

RETHINKING IRREGULAR SCENE TEXT RECOGNITION

TECHNICAL REPORT FOR PARTICIPATION IN ICDAR2019-ART RECOGNITION TRACK

Shangbang Long[†]
Department of Machine Learning
Carnegie Mellon University
Pittsburgh, USA
longshangbang@cmu.edu

Yushuo Guan
School of EECS
Peking University
Beijing, China
david.guan@pku.edu.cn

Bingxuan Wang
Yuanpei College
Peking University
Beijing, China
wangbx@pku.edu.cn

Kaigui Bian
School of EECS
Peking University
Beijing, China
bkg@pku.edu.cn

Cong Yao
MEGVII (Face++) Inc.
Beijing, China
yaocong2010@gmail.com

ABSTRACT

Reading text from natural images is challenging due to the great variety in text font, color, size, complex background and etc.. The perspective distortion and non-linear spatial arrangement of characters make it further difficult. While rectification based method is intuitively grounded and has pushed the envelope by far, its potential is far from being well exploited. In this paper, we present a bag of techniques that prove to significantly improve the performance of rectification based method. On curved text dataset, our method achieves an accuracy of 89.6% on CUTE-80 and 76.3% on Total-Text, an improvement over previous state-of-the-art by 6.3% and 14.7% respectively. Furthermore, our combination of techniques helps us win the ICDAR 2019 Arbitrary-Shaped Text Challenge (Latin script), achieving an accuracy of 74.3% on the held-out test set. We release our code as well as data samples for further exploration at <https://github.com/Jyouhou/ICDAR2019-ArT-Recognition-Alchemy>.

1 Introduction

Recently, the detection and recognition of irregular text from natural images have become a new popular research topic [20, 26, 39, 4, 21, 1, 32, 28, 16, 14]. However, most detection methods describe text as bounding boxes or groups of pixels in the form of semantic segmentation, without any indication of the shapes. Thus, the recognition models still need proper mechanism to deal with curved text.

Deep learning based text recognition algorithms have transitioned from *one-dimensional* methods, i.e. CRNN and CTC based methods [30, 18, 13, 2, 27, 5, 36], to *quasi-two-dimensional* methods using rectification [28, 29] which is adapted from Spatial Transformer Networks (STN) [10], and currently to *two-dimensional* methods with Fully Convolutional Networks [19] or spatial attention [35] such as [16, 14]. These methods have started and followed a standard and well accepted training and evaluation protocol, using images rendered with synthesis engines [9, 6] as training data and then evaluating on real world datasets. Besides, they share a standard pre-processing step by resizing all images to a fixed size, usually set as 64×256 pixels.

Despite these widely-adopted trends, the aforementioned common practices fall short when faced with irregular text. Specifically, synthetic data generated by existing algorithms [9, 6, 38, 15] largely consist of text aligned to straight lines. Although the 800K synthetic images generated by SynthText [6] do contain curved text, the proportion is small and they are only slightly curved. Besides, images with irregular text are usually seriously warped if they are resized to

[†]Work done while Shangbang Long was an intern at MEGVII (Face++) Inc., Beijing, China.

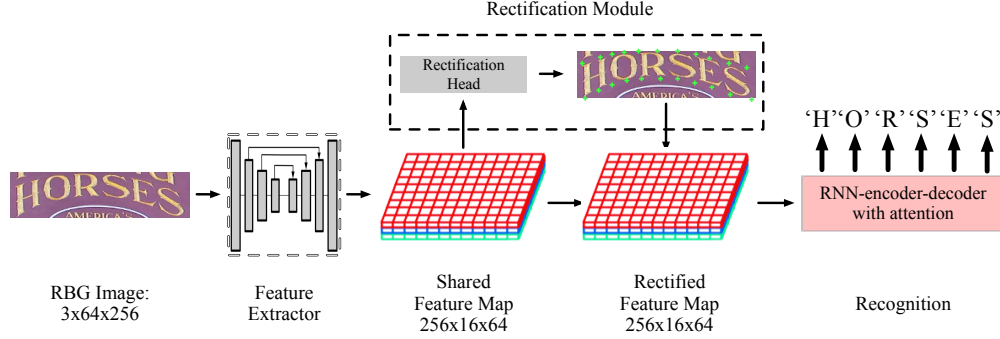


Figure 1: The pipeline of our implementation. The module marked with dotted rectangles is the rectification module. When we take off the rectification module, it becomes a vanilla RNN and CNN based model. The green dots inside represent predicted control points. For implementation details, we refer readers to our code.

64×256 . Therefore, it is important and worthwhile to study the bottleneck effect of these factors in building strong and robust text recognizers.

In this paper, we examine a set of techniques from different aspects that improve the performance of recognizers with focus on irregular text. These techniques may seem to be slight modification to models, data, and training procedures at first glance. However, they are reasonable and bring considerable improvement over baselines and even beat state-of-the-art models by a large margin. We design comprehensive experiments to analyse the effect of these techniques. Furthermore, we are also the first to evaluate on the newly published curved text dataset, Total-Text [3], which contains 2201 images, much larger than the CUTE80 dataset [25]. Images in Total-Text are characterized by even more challenging conditions, and thus should be a better performance indicator for curved text detectors.

Our paper is organized as follows. In Section 2, we first set up our baseline methods and experiment settings. The experiments throughout this paper follow the same settings for fair comparison, unless otherwise specified. In Section 3, we discuss the selection of training data. In Section 4, we evaluate several model modifications. In Section 5, we review our participation in the ICDAR 2019 ArT challenge, and describe our system used in the competition. In Section 6, we summarize our findings and contributions.

The code and training scripts are all included in our Github repository as shown in the abstract.

2 Baseline System and Experiment Settings

Basically, we follow the standard of training and evaluating scene text recognition algorithms as most previous works do. We give a brief description of our baseline recognition system in Section 2.1. We choose a rectification based method as our experiment tools. Then, we specify our experiments settings in Section 2.2 and Section 2.3. Details regarding to the models and experiments can be found in our code repository. Especially, we would like to thank Mingkun Yang for his repository¹ of the Aster algorithm [29].

2.1 Baseline Models with Rectification Layer

We follow and modify the PyTorch implementation of Aster [29] released by its authors as mentioned above. The pipeline is shown in Fig. 1, and is composed of the following steps:

Preprocessing Input images are first resized to 64×256 . Pixel values are subtracted and then divided by 128.0, thus normalized to a -1.0 to 0.99 base.

Feature Extractor We use a standard 50-layered ResNet [7] as backbone network and Feature Pyramid Networks [17] to extract features from images. The output feature maps have $\frac{1}{4}$ the resolution of input images, that is, 16×64 , and 256 channels. We believe that the widely-used ResNet+FPN feature extractor should serve as a fair playground for easy replication and controlled experiments.

Rectification Module To perform rectification, we first apply a control point header to localize the text with a 20-vertexed polygon. The header consists of a series of alternating convolutional layers and maxpooling layers that encode the $256 \times 16 \times 64$ feature maps from feature extractor to a 512-d vector. Two fully connected networks follow and

¹<https://github.com/ayumiymk/aster.pytorch>

output the polygon. Then, the feature maps are rectified to axis-aligned form with Thin-Plate-Spline (TPS) [34]. Details with regard to the grid generator is the same as that specified in Aster.

Recognition Module The input to the recognition modules is the rectified feature maps output from the rectification module, with a size of $256 \times 16 \times 64$. We follow the PyTorch implementation of Aster, and attach several convolutional layers, maxpooling layers and Bi-LSTM layers [8] layers which act as encoders, and then attentional decoders. The decoder is an LSTM with attention mechanism to predict a sequence of symbols from a predefined character list. To produce sequence of variable length, a special end-of-sentence symbol, denoted as *EOS*, is added to the character list and appended to the end of training target. During inference time, decoding ends once the *EOS* is emitted. Apart from *EOS*, 10 Arabic digits and 52 case-sensitive alphabets are predicted.

This model is referred to as **rectification baseline** in our paper or **Rect** for short. These two names will be used interchangeably in the following sections.

2.1.1 Non-Rectification Baseline

As shown in Fig.1, we can also take off the rectification module and it becomes a variant of the convolutional recurrent neural network (CRNN) [27], with RNN based transcription module stacked upon convolutional layers. The difference between CRNN and our model is that, the sequence output layer of CRNN is based on CTC, while our model uses the RNN-based encoder-decoder framework [31] which enables training and testing with variable lengths. We term our model without the rectification layer as **non-rectification baseline**, or **Non-Rect** for short. We only modify the rectification layer so that the Non-Rect model also serves as a benchmark for quantitative evaluation of the proposed techniques. The experiment results of Non-Rect can help attribute improvements to models and different techniques respectively. Also note that the input and output of the rectification module have the same size, and therefore, taking off it does not change the rest of the pipeline.

2.2 Training Settings

Training Target Given input image $I \in R^{H \times W \times 3}$ and ground truth text sequence $\{y_i\}_{t=1}^T$, the loss function is formulated as *cross-entropy loss averaged over time*:

$$L = -\frac{1}{T} \sum_{t=1}^T \log p(y_t | I)$$

Note that the last symbol of the ground truth sequence is always the special symbol *EOS*.

Optimization We use ADADELTA [37] with default hyper-parameters (rho=9e-1, eps=1e-6, weight decay=0) to minimize the aforementioned loss function. Gradients are estimated using mini-batches with 512 images that are randomly sampled from training set. We train 6 epochs in total. We initialize the learning rate to 1.0 for the first 4 epochs, 0.1 for the fifth epoch and 0.01 for the sixth epoch. All experiments are performed on a Ubuntu machine with 4 NVIDIA TITAN Xp graphics cards, each with 12GB memory.

2.3 Evaluation

We evaluate the models on a total of 9 datasets. These datasets are briefly described here:

IIIT 5K [23] contains 3000 horizontal testing images collected from the Internet that represent challenges in scene text recognition, such as the variety in font, size, and color.

Street View Text (SVT) [33] contains 647 testing images collected from the Google Street View. Many of them are corrupted by noise, blur and low resolution.

ICDAR 2003 (IC03) [22] recognition task contains 860 horizontal focused images. Following previous works [33], we only consider instances with at least 3 characters.

ICDAR 2013 (IC13) [12] inherit mostly from IC03 and contains 1015 images.

ICDAR 2015 (IC15) [11] is collected with Google Glasses without taking care of posing, position, and focus. The images are blurred, of low quality, and most of the text instances are oriented and small. We only consider instances with at least 3 characters. The total number of images is 2077.

SVT-Perspective (SVTP) [24] is collected in a similar way to SVT, with special focus on perspective text. It contains 645 images.



Figure 2: Selected samples of RectTotal produced by applying TextSnake with ground truth geometry attributes. Note that, not only the text themselves are rectified, but the image background is also largely eliminated.

CUTE 80 [25] is designed for curved text. It only contains 288 instances for testing. By far, CUTE80 is the most widely used dataset to evaluate curved text recognition, but it has a small sized compared to other datasets.

Total-Text [3] is a dataset devoted to curved text detection and recognition. It has 1000 scene images for training and 500 for testing. A large proportion of the instances are curved text and greatly oriented, making it difficult for both detection and recognition. We crop text instances from the test set, and obtain 2201 cropped text images.

Rectified Total-Text (RectTotal) We propose **RectTotal** which is obtained by using the TextSnake [21] algorithm to rectify the test images using ground truth geometry attributes. If the detection module can precisely capture the shape of text instances instead of predicting them with groups of unordered pixels, curved text recognition algorithms may be less necessary. We include the results here for future reference. We also release RectTotal and the whole RectTotal dataset can also be found in our code repository. We select some samples for demonstration as shown in Fig.2.

ICDAR 2019 ArT (IC19-ArT)² is a competition devoted to curved text detection and recognition. The dataset for the recognition track contains 50K images for training and testing respectively, and contain both Latin script and Chinese Script. In this paper, we only consider Latin script. The total number of Latin images in the training set is 36220. Among these images, we randomly split 3000 images as validation set, to match the size of other dataset. Also, as the test set is not available by the time this paper is written, we use this validation set as test set for evaluation.

Among the 9 dataset described above, the first 7 datasets have been widely used in research. Therefore, we can compare the performance of our methods with previous ones. For the latter 2 curved text dataset, we are the first to report experiment results. Without a reference point, we can only use them to evaluate the effectiveness of the proposed techniques in the form of ablation test. Nonetheless, they are much larger than the previous curved text baseline CUTE 80, and should act as better benchmarks. Therefore we release the results so that they can serve as a proper baseline for future research.

3 Select the Right Training Data

3.1 Synthetic Text

In the last few years, training solely on synthetic data has become a widely accepted norm. To be more precise, there are two synthetic datasets that are used by researchers: (1) the 800K synthetic images generated by the SynthText engine [6] that contain 7M cropped text images and (2) 9M cropped images from Synth90K [9]. Although SynthText has more realistic rendering, it is based on an imbalanced vocabulary. On the other hand, Synth90K has a balanced and large vocabulary. However, its images are monochrome, and the visual appearance is simpler than that of SynthText. Therefore, in most papers, the two datasets are combined to complement each other. Such a norm continues even after the recognition of curved text became a hot topic.

However, these two datasets only contain a low proportion of curved text, which may make it hard for recognizers to generalize to curved text. It is therefore reasonable and intuitive to generate synthetic curved text as a replacement. As SynthText provides an opensource code repository³ and Synth90K does not, we decide to work on SynthText.

3.1.1 Synthesizing curved text

The SynthText engine renders text as a curve when the sampled text line contains only one word and the word contains no more than 10 characters. Then, it samples a parabolic trajectory $y = \alpha x^2$ and places characters one by one symmetrically around the origin point. According to the code⁴, α is uniformly sampled from $[-0.50, -0.45] \cup [0.50, 0.55]$. The potential problem is that, however, the engine’s default parameters are set such that text source always attempts to sample a paragraph, i.e. multiple lines of text, and therefore it is only in rare condition that, there is only one word to

²<https://rrc.cvc.uab.es/?ch=14>

³<https://github.com/ankush-me/SynthText>

⁴https://github.com/ankush-me/SynthText/blob/6640bf6a0d07b01cbb108984814af2aeb6e30344/text_utils.py#L59



Figure 3: Randomly selected samples generated by our modified SynthText Engine.

Table 1: Experiments with rectification baseline and non-rectification baseline trained on different synthetic data. For each dataset including RectTotal, we mark the highest score in bold.

Model	Dataset	IIIT5K	SVT	IC03	IC13	IC15	SVTP	CUTE	Total-Text (RectTotal)	IC19-ArT
Non-Rect Baseline	<i>CurvedSynth</i>	0.906	0.836	0.927	0.888	0.698	0.695	0.847	0.692 (0.627)	0.653
	<i>SynthText</i>	0.912	0.849	0.929	0.915	0.731	0.722	0.799	0.629 (0.666)	0.649
	<i>Synth90K</i>	0.842	0.847	0.938	0.897	0.696	0.740	0.674	0.409 (0.661)	0.568
Rectification Baseline	<i>CurvedSynth</i>	0.920	0.850	0.922	0.893	0.722	0.740	0.837	0.724 (0.660)	0.671
	<i>SynthText</i>	0.916	0.853	0.927	0.907	0.727	0.735	0.792	0.636 (0.663)	0.660
	<i>Synth90K</i>	0.846	0.849	0.949	0.900	0.612	0.771	0.677	0.415 (0.665)	0.569

render when the selected location of text is small enough. As a result, with the current parameters, curved text are rare in the original 800K dataset.

To solve this issue, we revise the text rendering module of the engine. The main idea of the new text rendering module is two-fold: (1) we increase the proportion of single word text so that more text are rendered as curved; (2) we randomly sample a radius and place text on the corresponding circle instead. The parameters are set such that there are a larger proportion of curved text and the curvature of text is greater. We also opensource the modified code for future research⁵. For technical details, readers can refer to the code.

We obtain 7M image crops in total. We estimate that around 10% of the generated data are curved. We randomly select some samples from our modified SynthText for visualization, as shown in Fig.3. We refer to our version as *CurvedSynth*.

3.1.2 Experiments with synthetic data

Now we have three synthetic datasets: the original *SynthText*, *CurvedSynth*, and *Synth90K*. To compare the effectiveness of different datasets, we first conduct an experiment with these datasets separately. When trained with *Synth90K*, we transform the testing images to grey scale. We train the rectification baseline on these datasets respectively, and evaluate them on the 9 real-world datasets. We also carry out the same experiment with the Non-Rect model to see how much datasets can benefit models.

The experiment results are summarised in Tab.1. They show some consistency across models and dataset. We analyse these aspects, and inspect these evaluation datasets as well as model and training datasets one by one. We would present some very interesting findings.

Performance on straight text We first focus on the 6 horizontal datasets: IIIT5K, SVT, IC03, IC13, IC15, and SVTP. On IIIT5K and SVT, it is obvious that the rectification baseline performs better than the Non-Rect baseline. As for training datasets, models trained on *CurvedSynth* parallels *SynthText* overall, and *Synth90K* ranks the worst. It is reasonable as IIIT5K actually contain a small proportion of curved and oriented text. SVT also has a few curved sample, and remember that, 1 more correct image on SVT means an improvement by 0.15%. Therefore, the difference in the number of correctly recognized images may be little. We also note that, although models trained on *SynthText* performs slightly

⁵<https://github.com/Jyouhou/CurvedSynthText>



Figure 4: Randomly selected results by rectification baseline from IC03 and IC13 trained with different training datasets. The upper part of each sample is the input images which are resized to 64×256 and the bottom is the rectified images.

better when using Non-Rect, *CurvedSynth* soon catches up when using rectification layer. It is reasonable to attribute it to the compatibility between rectification layer and curved text. Other wise, the curved text are distorted greatly when resized to 64×256 , which makes it difficult to recognize.

On IC03 and IC13, things are different. As IC03 and IC15 contain few curved text, it is as expected that rectification layer brings no improvements. Besides, the fact that models trained with *CurvedSynth* performs worse than *SynthText* may indicate that redundant curved training examples may make the model less competent on purely horizontal text. It is worth-noting that, models trained on *Synth90K* benefits from the rectification layer on IC03 by a considerable margin. According to a recent survey [20], the state-of-the-art performance on IC03 is 94.5% when trained on *Synth90K*+*SynthText*, which is surpassed by ours. To understand what contributes to it, we inspect the evaluation results by checking each testing images one-by-by manually. We notice a outstanding phenomenon, select some representatives, and show them in Fig.4. We notice that the rectification layer actually shorten some text so that the aspect ratios of characters look better, although leaving some blank at the end. Such resizing is frequently found in results produced by models that are trained on *Synth90K*. This phenomenon also exists in models trained with *CurvedSynth*, but not in the original *SynthText*. It is also not found in IC13, while there is also little improvement on IC13. We assume that encouraging such phenomenon can be an important technical trick.

On IC15 and SVTP, we carry out similar in-depth inspection on testing images one-by-one and found similar phenomenon. Models trained with *Synth90K* seems to be better at resizing the text to a more normal aspect ratio. It remains a challenge to investigate the reason behind this. We suppose it is because *Synth90K* has a balanced vocabulary, and therefore has more long words, while the *SynthText* engine uses an imbalanced vocabulary, and in natural corpus, most high-frequency words are short. Thus, models trained on *Synth90K* would have to learn to resize characters so that words with varying lengths can fit the input size better.

Performance on curved text On CUTE80, Total-Text, and IC19-Art, we observe consistent improvements brought by our *CurvedSynth* over the original *SynthText*. It results in an improvement of 4-5% on CUTE80, around 6-9% on Total-Text, and about 1% on IC19-ArT, depending on the model we use. Especially, when we use rectification baseline, the use of *CurvedSynth* results in greater improvements over models trained with *SynthText* on Total-Text, from 6.3% to 8.8%, and IC19-ArT, from 0.4% to 1.1%. Although performance on CUTE80 drops after adding rectification layer, it can be attributed to the small size of CUTE80 and the variance it may have.

On the other hand, when trained on *SynthText* or *Synth90K* which contain few curved text, the rectification layer brings only insignificant improvements. The improvements are less than 0.3% on CUTE80, 0.6% on Total-Text, and 0.1-1.0% on IC19-ArT. These numbers are quite small compared to improvements by introducing our *CurvedSynth*. When using *CurvedSynth*, the rectification layer can boost performance by 3.2% on Total-Text and 1.8% on IC19-ArT. We can draw a conclusion that, our proposed *CurvedSynth* can help rectification layer reach its potential by providing appropriate data.

Performance on rectified Total-Text We also evaluate all models on the RectTotal dataset we create using the TextSnake algorithm as specified in Section 2.3. The results are marked by a set of brackets in the same table. We notice that the Non-Rect baseline trained on *SynthText* with few curved text instances achieve the highest score of 66.6%. The performance of models except Non-Rect trained on *CurvedSynth* is not significantly different from this best model. The reason why Non-Rect trained on *CurvedSynth* achieves the lowest accuracy may be attributed to the fact that, since the images in RectTotal are rectified, background is eliminated and the text compactly fit into and fill the images, as shown in the bottom row of Fig.2. This, as it turns out, makes these images visually more similar to the training images in *SynthText* and *Synth90K*. On the contrary, the curved text in *CurvedSynth* contain background and complex spatial alignment, which is actually out-of-distribution for RectTotal.

Another counter-intuitive phenomenon is that, models trained on *CurvedSynth* perform worse on RectTotal overall than on the original Total-Text. Note that, as the ground-truth polygon annotations of Total-Text is imprecise, the rectified images may be warped. Therefore, the performance gap may be due to the extra distortion brought up by this process.

Table 2: Experiments with our rectification baseline trained on multiple synthetic datasets compared with previous state-of-the-art methods. Performance scores marked with ‘-’ indicate that they are not reported. ‘*’ indicates results provided by asking the authors but not presented in the paper. Again, performance on RectTotal is included only for future reference. Best results are marked in **bold**. All scores here are obtained by training models solely on synthetic data.

Model	Dataset	IIIT5K	SVT	IC03	IC13	IC15	SVTP	CUTE	Total-Text (RectTotal)	IC19-ArT
SAR[14]	<i>SynthText</i> + <i>Synth90K</i>	0.915	0.845	-	0.910	0.692	0.764	0.833	-	-
CA-FCN[16]	<i>SynthText</i> + private data	0.919	0.864	-	0.915	-	-	0.799	0.616* (-)	-
Rectification Baseline	<i>CurvedSynth</i> + <i>Synth90K</i>	0.948	0.896	0.958	0.928	0.782	0.816	0.896	0.763 (0.734)	0.721

Table 3: Sizes of real world datasets with annotations. The last column entitled *sum* is the total amount of data for train/test respectively.

	IIIT5K	SVT	IC03	IC13	IC15	SVTP	CUTE	Total-Text	IC19-ArT	COCO	sum
train	2000	257	1156	848	4467	0	0	9267	33220	13943	65158
test	3000	647	860	1015	1811	645	288	2201	3000	0	13467

Basically, experiments on RectTotal lead us to the conclusion that, with a good detection model that can capture the shape of text and further rectify it, sophisticated model designs and tricks may be unnecessary. Specifically, we can observe absolute improvements in accuracy by 25% on Total-Text for models trained on *Synth90K* if we rectify them in advance. But for now, we still need them as IC19-ArT is not rectified.

3.1.3 Experiments with multiple synthetic data

As most recent methods are trained on *Synth90K* and *SynthText* jointly, we also present our results as trained on *Synth90K* and *CurvedSynth* jointly, which is shown in Tab.2. The total number of training data is 16M. With our *CurvedSynth*, we surpass recent baselines on all datasets by a large margin. We achieve significant improvements on CUTE80 by 6.3% and on Total-Text by 14.7%. The comparison verifies the effectiveness of our *CurvedSynth* dataset.

3.2 Mixing with Real World Data

As there are several annotated real world datasets, it seems a potential way to boost performance using real world images. However, the quantity of real world images is much smaller than synthetic data. It is then valuable to find out an optimal sampling scheme to balance real world data and synthetic data. This section is dedicated to this target.

We collect available real world data and summarize them in Tab.3. As IC19-ArT is only released recently, we do not add it into our training set for fair comparison with previous works that use real world data [14]. Together we obtain 31938 real world images. Since the number of synthetic data is much larger than that of real world images, we give real world data a larger sampling weight. We carry out experiments with different sampling weights such that the real world data account for 5%, 10%, 15%, 20% and 25% of the total training data. Also note that, we temporarily treat all real datasets the same and ignore the differences in their sizes.

As shown in Tab.4, our model outperforms previous methods that use real world data by a large margin consistently for different real world data ratios. We achieve improvements on all datasets except SVTP. Notably, IC15, SVTP, CUTE80, Total-Text and IC19-ArT all benefit significantly from real world data.

For an overall comparison, we compute the micro average score by taking the weighted average of all accuracy with dataset sizes as weights, and the average scores are shown in the last column labeled as Avg. In this sense, the rectification baseline performs best when the real world data should take up 15% of the total training data.

As for performance on curved text datasets, a ratio of 10% and 15% result in fairly comparative results. A proportion of 15% for real world data results in slightly higher accuracy in Total-Text and a 1.0% improvement is insignificant. Since the ratio is not making very large difference on most datasets, we would continue with 15% in later experiments on IC19-ArT due to its superiority both in curved text and in the overall way.

Table 4: Experiments with synthetic data mixed with real world data.

Model	Real World Data Ratio (%)	IIIT5K	SVT	IC03	IC13	IC15	SVTP	CUTE	Total-Text	IC19-ArT	Avg
SAR[14]	13.5	0.950	0.912	-	0.940	0.788	0.864	0.896	-	-	-
Rectification Baseline	0	0.948	0.896	0.958	0.928	0.782	0.816	0.896	0.763	0.721	0.834
Rectification Baseline	5	0.961	0.927	0.952	0.949	0.834	0.842	0.938	0.817	0.768	0.868
Rectification Baseline	10	0.958	0.924	0.964	0.953	0.832	0.851	0.931	0.814	0.762	0.866
Rectification Baseline	15	0.957	0.921	0.955	0.945	0.847	0.842	0.934	0.815	0.772	0.869
Rectification Baseline	20	0.956	0.926	0.966	0.948	0.839	0.840	0.938	0.807	0.759	0.864
Rectification Baseline	25	0.954	0.924	0.958	0.951	0.838	0.828	0.951	0.809	0.752	0.862

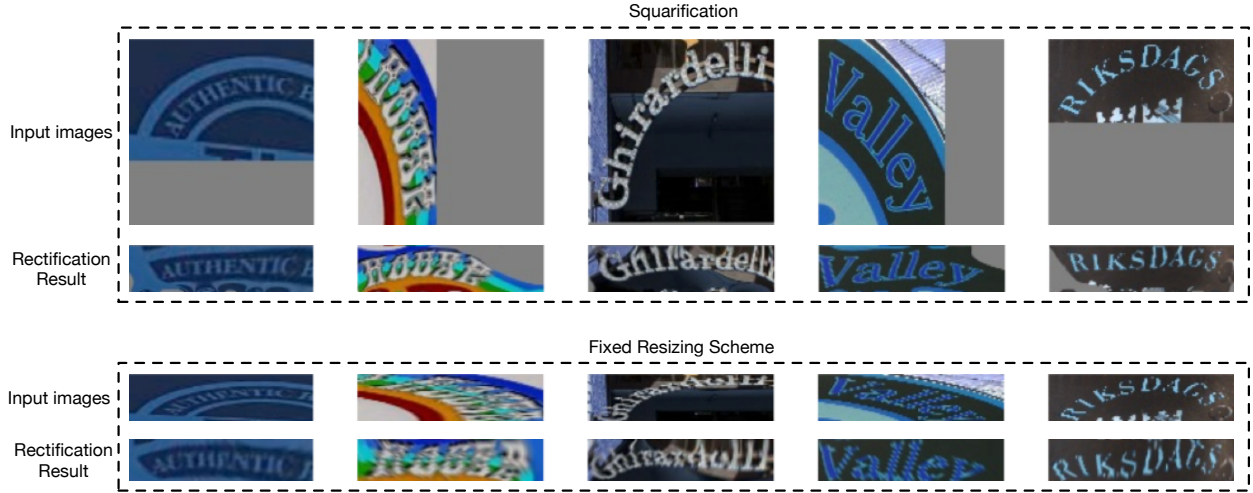


Figure 5: Visualization of rectification results for squarization compared with fixed resizing scheme. Within each dotted frame, in the upper part lies the input image, and in bottom part lies the rectification results. We can see that, rectification based on square images produce higher image quality, more clear rectified images and better overall rectification.

4 Model Design Tweaks

The previous sections mainly focus on the selection of data. In this part, we mainly talk about some model modifications that improve the performance of our rectification based methods. For fair comparison with most previous works as well as previous sections, we only train our method on synthetic data, i.e. *Synth90K* + *CurvedSynth*.

4.1 Keeping Aspect-Ratio for Input Images

As suggested in Section 3.1.2, the rectification layer learns to resizing the input images, and it is likely to be the reason why it performs better on those datasets. We suppose an aspect-ratio keeping input would benefit recognizers. Therefore, we design a new and intuitive pre-processing step as specified here.

For input images with different sizes, we first resize the long sides to 256 pixels, and short sides resized accordingly to keep the original aspect-ratios. Then, we pad *grey* borders to the short sides such that the pre-processed images are square. We select and visualize some samples in Fig.5. It is obvious that our pre-processing is more friendly to curved text. Compared with fixed-size scheme that resizes images to 64×256 uniformly, our method keeps the images semantically legible. We term our pre-processing as **squarization**.

Table 5: Results of experiments with model modifications.

Model	Modification	IIIT5K	SVT	IC03	IC13	IC15	SVTP	CUTE	Total-Text	IC19-ArT
Rectification Baseline	None	0.948	0.896	0.958	0.928	0.782	0.816	0.896	0.763	0.721
Rectification Baseline	Squarization	0.950	0.898	0.950	0.927	0.782	0.809	0.892	0.769	0.724
Rectification Baseline	Squarization + Random Rotation	0.949	0.898	0.948	0.925	0.781	0.817	0.889	0.781	0.728
Rectification Baseline	Squarization + Rectify Image	0.946	0.904	0.953	0.925	0.792	0.795	0.800	0.776	0.723
Rectification Baseline	All	0.942	0.881	0.948	0.920	0.771	0.803	0.903	0.761	0.720

4.1.1 Random Rotation

Another advantage of squarization is that, now images can be rotated freely in both training and testing stages. As instances in curved text dataset usually have oriented or even vertical arrangement, it is important to incorporate such situations in the training stages as well. Therefore, we propose to randomly rotate the images during training. Specifically, the images are randomly rotated 90, 180, or 270 degrees with a probability of 5% respectively.

4.2 Rectification on Images

While rectification layer can alleviate distortions to some extent, it only operates on the extracted feature maps and therefore the features may still be affected by the distortion in images. Therefore, we carry out experiments to inspect whether performing rectification on images can benefit recognition.

Specifically, the images are first fed forward to obtain the control points as our rectification baseline does. Then the control points are used to rectify the input images. Then the rectified images are again fed into the same network. The recognition module is applied to the feature maps extracted from the rectified images.

In implementation, such two-pass computation increases memory usage by nearly one time, and makes it impossible to fit in the GPUs we have. Therefore, in the first pass, we downsample the input images to half size using bilinear interpolation, but the second pass is performed on the images of original resolution. By reducing the input size of the first pass, the rectification module based on fully connected networks can also remain unchanged.

4.3 Experiments

We list the experiment results in Tab.5. We can see that the combination of squarization and random rotation boost performance on curved text recognition, with improvements on Total-Text by 1.8%, and IC19-ArT by 0.7%. Besides, squarization itself only results in slight improvements on nearly all datasets. However, we argue that squarization is still important because it enables the random rotation trick that can bring more improvements.

Unfortunately, the practice of rectifying images does not seem to be very effective. Although it scores highest on SVT with an improvement by 0.8% and IC15 by 1.0%, it fails on other datasets. We suppose rectification on images may make it difficult to train.

5 Evaluation on ICDAR-2019

In this part, we reveal details of how we build our recognition system for participation in IC19-ArT.

Data We pool all available data together: synthetic data including *CurvedSynth* and *Synth90K*, all real world data as listed in Tab.3 except the 3000 validation images we sample from IC19-ArT. We have 16M synthetic data in total, and 75625 real world images. During training, we sample real world data with a larger weight so that real world data take up 15% of total training data. We use the 3000 validation set to select the best models.

Model We train an ensemble of the following models on the aforementioned data: (1) rectification baseline; (2) squarified rectification model with random rotation; (3) rectification baseline with ResNet152; (4) rectification baseline with rectification on images. We use the validation set to select the best model parameters for each of the 4 models and combine the prediction results via a simple voting mechanism: we simply select the text that appear most from the 4 predictions.

The 4 models achieve validation accuracy from 76.5% and 78.5%. The final accuracy on the test set that is not yet published is 74.3%.

6 Conclusion

In this paper, we investigate several aspects that may affect the training and final performance of rectification based scene text recognizers on curved text datasets. We implement a rectification-based method with a standard ResNet+FPN as feature extractor, a rectification module, and a vanilla attentional RNN based encoder-decoder sequence learning module. We conduct comprehensive experiments to study the task of curved text recognition, and we finally arrive at the following conclusions:

1. **Training data that contain curved text are very important yet economical.** Despite the latest fashion of training solely on the standard SynthText and Synth90K for fair comparison, these two datasets contain few curved instances, and are therefore not suitable for curved text recognition. Algorithms trained on these datasets are not able to reach their full potential. We demonstrate this by training a previous state-of-the-art method on curved synthetic text, and boost its performance by as much as 6.3% on CUTE80, even surpassing the most recent state-of-the-art trained on both synthetic data and real world data. On the other hand, the synthesis of curved text is effortless and economical, and can replace existing datasets in a drop-in fashion. The proposed *CurvedSynth* is a better dataset to study the task of curved text recognition.
2. **Mixing synthetic data with real ones also makes a difference.** This rule of thumb is self-evident, but technical details are important. As shown in our experiments, it is important to find an appropriate proportion for the real world data. Besides, these datasets vary greatly in size, from a few hundreds to tens of thousands. Thus, it is also valuable to design a method to find an optimal ratio among real world datasets.
3. **Squarization works, and brings up new possibility.** Another standard norm we challenges is the fixed-size pre-processing that resizes all input images to the same preset size, which is usually 64×256 . Although the proposed squarization only results in slight improvements on curved datasets and only parallel results on straight text, it enables the use of random rotation, which can boost the performance on curved text to some extent. For future directions, it is also worthwhile to study inputs with variable lengths/widths, as done by the NLP community. This may be important for long text, such as Chinese, Japanese, Korean, and etc..
4. **Recognizers are more robust to shape when working on images rectified by select detectors.** We propose **RectTotal**, a dataset obtained by applying TextSnake to rectify test images of Total-Text using the ground truth polygon-based labels. Experiment results show that, even if algorithms are solely trained on straight text and not equipped with special mechanism for curved text recognition, they can achieve equally good results if the text are already rectified. This result places more importance on detectors that can capture the shape of text, such as TextSnake and CRAFT [1]. Text detection and recognition should be co-designed to achieve better results.

Finally, we release the training details for our participation in IC19-ArT. We hope it will help researchers in designing better algorithms and further mining new challenges that are not yet noticed by the community.

References

- [1] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9365–9374, 2019.
- [2] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5086–5094. IEEE, 2017.
- [3] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, pages 935–942. IEEE, 2017.
- [4] Deng Dan, Liu Haifeng, Li Xuelong, and Cai Deng. Pixellink: Detecting scene text via instance segmentation. In *Proceedings of AAAI, 2018*, 2018.
- [5] Yunze Gao, Yingying Chen, Jinqiao Wang, and Hanqing Lu. Reading scene text with attention convolutional sequence modeling. *arXiv preprint arXiv:1709.04303*, 2017.

- [6] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2315–2324, 2016.
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.
- [8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [9] 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.
- [10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.
- [11] Dimosthenis Karatzas, Lluís Gomez-Bigorda, Angelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 1156–1160. IEEE, 2015.
- [12] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluís Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluís Pere de las Heras. Icdar 2013 robust reading competition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1484–1493. IEEE, 2013.
- [13] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2231–2239, 2016.
- [14] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. *AAAI*, 2019.
- [15] Minghui Liao, Boyu Song, Minghang He, Shangbang Long, Cong Yao, and Xiang Bai. Synthtext3d: Synthesizing scene text images from 3d virtual worlds. *arXiv preprint arXiv:1907.06007*, 2019.
- [16] Minghui Liao, Jian Zhang, Zhaoyi Wan, Fengming Xie, Jiajun Liang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Scene text recognition from two-dimensional perspective. *AAAI*, 2019.
- [17] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [18] Wei Liu, Chaofeng Chen, Kwan-Yee K Wong, Zhizhong Su, and Junyu Han. Star-net: A spatial attention residue network for scene text recognition. In *BMVC*, volume 2, page 7, 2016.
- [19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
- [20] Shangbang Long, Xin He, and Cong Ya. Scene text detection and recognition: The deep learning era. *arXiv preprint arXiv:1811.04256*, 2018.
- [21] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *In Proceedings of European Conference on Computer Vision (ECCV)*, 2018.
- [22] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. Icdar 2003 robust reading competitions. In *null*, page 682. IEEE, 2003.
- [23] Anand Mishra, Karteek Alahari, and CV Jawahar. Scene text recognition using higher order language priors. In *BMVC-British Machine Vision Conference*. BMVA, 2012.
- [24] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. Recognizing text with perspective distortion in natural scenes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 569–576, 2013.
- [25] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41(18):8027–8048, 2014.
- [26] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

- [27] 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. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2017.
- [28] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4168–4176, 2016.
- [29] Baoguang Shi, Mingkun Yang, XingGang Wang, Pengyuan Lyu, Xiang Bai, and Cong Yao. Aster: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):855–868, 2018.
- [30] Bolan Su and Shijian Lu. Accurate scene text recognition based on recurrent neural network. In *Asian Conference on Computer Vision*, pages 35–48. Springer, 2014.
- [31] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [32] Zhuotao Tian, Michelle Shu, Pengyuan Lyu, Ruiyu Li, Chao Zhou, Xiaoyong Shen, and Jiaya Jia. Learning shape-aware embedding for scene text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4234–4243, 2019.
- [33] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011.
- [34] Fred L Bookstein Principal Warps. Thin-plate splines and the decompositions of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6), 1989.
- [35] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.
- [36] Fei Yin, Yi-Chao Wu, Xu-Yao Zhang, and Cheng-Lin Liu. Scene text recognition with sliding convolutional character models. *arXiv preprint arXiv:1709.01727*, 2017.
- [37] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [38] Fangneng Zhan, Shijian Lu, and Chuhui Xue. Verisimilar image synthesis for accurate detection and recognition of texts in scenes. 2018.
- [39] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: An efficient and accurate scene text detector. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.