Detecting Tampered Scene Text in the Wild (Supplementary Materials)

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
[2] Huawei Cloud
{wangyx58,metingx}@mail.ustc.edu.cn, {htxie,zhyd73}ustc.edu.cn, {wangjing105,zhushenggao}@huawei.com

1 The Detailed Modifications to Four-category Methods

The training objective of these four-category methods (EAST [15], PSENet [9], ContourNet [12] ATRR [11]) is defined in Eq. (1), where L_{tl} is the loss for text localization (TL) process and L_{gp} is the loss for geometric prediction (GP) process. Without special introduction, we use the loss function in the corresponding paper to formulate L_{tl} and L_{gp}.

$$L = L_{tl} + L_{gp} \tag{1}$$

1.1 The Modification details

The Modification to TL-SEG + GP-REG method EAST [15] constructs a parallel branch. The segmentation branch predicts the shrunk polygons (TL process) and the regression branch regresses the distance to four borders (GP process).

Modification Details. 1) Shared TL process. Firstly, we fuse the ground-truth shrink polygon maps of tampered and real-world texts through element-wise addition. Then, we construct a single segmentation branch for fused shrink polygon map prediction. Thus, the segmentation branch only locates the text regions without classifying the tampered and real-world texts. Finally, an embedding branch is introduced for classification. The embedding branch learns pixel-level embedding features in the training stage. To be specific, we update the mean of tampered and real-world embedding features in the training stage to represent these two classes. In the testing stage, we classify the tampered and real-world texts through calculating the distance from shrink polygons to the two-class mean embedding features. 2) Separated TL process. We use two different segmentation branches to predict the shrunk polygon maps of tampered and real-world texts respectively. 3) Shared GP process. Similarly, we firstly fuse the ground-truth distance regression maps of tampered and real-world texts through element-wise addition. Then, a single distance regression branch is implemented

⋆ H. Xie is the corresponding author.

to rebuild both tampered and real-world texts. 4) Separated GP process. We use two different regression branches to predict the distance regression maps of tampered and real-world texts respectively.

Training Details. 1) Shared TL process. In this implementation, $L_{tl} = L_{sp} + L_{em}$, where L_{sp} is the loss for fused shrink polygon map, and L_{em} is the embedding loss used in PAN [10]. 2) For separated TL/GP process, we firstly double the original prediction branches to generate the prediction for tampered and real-world texts respectively. Then we add the losses from both branches to train the network. 3) For shared GP process, the fused ground-truth distance map is used to guide the geometric learning.

The Modification to TL-SEG + GP-SEG method PSENet [9] predicts shrink polygons with different scales in TL and GP process. Specially, the shrink polygon map with minimum scale provides the coarse localization and the shrink polygon with other scales determines the final geometric information of text regions.

Modification Details. 1) For shared TL/GP process, we combine the corresponding ground-truth shrink polygon maps of the tampered and real-world texts to the single shrink polygon map (using element-wise addition), and use a single segmentation head for prediction. 2) In contrast, for separated TL/GP process, we use two different segmentation heads to predict the corresponding shrink polygon maps of tampered and real-world texts respectively.

Training Details. 1) For shared TL/GP process, the loss is calculated between the corresponding fused ground-truth shrink polygon maps and predicted shrink polygon maps. 2) For separated TL/GP process, the loss function of TL/GP process consists of two parts: $T_{tl/gp} = T_{tl/gp,tamper} + T_{tl/gp,real}$, where $T_{tl/gp,tamper}$ and $T_{tl/gp,real}$ are losses for tampered and real-world texts respectively in TL/GP process.

The Modification to TL-REG + GP-SEG method. As a method inspired from Mask R-CNN [4], ContourNet [12] contains two parts: region proposal network (RPN) and segmentation head. Firstly, in the TL process, RPN coarsely provides region of interests (RoIs). Then, the segmentation head predicts the contour points for geometric prediction.

Modification Details. 1) Shared TL process. As RPN naturally has the capability for multi-class prediction, we simply modify the output channel in the classification layer and use a same proposal regression layer for both tampered and real-world texts. 2) Separated TL process. We use two independent RPNs to localize tampered and real-world texts respectively. Specially, each RPN only needs to classify two classes (the background class + tampered/real-world class), and only calculate the regression loss of class-matched proposals in the related RPN. 3) Shared GP process. We simply use a single segmentation head to predict the contour points of both tampered and real-world texts. 4) Separated GP process. We use two independent segmentation heads to predict the contour points for tampered and real-world texts respectively.

Training Details. 1) For shared TL process, a simple RPN loss is used to formulate L_{tl}. 2) For separated TL process, $L_{tl} = L_{rpn,tamper} + L_{rpn,real}$. $L_{rpn,tamper}$ and $L_{rpn,real}$ are RPN loss [4] for tampered and real-world texts respectively. 3) For shared GP process, L_{gp} only focuses on whether the pixel is a contour point without considering the class difference between tampered and real-world texts. 4) For separated GP process, $L_{gp} = L_{gp,tamper} + L_{gp,real} + 0.1 * L_{rs}$. $L_{gp,tamper}$ and $L_{gp,real}$ are class-balanced cross-entropy losses [12] for tampered segmentation head and real-world segmentation head respectively. To explicitly distinguish the characteristics of tampered and real-world texts, L_{rs} aims to suppress the activation of real-world contour points in tampered segmentation head, and activation of tampered contour points in real-world segmentation head. To be specific, L_{rs} is a cross-entropy loss and guides the tampered segmentation head to predict real-world contour points as the background class. Similarly, L_{rs} also guides the real-world segmentation head to predict tampered contour points as the background class. In the testing stage, the class-specific contour points are used to represent tampered or real-world texts respectively.

The Modification to TL-REG + GP-REG method To simplify the structure of ATRR [11] and improve its generalization, we use several convolutional layers to replace the original RNN-based prediction layer. To be specific, we firstly reduce the size of RoIs from $H \times W$ to $1 \times W$. Then, we use two convolutional layers with channel $W + 1$ to regress the distance to top and bottom contour points in each column. Specially, the additional channel means that no contour points exist in the current column. During the inference, we sequentially link the contour points to reconstruct the text instances.

Modification Details. For shared and separated TL process, the modification of ATRR [11] is identical to previous detailed ContourNet [12]. For GP process, 1) shared GP process means that we use a single regression head to predict the distance to contour points for both tampered and real-world texts, while 2) in the separated GP process, we construct two independent regression heads for tampered and real-world texts respectively. Specially, we find that using L_{rs} in separated GP brings no improvement to the performance. As the distance regression aims to perceive the global semantic information (*e.g.* shape of texts), the identical semantics do not introduce new information. Thus, we simply fuse the loss of tampered and real-world regression heads to formulate L_{gp} without implementing L_{rs}.

1.2 The S3R Strategy on Four-category Methods

We visualize the modified structure of four-category methods under S3R strategy in Fig. 1. To be specific, 1) we use the $Head1$ and $Head2$ for TL prediction in EAST [15], which use shrink polygon maps to represent real-world and tampered texts respectively. $Head3$ is constructed to predict the geometric information of both real-world and tampered texts (including four distance and one angle maps). 2) For PSENet [9], we directly construct two heads to predict the shrink

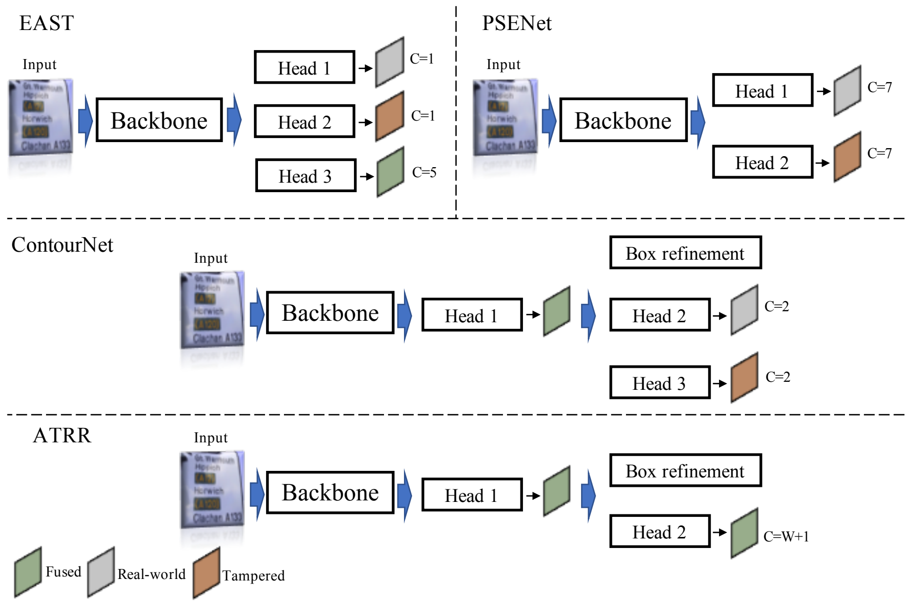

Fig. 1. The modified structure of four-category methods under S3R strategy. C is the channel of output features. W is the width of the input RoI features. Box refinement is used to optimize the proposals [11].

polygon maps of different scales (seven scales) for real-world and tampered texts respectively. 3) For ContourNet [12], we use $Head1$ (Region Proposal Network) for both real-world and tampered texts to predict location information. $Head2$ and $Head3$ are used to predict the contour points in two orthogonal directions for real-world and tampered texts respectively. 4) For ATRR [11], we share the $Head1$ and $Head2$ for both real-world and tampered texts to predict location and geometric information respectively.

2 Tampered-IC13 Dataset

Dataset Acquisition 1) The generation of paired tampered texts. We integrated a total number of 1551 text instances with more than 1 character from the IC13 training (737) and testing set (814), and these 1551 texts are regarded as source texts. The content of paired tampered texts is word with the opposite meaning or the similar number of characters as the source texts. 2) Tampering operation. Compared with traditional methods, deep generative methods require much less human cost in the real application, being welcome in the era of big data. SRNet [13] is used in our text tampering process. To train a powerful tampering network, we generate 50k paired synthetic training samples based on the prepared source-tampered word pairs. SRNet [13] is trained using 1 2080Ti

Table 1. The detection results of EAST with/without pre-training.

	TL	Accuracy						
	Pre-train	R-T	P-T	F-T	R-R	P-R	F-R	mF
	-	69.97	70.23	69.94	27.32	50.46	35.45	52.70
EAST	✓	**74.54**	**70.25**	**72.33**	**40.23**	**60.90**	**48.45**	**60.39**

GPU for 10w iterations. Finally, we use ground-truth bounding box annotations to crop the text patches from IC13 images, and use our well-trained SRNet to generate tampered text patches. 3) Refinement. Firstly, the coarse tampered results are generated by placing tampered text patches back to the original image. Considering that some hard samples obtain low-quality visualization, then, we use photoshop (PS) software to refine the results. In the end, a total of 507 out of 849 text instances in the training set and 488 out of 1095 text instances in the test set are marked as tampered classes.

Diversity. The diversity of the dataset is reflected in the following two aspects. 1) The diversity of text styles. Due to the different scenes in which the texts in Tampered-IC13 are distributed, correspondingly, the text fonts are also more diverse. In addition, the text instances in the dataset range from book subscript texts to large advertising slogans, showing a variety of font sizes. 2) The diversity of background texture. There are a large number of complex textures such as fences and grass in the background area in the dataset, which poses a greater challenge to the discriminative ability of the model.

3 Experiment

3.1 The Effectiveness of Pre-training

As the most common trick in STD methods for performance boosting, we claim that *"the pre-training trick is also suitable for TSTD task"*. To be specific, we firstly use SynthText [3] to pre-train EAST [15] and then finetune the model on Tampered-IC13. As the tampered texts are not included in the pre-training stage, we fix the parameter of tampered branches and only update them in the finetune stage. As shown in Tab. 1, the model implemented with pre-training process provides significant improvement in both tampered and real-world detection results. To explore the reason for the impressive improvement, we visualize the convergency process in the fine-tuning stage in Fig. 2. The global semantic learning (*e.g.* text position, etc) in backbone helps to converge the network in the fine-tuning stage, achieving a faster and better convergency compared with the model trained from scratch.

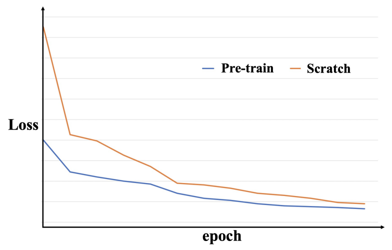

Fig. 2. The comparisons about convergency between the model with/without pre-training stage.

Table 2. The detection results of models with/without introducing parallel-branch feature extractor. Specially, EAST [15] and PSENet [9] are pre-trained on the SynthText [3] and MLT [1] respectively for a better convergency.

		Accuracy						
	Frequency	R-T	P-T	F-T	R-R	P-R	F-R	mF
EAST [15]	-	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**
PSENet [9]	-	83.10	83.61	83.35	42.55	60.76	50.05	66.70
	✓	**83.91**	**84.95**	**84.43**	**43.38**	**62.38**	**51.17**	**67.80**
ContourNet [12]	-	91.45	86.68	88.99	54.80	**77.88**	64.33	76.66
	✓	**91.85**	**88.43**	**90.11**	**56.61**	75.83	**64.83**	**77.47**
ATRR [11]	-	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**

3.2 The Effectiveness of Parallel-branch Feature Extractor

The Quantitative Analysis. A detailed comparison conducted on four-category models is shown in Tab. 2. To be specific, the proposed parallel-branch feature extractor effectively improves the detection performance (mF) by 2.57% in EAST [15], 1.1% in PSENet [9], 0.81% in ContourNet [12] and 0.67% in ATRR [11] respectively. Besides, we also prove that our parallel-branch feature extractor achieves significant performance in data-dependency reduction. Based on the above analyses, the proposed parallel-branch feature extractor shows its significance in TSTD task, and will give lots of insights to the TSTD community.

The Qualitative Analysis. To demonstrate how the parallel-branch feature extractor helps the TSTD task, we use CAM [14] to visualize the activation map in tampered class. As shown in Fig. 3, the activation map of the model constructed with parallel-branch feature extractor performs a more consistent activation in the tampered text areas, and is able to give a more accurate classification result.

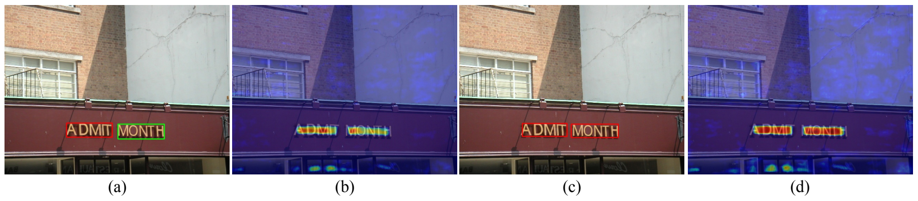

Fig. 3. The qualitative analysis of proposed parallel-branch feature extractor. To be specific, PSENet [9] is used in this visualization. (a) (b): the detection results and activation map without parallel-branch feature extractor; (c) (d): the detection results and activation map with parallel-branch feature extractor.

Table 3. The evaluation of generalization capability of different TSTD methods.

		Accuracy						
Compress	Method	R-T	P-T	F-T	R-R	P-R	F-R	MF
C23	EAST [15]	**74.54**	70.25	72.33	40.23	60.90	48.45	60.39
	PSENet [9]	56.01	74.53	63.95	28.64	43.47	34.53	49.24
	ATRR [11]	66.80	82.00	73.63	43.05	62.05	50.83	62.23
	ContourNet [12]	70.26	**83.74**	**76.41**	**43.38**	**63.44**	**51.52**	**63.97**
C31	EAST [15]	**65.58**	68.51	67.01	37.42	56.22	44.93	55.97
	PSENet [9]	51.12	71.51	59.62	25.00	37.47	29.99	44.81
	ATRR [11]	61.10	**81.74**	69.93	42.38	54.94	47.85	58.89
	ContourNet [12]	61.30	81.57	**70.00**	**42.55**	**59.35**	**49.47**	**59.74**

3.3 The Generalization on Detecting Low-quality Images

We use the ffmpeg compression algorithm to reduce the image quality, by setting the compression level to obtain images of different quality. The compression level is a positive integer from 2 to 31, and a larger value indicates a higher compression level. In this paper, the positive integers 23 and 31 are used. A detailed table of performance is available in Tab. 3, the detection performance of all the models degrades with lower image quality. We suggest that future TSTD methods should also consider the generalization capability on low-quality images to show their significance.

4 Discussion

4.1 The Differences Between Tampered Text Classification and Detection

There are two main differences: 1) Different targets. Referring to text recognition and end-to-end text recognition tasks, the defense against tampered texts contains two tasks: **tampered text classification** (TTC) [7,6,8,2] and **tampered text detection** (TSTD in our work). Different from the pure classification task

(TTC), TSTD successfully integrates text localization and tampered text classification into an end-to-end manner. 2) Better performance. Benefiting from the gradient spread and shared features between detection and classification branches, TSTD performs better in both detection accuracy (mF) and model complexity (Tab. 4).

Table 4. The comparison between TTC and TSTD.

Method	mF	extra parameters for classification
PSENet [18] + TTC	60.2	0.92M
TSTD	64.8	0.0018M

4.2 The Differences Between Forgery Detection in Document Images and Scene Text Images

Compared with tampered document text detection, the detection in scene image has two challenges: 1) Background diversity. The background textures are more complex in scene images. 2) Text diversity. The texts in scene images contain more complex fonts, color, etc. Actually, 1) and 2) are acknowledged in general scene text detection task. We collect tampered bill document images (SROIE [5]) following Tampered-IC13 for the quantitative analysis. There exists 33.4% gap in mF for EAST [15]. Thus, TSTD can better reflect the robustness of tampered text detectors and has important research value.

Table 5. The comparison between tampered document text detection and tampered scene text detection.

Dataset	R-T	P-T	F-T	R-R	P-R	F-R	mF
Scene (TSTD)	75.2	73.2	74.2	44.0	62.7	51.8	63.0
Document	96.6	97.2	96.9	96.3	95.7	96.0	96.4

References

1. Icdar2017 competition on multi-lingual scene text detection and script identification, `http://rrc.cvc.uab.es/?ch=8&com=introduction`
2. 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)

3. 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)
4. 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)
5. Huang, Z., Chen, K., He, J., Bai, X., Karatzas, D., Lu, S., Jawahar, C.: Icdar2019 competition on scanned receipt ocr and information extraction. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1516–1520. IEEE (2019)
6. 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: Asian Conference on Pattern Recognition. pp. 136–150. Springer (2019)
7. Nandanwar, L., Shivakumara, P., Mondal, P., Raghunandan, K.S., Pal, U., Lu, T., Lopresti, D.: Forged text detection in video, scene, and document images. IET Image Processing **14**(17), 4744–4755 (2020)
8. 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)
9. 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)
10. Wang, W., Xie, E., Song, X., Zang, Y., Wang, W., Lu, T., Yu, G., Shen, C.: 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)
11. 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)
12. 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)
13. 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)
14. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)
15. Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., Liang, J.: East: an efficient and accurate scene text detector. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 5551–5560 (2017)