Wolfgang Lehrach, CC Laan, Bhaskara Marthi, D. Phoenix, and Dileep George. Generative shape models: Joint text recognition and segmentation with very little training data. In *NeurIPS*, 2016. 2

[23] Anand Mishra, Karteek Alahari, and CV Jawahar. Enhancing energy minimization framework for scene text recogni-

tion with top-down cues. *Computer Vision and Image Understanding*, 2016. 2

[24] Yongqiang Mou, Lei Tan, Hui Yang, Jingying Chen, Leyuan Liu, Rui Yan, and Yaohong Huang. Plugnet: Degradation aware scene text recognition supervised by a pluggable super-resolution unit. In *ECCV*, 2020. 2, 9

[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018. 7

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 8

[27] Yash Patel, Tomas Hodan, and Jiri Matas. Learning surrogates via deep embedding. In *ECCV*, 2020. 7

[28] Zhi Qiao, Yu Zhou, Dongbao Yang, Yucan Zhou, and Weiping Wang. Seed: Semantics enhanced encoder-decoder framework for scene text recognition. In *CVPR*, 2020. 2, 9

[29] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *TPAMI*, 2016. 2, 7, 10

[30] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *CVPR*, 2016. 7

[31] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Aster: An attentional scene text recognizer with flexible rectification. *TPAMI*, 2018. 2, 7, 8, 9, 10

[32] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. *AI/ML for MDO*, 2019. 8

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 10

[34] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 7

[35] Zhaoyi Wan, Mingling He, Haoran Chen, Xiang Bai, and Cong Yao. Textscanner: Reading characters in order for robust scene text recognition. In *AAAI*, 2020. 2, 8, 9

[36] Tianwei Wang, Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen, Yaqiang Wu, Qianying Wang, and Mingxiang Cai. Decoupled attention network for text recognition. In *AAAI*, 2020. 2, 9

[37] Xing Xu, Jiefu Chen, Jinhui Xiao, Lianli Gao, Fumin Shen, and Heng Tao Shen. What machines see is not what they get: Fooling scene text recognition models with adversarial text images. In *CVPR*, 2020. 7

[38] Xing Xu, Jiefu Chen, Jinhui Xiao, Zheng Wang, Yang Yang, and Heng Tao Shen. Learning optimization-based adversarial perturbations for attacking sequential recognition models. In *ACMMM*, 2020. 7

[39] Mingkun Yang, Yushuo Guan, Minghui Liao, Xin He, Kaigui Bian, Song Bai, Cong Yao, and Xiang Bai. Symmetry-constrained rectification network for scene text recognition. In *ICCV*, 2019. 2, 8, 9

[40] Xiaoyu Yue, Zhanghui Kuang, Chenhao Lin, Hongbin Sun, and Wayne Zhang. Robustscanner: Dynamically enhancing positional clues for robust text recognition. In *ECCV*, 2020. 2, 8, 9

[41] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv:1212.5701*, 2012. 8

[42] Fangneng Zhan and Shijian Lu. Esir: End-to-end scene text recognition via iterative image rectification. In *CVPR*, 2019. 2, 9

[43] Jinjin Zhang, Wei Wang, Di Huang, Qingjie Liu, and Yunhong Wang. A feasible framework for arbitrary-shaped scene text recognition. *arXiv:1912.04561*, 2019. 2