Supplementary Material: Text Grouping Adapter: Adapting Pre-trained Text Detector for Layout Analysis

Tianci Bi[1*] Xiaoyi Zhang[2] Zhizheng Zhang[2] Wenxuan Xie[2] Cuiling Lan[2]
Yan Lu[2‡] Nanning Zheng[1‡]
[1] National Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
National Engineering Research Center for Visual Information and Applications,
and Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[2] Microsoft Research Asia
tiancibi@stu.xjtu.edu.cn,
xiaoyizhang, zhizzhang, wenxie, culan, yanlu@microsoft.com, nnzheng@mail.xjtu.edu.cn

In this supplementary material, we provide more detailed information about our Text Grouping Adapter (TGA) in Section 1, including the cost function of Hungarian Matching and the generation of the pixel embedding map. Moreover, we demonstrate more details about the experiment setup in Section 2, including detailed datasets in Section 2.1 and metrics in Section 2.2, and detailed configuration of training in Section 2.3. Lastly, additional experiments are presented in Section 3, including comparison of different fine-tuning strategies under longer fine-tuning times in Section 3.1, comparison of different padding strategies in Section 3.2, ablation studies of the pre-training in Section 3.3, ablation studies of the hyperparameters of TGA in Section 3.4, generalization of TGA at different annotation levels in Section 3.5, PCA visualization of TGA's output features with and without GMP in Section 3.6, and more qualitative results in Section 3.7.

1. Detailed Settings of TGA

Cost function of Hungarian Matching. As mentioned in Section 3.2 of our main paper, in Group Mask Prediction (GMP), we utilize Hungarian Matching to match ground-truth instances with predicted instances and assign group masks into instances as prediction target. Here we provide the details of the matching cost function. The pair-wise matching cost mainly consider two parts, the loss of predicted class confidence and the loss of similarity between predicted and ground-truth masks. To adapt with different text detectors, we keep our cost function and weight coefficient consistent with the loss of original model if it involves masks.

For example, KNet [25] with TGA uses a weighted combination of focal loss for predicted class confidence, dice loss and binary cross-entropy loss for masks, as the pair-wise matching cost, with corresponding weights of $\{2.0, 4.0, 1.0\}$. Besides using the above three losses with corresponding weights of $\{4.0, 5.0, 5.0\}$, MaskDINO [10] with TGA additionally takes Euclidean distance loss and Intersection over Union (IoU) loss for boxes, as the pair-wise matching cost, with corresponding weights of $\{5.0, 2.0\}$. DBNetpp [11] with TGA, following the setting of original text detection loss, utilizes the dice loss and binary cross-entropy loss for masks, as the pair-wise matching cost, with corresponding weights of $\{1.0, 4.0\}$. In cases where the original text detector does not include the mask loss, *e.g.*, DeepSolo [23], we consider a weighted combination of focal loss for class scores, dice loss for text instance masks, and binary cross-entropy loss for text instance masks as the pair-wise matching cost, with corresponding weights of $\{1.0, 2.0, 4.0\}$.

Details of pixel embedding map generation. In Text Instance Feature Assembling (TIFA), to obtain pixel embedding map, we transform multi-scale image features $\{\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\}$ into $\{\mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \mathbf{P}_5\}$ by applying a series of convolutional networks. Here, we explain more details of generating pixel embedding map.

$\mathbf{P}_2$ is obtained by applying a 3×3 stride-2 convolution to $\mathbf{X}_2$, and $\mathbf{P}_3$ is obtained by applying a 3×3 stride-1 convolution to $\mathbf{X}_3$, after adding positional encoding to it. To process $\mathbf{X}_4$ and $\mathbf{X}_5$, an amplification module is defined, which comprises a 3×3 stride-1 convolution and a $2\times$ upsampling. $\mathbf{P}_4$ is computed by applying the amplification module once to $\mathbf{X}_4$, and $\mathbf{P}_5$ is computed by applying the amplification module twice to $\mathbf{X}_5$. For certain models, additional feature

*Work done during the internship at Microsoft Research Asia.
‡Corresponding authors.

Datasets	Annotation Level			#Image		
	Word	Line	Paragraph	Train	Val	Test
Synth150K [13]	✔			150,000	0	0
MLT17 [16]	✔			9,000	0	9,000
IC13 [8]	✔			229	0	233
IC15 [9]	✔			1,000	0	500
TextOCR [19]	✔			21,778	3,124	3,232
CTW-1500 [24]		✔		1,000	0	500
MSRA-TD500 [22]		✔		300	0	200
IC19-LSVT [20]		✔		30,000	0	20,000
HierText [15]	✔	✔	✔	8,281	1,724	1,634

Table 1. Details of the datasets used in the experiments. Here, *#Image* represents the number of images corresponding to the training, validation and test sets.

$\mathbf{X}_6$ with a size of $\frac{1}{64}$ of the original input size will be fused with $\mathbf{X}_5$ directly through an amplification module. We sum $\{\mathbf{P}_l | i = 1, ..., n\}$ together into a 1x1 convolution to get the pixel embedding map:

$$\mathbf{P} = \sum_{l=1}^{n} conv_{1\times1}(\mathbf{P}_l), \tag{1}$$

where n is the number of multi-scale features, and $\mathbf{P}_l \in \mathbb{R}^{D_l \times H_l \times W_l}$ refers to the l^{th} D_l-dimensional feature with size (H_l, W_l), and $\mathbf{P} \in \mathbb{R}^{D \times H \times W}$ refers to the pixel embedding map in dimension D with size (H, W). In the experiments presented in the main paper, to be consistent with the multi-scale feature design of the original text detector, the scales are slightly different, as detailed below.

For KNet and DBNetpp, $\{\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\}$ are both extracted from the FPN network following the backbone of the text detector. For MaskDINO-R50, $\mathbf{X}_2$ is extracted from the backbone of the text detector and $\{\mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\}$ are extracted from the output of the subsequent encoder. For MaskDINO-Swin-B, $\{\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5, \mathbf{X}_6\}$ are both extracted from the output of the subsequent encoder. For DeepSolo, $\mathbf{X}_2$ is extracted from the backbone of the text detector and $\{\mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5, \mathbf{X}_6\}$ are extracted from the output of the subsequent encoder.

2. More Details about the Experiment Setup

2.1. Detailed Datasets

Datasets for text detection. In the main paper, we select KNet, MaskDINO, DBNetpp, and DeepSolo as the text detectors used in the experiments. Among them, KNet and MaskDINO are pre-trained for line-level text detection, while DBNetpp and DeepSolo focus on word-level text detection. Here we provide more details of these text detection datasets. As shown in Table 1, for line-level datasets, we choose CTW-1500 [24], MSRA-TD500 [22], and IC19-LSVT [20] datasets, while for word-level datasets, we choose Synth150K [13], TotalText [6], MLT17 [16], IC13 [8], IC15 [9], and TextOCR [19] datasets.

Different representations of line and paragraph mask. In particular, the HierText [15] Dataset provides a hierarchical annotation structure covering words, lines, and paragraphs and their relationships. This means that *line and paragraph annotations can be either masks of themselves or the collections of masks for the words composing them*, where the slight difference is that the former covers the background space between words but the latter does not. In the original paper [15] which proposes the HierText Dataset, a paragraph mask is represented and evaluated by the collection of word masks composing the paragraph. We follow the original evaluation setting in all our experiments except the Table 2 in our main paper. Table 2 takes the whole line and paragraph masks instead of word mask collections as the ground truths to be consistent with other line-level dataset evaluations like CTW-1500, MSRA-TD500 and IC19-LSVT. This explains the slightly different value from the same model between Table 1 and Table 2 in our main paper.

2.2. Detailed Metrics

Text detection metrics. In the comparison of fine-tuning strategies presented in the main paper, we focus on the model's ability in line-level detection and layout analysis. Therefore, text detectors with TGA are evaluated on the CTW-1500, MSRA-TD500, IC19-LSVT, and HierText datasets. Among them, the CTW-1500 Dataset and the MSRA-TD500 Dataset use Average Precision (AP) as the metric, the IC19-LSVT Dataset uses Harmonic Mean (H-mean) as the metric, and the HierText Dataset uses PQ as the metric. AP is primarily used to measure the model's accuracy at different confidence thresholds by calculating the area under the Precision-Recall curve. A high AP value typically indicates that the model can detect more positive samples while maintaining high precision. H-mean is commonly used to evaluate whether a model can effectively identify text regions while maintaining high detection accuracy. The value of H-mean is the harmonic mean of Precision and Recall:

$$\begin{aligned} \text{Precision} &= \frac{|TP|}{|TP| + |FP|}, \\ \text{Recall} &= \frac{|TP|}{|TP| + |FN|}, \\ \text{H} - \text{mean} &= \frac{2\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \end{aligned} \tag{2}$$

where $\{TP, FP, FN\}$ means the set of True Positive, False Positive, and False Negative, respectively. $|\cdot|$ means the corresponding number of it.

Layout analysis metrics. In the main comparison presented in the main paper, text detectors with TGA and base-

lines are evaluated and tested on the HierText Dataset using Precision (P), Recall (R), F1 score (F), and Panoptic Quality (PQ) as the metrics. The reason for using PQ is that it provides a unified performance evaluation for segmentation at the levels of words, lines, and paragraphs. For text detection, PQ is evaluated at the corresponding word or line levels, while for layout analysis, PQ is evaluated at the paragraph level. The value of PQ is equal to the product of the F1 score and the average IoU of all True Positive pairs:

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|} \times \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}. \quad (3)$$

2.3. Detailed Configuration of Training

Training schedules. KNet and MaskDINO do not have pre-trained parameters specifically tailored for text detection. Although DBNetpp does possess pre-trained parameters for text detection, its performance on the HierText Dataset is notably poor. Therefore, it is necessary to pre-train the text detectors before fine-tuning specifically for layout analysis.

In the comparisons of the main paper, we make efforts to ensure that the training processes of each model are as consistent as possible, as shown in Table 2. For line-level detection, KNet and MaskDINO are pre-trained on the CTW-1500, MSRA-TD500, IC19-LSVT, and HierText datasets for 36 and 24 epochs, respectively. In Table 2 of the main paper, the corresponding pre-trained parameters of KNet and MaskDINO are directly fine-tuned on the HierText Dataset for layout analysis. In Table 1 of the main paper, the corresponding pre-trained parameters of KNet and MaskDINO undergo alignment training on the HierText Dataset, transitioning from line annotations with the masks of themselves to annotations with the collections of masks for the words composing them, lasting for 50 epochs. Subsequently, they are separate fine-tuned on the HierText Dataset for layout analysis. Specifically, for the layout analysis fine-tuning on the HierText Dataset, KNet executes 60 epochs, while MaskDINO executes 100 epochs. Regarding TGA + MaskDINO-R50 in Table 1 of the main paper, there is a gradient explosion issue during the alignment process after pre-training. Therefore, we modify its pre-training to 27 epochs on the IC19-LSVT Dataset and add an auxiliary semantic segmentation loss during alignment training. For word-level detection, DBNetpp is pre-trained on the HierText Dataset for 120 epochs, building upon off-the-shelf pre-trained parameters from oCLIP, to enhance its text detection performance. Following this, it is fine-tuned on the HierText Dataset for layout analysis for 600 epochs. DeepSolo directly utilizes off-the-shelf pre-trained parameters from Synth150K, MLT17, IC13, IC15, and TextOCR datasets, undergoing fine-tuning on the HierText Dataset for layout analysis for 435,000 iterations.

In the ablation experiments, for fast implementation, KNet performs fine-tuning for layout analysis after 120 epochs of direct pre-training on the HierText Dataset.

Devices. In all experiments, text detectors using ResNet-50 [7] or ViTAE-S [21] as the backbone are trained with 8 * V100-16G, while those using Swin-Base [14] are trained with 8 * V100-32G.

Training hyperparameters. As shown in Table 2, we avoid extensive hyperparameter search and keep simple and consistent with the original text detectors' schedules. In the fine-tuning phase, the number of text instances N of KNet, MaskDINO, DBNetpp, and Deepsolo are set to $\{256, 300, 384, 100\}$ respectively, while the batch sizes of KNet, MaskDINO, DBNetpp, and Deepsolo are set to $\{16, 8, 16, 16\}$ respectively. The optimizer is Adam and the learning rate lr defaults to $1e^{-4}$, and the learning rate is set to $5e^{-5}$ only for TGA + MaskDINO-Swin-B, which is used in the Table 2 of the main paper.

3. More Experiment Results

3.1. Comparison of Different Fine-tuning Strategies under Longer Fine-tuning Times

To verify the effectiveness of TGA under different fine-tuning strategies, in Table 3, we extend the fine-tuning time of TGA + MaskDINO-R50, based on the setup of the comparison of fine-tuning strategies in the main paper.

The results demonstrate that a longer full fine-tuning time can improve the model's performance in text detection without compromising the ability of layout analysis. We attribute this to the fact that the encoder and decoder of the model do not explicitly learn features related to layout analysis during pre-training. Throughout the full fine-tuning period, TGA optimizes the parameters of the encoder and even the backbone, thereby influencing the subsequent processing of the decoder, i.e., the effect of text detection. Over time, the decoder gradually learns to handle features related to layout analysis, but the learning time is positively correlated with the number of decoder parameters. The decoder of TGA + MaskDINO-R50 is large, consisting of cross-attention layers, hence requiring a long fine-tuning time. Possibly due to insufficient fine-tuning time, the text detection performance of TGA + MaskDINO-R50 does not fully recover during the full fine-tuning period.

In addition, from the results, it can be observed that under the frozen text detector strategy, a longer fine-tuning time can enhance the ability of layout analysis. The frozen text detector strategy, while saving training costs, not only preserves the generalization of the text detector on the pre-training datasets but also maintains powerful performance

		KNet	MaskDINO	DBNetpp	DeepSolo
Initial Weight		COCO [1, 12] for R50 ImageNet [3, 18] for Swin-B	COCO [2, 12] for R50 ImageNet [3, 18] for Swin-B	oCLIP [4, 17]	Official release [5]
Pre-training	Dataset	Line-level datasets	Line-level datasets	HierText	-
	Schedule	36 epochs	24 epochs	120 epochs	-
	Learning Rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	-
Fine-tuning	Dataset	HierText	HierText	HierText	HierText
	Schedule	60 epochs	100 epochs	600 epochs	435,000 iters
	Learning Rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$

Table 2. Training configuration of the text detectors in all comparison experiments in the main paper. Line-level datasets denotes the combination of CTW-1500, MSRA-TD500, IC19-LSVT, and HierText. For the special cases of TGA + MaskDINO-R50 in Table 1 of the main paper and TGA + MaskDINO-Swin-B in Table 2 of the main paper, please refer to the Section 2.3 for details.

Frozen Text Detector	Epochs	CTW-1500	MSRA-TD500	IC19-LVST	HierText	
		Line AP	Line AP	Line H-mean	Line PQ	Paragraph PQ
✘	100	**41.46**	**51.80**	**59.19**	59.88	59.19
	200	39.88	50.23	58.04	62.52	59.91
	300	39.03	50.39	57.94	**64.39**	**59.91**
✔	100					54.95
	200	**61.18**	**59.54**	**82.62**	**65.83**	53.76
	300					**58.69**

Table 3. Results of TGA + MaskDINO-R50 under longer fine-tuning time on line-level datasets. In the main content, the first three rows are the results under the full fine-tuning strategy, and the last three rows are the results under the frozen text detector strategy.

Pre-training	Line		Paragraph	
	F	PQ	F	PQ
✘	73.85	56.17	69.68	53.14
✔	**77.04**	**59.08**	**71.90**	**55.27**

Table 4. Ablation studies of the pre-training for the text detector, utilizing TGA + KNet-R50, under the full fine-tuning strategy. The results are evaluated on the HierText Validation Set.

Attention Num	Line PQ	Paragraph			
		P	R	F	PQ
1		72.70	61.90	66.86	50.69
3	**58.40**	76.70	66.23	71.07	54.27
5		**77.47**	**66.34**	**71.48**	**54.64**

Table 5. Ablation studies of the number of self-attention layers for TGA, utilizing TGA + KNet-R50, under the frozen text detector strategy. The results are evaluated on the HierText Validation Set.

on the layout analysis dataset. It proves to be an economically effective fine-tuning strategy for TGA.

Weight of GMP \ Weight of AMP	1.0	2.0
2.0	53.95	54.26
4.0	53.88	**54.27**
8.0	53.60	53.99

Table 6. Results of Paragraph PQ with different loss weights for TGA, utilizing TGA + KNet-R50, under the frozen text detector strategy. The results are evaluated on the HierText Validation Set.

3.2. Comparison of Different Padding Strategies

As mentioned in Section 3.1 of our main paper, we extract instances $\mathbf{I} = \{I_i\}_{i=1}^{N}$ from the output of pre-trained text detectors. Masks are padded when less than N instances are found in the output of text detectors. Here we ablate how different padding strategies affect the final performance under the frozen text detector strategy, including empty padding, prediction padding, ground-truth padding and mixed padding. Empty padding denotes masks filled with all zero values are padded into $\hat{\mathbf{M}}^I$. Similarly, prediction or ground-truth padding means predicted or ground-truth instance masks are randomly selected and padded. The mixed padding denotes the padded masks are randomly select from a mixed set of predicted and ground-truth instance

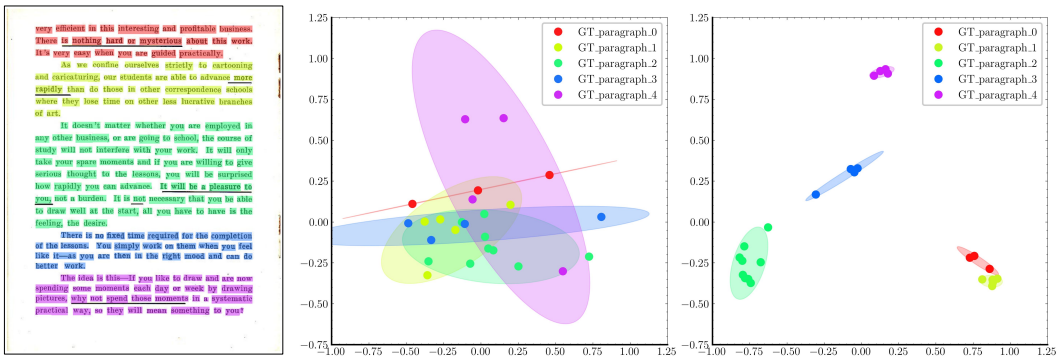

Figure 1. PCA visualization of text instance features with the input image **(a)**: **(b)** without GMP and **(c)** with GMP under the frozen text detector strategy. Here, The ellipses represent 95% confidence intervals for the features, and the explained variance ratios on the horizontal and vertical axes are {27.29%, 13.80%} in **(b)**, and {44.77%, 21.10%} in **(c)**, respectively.

Padding	Paragraph			
	P	R	F	PQ
empty	**47.37**	35.89	40.81	29.56
prediction	42.76	33.53	37.59	27.10
ground-truth	45.93	36.09	40.42	29.24
mixed	45.94	**37.22**	**41.12**	**29.72**

Table 7. Comparison of different padding strategies. The experiments perform fine-tuning with frozen text detector on single TGA + DBNetpp-R50 for 60 epochs on the HierText Dataset. Note that the number is lower than the main comparison for the short training epochs. The results are evaluated on the HierText Validation Set.

Models	Instance		Paragraph	
	F	PQ	F	PQ
Word-based				
TGA + KNet-R50	64.62	48.53	58.20	43.00
TGA + MaskDINO-R50	72.25	55.84	55.24	42.37
Line-based				
TGA + DBNetpp-R50	71.13	53.45	62.35	46.49

Table 8. Results of different models with TGA at different annotation levels, under the frozen text detector strategy. The results are evaluated on the HierText validation set.

masks. As shown in Table 7, the mixed one outperforms others. It demonstrates that the accurate instance masks help the layout analysis performance but directly injecting the ground-truth masks causes gap between predicted masks and ground-truth ones. Hence, the mixed padding strategy is most beneficial one for layout analysis.

3.3. Ablation Studies of Pre-training

To verify the necessity of pre-training, in Table 4, we perform ablation studies on it. The results show that pre-training not only improves the performance of text detection, but also further enhances the ability of layout analysis.

3.4. Ablation Studies of Hyperparameters in TGA

Number of self-attention layers. In Table 5, we experiment with the number of self-attention layers of TGA. It can be observed that as the number of layers increases, the gains decrease. To balance the computational cost with performance, we set the number of self-attention layers to be 3 in all our experiments.

Weights of losses. In Table 6, we experiment with different weight coefficients of losses for TGA on KNet-R50. It's noteworthy that our performance is robust when we adjust these weight coefficients, which demonstrates the robustness of our TGA.

3.5. Generalization of TGA at different annotation levels

To demonstrate that TGA performs well at different annotation levels, in Table 8, we fine-tuned TGA + KNet-R50 and TGA + MaskDINO-R50 on the HierText Dataset at the word level and TGA + DBNetpp-R50 on the HierText Dataset at the level of line annotations with the masks of themselves. All of the above fine-tuning is for the layout analysis phase and freezes the text detector. Since there are more word annotations than line annotations per image on average in the HierText Dataset, the number of text instances of KNet and MaskDINO should be increased appropriately. Meanwhile, the setting of the fine-tuning time should take into account the difficulty of the layout analysis task, the capability of the text detector, and the number of parameters of the TGA module. Thus, in the case

of all three models, namely TGA + KNet-R50, TGA + MaskDINO-R50, and TGA + DBNetpp-R50, the number of text instances N is set to 384, and each is fine-tuned for $\{120, 40, 1200\}$ epochs, respectively.

It can be observed that TGA + KNet-R50 and TGA + MaskDINO-R50 continue to perform well at the word annotation level. However, TGA + DBNetpp-R50 performs mediocrely at the level of text line annotation, which is at variance with the performance at the word annotation level. This is due to the fact that the ground truths of the polygons undergo a shrink operation in the loss computation during the text detection training of the DBNetpp model. The shrink operation, whose shrink offset is proportional to the ratio of the area and perimeter of the polygon. Since the aspect ratio of the text line mask is larger than that of the word mask, the former loses more features of the edges, which hinders fine-tuning of the layout analysis.

3.6. PCA Visualization

In the ablation studies presented in the main paper, we propose how GMP assists in predicting the affinity matrix when both group masks and affinity are derived from group annotations. The answer lies in the design of GMP, which can capture more comprehensive and holistic information about group instances. To substantiate this conclusion, in Figure 1, we present the PCA visualization of text instance features with and without GMP. The visualization clearly shows that instances incorporating GMP exhibit tighter clustering, validating the effectiveness of our design.

3.7. More Qualitative Results

To demonstrate the generality and effectiveness of TGA, we present more visual results of layout analysis in Figure 2. It can be observed from the results in the fourth row that the performance of text detection indeed has a significant impact on the layout analysis: the poor text detection results of TGA + DeepSolo-ViTAE-S, as it is not pre-trained on the HierText Dataset, lead to inferior layout analysis compared to TGA + MaskDINO-Swin-B. In the face of this situation, the frozen text detector strategy allows for the decoupling of the text detector and TGA, enabling them to focus on their respective tasks during pre-training and fine-tuning stages, namely text detection and layout analysis.

References

[1] Pre-training parameters of knet with res-50 as backbone on the coco dataset. https://download.openmmlab.com/mim-example/knet/det/knet/knet_s3_r50_fpn_1x_coco/knet_s3_r50_fpn_1x_coco_20211016_113017-8a8645d4.pth, . Accessed: 2023-11-27. 4

[2] Pre-training parameters of maskdino with res-50 as backbone on the coco dataset. https://github.com/IDEA-Research/detrex-storage/releases/download/maskdino-v0.1.0/maskdino_r50_50ep_300q_hid1024_3sd1_instance_maskenhanced_mask46.1ap_box51.5ap.pth, . Accessed: 2023-11-27. 4

[3] Pre-training parameters for swin-base on the imagenet dataset. https://github.com/SwinTransformer/storage/releases/download/v1.0.0/swin_base_patch4_window7_224_22k.pth. Accessed: 2023-11-27. 4

[4] Pre-training parameters of dbnetpp with res-50 as backbone from oclip. https://download.openmmlab.com/mmocr/backbone/resnet50-oclip-7ba0c533.pth. Accessed: 2023-11-27. 4

[5] Pre-training parameters of deepsolo with vitae-s as backbone on the synth150k, mlt17, ic13, ic15, and textocr datasets. https://1drv.ms/u/s!AimBgYV7JjTlgcdqw1UUnbSAG4qoWA?e=Co1prY. Accessed: 2023-11-27. 4

[6] Chee Kheng Ch'ng and Chee Seng Chan. Total-text: A comprehensive dataset for scene text detection and recognition. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, pages 935–942. IEEE, 2017. 2

[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*, pages 770–778, 2016. 3

[8] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th international conference on document analysis and recognition*, pages 1484–1493. IEEE, 2013. 2

[9] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos 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 *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 1156–1160. IEEE, 2015. 2

[10] Feng Li, Hao Zhang, Huaizhe Xu, Shilong Liu, Lei Zhang, Lionel M Ni, and Heung-Yeung Shum. Mask dino: Towards a unified transformer-based framework for object detection and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3041–3050, 2023. 1

[11] Minghui Liao, Zhisheng Zou, Zhaoyi Wan, Cong Yao, and Xiang Bai. Real-time scene text detection with differentiable binarization and adaptive scale fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):919–931, 2022. 1

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In

Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13, pages 740–755. Springer, 2014. 4

[13] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang. Abcnet: Real-time scene text spotting with adaptive bezier-curve network. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9809–9818, 2020. 2

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 3

[15] Shangbang Long, Siyang Qin, Dmitry Panteleev, Alessandro Bissacco, Yasuhisa Fujii, and Michalis Raptis. Towards end-to-end unified scene text detection and layout analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1049–1059, 2022. 2

[16] Nibal Nayef, Fei Yin, Imen Bizid, Hyunsoo Choi, Yuan Feng, Dimosthenis Karatzas, Zhenbo Luo, Umapada Pal, Christophe Rigaud, Joseph Chazalon, et al. Icdar2017 robust reading challenge on multi-lingual scene text detection and script identification-rrc-mlt. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, pages 1454–1459. IEEE, 2017. 2

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 4

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 4

[19] Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8802–8812, 2021. 2

[20] Yipeng Sun, Zihan Ni, Chee-Kheng Chng, Yuliang Liu, Canjie Luo, Chun Chet Ng, Junyu Han, Errui Ding, Jingtuo Liu, Dimosthenis Karatzas, et al. Icdar 2019 competition on large-scale street view text with partial labeling-rrc-lsvt. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1557–1562. IEEE, 2019. 2

[21] Yufei Xu, Qiming Zhang, Jing Zhang, and Dacheng Tao. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *Advances in neural information processing systems*, 34:28522–28535, 2021. 3

[22] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1083–1090. IEEE, 2012. 2

[23] Maoyuan Ye, Jing Zhang, Shanshan Zhao, Juhua Liu, Tongliang Liu, Bo Du, and Dacheng Tao. Deepsolo: Let transformer decoder with explicit points solo for text spotting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19348–19357, 2023. 1

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

[25] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. *Advances in Neural Information Processing Systems*, 34:10326–10338, 2021. 1

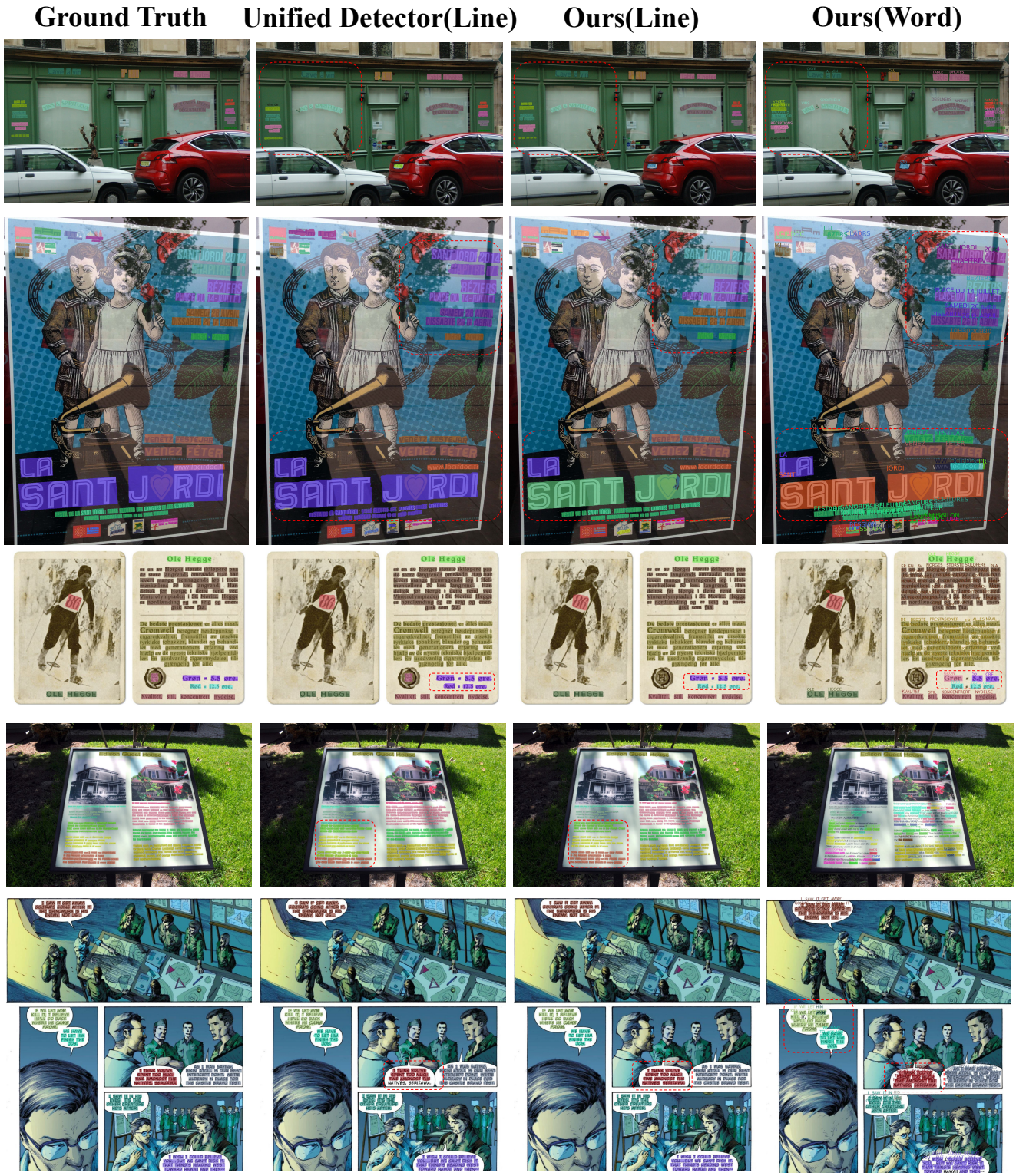

Figure 2. Visualization of results on the validation set of the HierText Dataset: from left to right, the sequence includes the ground truth, line-based Unified Detector, TGA + MaskDINO-Swin-B and TGA + DeepSolo-ViTAE-S. The red dashed boxes in the figure represent areas that require additional attention.