Hi-SAM: Marrying Segment Anything Model for Hierarchical Text Segmentation

Maoyuan Ye, Jing Zhang, *Senior Member, IEEE*, Juhua Liu, *Member, IEEE,* Chenyu Liu, Baocai Yin, Cong Liu, Bo Du, *Senior Member, IEEE*, Dacheng Tao, *Fellow, IEEE*

Abstract—The Segment Anything Model (SAM), a profound vision foundation model pretrained on a large-scale dataset, breaks the boundaries of general segmentation and sparks various downstream applications. This paper introduces **Hi-SAM**, a unified model leveraging SAM for hierarchical text segmentation. Hi-SAM excels in segmentation across four hierarchies, including *pixel-level text*, *word*, *text-line*, and *paragraph*, while realizing *layout analysis* as well. Specifically, we first turn SAM into a high-quality pixel-level text segmentation (TS) model through a parameter-efficient fine-tuning approach. We use this TS model to iteratively generate the pixel-level text labels in a semi-automatical manner, unifying labels across the four text hierarchies in the HierText dataset. Subsequently, with these complete labels, we launch the end-to-end trainable Hi-SAM based on the TS architecture with a customized hierarchical mask decoder. During inference, Hi-SAM offers both automatic mask generation (AMG) mode and promptable segmentation (PS) mode. In the AMG mode, Hi-SAM segments pixel-level text foreground masks initially, then samples foreground points for hierarchical text mask generation and achieves layout analysis in passing. As for the PS mode, Hi-SAM provides word, text-line, and paragraph masks with a single point click. Experimental results show the state-of-the-art performance of our TS model: 84.86% fgIOU on Total-Text and 88.96% fgIOU on TextSeg for pixel-level text segmentation. Moreover, compared to the previous specialist for joint hierarchical detection and layout analysis on HierText, Hi-SAM achieves significant improvements: 4.73% PQ and 5.39% F1 on the text-line level, 5.49% PQ and 7.39% F1 on the paragraph level layout analysis, requiring $20\times$ fewer training epochs. The code is available at Hi-SAM.

Index Terms—Hierarchical Text Segmentation, Unified Model, Segment Anything Model

Segmens across:
1. Pixel level
2. word level
3. Line level
4. Paragraph level

During inference:
- AMG - automatic mask generation
- PS - promptable segmentation

1 Introduction

Text effectively conveys rich high-level semantics through various hierarchies, spanning from pixel-level text to word, text-line, and paragraph. These hierarchies play crucial roles in diverse applications. For instance, pixel-level text segmentation (TS)[1] [1], [2], [3] is useful for font style transfer, scene text removal, and text editing. Additionally, word and text-line detection are important for text entity extraction and recognition, while visual text understanding [4] may necessitate a geometric layout, such as paragraph grouping results [5], [6]. Existing works [1], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] typically concentrate on specific hierarchies, lacking a comprehensive approach to handle multi-grained textual information within a unified system. This raises the question: ***Is it feasible to devise a cohesive segmentation framework capable of effectively managing diverse text hierarchies, encompassing pixel-level text, word, text-line, and paragraph?*** A major challenge lies in the limited availability of real-world data annotated with all four text hierarchies. Table 1 briefly summarizes existing datasets, revealing a predominant focus on word level. Only HierText [5] provides hierarchical annotations of word, text-line, and paragraph, yet it lacks pixel-level text annotations and the training set volume is also limited. This suggests the need for data annotation of all required text hierarchies as well as data-efficient training in hierarchical text segmentation.

Recently, the Segment Anything Model (SAM) [24] emerged as a foundational vision model for general image

TABLE 1 – Current datasets lack comprehensive and cohesive text hierarchies. Our study fills this gap by introducing pixel-level text labels to HierText, establishing it as the pioneering dataset with four distinct text hierarchies. The statistics of words are achieved from training and validation sets.

Datasets	Training Images	Words (avg/total)	Hierarchies			
			Pixel-level Text	Word	Line	Paragraph
MSRA-TD500 [17]	300	6.9/3.5K	✘	✘	✔	✘
IC15 [18]	1,000	4.4/6.5K	✘	✔	✘	✘
CTW1500 [19]	1,000	6.7/10K	✘	✘	✔	✘
Total-Text [20]	1,255	7.4/11K	✔	✔	✘	✘
TextSeg [1]	2,646	3.9/12K	✔	✔	✘	✘
IC19-ArT [21]	5,603	8.9/50K	✘	✔	✘	✘
IC19-LSVT [22]	30,000	8.1/243K	✘	✘	✔	✘
TextOCR [23]	21,778	32.1/903K	✘	✔	✘	✘
HierText [5]	8,281	103.8/1.2M	✘	✔	✔	✔
HierText* (w/ SAM-TS)	8,281	103.8/1.2M	✔	✔	✔	✔

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFC2705700 and 2022YFB4500600, in part by the National Natural Science Foundation of China under Grants U23B2048, 62076186 and 62225113, in part by the Innovative Research Group Project of Hubei Province under Grant 2024AFA017, and in part by the Science and Technology Major Project of Hubei Province under Grant 2024BAB046. The numerical calculations in this paper have been done on the supercomputing system in the Supercomputing Center of Wuhan University. Corresponding authors: Juhua Liu, Bo Du (e-mail: {liujuhua, dubo}@whu.edu.cn).

M. Ye, J. Zhang, J. Liu, and B. Du are with the School of Computer Science, National Engineering Research Center for Multimedia Software, Institute of Artificial Intelligence, and Hubei Key Laboratory of Multimedia and Network Communication Engineering, Wuhan University, Wuhan, China (e-mail: {yemaoyuan, liujuhua, dubo}@whu.edu.cn, jingzhang.cv@gmail.com).

C. Liu, B. Yin and C. Liu are with the iFLYTEK Research, IFLYTEK CO. LTD., China (email: {cyliu7, bcyin, congliu2}@iflytek.com).

D. Tao is with the College of Computing & Data Science at Nanyang Technological University, #32 Block N4 #02a-014, 50 Nanyang Avenue, Singapore 639798 (e-mail: dacheng.tao@ntu.edu.sg).

1. We view *text segmentation* in previous works [1], [2], [3] and *pixel-level text segmentation* as the same task. To avoid confusion with other text hierarchies, we always use *pixel-level text segmentation* in this paper.

Fig. 1 – Hi-SAM can perform pixel-level text segmentation, word segmentation, text-line segmentation, paragraph segmentation, and layout analysis in automatic mask generation mode. Hi-SAM also supports promptable segmentation. Given a single-point click on one word, Hi-SAM predicts the corresponding word, text-line, and paragraph masks.

segmentation, leveraging billion-scale class-agnostic mask labels for training. SAM exhibits promptable and multi-grained segmentation capabilities, generating object masks based on point, bounding box, or coarse mask inputs. Notably, SAM demonstrates robust transferability in zero-shot scenarios and task adaptation across diverse domains, including common image segmentation [24], [25], medical image analysis [26], [27], and video processing [28], [29]. Additionally, SAM facilitates scalable data annotation [30]. Motivated by SAM's excellence, we aim to answer the above question by harnessing its capabilities to build a strong hierarchical text segmentation model.

To this end, we propose Hi-SAM, a text-centric hierarchical segmentation model that leverages the inherent knowledge of SAM as well as existing fragmented training data with incomplete text hierarchy annotations. In the automatic mask generation (AMG) mode, SAM employs class and object-agnostic grid points as prompts for segmenting objects, ranging from groups to individual parts, based on the best confidence across scales. However, for hierarchical text segmentation, the text must be considered as the sole foreground and it is required to explicitly and simultaneously generate word, text-line, and paragraph masks. To address this issue, we propose to segment the pixel-level text masks at first, and then sample points from them as prompts for generating word, text-line, and paragraph masks subsequently. For the first step, we surprisingly discovered that SAM can function as a competitive TS model with minimal customizations. Unlike SAM's original mask decoder which takes local point or box-based prompts for segmentation, we propose to use global prompts, enabling the mask decoder to perceive the overall context of image texts. Technically, we convert the image encoder features to a sequence of tokens and use them as implicit prompts to tune the mask decoder. Furthermore, we identify the bottleneck in SAM's mask decoder for segmenting text with fine details as the small mask feature size. Consequently, we introduce a simple yet effective module to empower the mask decoder in delivering text features at the original input size. We call this simple baseline model SAM-TS and demonstrate that it surpasses state-of-the-art (SOTA) TS models across various existing datasets. In addition, inspired by SAM, we adopt an iterative strategy to first train SAM-TS on HierText [5] using a small set of manually labeled data, and then annotate additional pixel-level text in HierText by leveraging the trained SAM-TS with minimal manual intervention.

Next, we launch Hi-SAM, an end-to-end trainable model for hierarchical text segmentation, by extending SAM-TS. Hi-SAM incorporates a hierarchical mask decoder (H-Decoder) parallel to SAM-TS's mask decoder (S-Decoder), enabling segmentation at word, text-line, and paragraph levels. Foreground points are randomly sampled from the predicted pixel-level text mask and transformed into point prompts using SAM's prompt encoder. For each point prompt that belongs to a certain text, the H-Decoder uses three output tokens to predict its hierarchical masks. Hi-SAM supports both the AMG mode and promptable segmentation (PS) mode as shown in Fig. 1. In the AMG mode, Hi-SAM initially segments the pixel-level text mask, samples point prompts, and generates hierarchical masks. In the PS mode, a single-point click on a word prompts Hi-SAM to generate the word mask, along with text-line and paragraph masks at the word's location. Additionally, Hi-SAM provides layout analysis as a by-product by calculating Intersection-over-Union (IoU) matrix of paragraph masks and using it for clustering words.

In summary, our main contributions are three-fold:

- We propose a simple yet effective method to render SAM as a competitive TS baseline model, *i.e.*, SAM-TS. It surpasses SOTA TS methods, producing high-resolution masks logits. We further employ it to semi-automatically annotate pixel-level text labels for HierText, making it the first real dataset featuring mask annotations across four text hierarchies.
- We introduce Hi-SAM that builds upon SAM-TS with simple add-ons and is trained end-to-end on the enhanced HierText. Hi-SAM is the first method for hierarchical text segmentation, covering pixel-level text, word, text-line, and paragraph levels, and supporting both AMG and PS modes.
- Our SAM-TS and Hi-SAM offer compelling performance on public and challenging datasets. SAM-TS sets new records on Total-Text (84.86% fgIOU) and TextSeg (88.96% fgIOU). Hi-SAM surpasses previous

specialists for joint text detection and layout analysis on HierText by a large margin while requiring 20× fewer training epochs.

The paper is structured as follows: Section 2 provides a brief review of related works. Section 3 introduces the proposed method. Extensive experimental results are presented in Section 4, followed by a discussion on Hi-SAM's limitations in Section 5. The paper concludes in Section 6.

2 Related Work

2.1 Specialist Models for Different Text Hierarchies

(i) Pixel-level Text. Pixel-level text segmentation requires segmenting all text strokes from the background. It presents a fine-grained per-pixel binary segmentation challenge. The pixel of text is considered as the foreground. Existing datasets primarily target scene text and designed text of large or medium sizes, such as Total-Text [20] and TextSeg [1]. To address the limited training data, some methods integrate text recognizers to enhance multi-scale features [8], [9], [31]. Additionally, the recent approach [10] adopts a weakly supervised approach by combining pixel-level text segmentation with text recognition. In contrast to prior studies, our work demonstrates the utilization of SAM's knowledge, leveraging single-scale features from the ViT encoder to achieve SOTA performance. **(ii) Word and Text-line.** Existing models predominantly focus on word-level scene text detection, with early approaches [32], [33] employing axis-aligned or multi-oriented anchor boxes for word localization. To address arbitrarily-shaped text representation, some methods [11], [34], [35], [36], [37], [38] utilize pixel-level segmentation masks. For example, DB [11] segments text kernel regions to distinguish instances, performs an adaptive binarization process, and employs post-processing to expand kernel contours for final results. Alternatively, some methods regress control points [13], [39], [40] or predict parameterized curves [41], [42] to fit text contours. However, these approaches often require two sets of weights for word and text-line detection, lacking a unified model. **(iii) Layout Analysis and Document Understanding.** Layout analysis primarily focuses on document scenarios, treating visually and semantically cohesive text blocks as either detection [43] or segmentation [44] objects. In addition to text, some document datasets [45] require distinguishing different elements, including title, list, table, and figure. Biswas *et al*. [46] establish a Mask-RCNN [47] based model for complex document layouts. DocSegTr [48] is presented as a Transformer-based model. Language-model-based approaches [49], [50] leverage Optical Character Recognition (OCR) tokens to group words into segments for semantic parsing. Recently, Long *et al*. [5] introduce a Unified Detector (UD) that addresses joint text detection and layout analysis in both natural and document scenarios. Since the elements like table scarcely exist in natural images, text is considered as the sole object. In UD, each object query is trained to segment words within a text line, and a layout branch generates an affinity matrix between different object queries for clustering into paragraphs. On the other hand, there are several segmentation-free methods for document recognition and understanding. For example, Donut [51], an OCR-free Transformer-based model, predicts a sequence of tokens that can be converted into target information in a structured form to realize visual document understanding. Similarly, Dessurt [52] also achieves document understanding with Transformer in the autoregressive text generation manner. DAN [53] is the first architecture for handwritten document recognition that is able to recognize text at document level while labeling logical layout information. DAN sequentially outputs characters, as well as logical layout tokens, labeling text parts with begin and end tags in an XML-like fashion. Pix2Struct [54] is proposed with a screenshot parsing pretraining objective based on the HTML format of web pages, enabling more effective pixel-to-text transformation for general-purpose visually-situated language understanding. To summarize, for high-level visual-language understanding tasks, such as visual question answering and information extraction, segmentation is not a necessity. However, recognition and visual-language understanding are not the focus of this work. In contrary, we concentrate on the pixel-level OCR tasks and concern the multi-granularity problem [24], [55] which is crucial in computer vision community.

Although text naturally exhibits multi-granularity across various hierarchies from pixel-level text, to word, text-line, and paragraph, prior to this research, a unified segmentation framework had not been explored. Previous studies have predominantly focused on specific text hierarchies. Their differing paradigms, architectures, and training strategies add complexity, raising both deployment and maintenance costs. In this work, we aim to explore the first unified framework for these text hierarchies, enabling one model to automatically complete these tasks. Furthermore, there is currently no text segmentation model that supports prompts. A promptable model could assist in annotation and interactive applications. For instance, with a single prompt, the model can segment an object according to designated text hierarchies, thereby reducing annotation time. Moreover, it can facilitate interactive pixel-level image editing at specified locations and text hierarchies. Hence, we investigate this automatic and promptable model, which opens up new application possibilities.

2.2 Adapting Vision Foundation Model for Text Tasks

Recent studies [56], [57], [58], [59] have delved into leveraging CLIP [60] to improve backbone representations for scene text detection and spotting. Specifically, oCLIP [57] introduces a weakly supervised pretrained network aligning visual and partial textual information for scene text detection and spotting. In contrast, TCM [58] adapts the CLIP model for scene text detection through visual prompt tuning. Additionally, for enhanced scene text recognition, CLIP-OCR [61] proposes a symmetrical distillation strategy capturing linguistic knowledge in the CLIP text encoder. In our work, we advance this field by developing a hierarchical and promptable segmentation framework using SAM [24].

2.3 Segment Anything Model and Follow-ups

SAM [24] is a pioneering vision model for general image segmentation, exhibiting remarkable generalization capabilities through large-scale pretraining. It extends its impact

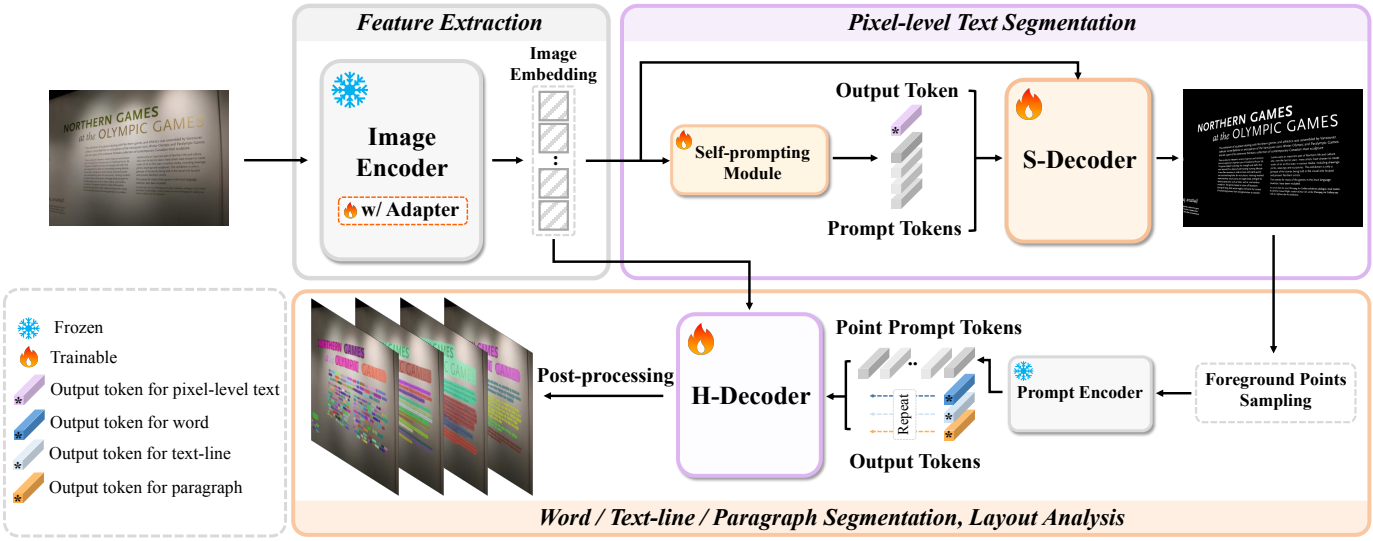

Fig. 2 – The overview of Hi-SAM. We show the automatic mask generation mode here. With the image embedding, for pixel-level text segmentation, self-prompting module generates implicit prompt tokens for the mask decoder (S-Decoder). Based on the pixel-level text mask, a certain number of foreground points are sampled and then embedded by the frozen prompt encoder. A customized hierarchical mask decoder (H-Decoder) segments word, text-line, and paragraph masks for each point prompt. Layout analysis can be achieved with the hierarchical outputs from the H-Decoder in passing.

to diverse downstream tasks such as image matting [62], 3D segmentation [63], video tracking [28], [29], and medical image segmentation [27], [64]. For example, SAM faces challenges in medical image segmentation due to domain gaps. Some methods [26], [65] leverage adapter tuning on SAM's ViT encoder to better adapt it to the medical imaging domain. PerSAM [66] and Matcher [67] introduce training-free segmentation frameworks based on SAM with one-shot learning. HQ-SAM [25] enhances SAM's object segmentation quality while maintaining zero-shot generalizability by applying a learnable high-quality output token on fused features of size 256×256 to improve mask details. Despite extensive studies in various domains, there is a notable gap in research on text-centric segmentation. In this work, we introduce the first unified segmentation framework for four text hierarchies. We identify the primary limitation of applying SAM to fine-grained text segmentation in the mask feature size. Our approach employs a simple yet effective method to achieve high-quality pixel-level text segmentation by providing high-resolution mask features.

3 Methodology

In this work, we propose Hi-SAM which excels in unified text segmentation spanning four hierarchies, including pixel-level text, word, text-line, and paragraph, while realizing layout analysis in passing. In the following subsections, we first briefly review the preliminaries about SAM in Sec. 3.1. Then, we provide a detailed description of Hi-SAM.

3.1 Preliminary

SAM [24] consists of a ViT backbone, a prompt encoder, and a lightweight mask decoder. The backbone (ViT-B/ViT-L/ViT-H) extracts image embeddings of size 64×64. SAM initializes the ViT with MAE [68] pretrained weights before training. The prompt encoder embeds different types of interactive positional cues into prompt tokens. The two-way Transformer-based mask decoder takes image embeddings, output tokens, and prompt tokens as inputs. The output tokens are similar to the object queries in DETR [69]. The mask decoder uses a multi-layer perceptron (MLP) to predict dynamic weights based on the output tokens, then applies the dynamic weights on up-sampled mask features in 256×256 spatial shape to generate mask results. Although SAM shows impressive generalization capabilities, it remains a class-agnostic model. How to leverage SAM for text-centric hierarchical segmentation is a pending issue.

3.2 Overview of Hi-SAM

Hi-SAM employs a unified framework for hierarchical text segmentation, initially tackling the global TS task and subsequently addressing local segmentation tasks. The overall architecture is depicted in Fig. 2. Hi-SAM contains five modules: 1) an image encoder from SAM, 2) a plug-in self-prompting module, 3) a pixel-level text mask decoder (S-Decoder) derived from SAM's mask decoder, 4) a frozen prompt encoder from SAM, and 5) a hierarchical mask decoder (H-Decoder) customized from SAM's mask decoder. For TS, we use the self-prompting module and S-Decoder to process the image embedding. Recognizing the fine details inherent in pixel-level text, we also devise a simple yet effective module to generate high-resolution pixel-level text mask features, thereby significantly enhancing performance. This base component of Hi-SAM for TS, *i.e.*, including the image encoder, self-prompting module, and S-Decoder, is referred to as SAM-TS. It is used to generate the labels of pixel-level text for HierText in a semi-automatic way. The details of SAM-TS are presented in Sec. 3.4. After obtaining the masks of pixel-level text, a certain number of foreground points are sampled as the bridge to word, text-line, and paragraph segmentation. With these foreground points, the prompt encoder and H-Decoder are used to predict masks

for the remaining three text hierarchies. This setting is referred to as the AMG mode while Hi-SAM also retains the PS mode of SAM. We introduce the details of the H-decoder in Sec. 3.5. Given the paragraph masks, Hi-SAM is capable of conducting layout analysis, which is described in Sec. 3.6. We detail the training and inference process of Hi-SAM in Sec. 3.7 and Sec. 3.8.

3.3 Feature Extraction

Since SAM's image encoder is not fully aware of text with fine details, directly applying the frozen image encoder for pixel-level text segmentation cannot achieve satisfactory results. To efficiently enhance the adaptability of the image encoder, an adapter-tuning approach [65] is adopted but is not constrained as the sole choice. The original parameters of SAM's image encoder are maintained frozen while two trainable adapter modules are inserted in each ViT block. Each adapter consists of a down-projection, a ReLU activation, and an up-projection sequentially. We use the default setting in [65]. Next, we devise a self-prompting module to generate implicit prompt tokens from the image embedding and turn SAM's mask decoder into the S-Decoder with high-resolution mask features.

3.4 Pixel-level Text Segmentation

Self-prompting Module. SAM's mask decoder takes prompt tokens with explicit positional cues as inputs. To steer the S-Decoder to segment pixel-level text within the input image, we propose to use the image embedding itself to generate implicit prompt tokens. The idea is that the required prompts for pixel-level text are already stored in image embedding, we just need to convert them into a series of representative prompt tokens. To achieve this, in the self-prompting module, we start by formulating an image tokenizer function T:

$$\boldsymbol{t} = T(\boldsymbol{I}), \tag{1}$$

which converts the image embedding $\boldsymbol{I} \in \mathbb{R}^{64\times64\times256}$ into tokens $\boldsymbol{t} \in \mathbb{R}^{N\times256}$. 64×64 denotes the spatial shape of image embedding after the ViT backbone, 256 is the feature dimension. N is the token length. Inspired by TokenLearner [70], we use a convolution block $ConvBlock$ and a $sigmoid$ function to process $\boldsymbol{I}$, generating a spatial attention map $\boldsymbol{A} \in \mathbb{R}^{64\times64\times N}$:

$$\boldsymbol{A} = sigmoid(ConvBlock(\boldsymbol{I})), \tag{2}$$

where $ConvBlock$ contains four convolution layers (kernel size 3×3, stride 1, padding 1) with the first one decreasing the dimension from 256 to N. GELU activation is inserted between convolution layers. Then we use $\boldsymbol{A}$, $\boldsymbol{I}$, and the spatial global average pooling operation ρ to obtain the tokens. We rewrite Eq. (1) as following:

$$\boldsymbol{t} = T(\boldsymbol{I}) = \rho(\gamma_A(\boldsymbol{A}) \odot \gamma_I(\boldsymbol{I})), \tag{3}$$

where $\odot$ is the element-wise product, γ_A and γ_I stand for reshape and broadcast operations on spatial attention map $\boldsymbol{A}$ and image embedding $\boldsymbol{I}$. To be more specific, for each channel of the spatial attention map $\boldsymbol{A}$, it is broadcasted to 256 dimension and multiplied with $\boldsymbol{I}$, achieving the spatially focused image embedding $\boldsymbol{I}' \in \mathbb{R}^{64\times64\times256}$. Then, a token with the shape of 1×256 can be calculated with the pooling operation ρ on $\boldsymbol{I}'$. In this way, with $\boldsymbol{A}$ owning N channels, the tokens $\boldsymbol{t} \in \mathbb{R}^{N\times256}$ can be obtained. The tokens $\boldsymbol{t}$ are further sent into a single Transformer decoder layer $TDecLayer$ for enhancing their representations:

$$\boldsymbol{t_{prompt}} = TDecLayer(Q = \boldsymbol{t}, K = \boldsymbol{I}, V = \boldsymbol{I}), \tag{4}$$

where $\boldsymbol{t_{prompt}} \in \mathbb{R}^{N\times256}$ denotes the final prompt tokens that will be fed into the S-Decoder. $TDecLayer$ takes $\boldsymbol{t}$ as query, $\boldsymbol{I}$ as key and value.

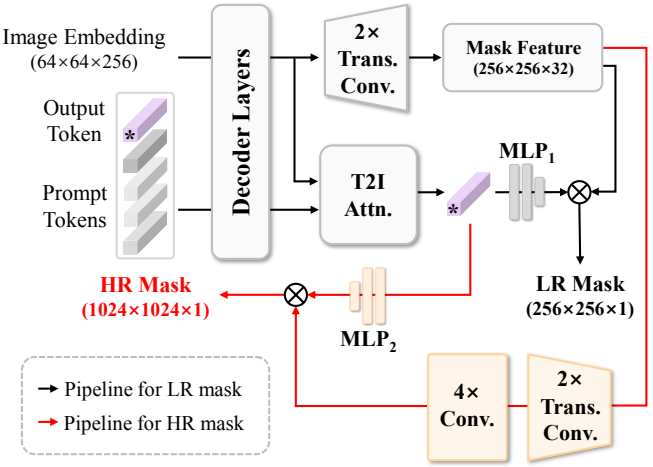

Fig. 3 – The structure details of S-Decoder. 'Trans. Conv.' and 'T2I Attn.' are transposed convolution and token-to-image attention, respectively. 'LR Mask' and 'HR Mask' denote the predicted low- and high-resolution mask logits, respectively.

S-Decoder with High-resolution Mask Features. We observe that the mask feature resolution mainly constrains the pixel-level text segmentation quality. Therefore, we design a simple yet effective method to provide high-resolution mask feature, significantly boosting the performance of pixel-level text with fine structures (See ablation results in Tab. 5).

The structure of the S-Decoder is presented in Fig. 3. The modules for predicting low-resolution masks in Fig. 3 inherit the parameters of SAM's mask decoder. Let $\boldsymbol{t_{s\_out}} \in \mathbb{R}^{1\times256}$ denote the inherited output token, which is the first slice of SAM's output tokens. For ease of description, we omit another token which is used for IoU prediction here. The concatenated output and prompt tokens $[\boldsymbol{t_{s\_out}}; \boldsymbol{t_{prompt}}] \in \mathbb{R}^{(1+N)\times256}$ are fed into S-Decoder. After the final token-to-image attention and transposed convolution, we obtain the updated output token $\hat{\boldsymbol{t}}_{\boldsymbol{s\_out}} \in \mathbb{R}^{1\times256}$ and up-sampled mask features $\boldsymbol{F} \in \mathbb{R}^{256\times256\times32}$, where 256×256 is the default mask feature size in SAM, 32 is the feature dimension. Then, $\hat{\boldsymbol{t}}_{\boldsymbol{s\_out}}$ is passed to an MLP that outputs a vector matching the dimension of $\boldsymbol{F}$. The mask logits $\boldsymbol{M}$ for pixel-level text is predicted by a spatially point-wise product between $\boldsymbol{F}$ and the MLP's output. During inference, $\boldsymbol{M}$ will be interpolated into the input image size and further processed with a threshold.

For high-resolution mask logits, we reuse the updated output token $\hat{\boldsymbol{t}}_{\boldsymbol{s\_out}}$ and mask features $\boldsymbol{F}$, while introducing a few lightweight modules. As illustrated in Fig. 3, we further up-sample $\boldsymbol{F}$ with another two transposed convolutional layers (kernel size 2×2, stride 2) and use

four subsequent convolutional layers (kernel size 3×3, stride 1, padding 1) for refining the feature. After this step, we achieve the high-resolution mask feature $\boldsymbol{F_{hr}}$ with 1024×1024 spatial size and 16 dimensions. We initialize a three-layer MLP and apply it on $\hat{\boldsymbol{t}}_{\boldsymbol{s\_out}}$ to produce a new vector. Similar to acquiring $\boldsymbol{M}$, S-Decoder predicts the final high-resolution mask logits $\boldsymbol{M_{hr}}$ of size 1024×1024 based on $\boldsymbol{F_{hr}}$ and the updated vector.

In contrast to image super-resolution [71], where the input RGB image resolution is increased at the encoder stage, our approach focuses on enhancing the resolution specifically at the mask feature level within the decoder. This strategy reduces the computational overhead typically incurred by processing high-resolution images in the encoder. Using SAM-TS-L as a case study, our method introduces high-resolution mask features at a minimal cost of 0.1 FPS (reducing from 2.9 FPS to 2.8 FPS on a single V100 GPU) and 0.79% FLOPs (increasing from 1426.6G to 1437.8G), resulting in a 14.24 fgIOU improvement on HierText (see Tab. 5). Without high-resolution mask features, the ViT encoder would require processing 4096×4096 resolution input images to achieve the same segmentation mask resolution, leading to unaffordable out-of-memory issues.

Semi-automatic Annotation of Pixel-level Text Masks. We leverage the obtained SAM-TS model to generate mask labels for HierText in a semi-automatic way. Initially, we select some samples in document, poster, and handwritten scenarios with dense and fine-grained texts. We annotate them in Adobe Photoshop with tools such as contrast adjustment, reverse, brush, and binarization. After annotating 418 images in the HierText training set, we combine these images and the training images of Total-Text to train a SAM-TS model with the ViT-L backbone, *i.e.*, SAM-TS-L. During training, we use augmentations including color jittering, random blur, random rotation, and large-scale jittering [72]. We use this model to label the remaining training images of HierText. During inference, we leverage a sliding window strategy to improve segmentation quality. We set the window size to 512×512 with a stride of 384. False positive and false negative regions are modified manually. Specifically, we retrain the model and update the segmentation masks for the remaining images when the ratio between annotated images and total HierText training images reaches $1/8$, $1/4$, and $1/2$. We only need to remove a few false positive segments as the model evolves. Finally, we use the latest model to label all the remaining images in the HierText training set. We also use SAM-TS-L to label the validation and testing images before manual inspection and modification. Fig. 4 shows some annotation samples generated by SAM-TS-L.

3.5 Word, Text-line, and Paragraph Segmentation

Different from the S-Decoder which takes implicit prompts, the H-Decoder receives point prompts from SAM's prompt encoder. We obtain word, text-line, and paragraph masks in a single H-Decoder that is customized from SAM's mask decoder. Supposing we have K foreground points on text, the prompt encoder processes them into prompt tokens $\boldsymbol{t_{point}} \in \mathbb{R}^{K \times N_p \times 256}$, where N_p is a newly introduced feature dimension for prompt tokens, following the default prompt encoding process as in SAM. H-decoder takes the

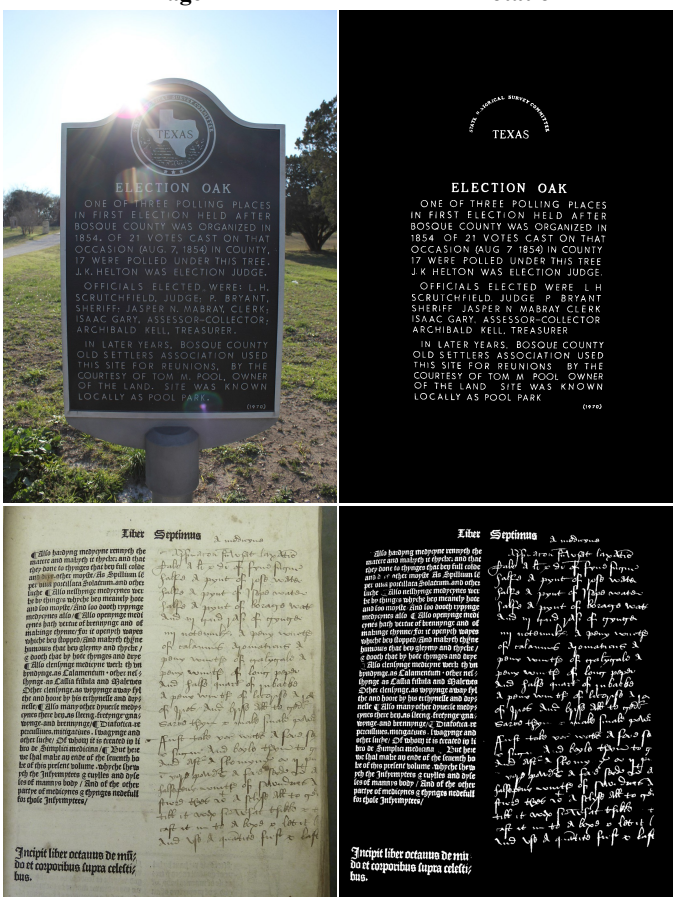

Fig. 4 – Annotation samples in HierText generated by SAM-TS automatically. Best view on screen with zooming in.

broadcasted output tokens $\boldsymbol{t_{h\_out}} \in \mathbb{R}^{K \times N_{out} \times 256}$ and point prompt tokens by concatenation ($[\boldsymbol{t_{h\_out}}; \boldsymbol{t_{point}}] \in \mathbb{R}^{K \times (N_{out}+N_p) \times 256}$) as input. N_{out} is the same output token number as in SAM's mask decoder. After the final token-to-image attention, we slice the last three output tokens and get $\hat{\boldsymbol{t}}_{\boldsymbol{h\_out}} \in \mathbb{R}^{K \times 3 \times 256}$. In the original SAM's mask decoder, the last three output tokens are used for multi-mask output, without specific task assignment. In H-Decoder, for each point prompt, we employ the three output tokens to take charge of the word, text-line, and paragraph segmentation in order. Applying point-wise product between output tokens and mask features, we get the mask logits $\boldsymbol{M_{w,l,p}}$ in the shape of $K \times 3 \times 256 \times 256$, containing the corresponding word, text-line, and paragraph masks where the points are located in. Note that for each point prompt, the word mask contains not only the selected word but also other words in the same text line. This is designed to ensure word recall during automatic mask generation for scenarios with dense text. UD [5] also adopts a similar design to ensure detection recall in its framework. In addition, to better separate adjacent word instances, the word output token is guided to segment word kernel regions. During inference, we follow the post-processing method in DB [11] to get complete word regions. In contrast, we do not conduct processing on text-line and paragraph labels.

Moreover, to improve the quality of word segmentation, we adopt a similar approach as in S-Decoder, *i.e.*, generating

mask features in higher resolution. Concretely, to save GPU memory usage, we use a convolutional layer with 1×1 kernel size to decrease the dimension of mask features from 32 to 16 at first. Then, we directly interpolate the mask features to have a spatial size of 384×384. We also use four convolutional layers with 3×3 kernel size to refine the mask features. A three-layer MLP is used to generate dynamic weights from the word-level output token. Finally, using the dynamic weights and refined mask features, we can get the word mask logits in higher resolution.

3.6 Layout Analysis

Layout analysis in this work aims at clustering words into different blocks, where texts are visually and semantically coherent in each block. Intuitively, layout analysis requires mining relationships between text objects. Since the outputs from the H-Decoder inherently present a hierarchical relationship, layout analysis can be accomplished in passing, without the need of a specially designed branch. Specifically, in the predictions of the H-Decoder, each text-line mask is accompanied by a word mask (containing words in this text-line) and a paragraph mask. If the corresponding paragraph masks of different text-lines have high IoU, we regard these text-lines and their corresponding words in the same paragraph, thereby achieving layout analysis. To be specific, we calculate the IoU matrix ($\in [0,1]^{K\times K}$) using paragraph masks in the shape of $K \times 256 \times 256$. Then, we group each pair of objects if their IoU metric exceeds 0.5. A union-find algorithm used in [5] is adopted to merge these connected nodes into clusters.

3.7 Training of Hi-SAM

During training, the learnable modules in Hi-SAM contain 1) the adapters in the ViT encoder, 2) the self-prompting module, 3) the S-Decoder, and 4) the H-Decoder. We optimize them in an end-to-end manner. In terms of the S-Decoder, we supervise both the low-resolution and high-resolution mask predictions. The loss function $\mathcal{L}_{lr}$ for low-resolution part is a linear combination of Focal loss [73], Dice loss [74], and IoU mean-square-error (MSE) loss in a ratio of 20:1:1, following [24]. Similarly, the loss function $\mathcal{L}_{hr}$ for the high-resolution part is also a linear combination of Focal loss, Dice loss, and MSE loss in the same ratio. For pixel-level text segmentation, the overall loss: $\mathcal{L}_{text} = \mathcal{L}_{lr} + \mathcal{L}_{hr}$.

In terms of the H-Decoder, we employ a training strategy contrapuntally. For each image which may contain up to hundreds of text-lines, we assume that some text-lines are similar to each other in appearance. We only use some samples for training the H-Decoder. Specifically, we randomly sample at most 10 text-lines in each image. Then, we calculate the intersection of text-line masks and the pixel-level text masks to determine the foreground points in each text-line. For each text-line, we randomly sample 2 foreground points. At last, we can get at most 20 foreground points along with the ground-truth word, text-line, and paragraph masks. We feed foreground points into the prompt encoder to generate prompt tokens for the H-Decoder. If there is no foreground point, the H-Decoder is guided to segment blank masks. Overall, for word segmentation, the loss function is defined as $\mathcal{L}_{word} = \mathcal{L}_{w\_lr} + \mathcal{L}_{w\_hr}$, where $\mathcal{L}_{w\_lr}$ and $\mathcal{L}_{w\_hr}$ are used to supervise the low-resolution and high-resolution word mask predictions. Each of them contains binary-cross-entropy (BCE) loss and Dice loss in a ratio of 1:1, following [25]. For text-line, the loss $\mathcal{L}_{line}$ is a linear combination of BCE loss, Dice loss, and IoU MSE loss in a ratio of 1:1:1. We use the same setting for paragraph loss $\mathcal{L}_{para}$. The predicted scores from IoU heads can be used to conduct mask non-maximum-suppression (NMS) during auto-mask generation. The final loss function $\mathcal{L}$ is formulated as:

$$\mathcal{L} = \mathcal{L}_{text} + \mathcal{L}_{word} + \mathcal{L}_{line} + 0.5 \times \mathcal{L}_{para}. \tag{5}$$

3.8 Inference of Hi-SAM

Automatic Mask Generation. For each input image, SAM-TS first generates pixel-level text mask, where the pixels of text are considered as foreground. Then, we randomly sample P foreground points, which are embedded into prompt tokens via the prompt encoder. H-Decoder uses them to predict dense word, text-line, and paragraph masks. We regard text-line as a pivot. We filter out the text-line masks whose predicted IoU scores are lower than 0.5. Their corresponding word and paragraph masks are also discarded. Next, we conduct Matrix NMS [75] on GPU using the remaining text-line masks and their IoU scores. The threshold for Matrix NMS is set to 0.5 by default. After NMS, we gather the final text-line masks, and their corresponding word and paragraph masks. Finally, we accomplish layout analysis based on the paragraph masks as described in Sec. 3.6. An additional NMS at the paragraph level can be applied if non-redundant paragraph masks are required. All the predicted masks can be interpolated into the same resolution as the input image. In this way, we achieve hierarchical text segmentation in a unified framework, and realize layout analysis in passing.

Promptable Segmentation. In this mode, the inference pipeline only involves the image encoder, prompt encoder, and H-decoder. The image encoder is used to extract embedding at first. For the points clicked by a user, the prompt encoder embeds them into prompt tokens and the H-Decoder generates word, text-line, and paragraph masks for each point. Since each word mask contains the words in one text-line as mentioned in Sec. 3.5, we compare the proximity between the clicked point and word instances to determine the selected word.

4 Experiments

4.1 Datasets and Evaluation Protocols

Total-Text [20] includes 1,255 training images and 300 testing images. It contains scene texts with arbitrary shapes. Total-Text provides binary mask labels of pixel-level text and polygon annotations of word instances.

TextSeg [1] contains 4,024 images, which are split into training, validation, and testing sets with 2,646, 340, and 1,038 images, respectively. It focuses on both scene texts and designed texts for pixel-level text segmentation.

COCO_TS [76] has 14,690 images from COCO-Text [77]. The COCO_TS pixel-level text masks are derived automatically from bounding box annotations in COCO-Text using a weakly supervised segmentation model trained on

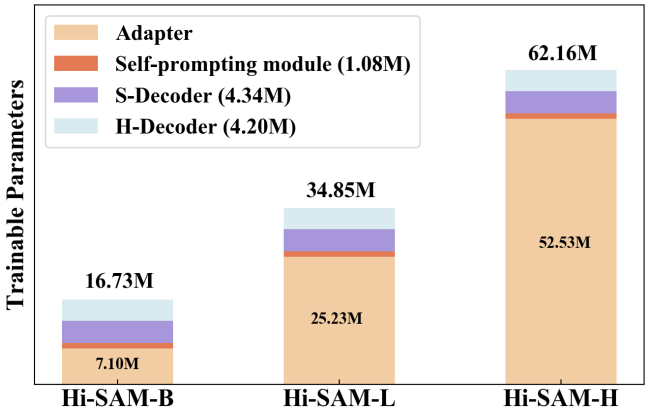

Fig. 5 – Trainable parameter statistics of Hi-SAM with ViT-B, ViT-L, ViT-H backbones, respectively.

extensive synthetic data. Due to the limited labeling quality, we exclusively employ COCO_TS as a reference to evaluate the automatic labeling capability of our SAM-TS model.

HierText [5] consists of 8,281 training images, 1,724 validation images, and 1,634 testing images, with dense and small texts in scene, document, *etc*. It contains hierarchical word, text-line, and paragraph location annotations. Word locations are annotated with polygons while text-line locations are annotated with quadrilateral boxes. Paragraph locations are represented by coarse polygons. HierText features scene, designed, printed, and handwritten texts. We contribute the annotations for pixel-level text segmentation in this paper.

We use the IOU and F-score metrics of foreground pixels for pixel-level text segmentation [1]. We follow HierText [5] to evaluate word, text-line, paragraph (layout) tasks with Panoptic Quality (PQ) [78], F-score, Precision (P), Recall (R), and Tightness (T). PQ and F-score are primary metrics.

4.2 Implementation Details

Training and Inference on Benchmarks. Hi-SAM takes an input image size of 1024×1024, which is the same as SAM [24]. In our default setting, the self-prompting module generates 12 prompt tokens for the S-Decoder. We use AdamW [79] ($\beta_1 = 0.9, \beta_2 = 0.999, weight\_decay = 0.05$) as the optimizer. The learning rate is set to $1e^{-4}$. The batch size is 8. We augment the input image with color jittering, random rotation, and large-scale jittering with a scale range of [0.5, 2.0]. For TS, we compare SAM-TS with existing representative TS methods. During training, for each dataset, we only use its training set following previous methods. On Total-Text and TextSeg, we train SAM-TS for 70 epochs without a learning rate drop. On HierText, we train SAM-TS for 80 epochs with the learning rate divided by 10 at 70 epochs. During inference, the input image size is also 1024×1024. For hierarchical text segmentation, we only use the training set of HierText for training. We train Hi-SAM for 150 epochs with the learning rate divided by 10 at 130 epochs. During training, we randomly sample at most 10 text-lines in each image and 2 points on each text-line. The sampling point number P in Sec. 3.8 is set to 1,500. In practice, we split these points into batches, with 100 points per batch in the default setting. We train the models on 8 NVIDIA Tesla V100 (32GB) GPUs.

TABLE 2 – Influence of the adapter in backbone and the Transformer decoder layer in self-prompting on Total-Text.

Backbone	Adapter	TDecLayer	fgIOU	F-score
ViT-B			68.80	78.31
	✓		79.73	86.25
	✓	✓	**80.93**	**86.25**
ViT-L			69.17	79.86
	✓		84.02	88.60
	✓	✓	**84.59**	**88.69**

TABLE 3 – Influence of prompt token number on Total-Text.

Backbone	Tokens	fgIOU	F-score
ViT-B	8	80.02	85.98
	12	80.93	**86.25**
	16	**81.04**	86.22
ViT-L	8	83.92	88.64
	12	**84.59**	**88.69**
	16	83.94	88.44

Trainable Parameter Statistics of Hi-SAM. We provide the trainable parameter statistics of Hi-SAM with different backbones in Fig. 5. Hi-SAM-B, Hi-SAM-L, and Hi-SAM-H employ ViT-B, ViT-L, and ViT-H backbones, respectively. Specifically, in Hi-SAM-B, there are 16.73M trainable parameters in total, with 7.10M in the adapter. In the three Hi-SAM variants, the self-prompting module only has 1.08M trainable parameters, while the S-Decoder and H-Decoder account for 4.34M and 4.20M trainable parameters, respectively. Note that SAM-TS-B for TS does not need an H-Decoder and has 12.53M trainable parameters in total, while SAM-TS-L and SAM-TS-H have 30.65M and 57.95M trainable parameters, respectively.

4.3 Ablation Studies

4.3.1 *Pixel-level Text Segmentation*

Influence on the Adapter and Transformer Decoder Layer in the Self-prompting Module. As shown in Tab. 2. We find that directly applying the frozen image encoder of SAM failed to yield satisfactory segmentation results, suggesting that the original SAM lacks awareness of fine text details. By introducing a few trainable parameters in the ViT blocks, both ViT-B and ViT-L models achieve significantly better performance than the baseline, validating the importance of adapting SAM to the text image domain. Moreover, using a single Transformer decoder layer to refine the tokens in the self-prompting module can further bring improvements.

Influence of Prompt Token Number. We investigate the influence of prompt token number on ViT-B and ViT-L models in Tab. 3. For ViT-B, using 12 prompt tokens achieves the best F-score and comparable fgIOU as using 16 prompt tokens, while using 12 prompt tokens for ViT-L achieves the best performance. For a better trade-off between complexity and performance, we adopt 12 prompt tokens by default for pixel-level text segmentation.

Comparison of Different Token Representation Methods. In the self-prompting module, we adopt a lightweight tokenizer [70] based on spatial attention to extract initial prompt tokens from image embeddings. As presented in Tab. 4, we compare it with vanilla embedding, which uses

TABLE 4 – Comparison of different token representations on Total-Text.

Backbone	Method	fgIOU	F-score
ViT-B	vanilla embedding [69]	80.78	85.95
	spatial attention [70]	**80.93**	**86.25**
ViT-L	vanilla embedding [69]	83.92	88.32
	spatial attention [70]	**84.59**	**88.69**

TABLE 5 – Effectiveness of introducing high-resolution mask features in S-Decoder. 'HR' denotes whether equipped with high-resolution mask features. 'fgIOU-LR' means the fgIOU evaluated with the outputs from low-resolution mask features while 'fgIOU-HR' is achieved from high-resolution features.

Backbone	HR	Total-Text		HierText	
		fgIOU-LR	fgIOU-HR	fgIOU-LR	fgIOU-HR
ViT-B		79.70	–	61.58	–
	✓	79.91	80.93 (+1.23)	61.63	73.39 (+11.81)
ViT-L		82.28	–	64.13	–
	✓	83.40	84.59 (+2.31)	63.75	78.37 (+14.24)

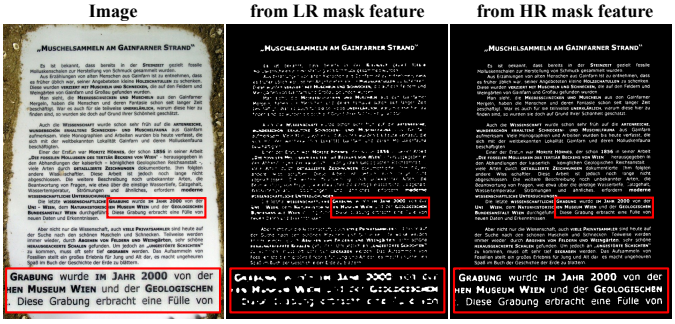

Fig. 6 – Qualitative comparison between the predictions from low-resolution (LR) mask feature and high-resolution (HR) mask feature.

initialized embeddings as prompt tokens, similar to the object queries in DETR [69]. After training, the prompt tokens derived from vanilla embedding are fixed while the ones from spatial attention are related to image embeddings. We observe that the latter method performs better, especially when image embeddings become more representative. Specifically, compared with the vanilla embedding, the latter gets 0.15% and 0.30% improvements in terms of fgIOU and F-score with ViT-B, while achieving 0.67% and 0.37% improvements on fgIOU and F-score with ViT-L.

Effectiveness of High-resolution Mask Features in S-Decoder. In Tab. 5, we validate the effectiveness of using high-resolution (HR) mask features for pixel-level text segmentation. We observe that introducing HR mask features significantly boosts the segmentation quality. With ViT-L, the fgIOU performance is improved by 2.31% on Total-Text. Moreover, on HierText which requires segmenting dense texts in scenes, documents, and handwritten materials, leveraging HR mask features delivers remarkable 14.24% fgIOU improvements. Some qualitative results are provided in Fig. 6. The result of using the LR mask feature almost loses all stroke details. In comparison, the prediction from the HR mask reserves most details.

Influence of the tuning method and pretrained weights for initialization. In Tab. 6, we demonstrate that adapter-tuning significantly enhances model adaptation to TS on smaller datasets, particularly with the larger ViT-L backbone. Additionally, when using MAE pretrained ViT weights instead of SAM's weights (for both the image encoder and mask decoder) for initialization, the fgIOU of SAM-TS-B and SAM-TS-L decreases by 7.21% and 5.13%, respectively. The results indicate that SAM effectively learns general segmentation knowledge through large-scale training, which can be successfully transferred to pixel-level text segmentation to improve performance.

TABLE 6 – Influence of the tuning method and pretrained weights for initialization on Total-Text. 'model-tuning' means all parameters are fully fine-tuned, without the adapter plugged in. '–' means the whole model is trained from scratch.

Backbone	Method	Pretrained Weights	fgIOU	F-score
ViT-B	adapter-tuning	SAM	**80.93**	**86.25**
	model-tuning	SAM	80.27	85.98
	model-tuning	MAE pretrained ViT	73.06	83.24
	model-tuning	–	41.03	57.70
ViT-L	adapter-tuning	SAM	**84.59**	**88.69**
	model-tuning	SAM	80.52	86.68
	model-tuning	MAE pretrained ViT	75.39	84.55
	model-tuning	–	41.86	58.85

TABLE 7 – Effectiveness of introducing high-resolution mask features in the H-Decoder. 'Feature Size' denotes the word mask feature size. The mask feature size for text-line and paragraph segmentation are both set to 256×256.

Feature Size	Word		Text-line		Layout Analysis	
	PQ	F-score	PQ	F-score	PQ	F-score
256×256	60.15	78.83	64.72	83.71	58.55	75.80
384×384	**62.60**	**80.99**	**65.62**	**84.04**	**58.83**	**75.84**
Δ	+2.45	+2.16	+0.90	+0.33	+0.28	+0.04

4.3.2 *Hierarchical Text Segmentation and Layout Analysis*

In this part, we focus on the ablation studies about the H-Decoder, training, and inference setting of Hi-SAM. We conduct experiments on the validation set of HierText using ViT-L as the backbone.

Effectiveness of High-resolution Mask Features in H-Decoder. As described in Sec. 3.5, we also introduce the mask feature with higher resolution to improve the quality of word segmentation in the H-Decoder. The performance comparison between different feature resolutions is listed in Tab. 7. As can be seen, utilizing the high-resolution mask feature in the H-Decoder leads to better capability in distinguishing adjacent word instances on book pages and document materials, bringing ideal improvement on word segmentation, *i.e.*, 2.45% PQ and 2.16% F-score. Note that the metrics for text-line and layout analysis are also enhanced because these tasks regard word instance as the basic unit.

Impact of Different Sample Strategies. During training, we randomly sample 10 text-lines in each image and 2 points on each text-line. We also examine the influence of increasing the point number on each text-line or adding text-line samples. The comparison results and corresponding training costs are shown in Tab. 8. As we can see while keeping the text-lines at 10 and increasing the point amount

TABLE 8 – Comparison of different training strategies. '10Ls × 2Pts' means sampling 10 text-lines in each image and 2 points on each text-line, resulting in at most 20 foreground points for prompt encoder and H-Decoder. 'Time' denotes the minutes per epoch. 'Mem.' denotes the peak memory footprint per GPU.

Strategy	Word		Text-line		Layout Analysis		Time (Mem.)
	PQ	F-score	PQ	F-score	PQ	F-score	
10Ls × 2Pts	62.60	80.99	65.62	84.04	58.83	75.84	20.5min (16GB)
10Ls × 5Pts	**63.14**	**81.72**	**65.95**	**84.25**	59.46	76.32	24.0min (22GB)
25Ls × 2Pts	63.02	81.57	65.81	84.12	**60.03**	**77.10**	32.0min (22GB)

TABLE 9 – Comparison of different point sampling number P during automatic mask generation.

Points	Word		Text-line		Layout Analysis	
	PQ	F-score	PQ	F-score	PQ	F-score
500	61.83	79.82	63.96	81.63	57.49	74.01
1,000	62.46	80.76	65.28	83.52	58.55	75.44
1,500	62.60	80.99	65.62	84.04	58.83	75.84
2,000	**62.64**	**81.07**	**65.74**	**84.24**	**58.92**	**75.97**

TABLE 10 – Influence of the tuning method and pretrained weights for initialization on Hi-SAM.

Method	Pretrained Weights	Word		Text-line		Layout Analysis	
		PQ	F-score	PQ	F-score	PQ	F-score
adapter-tuning	SAM	**62.60**	**80.99**	**65.62**	**84.04**	**58.83**	**75.84**
model-tuning	SAM	60.85	78.99	63.55	81.71	58.12	75.03
model-tuning	MAE pretrained ViT	57.67	75.42	61.39	79.95	56.05	73.21
model-tuning	–	47.13	63.57	52.12	70.21	47.44	63.83

to 5, the PQ and F-score improved by 0.54% and 0.73% at the word level, 0.33% and 0.21% at the text-line level, and 0.63% and 0.48% for layout analysis. In comparison, increasing text-line samples achieves more gains in layout analysis. However, increasing training samples requires more training resources. For example, only increasing the point amount on each text-line from 2 to 5 results in about 1.2× training time and 6GB more memory footprint on each GPU. Sampling more text-lines also leads to more data processing time. Therefore, we use 10 text-lines per image and 2 points per line as the default setting.

Influence of Point Sampling Number P. As mentioned in Sec. 3.8, we randomly sample P foreground points as the inputs for the prompt encoder. We evaluate the influence of using different point sampling numbers in Tab. 9. The performance of sampling 500 points is relatively low since it is insufficient to represent texts in HierText which has an average of 103.8 word instances per image. When the sampling points are doubled, the F-score improves by 0.94%, 1.89%, and 1.43% on word, text-line, and layout analysis, respectively. The performance could be further improved by using more points but it saturates at 2,000 points. For a better trade-off between performance and inference overhead, we set the point sampling number P to 1,500 by default.

Influence of the tuning method and pretrained weights for initialization. In Tab. 10, we further analyze the impact of pretrained weights on Hi-SAM. It is evident that adapter tuning outperforms model tuning. Using pretrained weights from SAM results in significant improvements compared to solely adopting the MAE pretrained ViT.

TABLE 11 – Comparison results of pixel-level text segmentation on Total-Text. The result of SegFormer is cited from TFT [2]. The best and second best are marked with **bold** and underline. 'Tr.Params' denotes the trainable parameters. 'To.Params' denotes the total parameters. They have the same meanings for other tables. Previous methods do not leverage the foundation model. SAM-TS leverages SAM in a parameter-efficient training manner, thus it owns more total parameters but fewer trainable parameters.

Method	Tr.Params	To.Params	fgIOU	F-score
DeepLabV3+ [80]	59.3M	59.3M	74.44	82.40
HRNetV2-W48 [81]	65.9M	65.9M	75.29	82.50
HRNetV2-W48 + OCR [81]	70.3M	70.3M	76.23	83.20
TexRNet + DeeplabV3+ [1]	59.9M	59.9M	76.53	84.40
TexRNet + HRNetV2-W48 [1]	67.1M	67.1M	78.47	84.80
SegFormer [82]	–	–	73.31	84.60
PGTSNet [9]	–	–	79.10	84.70
TFT [2]	–	–	82.10	**90.20**
SAM-TS-B	12.5M	102.2M	80.93	86.25
SAM-TS-L	30.7M	338.9M	84.59	88.69
SAM-TS-H	58.0M	695.0M	**84.86**	89.68

TABLE 12 – Comparison results of pixel-level text segmentation on TextSeg. The result of SegFormer is cited from TFT [2].

Method	Tr.Params	To.Params	fgIOU	F-score
DeepLabV3+ [80]	59.3M	59.3M	84.07	91.40
HRNetV2-W48 [81]	65.9M	65.9M	85.03	91.40
HRNetV2-W48 + OCR [81]	70.3M	70.3M	85.98	91.80
TexRNet + DeeplabV3+ [1]	59.9M	59.9M	86.06	92.10
TexRNet + HRNetV2-W48 [1]	67.1M	67.1M	86.84	92.40
SegFormer [82]	–	–	84.59	91.60
TFT [2]	–	–	87.11	93.10
SAM-TS-B	12.5M	102.2M	87.15	92.81
SAM-TS-L	30.7M	338.9M	88.77	93.79
SAM-TS-H	58.0M	695.0M	**88.96**	**93.87**

TABLE 13 – Comparison results of pixel-level text segmentation on HierText. We train TextRNet [1] with their official codes. 'S: 1024': the shorter side of each image is resized to 1024 and the aspect ratio is kept. '†': applying sliding window strategy (window size: 512 × 512, stride: 384).

Method	Tr.Params	To.Params	fgIOU	F-score
TexRNet + HRNetV2-W48 [1] (S: 1024)	67.1M	67.1M	55.50	65.64
TexRNet + HRNetV2-W48 [1] (S: 1536)	67.1M	67.1M	65.72	75.19
TexRNet + HRNetV2-W48 [1] (S: 2048)	67.1M	67.1M	70.77	80.32
SAM-TS-B	12.5M	102.2M	73.39	81.34
SAM-TS-L	30.7M	338.9M	78.37	84.99
SAM-TS-H	58.0M	695.0M	**79.27**	**85.63**
SAM-TS-B†	12.5M	102.2M	85.80	92.21
SAM-TS-L†	30.7M	338.9M	89.04	94.26
SAM-TS-H†	58.0M	695.0M	88.59	93.79

4.4 Comparison with State-of-the-art Methods

4.4.1 *Pixel-level Text Segmentation*

Total-Text Benchmark. Compared to other methods, our approach stands out as the first to leverage the vision foundation model [24] and realizes high-resolution pixel-level text segmentation. Although previous methods resort to utilizing additional text recognizers to enhance the semantic features [9], or extra text detectors and annotations [2], SAM-TS still outperforms them and achieves the best fgIOU performance on Total-Text. As shown in Tab. 11, compared

TABLE 14 – Label quality comparison on COCO_TS. While the original labels of COCO_TS are obtained using a weakly supervised segmentation model as described in Sec. 4.1, we use SAM-TS-L trained on different datasets to automatically generate its labels for comparison. 'COCO_TS → Target' means train the model on COCO_TS and evaluate it on Target (*e.g.*, Total-Text, TextSeg, and HierText) directly, without fine-tuning.

Model for Auto-labeling COCO_TS	COCO_TS → Total-Text				COCO_TS → TextSeg				COCO_TS → HierText			
	fgIOU	Δ	F-score	Δ	fgIOU	Δ	F-score	Δ	fgIOU	Δ	F-score	Δ
Original [76]	65.45	–	75.84	–	58.54	–	84.48	–	53.83	–	66.21	–
SAM-TS-L trained on Total-Text	80.02	+14.57	87.33	+11.49	56.93	-1.61	84.13	-0.35	63.64	+9.81	71.63	+5.42
SAM-TS-L trained on TextSeg	76.09	+10.64	84.10	+8.26	81.12	+22.58	91.51	+7.03	54.75	+0.92	61.26	-4.95
SAM-TS-L trained on HierText	76.23	+10.78	86.33	+10.49	71.23	+12.69	86.87	+2.39	69.95	+16.12	79.67	+13.46

Fig. 7 – Qualitative results on Total-Text and TextSeg. The predictions are made by SAM-TS-L.

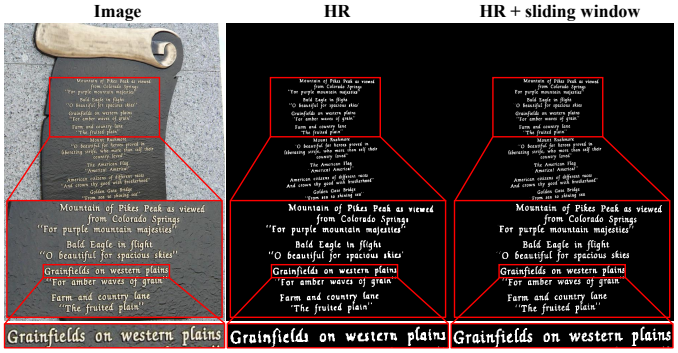

Fig. 8 – Visualizations on HierText. 'HR': predictions from high-resolution mask feature. Using the sliding window strategy substantially improves the segmentation quality on extremely small texts. For example, in the bottom row of sliced patches, the tiny hollows in character 'e' are distinguished when using sliding window. And the pixel-level text segmentation delivers more correct and clearer structure details. Best view on screen with zooming in.

to TFT [2], SAM-TS-L achieves a gain of 2.49% fgIOU and SAM-TS-H presents state-of-the-art 84.86% fgIOU. Compared to PGTSNet [9], SAM-TS-L significantly outperforms it by 5.49% fgIOU.

TextSeg Benchmark. As shown in Tab. 12, our method delivers commendable performance on TextSeg. SAM-TS-B achieves 87.15% fgIOU and SAM-TS-L further improves the fgIOU performance by 1.62%. Moreover, SAM-TS-H reaches 88.96% fgIOU. Some visualization examples on Total-Text and TextSeg are shown in Fig. 7.

HierText Benchmark. HierText presents a significant challenge in segmenting text with fine details on both natural and document scenarios. We evaluate TextRNet [1] with the HRNetV2-W48 [81] backbone on HierText for comparison. As can be seen in Tab. 13, TextRNet only achieves 55.50% fgIOU, running at 9.3 FPS tested on one V100 GPU with one batch size. In comparison, SAM-TS-B achieves 73.39% fgIOU with 6.4 FPS. SAM-TS-L achieves 78.37% fgIOU with 2.8 FPS and SAM-TS-H achieves 79.27% fgIOU with 1.6 FPS. Different from Total-Text and TextSeg, segmenting texts in high-quality for HierText requires larger feature size. With the shorter side of image resized to 2,048, TextRNet achieves 70.77% fgIOU with 2.2 FPS. We apply the sliding window strategy and the fgIOU performance of SAM-TS-L is further improved by 10.67%, outperforming TextRNet by 18.27%. We provide some qualitative comparison with and without the sliding window strategy in Fig. 8. Using the sliding window strategy substantially improves the segmentation quality on extremely small texts.

Auto-labeling on COCO_TS. We further compare the quality of labels generated by our SAM-TS-L and the original labels on COCO_TS. We train three SAM-TS-L models on Total-Text, TextSeg, and HierText to automatically label the images in COCO_TS using the sliding window strategy (window size: 512 × 512, stride: 384). Then, we train SAM-TTS using these different labels on COCO_TS for 40 epochs and evaluate them on other datasets without fine-tuning. The ability to generalize performance across various datasets serves as a reliable indicator of the quality of labels. Note that the original labels of COCO_TS contain a lot of uncertain regions, we involve them for training the baseline model following previous methods [1], [76].

The results are reported in Tab. 14. As can be seen, the models trained with our generated labels show significantly better generalization performance in general, which indicates that our model has an excellent auto-labeling ability. Note that employing labels obtained from the model trained on Total-Text does not result in improved performance on TextSeg. This is likely due to the fact that TextSeg encompasses both scenes and designed texts, whereas Total-Text predominantly emphasizes scene texts.

TABLE 15 – Comparison between Unified Detector [5] and Hi-SAM. 'TL': text-line. 'WG': word grouping. 'Para.': paragraph. 'VFM': whether the vision foundation model is used. As for the model parameter statistics, Unified Detector has 88.2M trainable and also total parameters. Hi-SAM-B has 16.7M trainable parameters and 106.4M parameters in total. Hi-SAM-L owns 34.9M trainable parameters and 343.1M parameters in total. Hi-SAM-H has 62.2M trainable parameters and 699.2M parameters in total.

Method	Pixel-level Text	Word	TL	WG in TL	Para.	Layout (WG in Para.)	Promptable	Training on HierText	VFM
Unified Detector [5]	✘	✔	✘	✔	✘	✔	✘	128 TPUv3 cores, ∼3091 epochs	No
Hi-SAM (Ours)	✔	✔	✔	✔	✔	✔	✔	8 V100 GPUs, 150 epochs	Yes

TABLE 16 – Comparison results on HierText test set. fgIOU, PQ, and F-score are the primary metrics.

Method	Pixel-level Text		Word					Text-line					Layout Analysis				
	fgIOU	F-score	PQ	F-score	P	R	T	PQ	F-score	P	R	T	PQ	F-score	P	R	T
Unified Detector [5]	–	–	48.21	61.51	67.54	56.47	**78.38**	62.23	79.91	79.64	80.19	77.87	53.60	68.58	76.04	62.45	**78.17**
Hi-SAM-B	70.94	79.78	59.74	78.34	83.97	73.41	76.25	63.34	82.15	89.17	76.16	77.10	54.46	71.15	78.53	65.03	76.54
Hi-SAM-L	75.22	82.90	63.10	81.83	87.22	77.06	77.11	66.17	84.85	90.66	79.74	77.99	57.61	74.49	80.43	69.37	77.34
Hi-SAM-H	**75.73**	**83.36**	**64.30**	**82.86**	**87.66**	**78.56**	77.60	**66.96**	**85.30**	**91.09**	**80.20**	**78.50**	**59.09**	**75.97**	**81.52**	**71.13**	77.79

TABLE 17 – Comparison results on HierText validation set. '‡' indicates that we calculate centroid points of the polygon labels of legible words and adopt them as prompts during inference.

Method	Pixel-level Text		Word					Text-line					Layout Analysis				
	fgIOU	F-score	PQ	F-score	P	R	T	PQ	F-score	P	R	T	PQ	F-score	P	R	T
Unified Detector [5]	–	–	48.47	61.65	67.18	56.96	**78.47**	61.29	78.66	78.98	78.34	77.92	52.86	67.73	76.17	60.98	78.04
Hi-SAM-B	70.35	79.65	59.20	77.48	83.51	72.26	76.41	62.89	81.44	88.89	75.13	77.22	55.66	72.64	80.06	66.48	76.63
Hi-SAM-L	74.86	82.87	62.60	80.99	86.48	76.16	77.29	65.62	84.04	90.29	78.59	78.08	58.83	75.84	81.72	70.75	77.57
Hi-SAM-H	**75.45**	**83.21**	**63.83**	**82.08**	**87.02**	**77.67**	77.76	**66.70**	**84.82**	**90.97**	**79.45**	**78.64**	**60.34**	**77.22**	**83.47**	**71.84**	**78.14**
Hi-SAM-B‡	70.35	79.65	61.17	80.55	84.35	77.08	75.93	67.32	88.17	91.21	85.32	76.35	57.85	75.88	81.43	71.03	76.24
Hi-SAM-L‡	74.86	82.87	64.63	84.08	87.79	80.68	76.87	69.58	90.06	92.28	87.94	77.27	60.42	78.16	82.35	74.37	77.30
Hi-SAM-H‡	75.45	83.21	65.73	85.06	88.18	82.15	77.28	70.61	90.81	92.85	88.85	77.76	61.81	79.40	83.86	75.38	77.85

4.4.2 *Hierarchical Text Segmentation and Layout Analysis*

We evaluate the hierarchical text segmentation and layout analysis performance of Hi-SAM on HierText. We first compare the functionality and training cost between the SOTA Unified Detector (UD) [5] and our Hi-SAM.

Comparison with UD. UD uses the MaX-DeepLab-S backbone [83], which has a relatively similar trainable parameter amount as Hi-SAM-H. The input image size of UD is the same as our Hi-SAM. Each object query in UD is trained to segment words in one text-line as opposed to a single word. Thus, as shown in Tab. 15, UD can perform word grouping at the text-line level naturally and also word instance segmentation with post-processing. UD leverages a layout branch to cluster object queries into different groups, thereby achieving layout analysis (word grouping at paragraph level). However, UD is not promptable and cannot segment pixel-level text masks and achieve contact text-line and paragraph masks. Besides, UD is trained on 128 TPUv3 cores with a batch size of 256 for 100K steps (about 3,091 epochs). In comparison, Hi-SAM is the first text-centric model that can perform all tasks mentioned above and is promptable. Moreover, Hi-SAM requires 20× fewer training epochs and is trained on eight NVIDIA V100 GPUs.

Results on Test Set. As presented in Tab. 16, compared to UD, our method achieves significant improvements in terms of PQ and F-score for word level, text-line level, and layout analysis. For example, Hi-SAM-H outperforms UD by 16.09% PQ and 21.35% F-score at word level, 4.73% PQ and 5.39% F-score at text-line level word grouping, 5.49% PQ and 7.39% F-score at paragraph level layout analysis. We notice that UD performs poorly at the word level, probably because UD cannot separate adjacent instances better in dense text scenarios. Whereas, this does not affect the performance at the text-line and paragraph level. Note that UD utilizes all samples in each image for training and performs layout analysis by calculating affinity scores among object queries. In contrast, we design completely different training and inference pipelines, where we only sample a few samples in each image for training and implement layout analysis without an additional layout branch. Even with 20× fewer training epochs, our method still achieves significantly better performance. Comparing different Hi-SAM variants, we observe that Hi-SAM-L is significantly better than Hi-SAM-B, *i.e.*, achieving improvement by 3.36% PQ and 3.49% F-score at the word level, by 2.83% PQ and 2.70% F-score on text-line level, and by 3.15% PQ and 3.34% F-score at the paragraph level. Comparing Hi-SAM-H to Hi-SAM-L, the main improvement lies in the layout analysis results, owing to the better image embedding.

Results on Validation Set. We also report the results on the HierText validation set, as shown in Tab. 17. The results of UD are collected from the official GitHub repository[2]. As can be seen, compared to UD, our method also achieves similar performance improvements as on the test set. Hi-SAM-H outperforms UD by 15.36% PQ and 20.43% F-score at word level, 5.41% PQ and 6.16% F-score at text-line level word grouping, 7.48% PQ and 9.49% F-score at paragraph level layout analysis. In addition, we use the polygon annotation of legible words to calculate centroid points and adopt these

2. https://github.com/google-research-datasets/hiertext

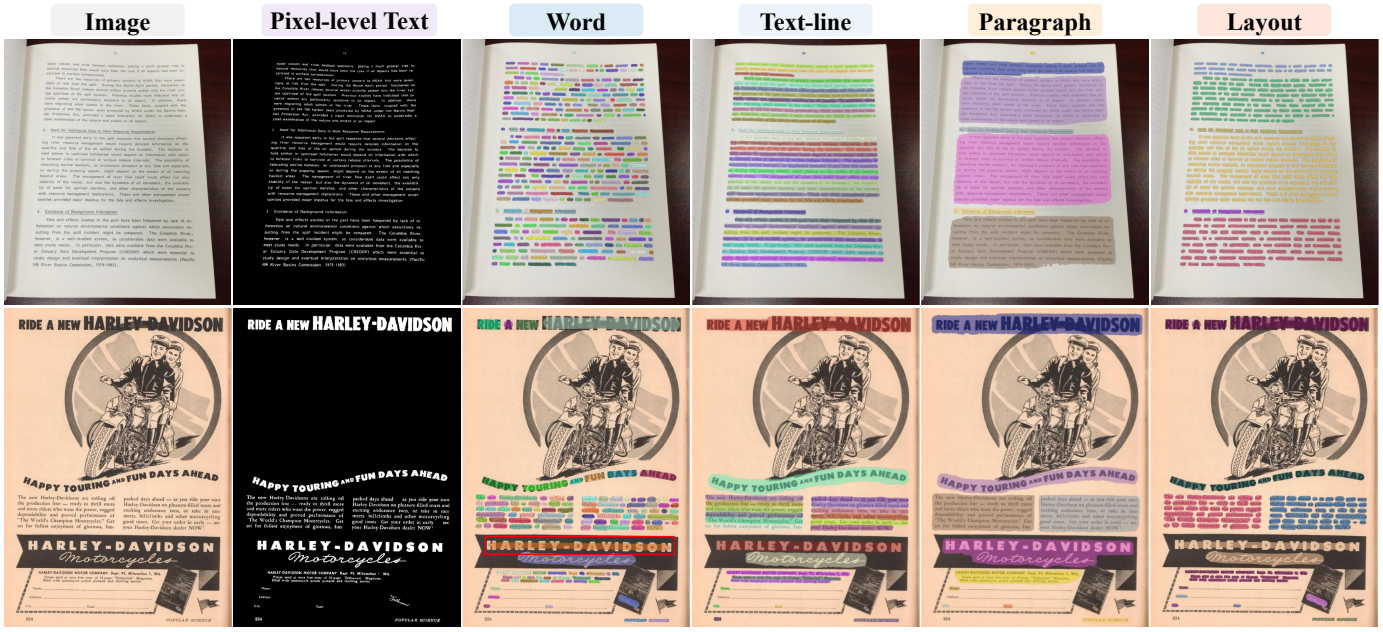

Fig. 9 – Visualizations of hierarchical text segmentation and layout analysis results. Some flaws are marked with red boxes.

Fig. 10 – Visualizations of promptable segmentation. Green stars in images are point prompt locations. Word, text-line, and paragraph masks are in green, blue, and orange color.

points as prompts for generating hierarchical masks during inference. We observe that using these centroid points as prompts mainly enhances the recall (R) scores, especially at the text-line level. For instance, for Hi-SAM-B, the recall scores are improved by 4.82%, 10.19%, and 4.55% at the word level, text-line level, and layout analysis, respectively. For Hi-SAM-H, the recall metrics are enhanced by 4.49%, 9.40%, and 3.54% at the word level, text-line level, and layout analysis, respectively. It indicates that there is still room for developing a better sampling strategy. Perhaps leveraging a simple counting module for providing a text center heat map may help. In addition, since we apply NMS on text-line masks and discard corresponding word-level masks, the remained word-level masks may not achieve the best segmentation.

TABLE 18 – AP comparison on HierText validation set.

Method	Text-line			Layout Analysis		
	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
Unified Detector [5]	40.2	73.6	40.5	22.3	41.3	22.9
Hi-SAM-B	40.0	73.3	41.9	25.2	48.3	24.9
Hi-SAM-L	43.1	76.2	46.4	28.5	52.5	29.6
Hi-SAM-H	**44.7**	**77.3**	**50.0**	**29.4**	**53.2**	**32.1**

On the validation set, we compared the AP metric at IoU thresholds ranging from 0.5 to 0.95, as shown in Tab. 18. At the text-line level, the Hi-SAM models significantly outperform the UD model, particularly at higher IoU thresholds. For instance, Hi-SAM-B, Hi-SAM-L, and Hi-SAM-H achieve AP_{75} improvements of 1.4%, 5.9%, and 9.5%, respectively. Moreover, Hi-SAM shows even greater improvements over UD in layout analysis compared to the text-line level. Specifically, Hi-SAM-L outperforms UD by 2.9% AP on the text-line level and by 6.2% AP in layout analysis, indicating Hi-SAM's superior segmentation accuracy, especially at the paragraph level. Among the Hi-SAM models, Hi-SAM-L notably outperforms Hi-SAM-B, with a significantly better AP_{75}, demonstrating the advantages of a larger backbone.

Visualizations. We provide some visualizations of automatic mask generation and promptable segmentation in Fig. 9 and Fig. 10. Some failure cases are marked with red boxes. For example, the word segments for long words are sometimes incomplete. The text-line segmentation results in extremely curved entities are not promising, which is caused by the quadrilateral labels at the text-line level.

5 Limitation and Discussion

While Hi-SAM has shown promising performance, there is still potential for further improvement. 1) Due to the heavy backbone, the existing Hi-SAM models cannot reach real-time inference speed. As some follow-ups of SAM [84], [85] have achieved great progress in developing lightweight SAM models, it is promising to develop efficient Hi-SAM models by exploring a more lightweight structure. 2) Since we freeze the pretrained weights of SAM's image encoder and only use HierText to train Hi-SAM, the generalization ability of Hi-SAM for unseen domains can be constrained. 3) Hi-SAM requires labels across four text hierarchies, which are scarce in real datasets. It is worth trying to explore synthetic or diffusion-based [86] methods to generate large-scale hierarchical text data with complex layouts and accurate text contents. 4) We observe that the pixel-level text segmentation results of Hi-SAM in Tab. 16 are lower than those in Tab. 13. How to better synergize different text hierarchies during training and inference is also an open issue. 5) As a text-centric model, Hi-SAM falls short in conducting the multi-class (including text, title, list, table, and figure) layout analysis in some document-only datasets, such as PubLayNet [45].

6 Conclusion

In this study, we demonstrate the transformation of SAM into a cutting-edge pixel-level text segmentation model, SAM-TS, through minimal customizations. It is capable of generating high-quality labels semi-automatically, helping unify annotations across four text hierarchies in HierText. Building upon these advancements, we introduce Hi-SAM – a pioneering unified model for hierarchical text segmentation. It seamlessly operates across multiple levels, spanning from the finest pixel level to paragraph, while conducting layout analysis without the need for any dedicated modules. Extensive experimental results unequivocally highlight the superior performance of our proposed method compared to existing representative approaches. We hope this work can pave the way for practical hierarchical text segmentation in real-world text images and inspire further exploration in this field.

References

[1] X. Xu, Z. Zhang, Z. Wang, B. Price, Z. Wang, and H. Shi, "Rethinking text segmentation: A novel dataset and a text-specific refinement approach," in *CVPR*, 2021, pp. 12 045–12 055. 1, 3, 7, 8, 10, 11

[2] H. Yu, X. Wang, K. Niu, B. Li, and X. Xue, "Scene text segmentation with text-focused transformers," in *ACM MM*, 2023, pp. 2898–2907. 1, 10, 11

[3] D. Peng, Z. Yang, J. Zhang, C. Liu, Y. Shi, K. Ding, F. Guo, and L. Jin, "Upocr: Towards unified pixel-level ocr interface," in *ICML*, 2024. 1

[4] S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha, "Docformer: End-to-end transformer for document understanding," in *ICCV*, 2021, pp. 993–1003. 1

[5] S. Long, S. Qin, D. Panteleev, A. Bissacco, Y. Fujii, and M. Raptis, "Towards end-to-end unified scene text detection and layout analysis," in *CVPR*, 2022, pp. 1049–1059. 1, 2, 3, 6, 7, 8, 12, 13

[6] S. Long, S. Qin, Y. Fujii, A. Bissacco, and M. Raptis, "Hierarchical text spotter for joint text spotting and layout analysis," in *WACV*, 2024, pp. 903–913. 1

[7] J. Liu, Z. Chen, B. Du, and D. Tao, "Asts: A unified framework for arbitrary shape text spotting," *IEEE TIP*, vol. 29, pp. 5924–5936, 2020. 1

[8] Y. Ren, J. Zhang, B. Chen, X. Zhang, and L. Jin, "Looking from a higher-level perspective: Attention and recognition enhanced multi-scale scene text segmentation," in *ACCV*, 2022, pp. 3138–3154. 1, 3

[9] X. Xu, Z. Qi, J. Ma, H. Zhang, Y. Shan, and X. Qie, "Bts: a bi-lingual benchmark for text segmentation in the wild," in *CVPR*, 2022, pp. 19 152–19 162. 1, 3, 10, 11

[10] X. Zu, H. Yu, B. Li, and X. Xue, "Weakly-supervised text instance segmentation," in *ACM MM*, 2023, pp. 1915–1923. 1, 3

[11] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," in *AAAI*, vol. 34, no. 07, 2020, pp. 11 474–11 481. 1, 3, 6

[12] S.-X. Zhang, C. Yang, X. Zhu, and X.-C. Yin, "Arbitrary shape text detection via boundary transformer," *IEEE TMM*, vol. 26, pp. 1747–1760, 2023. 1

[13] M. Ye, J. Zhang, S. Zhao, J. Liu, B. Du, and D. Tao, "Dptext-detr: Towards better scene text detection with dynamic points in transformer," in *AAAI*, vol. 37, no. 3, 2023, pp. 3241–3249. 1, 3

[14] M. Ye, J. Zhang, S. Zhao, J. Liu, T. Liu, B. Du, and D. Tao, "Deepsolo: Let transformer decoder with explicit points solo for text spotting," in *CVPR*, 2023, pp. 19 348–19 357. 1

[15] Y. Kittenplon, I. Lavi, S. Fogel, Y. Bar, R. Manmatha, and P. Perona, "Towards weakly-supervised text spotting using a multi-task transformer," in *CVPR*, 2022, pp. 4604–4613. 1

[16] J. Liu, H. Ding, Z. Cai, Y. Zhang, R. K. Satzoda, V. Mahadevan, and R. Manmatha, "Polyformer: Referring image segmentation as sequential polygon generation," in *CVPR*, 2023, pp. 18 653–18 663. 1

[17] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, "Detecting texts of arbitrary orientations in natural images," in *CVPR*, 2012, pp. 1083–1090. 1

[18] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu *et al.*, "Icdar 2015 competition on robust reading," in *ICDAR*, 2015, pp. 1156–1160. 1

[19] Y. Liu, L. Jin, S. Zhang, C. Luo, and S. Zhang, "Curved scene text detection via transverse and longitudinal sequence connection," *PR*, vol. 90, pp. 337–345, 2019. 1

[20] C. K. Ch'ng and C. S. Chan, "Total-text: A comprehensive dataset for scene text detection and recognition," in *ICDAR*, vol. 1, 2017, pp. 935–942. 1, 3, 7

[21] C. K. Chng, Y. Liu, Y. Sun, C. C. Ng, C. Luo, Z. Ni, C. Fang, S. Zhang, J. Han, E. Ding *et al.*, "Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art," in *ICDAR*, 2019, pp. 1571–1576. 1

[22] Y. Sun, Z. Ni, C.-K. Chng, Y. Liu, C. Luo, C. C. Ng, J. Han, E. Ding, J. Liu, D. Karatzas *et al.*, "Icdar 2019 competition on large-scale street view text with partial labeling-rrc-lsvt," in *ICDAR*, 2019, pp. 1557–1562. 1

[23] A. Singh, G. Pang, M. Toh, J. Huang, W. Galuba, and T. Hassner, "Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text," in *CVPR*, 2021, pp. 8802–8812. 1

[24] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *ICCV*, 2023, pp. 4015–4026. 1, 2, 3, 4, 7, 8, 10

[25] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, and F. Yu, "Segment anything in high quality," in *NeurIPS*, vol. 36, 2023. 2, 4, 7

[26] J. Cheng, J. Ye, Z. Deng, J. Chen, T. Li, H. Wang, Y. Su, Z. Huang, J. Chen, L. Jiang *et al.*, "Sam-med2d," *arXiv:2308.16184*, 2023. 2, 4

[27] W. Yue, J. Zhang, K. Hu, Y. Xia, J. Luo, and Z. Wang, "Surgicalsam: Efficient class promptable surgical instrument segmentation," in *AAAI*, 2023. 2, 4

[28] Y. Cheng, L. Li, Y. Xu, X. Li, Z. Yang, W. Wang, and Y. Yang, "Segment and track anything," *arXiv:2305.06558*, 2023. 2, 4

[29] J. Yang, M. Gao, Z. Li, S. Gao, F. Wang, and F. Zheng, "Track anything: Segment anything meets videos," *arXiv:2304.11968*, 2023. 2, 4

[30] D. Wang, J. Zhang, B. Du, M. Xu, L. Liu, D. Tao, and L. Zhang, "SAMRS: Scaling-up remote sensing segmentation dataset with segment anything model," in *NeurIPS Datasets and Benchmarks Track*, 2023. [Online]. Available: https://openreview.net/forum?id=jHrgq55ftl 2

[31] X. Wang, C. Wu, H. Yu, B. Li, and X. Xue, "Textformer: Component-aware text segmentation with transformer," in *ICME*, 2023, pp. 1877–1882. 3

[32] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," in *AAAI*, vol. 31, no. 1, 2017. 3

[33] M. Liao, B. Shi, and X. Bai, "Textboxes++: A single-shot oriented scene text detector," *IEEE TIP*, vol. 27, no. 8, pp. 3676–3690, 2018. 3

[34] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *CVPR*, 2019, pp. 9365–9374. 3

[35] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape robust text detection with progressive scale expansion network," in *CVPR*, 2019, pp. 9336–9345. 3

[36] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen, "Efficient and accurate arbitrary-shaped text detection with pixel aggregation network," in *ICCV*, 2019, pp. 8440–8449. 3

[37] J. Ye, Z. Chen, J. Liu, and B. Du, "Textfusenet: Scene text detection with richer fused features." in *IJCAI*, vol. 20, 2020, pp. 516–522. 3

[38] S.-X. Zhang, X. Zhu, C. Yang, H. Wang, and X.-C. Yin, "Adaptive boundary proposal network for arbitrary shape text detection," in *CVPR*, 2021, pp. 1305–1314. 3

[39] P. Dai, S. Zhang, H. Zhang, and X. Cao, "Progressive contour regression for arbitrary-shape scene text detection," in *CVPR*, 2021, pp. 7393–7402. 3

[40] X. Zhang, Y. Su, S. Tripathi, and Z. Tu, "Text spotting transformers," in *CVPR*, 2022, pp. 9519–9528. 3

[41] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, "Abcnet: Real-time scene text spotting with adaptive bezier-curve network," in *CVPR*, 2020, pp. 9809–9818. 3

[42] Y. Zhu, J. Chen, L. Liang, Z. Kuang, L. Jin, and W. Zhang, "Fourier contour embedding for arbitrary-shaped text detection," in *CVPR*, 2021, pp. 3123–3131. 3

[43] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *ICDAR*, vol. 1, 2017, pp. 1162–1167. 3

[44] J. Lee, H. Hayashi, W. Ohyama, and S. Uchida, "Page segmentation using a convolutional neural network with trainable co-occurrence features," in *ICDAR*, 2019, pp. 1023–1028. 3

[45] X. Zhong, J. Tang, and A. J. Yepes, "Publaynet: largest dataset ever for document layout analysis," in *ICDAR*, 2019, pp. 1015–1022. 3, 14

[46] S. Biswas, P. Riba, J. Lladós, and U. Pal, "Beyond document object detection: instance-level segmentation of complex layouts," *IJDAR*, vol. 24, no. 3, pp. 269–281, 2021. 3

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017, pp. 2961–2969. 3

[48] S. Biswas, A. Banerjee, J. Lladós, and U. Pal, "Docsegtr: an instance-level end-to-end document image segmentation transformer," *arXiv preprint arXiv:2201.11438*, 2022. 3

[49] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, "Layoutlmv3: Pre-training for document ai with unified text and image masking," in *ACM MM*, 2022, pp. 4083–4091. 3

[50] C. Li, B. Bi, M. Yan, W. Wang, S. Huang, F. Huang, and L. Si, "Structurallm: Structural pre-training for form understanding," in *ACL*, 2021, pp. 6309–6318. 3

[51] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park, "Ocr-free document understanding transformer," in *ECCV*, 2022, pp. 498–517. 3

[52] B. Davis, B. Morse, B. Price, C. Tensmeyer, C. Wigington, and V. Morariu, "End-to-end document recognition and understanding with dessurt," in *ECCV*, 2022, pp. 280–296. 3

[53] D. Coquenet, C. Chatelain, and T. Paquet, "Dan: a segmentation-free document attention network for handwritten document recognition," *IEEE TPAMI*, vol. 45, no. 7, pp. 8227–8243, 2023. 3

[54] K. Lee, M. Joshi, I. R. Turc, H. Hu, F. Liu, J. M. Eisenschlos, U. Khandelwal, P. Shaw, M.-W. Chang, and K. Toutanova, "Pix2struct: Screenshot parsing as pretraining for visual language understanding," in *ICML*, 2023, pp. 18 893–18 912. 3

[55] F. Li, H. Zhang, P. Sun, X. Zou, S. Liu, J. Yang, C. Li, L. Zhang, and J. Gao, "Semantic-sam: Segment and recognize anything at any granularity," in *ECCV*, 2024. 3

[56] S. Song, J. Wan, Z. Yang, J. Tang, W. Cheng, X. Bai, and C. Yao, "Vision-language pre-training for boosting scene text detectors," in *CVPR*, 2022, pp. 15 681–15 691. 3

[57] C. Xue, W. Zhang, Y. Hao, S. Lu, P. H. Torr, and S. Bai, "Language matters: A weakly supervised vision-language pre-training approach for scene text detection and spotting," in *ECCV*, 2022, pp. 284–302. 3

[58] W. Yu, Y. Liu, W. Hua, D. Jiang, B. Ren, and X. Bai, "Turning a clip model into a scene text detector," in *CVPR*, 2023, pp. 6978–6988. 3

[59] W. Yu, Y. Liu, X. Zhu, H. Cao, X. Sun, and X. Bai, "Turning a clip model into a scene text spotter," *IEEE TPAMI*, 2024. 3

[60] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021, pp. 8748–8763. 3

[61] Z. Wang, H. Xie, Y. Wang, J. Xu, B. Zhang, and Y. Zhang, "Symmetrical linguistic feature distillation with clip for scene text recognition," in *ACM MM*, 2023, pp. 509–518. 3

[62] J. Li, J. Jain, and H. Shi, "Matting anything," in *CVPR*, 2024, pp. 1775–1785. 4

[63] J. Cen, Z. Zhou, J. Fang, C. Yang, W. Shen, L. Xie, X. Zhang, and Q. Tian, "Segment anything in 3d with nerfs," in *NeurIPS*, vol. 36, 2023, pp. 25 971–25 990. 4

[64] W. Yue, J. Zhang, K. Hu, Q. Wu, Z. Ge, Y. Xia, J. Luo, and Z. Wang, "Part to whole: Collaborative prompting for surgical instrument segmentation," *arXiv:2312.14481*, 2023. 4

[65] J. Wu, R. Fu, H. Fang, Y. Liu, Z. Wang, Y. Xu, Y. Jin, and T. Arbel, "Medical sam adapter: Adapting segment anything model for medical image segmentation," *arXiv:2304.12620*, 2023. 4, 5

[66] R. Zhang, Z. Jiang, Z. Guo, S. Yan, J. Pan, H. Dong, P. Gao, and H. Li, "Personalize segment anything model with one shot," in *ICLR*, 2024. 4

[67] Y. Liu, M. Zhu, H. Li, H. Chen, X. Wang, and C. Shen, "Matcher: Segment anything with one shot using all-purpose feature matching," in *ICLR*, 2024. 4

[68] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022, pp. 16 000–16 009. 4

[69] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*, 2020, pp. 213–229. 4, 9

[70] M. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova, "Tokenlearner: Adaptive space-time tokenization for videos," in *NeurIPS*, vol. 34, 2021, pp. 12 786–12 797. 5, 8, 9

[71] X. Chen, X. Wang, J. Zhou, Y. Qiao, and C. Dong, "Activating more pixels in image super-resolution transformer," in *CVPR*, 2023, pp. 22 367–22 377. 6

[72] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *CVPR*, 2021, pp. 2918–2928. 6

[73] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988. 7

[74] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *3DV*, 2016. 7

[75] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2: Dynamic and fast instance segmentation," in *NeurIPS*, vol. 33, 2020, pp. 17 721–17 732. 7

[76] S. Bonechi, P. Andreini, M. Bianchini, and F. Scarselli, "Coco_ts dataset: pixel–level annotations based on weak supervision for scene text segmentation," in *ICANN*, 2019, pp. 238–250. 7, 11

[77] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "Coco-text: Dataset and benchmark for text detection and recognition in natural images," *arXiv:1601.07140*, 2016. 7

[78] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *CVPR*, 2019, pp. 9404–9413. 8

[79] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019. 8

[80] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018, pp. 801–818. 10

[81] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE TPAMI*, vol. 43, no. 10, pp. 3349–3364, 2020. 10, 11

[82] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," in *NeurIPS*, vol. 34, 2021, pp. 12 077–12 090. 10

[83] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "Max-deeplab: End-to-end panoptic segmentation with mask transformers," in *CVPR*, 2021, pp. 5463–5474. 12
[84] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv:2306.12156*, 2023. 14
[85] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv:2306.14289*, 2023. 14
[86] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, vol. 33, 2020, pp. 6840–6851. 14