Detecting Tampered Scene Text in the Wild

Yuxin Wang[1], Hongtao Xie[1(✉)], Mengting Xing[1], Jing Wang[2], Shenggao Zhu[2], and Yongdong Zhang[1]

[1] University of Science and Technology of China, Hefei, China
{wangyx58,htxie,metingx,zhyd73}@mail.ustc.edu.cn
[2] Huawei Cloud, Shenzhen, Guangdong, China
{wangjing105,zhushenggao}@huawei.com

Abstract. Text manipulation technologies cause serious worries in recent years, however, corresponding tampering detection methods have not been well explored. In this paper, we introduce a new task, named Tampered Scene Text Detection (TSTD), to localize text instances and recognize the texture authenticity in an end-to-end manner. Different from the general scene text detection (STD) task, TSTD further introduces the fine-grained classification, *i.e.* the tampered and real-world texts share a semantic space (text position and geometric structure) but have different local textures. To this end, we propose a simple yet effective modification strategy to migrate existing STD methods to TSTD task, keeping the semantic invariance while explicitly guiding the class-specific texture feature learning. Furthermore, we discuss the potential of frequency information for distinguishing feature learning, and propose a parallel-branch feature extractor to enhance the feature representation capability. To evaluate the effectiveness of our method, a new TSTD dataset (Tampered-IC13) is proposed and released at https://github.com/wangyuxin87/Tampered-IC13.

Keywords: Tampered scene text detection · Parallel-branch feature extractor · Deep learning

1 Introduction

As an important media for information transmission, scene text contains amounts of important and sensitive information [3,22,31,33]. With the development of text manipulation technologies [21,34,38], computers can automatically tamper with the important and sensitive content into fake information, being used in fraud, marketing or other illegal purposes. In contrast, methods for the tampered text detection field are currently blank. To fill such a research blank,

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19815-1_13.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
S. Avidan et al. (Eds.): ECCV 2022, LNCS 13688, pp. 215–232, 2022.
https://doi.org/10.1007/978-3-031-19815-1_13

Fig. 1. The visualization of TSTD task. TSTD methods need to firstly locate all the text regions and then determine whether the text has been tampered with. Left: original image; right: tampered image. Green box: real-world texts; red box: tampered texts. (Color figure online)

we propose a new task named Tampered Scene Text Detection (TSTD) in this paper. On the basis of face forgery detection task [12,17] that tampering detection approaches should not focus on only the tampered class, TSTD task needs to locate all the texts in scene images and determine whether the text has been tampered with (shown in Fig. 1).

TSTD task has two main challenges: 1) **Fine-grained perception.** The tampered and real-world texts share a semantic space (text position and geometric structure) but have local texture differences. As shown in Fig. 1, both tampered and real-world texts exist in the same position (*e.g.* board, bus, etc.) and have the identical geometric structure (*e.g.* horizontal/oriented posture and text shape), while tampered texts contain different local textures (*e.g.* smoothness) than real-world ones. Thus, TSTD methods need to maximize the discrimination of class-specific texture features while maintaining the semantic invariance. 2) **The limited size of high-quality annotated tampered text images.** At present, the results of existing text manipulation methods [34,38] are still a long way from practical application. In general, manual refinement in the post processing is necessary for the visualization improvement, which inevitably results in lots of human cost. Thus, how to construct a TSTD method with low data-dependency is necessary.

Firstly, to inherit the advantages (*e.g.* multi-scale text modeling [32,37], etc) of general scene text detection (STD) methods [43,44], we argue that TSTD methods should evolve from the STD approaches but not an entire new architecture. In this paper, we propose a Separating Segmentation while Sharing Regression (S3R) modification strategy to construct TSTD approaches based on existing STD ones. The S3R strategy follows the phenomenon that tampered and real-word texts only have local texture differences, but contain the same global semantics (text position and geometric structure). In the previous STD works, the pixel-level segmentation and distance regression show impressive performance to model local texture [33] and global semantics [37] respectively in STD task. Thus, the S3R strategy aims to maximize the discrimination of class-specific texture features while maintaining the semantic invariance by sharing/separating the segmentation/regression branches between tampered and real-world texts. On the one hand, S3R strategy separates the segmentation branches between tampered and real-world texts, and introduces a representation suppression loss L_{rs} for class-specific texture feature learning. On the

other hand, S3R strategy shares the regression branch between tampered and real-world texts to learn the invariant semantics (text position and geometric structure). The proposed S3R strategy can be effectively embedded into any current scene text detectors, migrating the general STD methods to TSTD task without introducing obvious speed decrease (detailed in Sect. 4.3).

Secondly, image manipulations are proved to leave high-frequency traces [17,41]. However, such variant high-frequency information is difficult to be captured in the RGB domain. Thus, the network needs amounts of tampered images for the better convergency on tampered textures, resulting in a high data-dependency. To this end, we introduce a parallel-branch feature extractor to capture the high-frequency information in frequency domain. Through aggregating the RGB and high-frequency features, our parallel-branch feature extractor can easily capture the high-frequency traces to assist the prediction and cause in low data-dependency (detailed in Sect. 4.4).

In addition, a new word-level tampered TSTD dataset named Tampered-IC13 is proposed (shown in Fig. 5). Tampered-IC13 is generated by tampering with the text in the most well-known scene text detection benchmark ICDAR2013 [10]. To the best of our knowledge, this is the first world-level TSTD dataset, which will greatly promote the development of TSTD task.

The contribution of this paper can be summarized as following three points:

- We introduce a new tampered scene text detection (TSTD) task to fill the research blank. Furthermore, the proposed united modification strategy (S3R) can be embedded into any current scene text detection methods, helping them to migrate to TSTD task without introducing obvious speed decrease.
- A parallel-branch feature extractor is constructed to capture both characteristics in RGB and frequency domains, which is first introduced in TSTD task. The exhaustive experiments prove its effectiveness in feature representation enhancement and data-dependency reduction.
- We construct a new word-level TSTD dataset (Tampered-IC13) to evaluate the effectiveness of our method, which is released publicly.

2 Related Work

2.1 Scene Text Detection

Scene text detection (STD) networks [27,35,37] contain two processes: text localization (TL) and geometric prediction (GP). TL process determines the position of text instances (*e.g.* center point/line) and GP process aims to accurately reconstruct the text regions (*e.g.* contour line). Previous methods [32,33] mainly divide the scene text detection methods based on the GP process, and classify detection methods to GP-REG and GP-SEG ones. Here, SEG means that pixel-level segmentation is utilized and REG refers to regressing the distance. As the TL process is also important in the detection process and has various implementations [26,40,43], we provide a more detailed classification of STD methods by taking the TL process into account: TL-SEG + GP-REG, TL-SEG + GP-SEG,

TL-REG + GP-SEG and TL-REG + GP-REG. In this section, we will detail the differences among these four-category methods respectively.

TL-SEG + GP-REG methods [23,27,37,43] use pixel-level segmentation to represent the text center location, *e.g.* shrinked polygons, while regressing the distance to the border for geometric prediction. EAST [43] predicts the shrinked polygon for text center localization to avoid the overlapping of adjacent texts. Then, distance regression to the four borders is utilized to reconstruct the text regions. Similarly, to further handle the arbitrary-shaped texts, MSR [37] regresses the distance from the closest border point to the center point. Thus, arbitrary-shaped polygon can be obtained by clustering all the border points. CentripetalText [23] proposes to use text kernels and centripetal shifts to present text instances. First, it generates shrinked polygons to coarsely locate text instances. The predicted centripetal shifts are then utilized to determine the boundaries of text instances. FCENet [44] models text instances in the Fourier domain via the Fourier transformation, and regresses compact Fourier signatures to represent the contour of arbitrary-shaped texts in GP process.

TL-SEG + GP-SEG approaches [26,28,29] regard scene text detection task as a pure pixel-level classification task. PSENet [28] predicts kernels with different scales for text reconstruction. To handle the adjacent text instances, the kernel map with minimum scale is used to separate adjacent texts, and other kernel maps constrain the geometric representation during the reconstruction process. To further simplify the reconstruction rules, PAN [29] only uses two different scales of kernel maps (shrinked polygon and full text region) for text localization and geometric prediction respectively, and proposes a low computational-cost segmentation head for real-time text detection. Similarly, Tian *et al.* [26] proposes a learnable post-processing (embedding feature) for accurate arbitrary-shaped text representation. To be specific, pixels of the full text are assigned to different text centers based on the pixel embedding distance.

TL-REG + GP-SEG methods [33,35,36] are mainly inspired by general two-stage [7,20] object detection methods, which firstly regress the coarse location of text region (*e.g.* text proposals). Then, pixel-level segmentation is implemented in the region of interests (RoIs) for geometric prediction. To handle the large-scale variance problem, ContourNet [33] firstly constructs an Adaptive-RPN to perceive the shape characteristics, which is supervised under the scale-invariance metric. Then, an orthogonal correction module is used to suppress the false-positive contour points. To further supplement the limited correction capability in 2D-space [33], LEMNet [36] distinguishes the false-positive samples in a high-level semantic dimension. Besides, SPCNet [35] introduces a semantic segmentation branch to enhance the feature representation capability and re-score the box confidence.

TL-REG + GP-REG methods [30,45] have the similar architecture to TL-REG + GP-SEG approaches. Instead of simply regarding geometric prediction as the pure segmentation task, these methods try to regress the text-specific geometric properties to handle the complex geometric variance. SLPR [45] uses slide lines to regress the outline points in horizontal and vertical directions respec-

tively. To iteratively refine the contour points, ATRR [30] uses RNNs [39] to regress the border points based on the region of interests (RoIs).

Special Cases. We discuss some specific cases that can not be easily distinguished: 1) Compared with TL process, GP process is usually more complex in scene text detectors. For example, LOMO [40] predicts both text segmentation and border distance for text reconstruction. Thus, the GP process in LOMO [40] can be regarded as a combination of GP-SEG and GP-REG. 2) For the detection methods constructed on the one-stage detection framework [14,15] without introducing an explicit TL process, we simply classify these methods based on the GP process.

2.2 Scene Text Editing and Tampering Detection

Scene text editing task aims to end-to-end tamper with text content in scene images. As deep learning becomes the most promising machine learning tool [2,4,13,42], scene text editing has achieved remarkable improvement in recent years. ETW [34] splits the text editing process into three sub-networks: text conversion network, background inpainting network and fusion network. The text transfer network learns to transform the style of input text image. Then, the background inpainting network erases the text content in the source image and reconstructs the background texture. Finally, the fusion network aggregates the style-transferred text images and text erased images to generate the final tampered sample. Based on ETW [34], SwapText [38] introduces TPS to handle the severe geometric distortion cases. To edit the specific character in text images, STEFANN [21] proposes a character-level text editing network. Though text manipulation technologies have been well developed in recent years, methods for tampered text detection field are almost blank. [1,11,18,24] regard tampered text detection task as a pure classification task, and the detection process is not included in their approaches. Benefiting from the gradient spread and shared features between detection and classification branches, the end-to-end detection framework is proved to be significant in both detection accuracy enhancement and model complexity reduction [16,19]. Thus, the end-to-end tampered text detection approaches need to be explored.

Although the face forgery detection methods achieve promising results in capturing tampered textures, it is impossible to directly use the face-specific texture learning (*e.g.* lips [6], eyes and nose [9]) to handle text-specific tampered samples. Thus, it is necessary to explore a text-specific tampering detection method for accurate tampered text detection.

3 Our Method

In this section, we firstly introduce the proposed S3R modification strategy in Sect. 3.1. Then, the parallel-branch feature extractor is detailed in Sect. 3.2. Finally, we introduce the proposed Tampered-IC13 dataset in Sect. 3.3.

3.1 The S3R Strategy

As tampered and real-word texts only have local texture differences but the same global semantics (appearance and geometric structure), the S3R strategy mainly focuses on one point: "*shall we separate the TL and GP processes of tampered text from the real-world one*?".

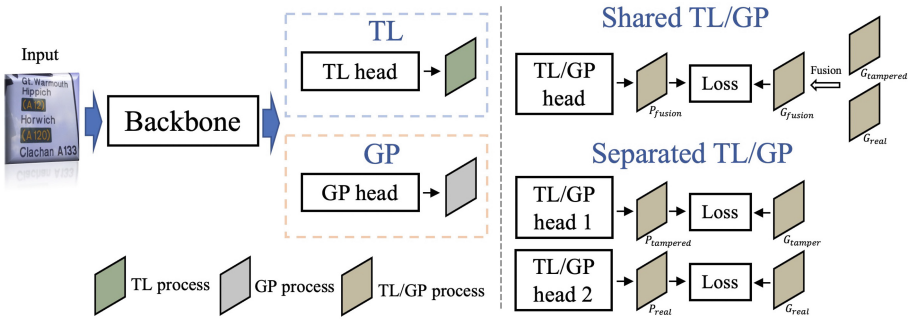

Fig. 2. The pipeline of our S3R strategy. TL is short for text localization process and GP is short for geometric prediction process. TL and GP process are parallel branches in [28,43], but are serial branches in [30,33]. P is short for prediction. $G_{tampered}$ and G_{real} are the ground truth of tampered and real-world texts respectively. $Fusion$ is the element-wise addition operation.

As shown in the right top of Fig. 2, the shared TL/GP process means that we aggregate the tampered and real-world ground truth in a single map and use one TL/GP head for prediction. In contrast, separated TL/GP (right bottom of Fig. 2) means that we use two independent heads to handle tampered and real-world texts respectively. In general, 1) for shared TL/GP process, we firstly aggregate the ground truth map of tampered texts $G_{tampered} \in R^{H\times W\times C}$ and real-world texts $G_{real} \in R^{H\times W\times C}$ to a single map $G_{fusion} \in R^{H\times W\times C}$ through element-wise addition. H, W and C mean height, width and the channel number respectively. Thus, the fused ground-truth map G_{fusion} guides the network to eliminate the class difference between tampered and real-world texts. Then, we use a single head to predict the shared feature map P_{fusion}, and calculate the loss between the P_{fusion} and G_{fusion}. 2) For separated TL/GP process, we simply use two independent heads for tampered ($P_{tampered}$) and real-world (P_{real}) feature map prediction. The final loss function is the sum of the losses in these two branches, calculated between $[P_{tampered}, G_{tampered}]$ and $[P_{real}, G_{real}]$ respectively.

In S3R strategy, we use the separated TL/GP process for segmentation, and use shared TL/GP process for regression. Specially, in order to guide the class-specific texture learning in the segmentation branch, we introduce a representation suppression loss L_{rs} in separated segmentation branches, which is formulated in Eq. (1). $L_{tampered,real}$ is the loss to suppress the representation

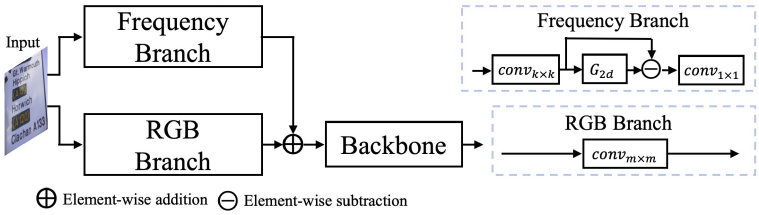

Fig. 3. The structure of our parallel feature extractor. *conv* is the convolutional layer. G_{2d} is the 2d Gaussian kernel.

of tampered prediction in the real-world segmentation branch, which punishes the tampered pixels activated in the real-world segmentation map. Similarly, $L_{real,tampered}$ suppresses the representation of real-world prediction in tampered segmentation branch. As the pixel-level segmentation and distance regression shows an impressive performance in modeling local texture [33] and global semantics [37], S3R helps the model to explicitly classify class-aware textures while keeping semantic invarance (*e.g.* text appearance, textual shape) between tampered and real-world texts (detailed in Sect. 4.3).

$$L_{rs} = L_{tampered,real} + L_{real,tampered} \tag{1}$$

3.2 The Parallel-branch Feature Extractor

Though image manipulations leave high-frequency traces [17,41] in the tampered image, it is difficult for the network to capture the variant high-frequency information in the RGB domain. According to this observation, we propose a parallel-branch feature extractor with the goal of considering the characteristics in both RGB and frequency domain. As shown in Fig. 3, our parallel-branch feature extractor contains two parts: frequency branch and RGB branch. Firstly, the input image is processed by these two branches respectively. Then, the information captured from two branches is fused by element-wise addition. Finally, the aggregated features are sent to the backbone.

For the frequency branch, inspired from [17,41], we adopt Laplacian of Gaussian (LoG) to capture high-frequency information. For input image I, a convolutional layer with size $k \times k$ is firstly utilized for feature enhancement. The value of k represents how much information can be perceived in the frequency branch, which is an ablation study in our experiments. Then, a 2d Gaussian kernel is used to smooth the features. Finally, a skip-connection and a 1×1 convolutional layer are used for generating high-frequency information and dimension alignment respectively. The detailed formulation of frequency branch is shown in Eq. (2):

$$F_{fre} = w_{1\times 1}(x - w_g x) \tag{2}$$

To be specific, w_g is the 2d Gaussian kernel, $w_{1\times 1}$ means the convolutional layer with size 1×1. $x = w_{k\times k} I$. I is the input image and $w_{k\times k}$ is the convolutional

layer with size $k \times k$. For RGB branch, we utilize an $m \times m$ convolutional layer to generate RGB information. In the real application, we use the first convolutional layer in the backbone to replace our RGB branch. For example, m is set to 3 for VGG16 [25] and 7 for ResNet50 [8] respectively.

Fig. 4. The visualization of feature maps in parallel-branch feature extractor. (a): input image; (b): the feature map in frequency branch; (c) the feature map in RGB branch.

Fig. 5. Some examples in Tampered-IC13. Left: original images; right: our tampered images. The tampered texts contain a high similarity to the real-world ones.

To better understand how frequency information helps the network for prediction, we visualize the extracted features from frequency branch and RGB branch. We normalize each map to [0, 1] for better visualization. As shown in Fig. 4, different from RGB branch mainly focuses on the text content in the RGB domain, the frequency branch effectively captures the high-frequency characteristics in the outline areas. By fusing the features captured from both frequency and RGB branches, the parallel-branch feature extractor is able to learn distinguishing features between tampered and real-world texts.

3.3 Tampered-IC13 Dataset

The difficulty of the TSTD task lies in how the network can better distinguish the tampered class from the real class, which puts higher requirements on the data generation process to ensure the texture consistency and background integrity in the tampered region. Considering that ICDAR2013 (IC13) dataset is one of

the most well-known datasets in scene text detection community during the past five years, we use IC13 dataset as the benchmark data to generate our tampered dataset. Some examples of Tampered-IC13 are shown in Fig. 5. The details of Tampered-IC13 dataset are available in the supplementaries.

Dataset Challenge. Benefiting from the sophisticated tampering process and effective later interventions, the proposed Tampered-IC13 achieves high-quality tampered text textures, consistent text-free areas and smooth transmission in the contour regions. Thus, it is quite a challenge to achieve the accurate detection on Tampered-IC13 and the detection results on Tampered-IC13 can well reflect the performance of TSTD detectors.

4 Experiment

In this section, we firstly introduce the evaluation metrics and implementation details in Sect. 4.1 and Sect. 4.2 respectively. Next, we evaluate the effectiveness of S3R strategy in Sect. 4.3. Finally, the performance of parallel-branch feature extractor and some discussions are shown in Sect. 4.4 and Sect. 4.5.

4.1 Evaluation Metric

Following the general scene text detection methods, we adopt precision (P), recall (R) and F-measure (F) to evaluate the detection results of tampered and real-world texts. To evaluate the average performance between tampered and real-world detection results, we propose a new mean F-measure (mF), which is inspired by mAP [20] Eq. (4). $F_{tampered}$ and F_{real} are F-measure calculated by Eq. (3) for tampered and real-world texts respectively. In order to treat tampered and real-world results equally, we set $\lambda_1, \lambda_2 = 1$.

$$F = 2 \times P \times R/(P + R) \tag{3}$$

$$mF = (\lambda_1 F_{tampered} + \lambda_2 F_{real})/2 \tag{4}$$

4.2 Implementation Details

We conduct the experiments on four well-known models to evaluate the effectiveness of our method. To be specific, EAST [43] (TL-SEG + GP-REG), PSENet [28] (TL-SEG + GP-SEG), ContourNet [33] (TL-REG + GP-SEG) and ATRR [30] (TL-REG + GP-REG) are used in our experiments. Specially, we simplify the prediction head in ATRR [30] and use a convolutional layer to replace RNNs [39] to improve its generalization. All the models are constructed based on the publicly released code[1,2,3,4]. As there is not a published code in ATRR [30], we

[1] https://github.com/SakuraRiven/EAST.
[2] https://github.com/whai362/PSENet.
[3] https://github.com/wangyuxin87/ContourNet.
[4] https://github.com/facebookresearch/maskrcnn-benchmark.

reconstruct the model based on the structure of Mask-RCNN [7]. The training settings follow the details in the corresponding paper. The training and testing sets in Tampered-IC13 are used to train and evaluate our model respectively. During the inference, we resize the image to 748×748, 736×736, 1200×2000 and 1200×2000 for EAST, PSENet, ContourNet and ATRR respectively, where each size reflects the best performance in the corresponding model.

Table 1. The detection performance of different modification strategies. Sep and Sha are shorts for separated and shared. TL and GP are the TL process and GP process. SEG+REG, SEG+SEG, REG+SEG and REG+REG mean TL-SEG+GP-REG, TL-SEG+GP-SEG, TL-REG+GP-SEG and TL-REG+GP-REG respectively. R-T, P-T and F-T are the recall, precision and F-measure of tampered detection results. R-R, P-R and F-R are the recall, precision and F-measure of real-world detection results. mF is the mean F-measure. To better understand the table, we set "separated" (Sep) and "segmentation" (SEG) to red, and set "shared" (Sha) and "regression" (REG) to blue.

	TL		GP		Accuracy						
	Sha	Sep	Sha	Sep	R-T	P-T	F-T	R-R	P-R	F-R	mF
EAST [43] (SEG+REG)	✓		✓		68.68	70.11	69.39	23.18	46.61	31.39	50.39
		✓	✓		**69.97**	**70.23**	**69.94**	**27.32**	**50.46**	**35.45**	**52.70**
		✓		✓	53.97	31.74	39.97	15.40	40.97	22.38	31.18
PSENet [28] (SEG+SEG)	✓			✓	77.39	77.24	77.31	38.25	51.45	43.88	60.60
		✓		✓	**79.43**	**79.92**	**79.67**	**41.89**	**61.56**	**49.85**	**64.76**
		✓	✓		65.58	53.76	59.08	21.69	12.46	15.83	37.46
ContourNet [33] (REG+SEG)	✓		✓		91.24	85.33	88.19	54.77	76.32	63.77	75.98
		✓	✓		79.23	75.39	77.26	46.85	66.75	55.06	66.16
	✓			✓	**91.45**	**86.68**	**88.99**	**54.80**	**77.88**	**64.33**	**76.66**
ATRR [30] (REG+REG)	✓		✓		**90.63**	**84.60**	**87.52**	**54.63**	**76.74**	**63.83**	**75.68**
	✓			✓	90.43	83.77	86.97	52.15	74.82	61.46	74.22
		✓	✓		86.97	72.87	79.29	51.29	66.93	58.08	68.69

4.3 The Evaluation of S3R Strategy

The detailed modifications to four models in Table 1 are available in the supplementary materials. As shown in Table 1, we summarize two conclusions: 1) Separating the SEG branch benefits the class-specific texture learning between tampered and real-world texts. For example, EAST, PSENet and ContourNet obtains 2.31%(52.7% vs 50.39%), 27.30% (64.76% vs 37.46%) and 0.68% (76.66% vs 75.98%) improvement in mF respectively. 2) Sharing the REG branches helps the learning of invariant semantics. For example, when we shared the GP process in EAST, there exists 21.52% improvement in the mF (52.70% vs 31.18%). Based on the above two conclusions, our S3R strategy shows significance in both

tampered and real-world text detection. As shown in Table 1, the model implemented with S3R strategy (separating segmentation while sharing regression) outperforms other modification strategies by a large margin. To be specific, the S3R strategy helps EAST [43], PSENet [28], ContourNet [33] and ATRR [30] to obtain 52.70%, 64.76%, 76.66% and 75.68% in mF respectively.

How Does S3R Strategy Achieve Class-specific Texture Learning in Separated Segmentation? We conduct an additional experiment to illustrate how does S3R achieve the class-specific texture learning. Specially, ContourNet [33] (TL-REG + GP-SEG) is used in this experiment. The results in Table 2 illustrate several conclusions: 1) Simply constructing the separated segmentation branches without representation suppression (L_{rs} in Eq. (1)) between two branches is even harmful to performance (74.18% of Separated GP vs 75.98% of Shared GP in mF). Thus, such an implicit separated structure has no capability to learn class-specific texture features, and the extra introduced parameters further increase the network learning difficulty. 2) By implementing a shared segmentation head and only separating the segmentation map in the last convolutional layer with the representation suppression, it obtains a slight improvement in mF (76.16% of Shared GP + Sup vs 75.98% of Shared GP). The implementation of Shared GP + Sup is similar to Softmax loss, predicting multi-class segmentation maps and suppressing the representation between each other. 3) When we separate the segmentation head and further use the representation suppression between two heads (Separated GP + Sup), the model obtains the best results (76.66% in mF). Compared with separating only the last convolutional layer (Shared GP + Sup), the separated segmentation heads have more powerful capability for class-specific texture learning. Furthermore, such explicit separated structure also benefits the model convergency (compared with Separated GP). 4) As EAST [43] and PSENet [28] segment the text regions in the entire image rather than RoI-based prediction (*e.g.* shared-TL sends class-independent proposals to both tampered and real-world heads), the ground-truth segmentation maps in separated branches are mutually exclusive. Thus, the natural representation suppression in separated segmentation branches of EAST [43] and PSENet [28] helps the model to learn class-specific texture features (Table 1). Based on above analyses, the separated structure and representation suppression between

Table 2. The evaluation of S3R strategy in class-specific texture learning. Sup is short for representation suppression. GP is short for geometric prediction process.

		Accuracy						
Method	Implementation	R-T	P-T	F-T	R-R	P-R	F-R	mF
ContourNet [33]	Shared GP	91.24	85.33	88.19	54.77	76.32	63.77	75.98
	Separated GP	86.35	85.66	86.00	52.12	77.62	62.36	74.18
	Shared GP + Sup	91.33	85.27	88.20	**54.80**	77.29	64.12	76.16
	Separated GP + Sup	**91.45**	**86.68**	**88.99**	**54.80**	**77.88**	**64.33**	**76.66**

two branches together promote the class-specific texture learning, and improve the detection performance to a new level. Specially, the implementation details of L_{rs} in ContourNet [33] are shown in the supplementaries.

Why is It Important to Maintain Semantic Invariance? We infer that the shared feature learning of global semantics helps the TSTD network converge. As the distance regression is proved to perform well in modeling the text position [33] and geometric structure [37], independently modeling these class-invariant semantics will not introduce new information and the additionally introduced parameters are harmful to the network convergency.

Table 3. The comparison of testing speed. As the original ATRR [30] shares TL and GP process, S3R strategy does not introduce extra computations.

Method	Original (FPS)	Modified (FPS)
EAST [43]	6.7	5.8
PSENet [28]	8.9	7.4
ATRR [30]	3.2	3.2
ContourNet [33]	3.7	3.6

Table 4. The comparison of parallel-branch feature extractor implemented with different k. Specially, EAST is pre-trained on the SynthText [5] for a better convergency.

	Accuracy						
kernel size ($k \times k$)	R-T	P-T	F-T	R-R	P-R	F-R	mF
5×5	74.75	72.39	73.55	41.39	**64.77**	50.50	60.03
7×7	**75.15**	**73.21**	**74.17**	**44.04**	62.74	**51.75**	**62.96**
9×9	74.74	72.37	73.55	43.38	61.93	51.10	62.33

The Influence in Testing Speed. We conduct several experiments to compare the testing speed between original and our modified models. As shown in Table 3, our S3R strategy only introduces a slight speed decrease to the original model.

4.4 The Effectiveness of Parallel-branch Feature Extractor

The Evaluation of k. We conduct several experiments to study the relationship between the k and the detection performance. The value of k determines how much information can be perceived in the frequency branch. As shown in Table 4, the kernel with size 7×7 obtains the best results. Thus, we set kernel size to 7×7 in the later experiments.

The Effectiveness in Performance Boosting. We embed parallel-branch feature extractor into existing detection methods to evaluate its effectiveness. As shown in Table 5, our parallel-branch feature extractor is able to improve both tampered and real-world text detection results. The relative improvement for EAST [43] and ATRR [28] are 2.57% and 0.67% in mF respectively. We summarize the impressive improvement to that the high-frequency information effectively assists the network to learn distinguishing features between tampered and real-world texts. More experiments conducted on other models are available in the supplementary materials.

Table 5. The detection results of models with/without introducing parallel-branch feature extractor. Specially, EAST is pre-trained on the SynthText [5] for a better convergency. The evaluation on more models are available in the supplementary materials.

		Accuracy						
	Frequency	R-T	P-T	F-T	R-R	P-R	F-R	mF
EAST [43]	–	74.54	70.25	72.33	40.23	60.90	48.45	60.39
	✓	**75.15**	**73.21**	**74.17**	**44.04**	**62.74**	**51.75**	**62.96**
ATRR [30]	–	90.63	84.60	87.52	54.63	76.74	63.83	75.68
	✓	**90.84**	**86.10**	**88.40**	**55.13**	**77.08**	**64.29**	**76.35**

Table 6. The evaluation of parallel-branch feature extractor in data dependency reduction. EAST [43] is used in this experiment. To ensure a better convergency, SynthText [5] is used to pre-train the model. $Half$ means that we only use a half of images in the training set.

		Accuracy							
Frequency	Data	R-T	P-T	F-T	R-R	P-R	F-R	mF	↓mF
–	Full	74.54	70.25	72.33	40.23	60.90	48.45	60.39	–
	Half	72.91	68.71	70.75	37.09	57.14	44.98	57.87	2.52
✓	Full	75.15	73.21	74.17	44.04	62.74	51.75	62.96	–
	Half	73.11	71.80	72.45	41.48	59.39	49.12	60.79	**2.17**

The Effectiveness in Data-dependency Reduction. We further reduce the training images to evaluate our effectiveness in data-dependency reduction. As show in Table 6, we summarize two conclusions: 1) Model implemented with our parallel-branch feature extractor obtains less performance decrease with fewer training images (2.17% vs 2.52% decrease in mF). 2) Compared with the model without implementing parallel-branch feature extractor, our method can obtain

comparable even better detection performance with only a half of training images (60.79% vs 60.39% in mF). Based on above analyses, our parallel-branch feature extractor effectively reduces the data dependency of the network.

4.5 Discussion

The Generalization on Detecting Low-quality Images. As images on the web have different quality, we think it is necessary to evaluate the generalization of TSTD methods on detecting low-quality images. Specifically, we use the ffmpeg compression algorithm to reduce the image quality. Details are available in the supplementaries.

The Qualitative Analysis. We visualize some detection results in Fig. 6. From the first and second rows, we find that the modified four models can effectively handle most cases in TSTD task. We further provide some failure cases to discuss the limitation of these methods. As shown in the third row of Fig. 6, methods implemented with TL-SEG process (EAST [43] and PSENet [28]) fail to handle the texts with extreme ratios (word "management" and string "002101"). We infer that the simplicity of these two methods makes them difficult to consider the long-range geometric structure [27].

Fig. 6. The visualization of detection results from four models. From left to right: ground truth, EAST [43], PSENet [28], ATRR [30], ContourNet [33]. Red boxes: tampered texts; green boxes: real-world texts. (Color figure online)

Limitation. In this paper, we focus on only the word-level tampered text detection and propose the relatively word-level detection method. The character-level and line-level tampered cases are not included in this paper. As the first work for TSTD task, we think that our word-level detection approach also gives lots of insights to TSTD community. Furthermore, we believe the methods proposed in this paper can also be used in character-level and line-level tampered text detection methods, *e.g.* the S3R strategy and parallel-branch feature extractor.

5 Conclusion

This paper introduces a new task, named Tampered Scene Text Detection (TSTD), to localize text instances and recognize the texture authenticity. We propose a unified modification (S3R) strategy to migrate the general STD method to TSTD task while keeping high detection performance and inference speed. The S3R strategy successfully maintains the semantic invariance and explicitly guides the class-specific texture feature learning between tampered and real-world texts. Furthermore, a parallel-branch feature extractor is constructed for the feature representation capability enhancement and data-dependency reduction. The exhaustive experiments on the proposed Tampered-IC13 demonstrate the effectiveness of our methods, and will give lots of insights to the TSTD community.

Acknowledgement. This work is supported by the National Nature Science Foundation of China (62121002, 62022076, U1936210), the Fundamental Research Funds for the Central Universities under Grant WK3480000011, the Youth Innovation Promotion Association Chinese Academy of Sciences (Y2021122). We acknowledge the support of GPU cluster built by MCC Lab of Information Science and Technology Institution, USTC.

References

1. Bibi, M., Hamid, A., Moetesum, M., Siddiqi, I.: Document forgery detection using printer source identification-a text-independent approach. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), vol. 8, pp. 7–12. IEEE (2019)
2. Du, Y., et al.: SVTR: scene text recognition with a single visual model. In: IJCAI (2022)
3. Fang, S., Xie, H., Wang, Y., Mao, Z., Zhang, Y.: Read like humans: autonomous, bidirectional and iterative language modeling for scene text recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7098–7107 (2021)
4. Ge, J., Xie, H., Min, S., Zhang, Y.: Semantic-guided reinforced region embedding for generalized zero-shot learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 1406–1414 (2021)
5. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2315–2324 (2016)
6. Haliassos, A., Vougioukas, K., Petridis, S., Pantic, M.: Lips don't lie: a generalisable and robust approach to face forgery detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5039–5049 (2021)
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

9. Hu, Z., Xie, H., Wang, Y., Li, J., Wang, Z., Zhang, Y.: Dynamic inconsistency-aware deepfake video detection. In: IJCAI (2021)
10. Karatzas, D., et al.: ICDAR 2013 robust reading competition. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1484–1493. IEEE (2013)
11. Kundu, S., Shivakumara, P., Grouver, A., Pal, U., Lu, T., Blumenstein, M.: A new forged handwriting detection method based on fourier spectral density and variation. In: Palaiahnakote, S., Sanniti di Baja, G., Wang, L., Yan, W.Q. (eds.) ACPR 2019. LNCS, vol. 12046, pp. 136–150. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41404-7_10
12. Li, J., Xie, H., Li, J., Wang, Z., Zhang, Y.: Frequency-aware discriminative feature learning supervised by single-center loss for face forgery detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6458–6467 (2021)
13. Li, P., Li, Y., Xie, H., Zhang, L.: Neighborhood-adaptive structure augmented metric learning. In: AAAI (2022)
14. Liao, M., Shi, B., Bai, X.: Textboxes++: a single-shot oriented scene text detector. IEEE Trans. Image Process. **27**(8), 3676–3690 (2018)
15. Liao, M., Shi, B., Bai, X., Wang, X., Liu, W.: Textboxes: a fast text detector with a single deep neural network. In: Thirty-first AAAI Conference on Artificial Intelligence (2017)
16. Liu, Y., Chen, H., Shen, C., He, T., Jin, L., Wang, L.: Abcnet: real-time scene text spotting with adaptive bezier-curve network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9809–9818 (2020)
17. Masi, I., Killekar, A., Mascarenhas, R.M., Gurudatt, S.P., AbdAlmageed, W.: Two-branch recurrent network for isolating deepfakes in videos. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12352, pp. 667–684. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58571-6_39
18. Nandanwar, L., et al.: Forged text detection in video, scene, and document images. IET Image Process. **14**(17), 4744–4755 (2020)
19. Qiao, L., et al.: Mango: a mask attention guided one-stage scene text spotter. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 2467–2476 (2021)
20. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. **28** (2015)
21. Roy, P., Bhattacharya, S., Ghosh, S., Pal, U.: STEFANN: scene text editor using font adaptive neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13228–13237 (2020)
22. Sheng, F., Chen, Z., Xu, B.: NRTR: a no-recurrence sequence-to-sequence model for scene text recognition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 781–786. IEEE (2019)
23. Sheng, T., Chen, J., Lian, Z.: Centripetaltext: an efficient text instance representation for scene text detection. Adv. Neural Inf. Process. Syst. **34**, 335–346 (2021)
24. da Silva Barbosa, R., Lins, R.D., De Lira, E.D.F., Camara, A.C.A.: Later added strokes or text-fraud detection in documents written with ballpoint pens. In: 2014 14th International Conference on Frontiers in Handwriting Recognition, pp. 517–522. IEEE (2014)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

26. Tian, Z., Shu, M., Lyu, P., Li, R., Zhou, C., Shen, X., Jia, J.: Learning shape-aware embedding for scene text detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4234–4243 (2019)
27. Wang, P., et al.: A single-shot arbitrarily-shaped text detector based on context attended multi-task learning. In: Proceedings of the 27th ACM International Conference on Multimedia, pp. 1277–1285 (2019)
28. Wang, W., Xie, E., Li, X., Hou, W., Lu, T., Yu, G., Shao, S.: Shape robust text detection with progressive scale expansion network. In: CVPR, pp. 9336–9345 (2019)
29. Wang, W., et al.: Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8440–8449 (2019)
30. Wang, X., Jiang, Y., Luo, Z., Liu, C.L., Choi, H., Kim, S.: Arbitrary shape scene text detection with adaptive text region representation. In: CVPR, pp. 6449–6458 (2019)
31. Wang, Y., Xie, H., Fang, S., Wang, J., Zhu, S., Zhang, Y.: From two to one: a new scene text recognizer with visual language modeling network. In: ICCV, pp. 14194–14203 (2021)
32. Wang, Y., Xie, H., Fu, Z., Zhang, Y.: DSRN: a deep scale relationship network for scene text detection. In: IJCAI, pp. 947–953 (2019)
33. Wang, Y., Xie, H., Zha, Z.J., Xing, M., Fu, Z., Zhang, Y.: Contournet: taking a further step toward accurate arbitrary-shaped scene text detection. In: CVPR, pp. 11753–11762 (2020)
34. Wu, L., Zhang, C., Liu, J., Han, J., Liu, J., Ding, E., Bai, X.: Editing text in the wild. In: ACM MM, pp. 1500–1508 (2019)
35. Xie, E., Zang, Y., Shao, S., Yu, G., Yao, C., Li, G.: Scene text detection with supervised pyramid context network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 9038–9045 (2019)
36. Xing, M., et al.: Boundary-aware arbitrary-shaped scene text detector with learnable embedding network. IEEE Trans. Multimedia **24**, 3129–3143 (2021)
37. Xue, C., Lu, S., Zhang, W.: Msr: multi-scale shape regression for scene text detection. arXiv preprint arXiv:1901.02596 (2019)
38. Yang, Q., Huang, J., Lin, W.: Swaptext: image based texts transfer in scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14700–14709 (2020)
39. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
40. Zhang, C., Liang, B., Huang, E.A.: Look more than once: an accurate detector for text of arbitrary shapes. In: CVPR, pp. 10552–10561 (2019)
41. Zhang, X., Karaman, S., Chang, S.F.: Detecting and simulating artifacts in GAN fake images. In: 2019 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1–6. IEEE (2019)
42. Zheng, T., Chen, Z., Fang, S., Xie, H., Jiang, Y.G.: Cdistnet: Perceiving multi-domain character distance for robust text recognition. arXiv preprint arXiv:2111.11011 (2021)
43. Zhou, X., et al.: East: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 5551–5560 (2017)

44. Zhu, Y., Chen, J., Liang, L., Kuang, Z., Jin, L., Zhang, W.: Fourier contour embedding for arbitrary-shaped text detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3123–3131 (2021)
45. Zhu, Y., Du, J.: Sliding line point regression for shape robust scene text detection. In: ICPR, pp. 3735–3740. IEEE (2018)