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

Abstract

Significant progress has been made in scene text detection models since the rise of deep learning, but scene text layout analysis, which aims to group detected text instances as paragraphs, has not kept pace. Previous works either treated text detection and grouping using separate models, or train a model from scratch while using a unified one. All of them have not yet made full use of the already well-trained text detectors and easily obtainable detection datasets. In this paper, we present Text Grouping Adapter (TGA), a module that can enable the utilization of various pre-trained text detectors to learn layout analysis, allowing us to adopt a well-trained text detector right off the shelf or just fine-tune it efficiently. Designed to be compatible with various text detector architectures, TGA takes detected text regions and image features as universal inputs to assemble text instance features. To capture broader contextual information for layout analysis, we propose to predict text group masks from text instance features by one-to-many assignment. Our comprehensive experiments demonstrate that, even with frozen pre-trained models, incorporating our TGA into various pre-trained text detectors and text spotters can achieve superior layout analysis performance, simultaneously inheriting generalized text detection ability from pre-training. In the case of full parameter fine-tuning, we can further improve layout analysis performance.

1. Introduction

With the rise of deep learning, text detection [17, 18, 22, 44], text recognition [3, 8, 16, 34], and end-to-end text spotting [19, 20, 38, 39] models have greatly improved the accuracy and efficiency of identifying text instances like words or text lines. To fully understand the text semantics in various applications [1, 2, 30, 41, 45], it is essential to determine how to organize these text instances into coherent semantic entities, *e.g.*, determining which detected words constitute a line and which detected lines form a paragraph. This problem, as visualized in the top of Figure 1, named scene text layout analysis [23], has not been advanced at the same pace as other scene text understanding tasks.

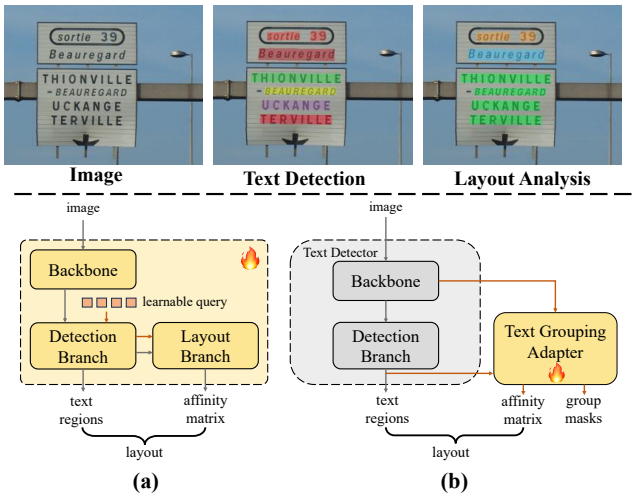

Figure 1. **Top**: Visualization of the scene text detection and layout analysis tasks. The mask with the same color denotes detected as a group. **Bottom**: Comparison between **(a)** the previous work Unified Detector [23] and **(b)** proposed TGA. TGA also provides the flexibility of freezing or fine-tuning the pre-trained text detector.

Previous layout analysis works [13, 29, 46] adopt separate semantic segmentation models to localize different high-level entities in scanned and digital document images, which only focus on limited scenarios instead of general-scope natural scenes and ignore low-level text instances like words and text lines. Recent design on a Unified Detec-

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

tor [23] is the first to address the layout analysis problem in this field and propose a unified model to detect text instances and their layouts. Different from previous works that model layout as high-level text entity localization, Unified Detector considers it from the perspective of the affinities between low-level text instances. This enables a direct prediction of the affinities between text instances. The masks of high-level text entities can be derived by connecting these low-level instance masks. However, this Unified Detector requires a query-based network structure and only considers jointly training its text detection branch with the layout branch from scratch. This implies a lack of flexibility in the detection network structure. More importantly, it only benefits from the layout analysis dataset, given the significantly smaller size of the existing layout analysis dataset, *e.g.*, 8,281 images in HierText [23] compared to 30,000 in the text detection dataset IC19-LSVT [31], which obviously limits the potential of text detection.

Given these limitations, we pose the question: *can we enable already pre-trained text detectors with a new module to learn layout analysis?* Nonetheless, answering this question presents nontrivial challenges. The first challenge arises from the diversity of existing text detectors in terms of architecture and text region representations. Specifically, these text detectors employ a wide array of network structures, ranging from query-based transformers [25, 38, 43] to fully convolutional networks [17, 22, 44] and dynamic convolutional networks [42]. Moreover, they also model the text region diversely, such as semantic mask [17, 44], instance mask [23] and parameterized curve [19, 48]. Another challenge is the lack of global features in pre-trained text detectors since it might overly focus on local features for text instance detection when pre-training.

To address these challenges, we introduce Text Grouping Adapter (TGA), a novel module that adapts diverse pre-trained text detectors to learn layout analysis. Not only does it provide flexibility and compatibility for network structures, it also empowers the model to inherit a robust text detection capability from pre-training on large-scale text datasets. Specifically, the TGA takes text regions and image features as input, and outputs affinities between regions to represent layouts following the approach of the Unified Detector. The TGA comprises two key components: Text Instance Feature Assembling (TIFA) and Group Mask Prediction (GMP). By seamlessly converting the text regions into text instance masks and embedding image feature into pixel embedding map, our TIFA ensures TGA can obtain text instance representations from the two inputs. It becomes the cornerstone of TGA's compatibility with various text detection architectures. GMP is designed as an additional task during training, to boost the learning of cohesive features for text instances within their respective groups, which helps the instances to understand the group region and aggregate global features necessary for layout analysis. By doing so, TGA effectively bridges the gap between pre-trained text detectors and scene text layout analysis.

Our extensive experiments reveal that even when freezing the pre-trained models, the integration of our TGA with various pre-trained text detectors and text spotters can significantly enhance performance on the layout analysis dataset. Moreover, it also allows for the inheritance of a generalized text detection ability from pre-training. Remarkably, when full parameter fine-tuning is applied, our TGA can further improve layout analysis performance. Varied ablation studies on the components of TGA are further conducted, demonstrating their substantial contribution to the improvement of layout analysis performance. We believe such a Text Grouping Adapter will accelerate the development and application of layout analysis via leveraging broader pre-trained models and datasets.

2. Related Works

2.1. Text Detection and Spotting

As an important and active topic in the computer vision field with numerous practical applications, text detection has been studied extensively. However, to date, these works exhibit a high degree of variance in terms of text region representations and network structures. For text representations, a series of works [17, 18, 22, 35, 44] model the text regions as semantic masks. As alternative approaches, [19, 25, 38, 43, 48] propose to leverage parameterized curves such as Bezier curves and Fourier contours to adaptively fit highly-curved text regions. With regards to network structures, some works [17, 18, 22, 35, 44] use fully convolutional networks to predict semantic masks, while others incorporate diverse approaches like query-based transformers [25, 38, 43] and region-based convolutional networks [19, 24].

2.2. Text Layout Analysis

While text detection has been extensively developed and studied, layout analysis remains in its nascent stages, particularly concerning scene text, due to the complex and challenging task of distinguishing relationships between text instances. Some works [13, 29, 46] propose to analyze the layout of document images, where the task is defined as to localize semantically coherent text blocks. These works only focus on document images and neglect word or line-level text instances. Long et al. [23] is the first work to study layout analysis in the scene text field and train a unified model to detect text instances and recognize their relationships as layout. Given the fact that layout analysis annotations are expensive and rare, training a unified scene text layout analysis model only based on the layout analysis dataset poses limitations in learning text instance detection capabilities. Our work aims to harness the advancements

made in scene text detection to enhance layout analysis.

2.3. Adapter

Adapters have garnered extensive utilization in the Natural Language Processing (NLP) field as an efficient tool to enable the adaptation of pre-trained models to downstream NLP tasks. In the field of computer vision, earlier works apply adapters with incremental learning [28] and domain adaptation [26, 27]. Recent works focus on leveraging adapters to transfer pre-trained vision transformers into downstream tasks, including dense prediction [6] and vision-language domains [9, 15, 47]. Similar to the previous adapter works on other domains, our TGA transfers the pre-trained text detector into the layout analysis task to inherit the knowledge learned from pre-training.

3. Methods

Figure 2 illustrates the pipeline of our proposed Text Grouping Adaptor (TGA), including two key components: Text Instance Feature Assembling (TIFA) and Group Mask Prediction (GMP). TGA takes image features and detected text regions as inputs, producing affinities between text regions. Within the TIFA component, text instance features are assembled, and GMP encourages these instance features to learn cohesive group features during training. Then, the learned text instance features are used to predict the final affinity matrix. We now discuss these components in detail.

3.1. Text Instance Feature Assembling

Text Instance Feature Assembling (TIFA) is designed to process diverse representations of text regions and produce text instance features as output. This requires a unified text region representation and a spatially correspondent approach to assemble instance features. Inspired by instance segmentation methods [7, 42], we employ a pixel embedding map and instance masks to assemble text instance features. We sequentially introduce this module as follows.

Unified text region representation. Various text detectors generate different representations of text regions, e.g., semantic masks, instance mask, and kinds of parameterized curves. For the compatibility of TGA and instance-level understanding, we select the text instance mask as our representation, which can be converted seamlessly from other representations. Specifically, for semantic masks, we employ binarization on each mask and utilize the algorithm [32] to identify text instances within them. In the case of parameterized curves, we transform each curve instance into a close polygon, then draw and fill the polygon to create the corresponding instance mask. We define $\mathbf{I} = \{I_i\}_{i=1}^{N}$ as the extracted text instance set, including N instances. Empty masks are padded when less than N instances are found in the output of text detectors. By this process, we unify disparate text region representations as text instance masks $\hat{\mathbf{M}}^I$ for later feature assembling.

Pixel embedding map. Grouping text instances necessitates both low-level features to discriminate small-size texts and high-level features to provide sufficient contextual information. For this, we derive a pixel embedding map from multi-scale features $\{\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\}$, of the text detector's backbone. The sizes of $\{\mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4, \mathbf{X}_5\}$ are $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ of the input image size. Through a series of convolutional layers and upsampling, each $\mathbf{X}_i$ is transformed into a corresponding $\mathbf{P}_i$, all standardized to $\frac{1}{8}$ of the input image size. The final pixel embedding map $\mathbf{P}$ is obtained by summing up $\{\mathbf{P}_l | i = 1, ..., n\}$ and then feeding the result to another convolution to refine the pixel features. With a simple fully convolutional network, we embed the multi-scale features into the pixel embedding map, ensuring the retention of distinct text features while capturing the necessary contextual information for layout analysis. We refer this simple fully convolutional network as Pixel Embedding Layers, shown in the Figure 2.

Feature assembling. As the converted instance mask prediction $\hat{\mathbf{M}}^I$ determines if a pixel of the image belongs to the text instance, we can assemble the instance feature by integrating the pixel embedding map $\mathbf{P}$ and the text instance mask $\hat{\mathbf{M}}^I$. This is accomplished through a multiplication operation between the two, which is equivalent to extracting the D-dimensional feature for each point corresponding to the pixel embedding map and sum pooling spatially.

$$\mathbf{F}^I = \hat{\mathbf{M}}^I \cdot \mathbf{P}^\top, \tag{1}$$

where $\hat{\mathbf{M}}^I \in \mathbb{R}^{N \times H \times W}$ refers to N text instance masks converted from the prediction of the text detector, and $\mathbf{F}^I \in \mathbb{R}^{N \times D}$ are N text instance features.

3.2. Group Mask Prediction

The text instance features assembled from individual instance masks in TIFA exclusively focus on their respective instance regions, overlooking the contextual information. Hence, we introduce Group Mask Prediction (GMP) subsequent to TIFA. GMP encourages text instances to learn cohesive features of their corresponding group, thereby realizing the implicit clustering, which is crucial for the accurate prediction of the final affinity matrix.

To facilitate interaction among text instances and extract contextual information in group mask prediction, we first incorporate self-attention layers into the process. We fed text instance features into the self-attention layers: $\hat{\mathbf{F}}^I = SA(\mathbf{F}^I, \mathbf{F}^I)$, where $\hat{\mathbf{F}}^I \in \mathbb{R}^{N \times D}$ denotes the updated text instance features and $SA(\cdot)$ denotes self-attention layers.

Then, we set the group mask with a one-to-many assignment as the learning target. Specifically, for each text group entity, *e.g.*, text paragraph, we assign the group's mask to all instances that belong to that group using Hungarian matching. To achieve a more accurate assignment, we avoid direct matching of the detected instance masks with the ground-truth group masks. Instead, we first match the detected in-

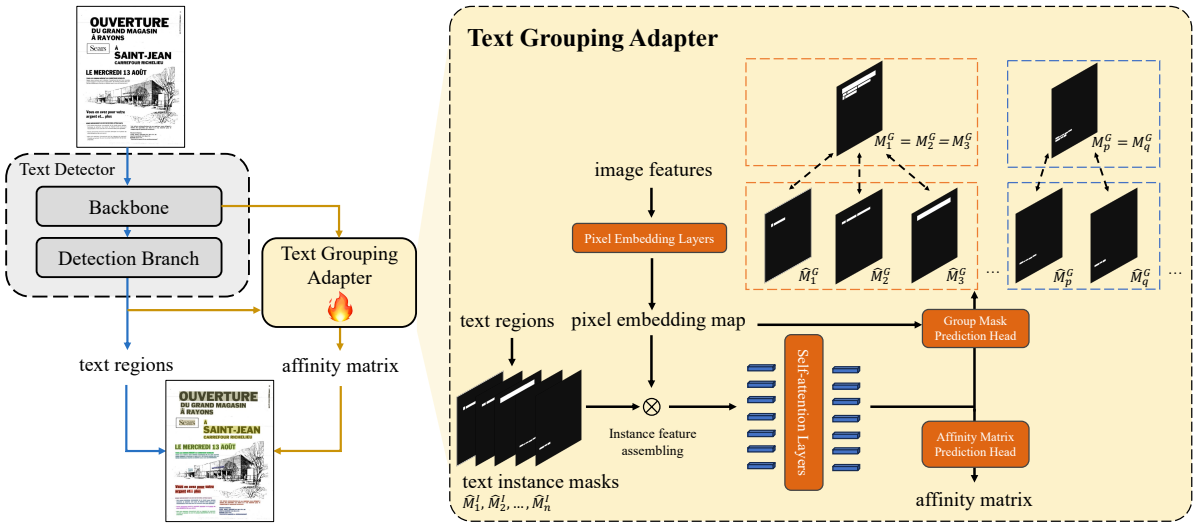

Figure 2. Overview of proposed Text Grouping Adapter. The dashed boxes denote the matched group and instance. The text detector can be frozen or fine-tuned together with TGA when training. $\hat{\mathbf{M}}_i^I$ is the predicted instance mask of I_i. $\hat{\mathbf{M}}_i^G$ and $\mathbf{M}_i^G$ are predicted group mask of I_i and assigned ground-truth one of I_i. To illustrate, a same group mask is duplicated as $\mathbf{M}_p^G$ and $\mathbf{M}_q^G$ and assigned to I_p and I_q.

stance masks with the ground-truth instance masks. Subsequently, we transform the matched ground-truth instance masks into the corresponding ground-truth group masks according to the provided annotations. The instance-level matching result are denoted as $\hat{\sigma}$:

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{match}(\mathbf{M}_i^I, \hat{\mathbf{M}}_{\sigma(i)}^I), \tag{2}$$

where we match for a permutation of N text instances $\sigma \in \mathfrak{S}_N$ with the lowest cost. Under certain σ, $\mathbf{M}_i^I$ is the ground-truth text instance mask, while $\hat{\mathbf{M}}_{\sigma(i)}^I$ is its matched predicted value. $\mathcal{L}_{match}$ is the pair-wise matching cost, which is designed to be consistent with the original loss of text detectors, detailed in our supplementary material.

We then obtain the predicted group mask $\hat{\mathbf{M}}^G$ via a dot-product between the updated text instance features and the pixel embedding map:

$$\hat{\mathbf{M}}^G = \hat{\mathbf{F}}^I \otimes \mathbf{P}, \tag{3}$$

where $\hat{\mathbf{M}}^G \in \mathbb{R}^{N \times H \times W}$ are the N predicted text group masks from the N text instances. Subsequently, we compute the Dice Loss between predicted text group masks and matched ground-truth group masks:

$$\mathcal{L}_{dice} = 1 - \frac{2 \sum_i^N \mathbf{M}_i^G \cdot \hat{\mathbf{M}}_{\hat{\sigma}(i)}^G}{\sum_i^N \left\|\mathbf{M}_i^G\right\|_F^2 + \sum_i^N \left\|\hat{\mathbf{M}}_{\hat{\sigma}(i)}^G\right\|_F^2}, \tag{4}$$

where the predicted instances without matched group-truth ones are excluded from the loss calculation.

This approach transforms DETR-style [4] one-to-one assignment to a one-to-many assignment by replacing matched instance masks with corresponding group masks, resulting in the same group mask being assigned to all text instances belonging to the group. It also differs from the one-to-many assignment concept referred to in other DETR variants [5, 12], which assign single instance supervision into many instance predictions for faster convergence rather than implicitly clustering instances for group entities.

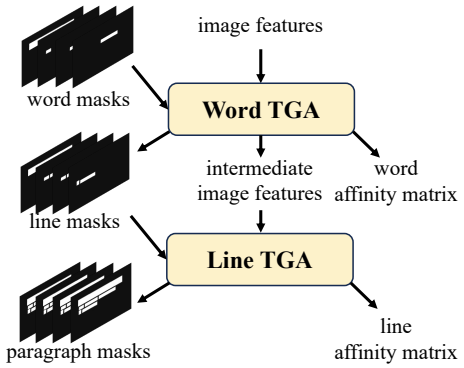

Figure 3. Details of Cascade TGA. Intermediate image features denotes the multi-scale features before summing in Word TGA.

3.3. Affinity Matrix Prediction

Following Unified Detector, we predict an affinity matrix between text instances to represent the layout. By multiplying the updated text instance features with their transpose, we can obtain the predicted affinity matrix $\hat{\mathbf{A}} = \hat{\mathbf{F}}^I \cdot (\hat{\mathbf{F}}^I)^\top$, where the element $\hat{\mathbf{A}}_{i,j}$ denotes the predicted affinity score between I_i and I_j, *i.e.*, the possibility of the instance pair belonging to the same group. We utilize the previously obtained instance matching result $\hat{\sigma}$ and annotations to construct ground truth binary affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ and the binary instance loss weight $\mathbf{C} \in \mathbb{R}^{N \times N}$. The element $\mathbf{A}_{i,j} \in \{0, 1\}$ is set to 1 if the pair, I_i and I_j, belongs to the same text group and 0 otherwise. $\mathbf{C}_{i,j}$ is used to exclude

the instances without matched ground truth under $\hat{\sigma}$.

Finally, we calculate the binary cross-entropy loss between the predicted and ground-truth affinity matrix as the loss function for text grouping:

$$\mathcal{L}_{group} = \sum_{i}^{N}\sum_{j}^{N} \mathbf{C}_{i,j}[\mathbf{A}_{i,j} \cdot \log(\hat{\mathbf{A}}_{i,j}) + (1 - \mathbf{A}_{i,j}) \cdot \log(1 - \hat{\mathbf{A}}_{i,j})]. \tag{5}$$

during the inference phase, we set a group threshold t on the affinity matrix $\mathbf{A}$ to determine whether pairs of texts are related. We then use the union-intersection approach to obtain the final text grouping.

3.4. Loss Function

The TGA's overall loss function is calculated as the weighted sum of the group mask prediction, affinity matrix, and optional original text detection loss functions:

$$\mathcal{L}_{TGA} = \alpha_1 \mathcal{L}_{dice} + \alpha_2 \mathcal{L}_{group} + \alpha_3 \mathcal{L}_{det}, \tag{6}$$

where $\mathcal{L}_{det}$ is the sum of original text detection loss functions. α_1, α_2 and α_3 represent the weights of GMP, AMP, and original detection loss functions respectively.

3.5. Cascade TGA for Word-based Text Detector

Different from directly grouping detected lines into paragraphs, grouping detected words into paragraphs is actually a hierarchical grouping task. It's more challenging as the words in a paragraph might exhibit greater diversity and be more spatially distant from each other. Thanks to the compatibility of the proposed TGA, we can simply cascade more than one TGA to address this problem, referred as Cascade TGA. As shown in Figure 3, we cascade two TGAs, named Word TGA and Line TGA. Detected text word regions and image features are first fed into Word TGA to predict line masks. The predicted line masks are subsequently fed into Line TGA to predict paragraph masks and the affinity matrix. This design complements the mid-level supervision and showcases the compatibility of TGA.

4. Experiments

In this section, we set up comprehensive experiments and analysis on TGA. We incorporate TGA into diverse pre-trained text detectors and compare their performance with a series of competitive baseline methods. To explore the possibility of parameter-effectively adapting the text detectors, we also compare the TGA's performance on diverse datasets under the condition that the parameters of the text detectors are either frozen or not. Finally, we validate the effectiveness of TGA's components by extensive ablation studies and visualizations.

4.1. Experiment Settings

Baselines. Since the layout analysis can build on word instances and text line instances, we compare methods separately on word-based layout analysis and line-based layout analysis. Since there is little literature specially studying word-based layout analysis, we solely draw on the Unified Detector [23] as our word-based baseline. As for line-based methods, we adapt baselines from [23], including Google Cloud OCR API [10], Max-DeepLab-Cluster [23, 33] and the Unified Detector [23]. Google OCR (GCP) API is a public commercial OCR engine. Max-DeepLab-Cluster builds two separate Max-DeepLab models to detect text lines and paragraphs and reassign the affinities between instances by post-processing. The Unified Detector stands out as the first unified model to detect text instances and layout simultaneously, achieving the previous state-of-the-art performance. Note that in the following comparison, we refer to the Unified Detector with 384 queries for its best performance.

Pre-trained text detectors. We applied TGA to the following four models pre-trained for text detection with diverse network structures and text region representations: DBNetpp [17], Deepsolo [38], KNet [42], and MaskDINO [14]. In terms of the network architecture, DBNetpp utilizes a fully convolutional network, while MaskDINO and DeepSolo employ query-based transformers. KNet introduces dynamic kernels with the convolutional network. Regarding the text region representations, DBNetpp produces semantic masks. KNet and MaskDINO model the text region as text instance masks. DeepSolo stands out as the state-of-the-art text spotting model, outputting Bezier Curve control points. Besides the differences resulting from their methods, in our experiments, DBNetpp is pre-trained as word-level text detectors, while DeepSolo serves as the word-level text spotting model. KNet and MaskDINO are pre-trained as line-level text detectors.

Datasets and metrics. HierText [23] is the scene text layout analysis dataset for our training and evaluation, which is well annotated with not only paragraph masks but also text line and word masks. It consists of 8,281, 1,724, and 1,634 images in training, validation, and test set, respectively. Following [23], we use Precision (P), Recall (R), F1 score (F), and Panoptic Quality (PQ) as our metrics for layout analysis and text detection on the HierText Dataset, where PQ is computed as the product of the F1 score and the average Intersection over Union of all True Positive pairs.

For text detection, CTW1500 [40], MSRA-TD500 [37], IC19-LSVT [31] and HierText, totally 39,581 images, are the datasets used for the pre-training of line-level text detector. As for word-level text detection, we proceed with the continual pre-training of the DBNetpp model and directly draw the off-the-shelf parameters from the DeepSolo model. We use Average Precision (AP) and the Harmonic Mean (H-mean) as the text detection metrics. We report more details of datasets, metrics and pre-training in our supplementary material.

Models	Frozen Text Detector	Trainable Params / Total Params	Val				Test			
			Instance		Paragraph		Instance		Paragraph	
			F	PQ	F	PQ	F	PQ	F	PQ
Word-based										
Unified Detector	✘	88.2M / 88.2M	**78.70**	**62.84**	50.11	39.26	**80.03**	**63.89**	51.54	40.33
TGA + Deepsolo-ViTAE-S	✔	8.7M / 44.8M	63.45	47.20	55.97	40.32	64.34	47.94	55.59	40.18
TGA + DBNetPP-R50	✔	8.0M / 34.1M	76.91	58.70	**56.23**	**42.23**	78.07	59.60	**56.32**	**42.23**
Line-based										
GCP API	-	-	-	-	-	-	-	56.17	-	46.33
Max-DeepLab-Cluster	-	-	-	-	-	-	-	62.23	-	52.52
Unified Detector	✘	88.2M / 88.2M	78.44	61.04	67.76	52.84	79.91	62.23	68.58	53.60
TGA + KNet-R50	✘	43.4M / 43.4M	77.04	59.08	71.90	55.27	77.88	60.03	71.37	55.04
TGA + MaskDINO-R50	✘	52.2M / 52.2M	76.41	58.91	75.46	58.28	77.43	59.89	75.41	58.33
TGA + KNet-R50	✔	5.9M / 43.4M	76.66	58.68	71.31	54.58	77.37	59.54	70.34	54.05
TGA + MaskDINO-R50	✔	5.9M / 52.2M	84.10	64.51	74.67	57.62	84.92	65.35	74.22	57.47
TGA + KNet-Swin-B	✔	5.9M / 106.2M	79.10	60.64	73.58	56.41	79.67	61.40	72.49	55.90
TGA + MaskDINO-Swin-B	✔	7.1M / 123.0M	**85.84**	**66.50**	75.45	58.72	**86.65**	**67.27**	75.11	58.65
TGA† + MaskDINO-R50	✔	12.0M / 58.2M	85.00	65.60	**78.00**	**60.28**	86.18	66.63	**77.61**	**60.05**

Table 1. Results of word and line detection on the HierText Dataset for all models. *Word-based* refers to detects and groups word regions as paragraphs while *Line-based* does the same for text line regions. The fine-tune strategy and trainable parameter size is inapplicable for API or non-unified model. Deepsolo-ViTAE-S means the Deepsolo with ViTAE-S [36] as its backbone. KNet-R50 means the KNet with ResNet-50 [11] as its backbone. Swin-B denotes Swin-Base [21]. Other models' naming follows the same approach. TGA† means an enhanced TGA by adding more layers in Pixel Embedding Layers for more trainable parameters while other settings keep the same.

Hyperparameter setting. Hyperparameters of TGA network structure are configured as follows: the feature dimension, D, is set to 256, the number of self-attention layers is 3, the number of attention heads is 4, and the dimension of the hidden layers is 512. During the inference phase, we set the group threshold t as 0.8 for all these models. As for training hyperparameters, we mainly refer to the original text detector training hyperparameters, which are provided in our supplementary material.

4.2. Main Comparison

In this main comparison, we emphasize Paragraph F and PQ as our main metrics since they represent the performance of layout analysis. Instance F and PQ also are considered as better instance detection helps layout analysis. As shown in Table 1, we incorporate TGA into various pre-trained text detectors, achieving impressive results on both word-based and line-based layout analysis. On word-based comparison with the Unified Detector, with much smaller trainable parameter size and total parameter size (around 10% trainable parameters and one-half total parameters), both frozen DBNetpp and DeepSolo combined with TGA achieve on-par, even superior performance on Paragraph PQ. It's also noteworthy that, due to the limitation of the text detector's parameter size, our layout analysis results are produced given this worse instance-level performance, which further demonstrates our advantages in grouping semantic entities.

On line-based layout analysis, while applying TGA into different text detectors, we also involve the settings of whether or not to freeze the text detector. As shown in the bottom section of Table 1, under all possible configurations, all our models consistently outperform all previous models on the Paragraph PQ of the test set to varying degrees. Among them, even the lightest model, TGA with frozen KNet-R50 outperforms Unified Detector on Paragraph PQ by 2.7% relatively. For the most powerful model, the TGA†, adding more trainable parameters in Pixel Embedding Layers to get the enhanced pixel embedding map, achieves 60.05 PQ at the paragraph level with frozen MaskDINO-R-50. It is a 12.0% relative improvement compared to the Unified Detector. For our models with ResNet-50, we notice the fully fine-tuned models gain slightly better performance than the models with frozen text detectors on paragraph-level metrics, which is further studied in the following section. We further show that TGA is robust to different annotation levels under the same model, detailed in supplementary material. In summary, TGA is seamlessly compatible with various pre-trained text detectors and achieves superior layout analysis performance, even given the limitation of parameter size and worse detected text instances.

4.3. Comparison of Fine-Tuning Strategies

We study the effect of different fine-tuning strategies when using TGA. We fine-tune the same pre-trained text detectors on the HierText Dataset for layout analysis and evaluate them on both text detection datasets and layout analysis datasets. We adopt two different fine-tune strategies: (1) only fine-tune TGA's parameters and freeze the orig-

Models	Frozen Text Detector	Trainable Params	Training Time	CTW1500	MSRA-TD500	IC19-LVST	HierText Val	
				Line AP	Line AP	Line H-mean	Line PQ	Paragraph PQ
TGA + KNet-R50	✘	43.4M	11.5h	43.12	47.54	69.89	**60.59**	**51.85**
	✔	5.9M	11.0h	**56.37**	**57.38**	**76.01**	57.80	49.83
TGA + KNet-Swin-B	✘	106.2M	20.2h	44.00	57.25	76.90	**62.97**	**54.17**
	✔	5.9M	13.2h	**58.28**	**59.81**	**78.69**	59.75	50.77
TGA + MaskDINO-R50	✘	52.2M	31.8h	41.46	51.80	59.19	59.88	**59.19**
	✔	5.9M	23.8h	**61.18**	**59.54**	**82.62**	**65.83**	54.95
TGA + MaskDINO-Swin-B	✘	123.0M		Out of Memory				
	✔	7.1M	27.3h	**61.25**	**59.58**	**82.63**	**66.37**	**58.56**

Table 2. Comparison between different fine-tuning strategies: frozen text detector and full fine-tuning. Out of Memory denotes this error occurs when fully fine-tuning the TGA + MaskDINO-SWin-B on a 8 * V100-32G workstation with minimal batch size. The values on HierText Val set slightly differ from ones reported in Table 1 for slightly different annotations, detailed in supplementary material.

Frozen Text Detector	Components		Line PQ	Paragraph PQ
	Pix. Emb. Map	GMP		
✘	✘	✘	**58.74**	49.99
	✘	✔	58.25	54.55
	✔	✔	58.55	**55.03**
✔	✘	✘	58.40	49.09
	✘	✔		51.20
	✔	✔		**54.27**
	✔	◯	58.40	47.32

Table 3. Ablation studies of all modules in TGA. ◯ means using line masks as the ground truth of GMP.

BCE Loss	Dice Loss	Paragraph			
		P	R	F	PQ
✔	✔	76.47	65.06	70.31	53.72
✔	✘	65.89	36.44	46.93	35.68
✘	✔	**76.68**	**66.23**	**71.07**	**53.88**

Table 4. Ablation studies of category for GMP loss function.

inal text detector, referred to as *frozen text detector*. (2) fine-tune all parameters, including original text detector and TGA, referred to as *full fine-tuning*. Results are shown in Table 2. We can obverse full fine-tuning strategy consistently enhances the layout analysis performance of all models. The most notable improvement is seen in MaskDINO-R50, where the full fine-tuning strategy surpasses the frozen text detector strategy by a significant 4.2 on Paragraph PQ.

On the other hand, the benefit of frozen text detector is also obvious in that it significantly saves the training time and GPU memory for the less trainable parameters, especially for the ones that have heavier backbones. Frozen detector detector strategy saves 34.7% training time for KNet-Swin-B and prevents out-of-memory error for MaskDINO-Swin-B fine-tuning. Another advantage of frozen text detector is that it can keep the generalized detection ability obtained from pre-training and prevent overfit on the layout analysis dataset, especially given the fact that the current size of the layout analysis dataset is much smaller than broad text detection datasets. As shown in the middle columns of Table 2, on CTW1500, MSRA-TD500 and LSVT detection dataset, the full fine-tuning models' performances rapidly drop on text detection metrics. Representatively, full fine-tuning KNet-R50 relatively drops 23.5%, 17.1%, and 8.1%, respectively on the three text detection datasets compared with frozen one. It shows the trade-off choice of full fine-tuning or freezing text detector, where the former helps boost better layout analysis performance on the specific dataset, and the latter saves computing resources and benefits robustness on broader scenarios.

4.4. Ablation Study

To validate the efficacy of the key components in our proposed TGA, we conducted a series of experiments where we respectively replace the Pixel embedding map with a single scale image feature and removed the Group Mask Prediction feature. These experiments were performed under two different fine-tuning strategies. We compared single TGA with Cascade TGA in the training process and analyze these components sequentially in the following paragraphs.

Pixel embedding map. Pixel embedding map is designed to encode high-level features for layout contextual information while preserving low-level features for text instance details. As shown in Table 3, the removal of the Pixel embedding map, regardless of whether the text detector is frozen or fully fine-tuned, negatively impacts the performance of layout analysis. This suggests that our proposed components are generally effective across different fine-tuning strategies. The results also indicate that components play different roles under different fine-tuning strategies. The removal of the pixel embedding map leads to a 3.03 drop in the Paragraph PQ under a frozen text detector, as compared to a 0.48 drop under full fine-tuning. It indicates distinct scale features are needed between text detection pre-training and layout analysis fine-tuning. Hence, the removal of the pixel embedding map under a frozen text detector results in the loss of global contextual information, which significantly impairs the layout analysis performance.

Group Mask Prediction. Group Mask Prediction (GMP) is a novel component in our Text Grouping Adapter, which

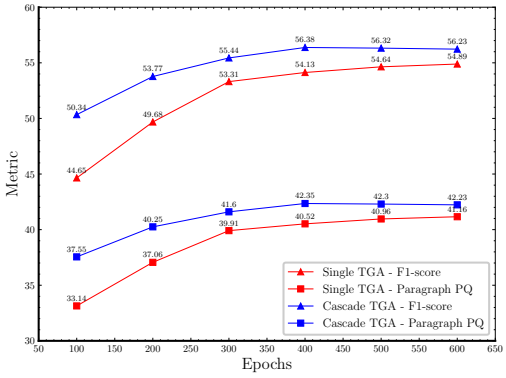

Figure 4. Comparison between different stages of single TGA and Cascade TGA training under frozen text detector strategy.

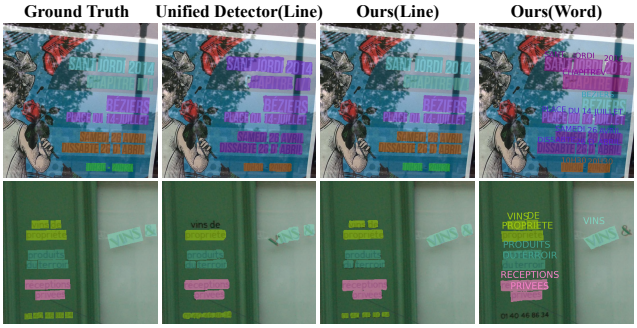

Figure 5. Visualization of results on the validation set of the Hier-Text Dataset: from left to right, the sequence includes the ground truth, line-based Unified Detector, TGA + MaskDINO-Swin-B and TGA + DeepSolo-ViTAE-S. (Zoom in for the best view)

leverages a one-to-many assignment to encourage text instances belonging to the same group to predict the same group mask. This process enables text instances to learn group-level features that encompass more contextual information. As shown in Table 3, models with GMP consistently outperform those without it. Particularly under the full fine-tuning setting, solely add Group Mask Prediction gains 4.56 on Paragraph PQ. When fine-tuned with the frozen text detector, the improvement is slightly less pronounced due to the limitation of detection-biased features. With the assistance of the pixel embedding map, we still observe a substantial improvement of 5.18 on Paragraph PQ.

Group mask v.s. affinity matrix. The question arises as to *why GMP helps the prediction of the affinity matrix given both group masks and affinity are derived from group annotations?* To answer this question, we initially ablate the supervision used in GMP. As shown in the last row of Table 3, replacing the one-to-many assigned group mask with the one-to-one assigned line mask in the prediction causes the drop back to baseline performance levels. This confirms the advantages of GMP from the unique group masks and the one-to-many assignment, not merely from simple mask prediction. Unlike the affinity matrix, which depicts layout through pairwise relationships, the group mask represents layout through the collective representation of all instances within the group, thereby optimizing the inter-instance distances globally. Our investigation into mask prediction loss combinations for GMP, shown in Table 4, demonstrates that relying solely on binary cross-entropy loss, computed pixel-wise, leads to a drastic decline in layout analysis performance. Conversely, employing dice loss, which evaluates the holistic statistical resemblance between predicted group masks and their true counterparts, significantly elevates layout analysis outcomes. When using dice loss, which focus on the holistic statistic similarity between predicted group masks and ground-truth ones, it greatly boosts the performance of layout analysis. These findings validate the GMP design's capacity to capture a more global and holistic representation of group instance information. We further visualize the clustering effect of GMP and provide more ablations in supplementary material.

Cascade TGA. In Figure 4, we evaluate DBNetpp with single TGA and Cascade TGA, respectively, at various training stages with the frozen text detector. The results indicate that introducing Cascade TGA not only accelerates convergence, but also produces superior results compared to single TGA. This structural prior introduced by Cascade TGA reduces the problem of clustering words into paragraphs to a two-stage problem: first clustering words into lines and then clustering lines into paragraphs.

4.5. Qualitative Results

We compare visualizations of generated layouts between Unified Detector, our line-based MaskDINO with TGA and our word-based DeepSolo with TGA in Figure 5. We observe that our line-based model performs better in the details, with more complete text masks and accurate grouping results. It is also noteworthy that our DeepSolo with TGA simultaneously produces the result of text detection, recognition, and layout analysis as a unified model. Facing more challenging in word-based layout analysis, it shows slight defects like losing the capture of small-size texts.

5. Conclusion

We present Text Grouping Adapter (TGA), a versatile module that enhances the capability of various pre-trained text detectors to serve for layout analysis. TGA takes text masks and image features as inputs to predict text group masks from text instance features. It facilitates the full exploitation of well-trained text detectors and easily obtainable text detection data. This work provides insights and a practical solution for aligning layout analysis with text detection and also has the potential to model general object relations.

Acknowledgements

Tianci Bi and Nanning Zheng were supported in part by NSFC under grant No. 62088102.

References

[1] Galal M Binmakhashen and Sabri A Mahmoud. Document layout analysis: a comprehensive survey. *ACM Computing Surveys (CSUR)*, 52(6):1–36, 2019. 1

[2] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven. Photoocr: Reading text in uncontrolled conditions. In *Proceedings of the ieee international conference on computer vision*, pages 785–792, 2013. 1

[3] Théodore Bluche and Ronaldo Messina. Gated convolutional recurrent neural networks for multilingual handwriting recognition. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, pages 646–651. IEEE, 2017. 1

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 4

[5] Qiang Chen, Xiaokang Chen, Jian Wang, Shan Zhang, Kun Yao, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group detr: Fast detr training with group-wise one-to-many assignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6633–6642, 2023. 4

[6] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. 3

[7] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. 2021. 3

[8] Daniel Hernandez Diaz, Siyang Qin, Reeve Ingle, Yasuhisa Fujii, and Alessandro Bissacco. Rethinking text line recognition models. *arXiv preprint arXiv:2104.07787*, 2021. 1

[9] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023. 3

[10] Google. Google cloud platform text detection api, 2023. 5

[11] 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. 6

[12] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19702–19712, 2023. 4

[13] Joonho Lee, Hideaki Hayashi, Wataru Ohyama, and Seiichi Uchida. Page segmentation using a convolutional neural network with trainable co-occurrence features. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1023–1028. IEEE, 2019. 1, 2

[14] 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. 5

[15] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 19730–19742. PMLR, 2023. 3

[16] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13094–13102, 2023. 1

[17] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. Real-time scene text detection with differentiable binarization. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11474–11481, 2020. 1, 2, 5

[18] 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, 2

[19] 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 *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[20] Yuliang Liu, Chunhua Shen, Lianwen Jin, Tong He, Peng Chen, Chongyu Liu, and Hao Chen. Abcnet v2: Adaptive bezier-curve network for real-time end-to-end text spotting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8048–8064, 2021. 1

[21] 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. 6

[22] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *Proceedings of the European conference on computer vision (ECCV)*, pages 20–36, 2018. 1, 2

[23] 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 *CVPR 2023*, pages 1049–1059, 2022. 1, 2, 5

[24] Siyang Qin, Alessandro Bissacco, Michalis Raptis, Yasuhisa Fujii, and Ying Xiao. Towards unconstrained end-to-end text spotting. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4704–4714, 2019. 2

[25] Zobeir Raisi, Mohamed A Naiel, Georges Younes, Steven Wardell, and John S Zelek. Transformer-based text detection in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3162–3171, 2021. 2

[26] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Ad-*

vances in neural information processing systems, 30, 2017. 3

[27] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018. 3

[28] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):651–663, 2018. 3

[29] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, pages 1162–1167. IEEE, 2017. 1, 2

[30] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2550–2558, 2017. 1

[31] 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, 5

[32] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985. 3

[33] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5463–5474, 2021. 5

[34] Tianwei Wang, Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen, Yaqiang Wu, Qianying Wang, and Mingxiang Cai. Decoupled attention network for text recognition. In *Proceedings of the AAAI conference on artificial intelligence*, pages 12216–12224, 2020. 1

[35] Wenhai Wang, Enze Xie, Xiang Li, Wenbo Hou, Tong Lu, Gang Yu, and Shuai Shao. Shape robust text detection with progressive scale expansion network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9336–9345, 2019. 2

[36] 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. 6

[37] 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. 5

[38] 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, 2, 5

[39] 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, 2023. 1

[40] 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. 5

[41] Chongsheng Zhang, Yuefeng Tao, Kai Du, Weiping Ding, Bin Wang, Ji Liu, and Wei Wang. Character-level street view text spotting based on deep multisegmentation network for smarter autonomous driving. *IEEE Transactions on Artificial Intelligence*, 3(2):297–308, 2022. 1

[42] 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. 2, 3, 5

[43] Xiang Zhang, Yongwen Su, Subarna Tripathi, and Zhuowen Tu. Text spotting transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9519–9528, 2022. 2

[44] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4159–4167, 2016. 1, 2

[45] Zhizheng Zhang, Xiaoyi Zhang, Wenxuan Xie, and Yan Lu. Responsible task automation: Empowering large language models as responsible task automators. *arXiv preprint arXiv:2306.01242*, 2023. 1

[46] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, 2019. 1, 2

[47] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 3

[48] Yiqin Zhu, Jianyong Chen, Lingyu Liang, Zhanghui Kuang, Lianwen Jin, and Wayne Zhang. Fourier contour embedding for arbitrary-shaped text detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3123–3131, 2021. 2