APSIPA Transactions on Signal and Information Processing, 2024, 13, e400

Original Paper

A Lightweight Enhancement Approach for Real-Time Semantic Segmentation by Distilling Rich Knowledge from Pre-Trained Vision-Language Model

Chia-Yi Lin[1], Jun-Cheng Chen[2] and Ja-Ling Wu[1,3*]

[1] *Department of Computer Science and Information Engineering, National Taiwan University*
[2] *Research Center for Information Technology Innovation, Academia Sinica*
[3] *Graduate Institute of Networking and Multimedia, National Taiwan University*

ABSTRACT

In this work, we propose a lightweight approach to enhance real-time semantic segmentation by leveraging the pre-trained vision-language models, specifically utilizing the text encoder of Contrastive Language-Image Pretraining (CLIP) to generate rich semantic embeddings for text labels. Then, our method distills this textual knowledge into the segmentation model, integrating the image and text embeddings to align visual and textual information. Additionally, we implement learnable prompt embeddings for better class-specific semantic comprehension. We propose a two-stage training strategy for efficient learning: the segmentation backbone initially learns from fixed text embeddings and subsequently optimizes prompt embeddings to streamline the learning process. The extensive evaluations and ablation studies validate our approach's ability to effectively improve the semantic segmentation model's performance over the compared methods.

*Corresponding author: wjl@cmlab.csie.ntu.edu.tw.

Received 01 March 2024; Revised 26 May 2024
ISSN 2048-7703; DOI 10.1561/116.20240015

Keywords: semantic segmentation, real-time, vision-language pre-training, CLIP

1 Introduction

Semantic segmentation is one of research hotspots in computer vision, which seeks to assign each pixel a semantic label in an image, thus segmenting the image into distinct and meaningful components. This task is fundamental for machines to comprehend the details of an image, enabling differentiation among different objects, regions, or parts in the image. It holds significant importance for applications demanding rapid processing, such as autonomous vehicles and augmented reality, where swift interaction and response are crucial. Consequently, the focus on creating efficient, real-time semantic segmentation models has recently emerged as a vital research area.

Numerous segmentation models proposed by Zhao *et al.* [31], Li *et al.* [13], Mehta *et al.* [16], Yu *et al.* [28], Chen *et al.* [3], Fan *et al.* [7] and Peng *et al.* [19] have been developed to address the need for low-latency algorithms in real-time semantic segmentation. While these models achieve significant progress, challenges remain, particularly in balancing the inference speed with the model accuracy. Although an effective approach to improve the performance is to exploit a more complex architecture, it is still an ongoing research problem to keep the latency as low as possible at the same time.

To address these challenges, we utilize the pre-trained vision-language foundation models developed by Radford *et al.* [21] and Jia *et al.* [10] to improve real-time semantic segmentation. Our approach, integrating the text encoder of the Contrastive Language-Image Pre-training (CLIP) proposed by Radford *et al.* [21], aims to transfer its substantial textual knowledge to the real-time semantic segmentation model, STDC-Seg developed by Fan *et al.* [7]. As shown in Figure 1, our framework aligns the information from the visual and textual domains by consolidating image and text embeddings, using cosine similarity to create a score map that reflects semantic correlations between different image regions and textual labels. The score map, combined with the image's feature maps, facilitates the integration of visual and textual information. Additionally, we introduce learnable prompt embeddings specific to each class, further enhancing the semantic understanding of the model. These learnable prompts combined with the label token embeddings are then processed through the CLIP text encoder, producing enriched class-specific prompt embeddings. The learnable prompts provide the model a flexible way to effectively learn the class-specific information.

Furthermore, to guarantee efficient learning, our methodology exploits a two-stage training process. Initially, the segmentation model's backbone is trained by aligning its image feature representations with the text embeddings

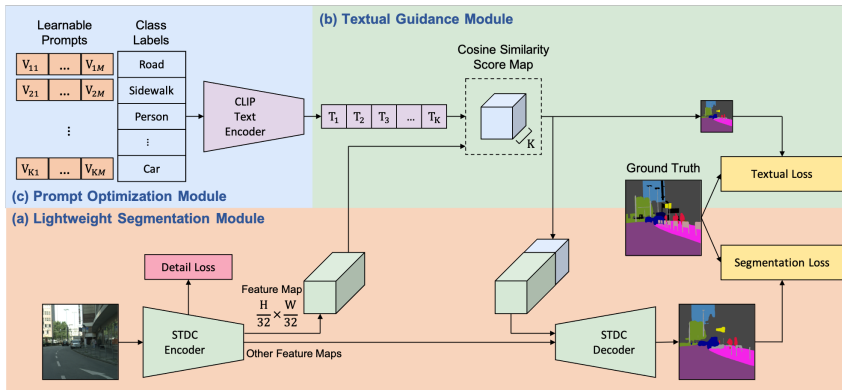

Figure 1: The overview of the proposed framework: (a) lightweight segmentation module; (b) textual guidance module; (c) prompt optimization module.

derived from the CLIP text encoder. This stage aims to distill the segmentation backbone with the comprehensive textual knowledge from the CLIP model, enriching its feature representation. In the subsequent stage, the focus shifts to optimize the class-specific learnable prompt embeddings while the backbone's parameters are held constant. This enables the model to refine the prompt embedding to capture class-specific nuances, thereby elevating the model's segmentation performance.

Our framework, encompassing the proposed components and training strategies, significantly enhances the segmentation performance of the STDC-Seg model while only introducing negligible additional latency at the inference stage. This makes our solution both efficient and practical for deployment in real-world applications.

Our primary contributions are summarized as follows:

- We present a novel approach to enhance the real-time semantic segmentation model STDC-Seg by efficiently distilling CLIP's textual knowledge, effectively boosting performance while maintaining low latency during inference.
- For real-time semantic segmentation, we develop a two-stage training methodology to effectively train both the segmentation backbone and class-specific learnable prompts, thereby elevating the segmentation performance.
- Through rigorous experimentation and ablation studies on two benchmark datasets for semantic segmentation, we demonstrate the effectiveness of our approach and offer insightful findings.

The structure of this paper is as follows: Section 2 delves into a thorough review of the related works, examining the current research landscape and previous studies in the domain. Section 3 details our proposed methodology, outlining the architecture, algorithms, and essential elements of our approach. Section 4 discusses the experimental findings, including the descriptions of the datasets used, implementation details, performance evaluations, and visualization, along with various ablation studies. Section 5 concludes the paper by summarizing the main outcomes and underscoring the contributions and impacts of our research.

2 Related Work

In this section, we briefly describe the recent relevant research related to real-time semantic segmenation, pre-trained vision-language model, prompt tuning, and semantic segmentation with pre-trained vision-language models.

2.1 Real-Time Semantic Segmentation

Real-time semantic segmentation has garnered significant attention as a research domain focusing on achieving precise image segmentation in real-time contexts. Various approaches have been introduced to address this challenge: Zhao *et al.* [31] proposed ICNet which introduces a multi-resolution cascaded network architecture designed for real-time performance on high-resolution images. Yu *et al.* [29] developed BiSeNetV1 which utilizes a dual-path network architecture that efficiently processes spatial information and global context, aiming for compactness and efficiency. Li *et al.* [13] proposed DFANet which leveraged multi-scale feature propagation and aggregation to achieve a balance between computational load and accuracy. Mehta *et al.* [16] presented ESPNetv2 which utilized grouped point-wise and depth-wise dilated separable convolutions to minimize computational demands. Yu *et al.* [28] further proposed BiSeNetV2 to enhance the original BiSeNetV1 architecture, achieving further performance improvements. Chen *et al.* [3] developed FasterSeg to apply neural architecture search (NAS) proposed by Zoph and Le [34] with fine-grained latency regularization, targeting enhanced accuracy for low-latency networks. Fan *et al.* [7] proposed STDC-Seg which introduces spatio-temporal decomposition along with context aggregation mechanisms, improving both accuracy and efficiency, establishing the STDC backbone as a robust, efficient option. Peng *et al.* [19] presented PP-LiteSeg to refine the decoder of STDC-Seg, resulting in speed and accuracy enhancements. Xu *et al.* [27] proposed PIDNet which exploits a three-branch network architecture to more effectively parse the detailed, context, and boundary information for better segmentation performance than other two-branch approaches. This study

specifically focuses on STDC-Seg, utilizing its innovative STDC backbone, which has demonstrated state-of-the-art performance, as the baseline for our research.

2.2 Pre-Trained Vision-Language Model

Vision-language pre-training models proposed by Radford *et al.* [21] and Jia *et al.* [10], Chen *et al.* [4], Zhang *et al.* [30] have emerged as a prominent area of research, focusing on developing models capable of learning combined representations of images and text. Among various methodologies, CLIP and Align proposed by Radford *et al.* [21] and Jia *et al.* [10] respectively stand out, particularly CLIP, which has received considerable attention for its innovative approach. CLIP employs a vision-language contrastive training mechanism to create a unified embedding space for both images and texts, facilitating zero-shot inference capabilities across a range of tasks. The effectiveness of CLIP is largely due to its ability to assimilate knowledge from a broad spectrum of data and its use of contrastive learning. This approach not only promotes the learning of distinct representations for correlated (positive) and uncorrelated (negative) pairs but also significantly enhances the model's ability for generalization. In our research, we harness the pre-trained CLIP text encoder to infuse textual knowledge into our semantic segmentation framework, leveraging its powerful representation.

2.3 Prompt Tuning

Prompt tuning methods proposed by Petroni *et al.* [20], Gao *et al.* [9], and Liu *et al.* [14] have evolved as a key approach to adapt the pre-trained language models to specific downstream tasks, thereby improving their performance. This approach involves adjusting the model to better suit the task requirements through various techniques, including the creation of hand-crafted prompt templates as demonstrated by Petroni *et al.* [20] or the automatic prompt generation as proposed by Gao *et al.* [9] using advanced sequence-to-sequence large language models, such as T5 developed by Raffel *et al.* [22]. Additionally, some methods focus on optimizing prompts directly within the embedding space as presented by Liu *et al.* [14], rather than formulating prompt templates in textual form. In the context of vision-language integration, innovative approaches like CoOp and CLIP-Adapter have been developed. Zhou *et al.* [32] proposed CoOp to introduce the learnable contexts to the text encoder of CLIP, aiming to boost performance in tasks like few-shot image classification. Conversely, Gao *et al.* [8] developed CLIP-Adapter which employed lightweight feature adapters that can be trained with a small amount of annotated data. Khattak *et al.* [11] further proposed Multi-modal Prompt Learning (MaPLe) for CLIP which performs joint prompt tuning upon both vision and language

branches for improved alignment between both representations. These advancements enhancing the model's efficiency and effectiveness in interpreting and generating descriptions for visual content. Given the relevance and efficiency of CoOp's prompt tuning mechanism, especially for real-time applications, our research incorporates it into our framework.

2.4 Semantic Segmentation with Pre-Trained Vision-Language Model

The integration of pre-trained vision-language models has significantly advanced semantic segmentation, with research demonstrating their capability to boost performance. Language-Driven Semantic Segmentation (LSeg) proposed by Li *et al.* [12] and DenseCLIP by Rao *et al.* [23] are notable examples. LSeg enabled zero-shot open-vocabulary semantic segmentation, as demonstrated by Bucher *et al.* [2] earlier, to combine the text encoder of CLIP with Vision Transformer-based segmentation models, which are stemmed from the base model proposed by Dosovitskiy *et al.* [6], through aligning their decoder output feature maps. Rao *et al.* [23] developed DenseCLIP to further enhance the alignment between image and text embeddings and introduced visual context-aware prompt tuning with an additional Transformer module proposed by Vaswani *et al.* [26], setting new benchmarks across multiple semantic segmentation datasets. Furthermore, Zhou *et al.* [33] proposed ZegCLIP for enhanced zero-shot semantic segmentation by combining the image-level prior into text embedding before text-patch matching upon corresponding embeddings from CLIP.

However, these methodologies typically do not cater to real-time processing demands due to their complex architectures. Our research is distinct in its focus on real-time semantic segmentation, leveraging the rich textual knowledge from the text encoder of CLIP to improve the STDC-Seg model's accuracy without significantly affecting inference speed. By incorporating textual knowledge, our goal is to achieve an optimal balance between accuracy and speed, addressing the critical need for efficiency in real-world semantic segmentation applications.

3 Methodology

This section presents a detailed description of our proposed framework for real-time semantic segmentation. Our framework consists of three main modules: (1) the Lightweight Segmentation Module (LSM), which is designed to ensure the segmentation process remains efficient and fast, crucial for real-time applications. It employs a streamlined architecture that minimizes computational overhead while maintaining high accuracy in segmentation tasks, such as STDC-Seg; (2) Textual Guidance Module (TGM), which leverages the rich semantic knowledge embedded in the text encoders of the pretrained

vision-language foundation models like CLIP. It aims to enrich the segmentation model's understanding by aligning its visual features with the textual representations from the pre-trained foundation model, facilitating a deeper semantic comprehension of the images being processed; (3) Prompt Optimization Module (POM), which focuses on refining the interaction between the segmentation model and the textual guidance provided by TGM. It employs class-specific learnable prompts, which are optimized to enhance the model's ability to utilize textual information effectively, thereby improving its segmentation performance. We aim to integrate these modules to enhance the performance of STDC-Seg, a real-time semantic segmentation model. In the following subsections, we provide a comprehensive elaboration of each module, followed by our designed training procedure.

3.1 Model Architectures

3.1.1 Lightweight Segmentation Module (LSM)

Our LSM module employs an encoder-decoder architecture tailored for efficient semantic segmentation. The encoder's role is to extract the multi-scale image features from the shallow to deep layers. This process generates multiple sets of feature maps at different levels, which are crucial for understanding the image's content at various scales. On the other hand, the decoder is responsible to combine these extracted features to construct the final segmentation result. The procedure begins with the high-level, semantically rich features, which are progressively upsampled and merged with lower-level, detail-oriented features through pixel-wise addition. This iterative feature aggregation from the deep to the shallow layers ensures that the resultant feature map is comprehensive to contain all necessary details for pixel-level classification. Thus, this enables precise segmentation as shown in Figure 1(a).

Formally, given an input image X and its corresponding ground-truth semantic mask Y, our objective is to acquire the predicted mask $Y' = D(E(X))$ which is derived from the encoder E and decoder D to closely approximate Y. For the purposes of this study, we have chosen STDC-Seg to serve as the LSM, capitalizing on its efficiency and effectiveness in real-time semantic segmentation tasks.

3.1.2 Textual Guidance Module (TGM)

Inspired by DenseCLIP [23], our TSM module aims to empower the LSM module, the STDC backbone, by infusing it with the rich textual knowledge from the CLIP text encoder. It processes the feature map output from the final stage of the STDC backbone and the text embeddings for each class label generated by the CLIP text encoder as shown in Figure 1(b).

We denote the output feature maps from the first stage to the fourth stage of the STDC backbone as $F^{(1)}, F^{(2)}, F^{(3)}, F^{(4)} \leftarrow E(X)$, where $F^{(1)} \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times d^{(1)}}$, $F^{(2)} \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times d^{(2)}}$, $F^{(3)} \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times d^{(3)}}$, $F^{(4)} \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times d^{(4)}}$, H and W represent the height and width of the input image, and $d^{(1)}$ to $d^{(4)}$ are the channel dimensions, respectively. The class labels (e.g., road, person, car), $L_1, L_2, ..., L_K$ corresponding to K classes, are encoded into text embeddings $T_1, T_2, ..., T_K \in \mathbb{R}^{d^{(4)}}$ by passing them through the CLIP text encoder.

The core operation of TGM involves calculating the cosine similarity between each pixel in $F^{(4)}$ and the text embeddings $T_1, T_2, ..., T_K$. We choose to calculate the similarity using $F^{(4)}$ instead of other encoded features due to the efficiency consideration that it possesses the smallest resolution. As a result, this computation yields a score map $S \in \mathbb{R}^{\frac{H}{32} \times \frac{W}{32} \times K}$, where each element $S_{i,j,k}$ represents the similarity between the feature map at location (i, j) and the text embedding for class k. The equation is as follows:

$$S_{i,j,k} = \frac{F^{(4)}_{i,j} \cdot T_k}{\|F^{(4)}_{i,j}\| \|T_k\|}, i = 1, ..., \frac{H}{32}, j = 1, ..., \frac{W}{32}, k = 1, ..., K \tag{1}$$

This score map, essentially a lower-resolution semantic prediction, is then used as an auxiliary task to refine the segmentation output by aligning it closer to the ground truth. Moreover, by computing cosine similarity, the image features of STDC backbone are aligned with the text embeddings from CLIP. This not only allows the model to integrate and utilize the rich textual knowledge from the CLIP text encoder but also boosts its semantic representation capabilities.

3.1.3 Prompt Optimization Module (POM)

The POM module, inspired from CoOp proposed by Zhou *et al.* [32], is designed to manage class-specific learnable prompt embeddings and aims to refine the discriminative power of the text embeddings for each class label. These embeddings are denoted as $V_{ij} \in \mathbb{R}^{d_t},\ i = 1, 2, ..., K,\ j = 1, 2, ..., M$, where K represents the number of classes, M represents the number of learnable embeddings for each class, and d_t is the dimension of the CLIP token embeddings. Additionally, the token embeddings of the class label text are represented by $E_{ik} \in \mathbb{R}^{d_t}$ for $i = 1, 2, ..., K$ classes and $k = 1, 2, ..., l$ tokens, with l indicating the token sequence length. The essence of incorporating class-specific learnable prompt embeddings lies in their ability to enable trainable modifications to the text embeddings provided by the CLIP text encoder, thereby enhancing the specificity and discriminative capabilities of the embeddings for each class label as shown in Figure 1(c).

To achieve this, for each class i, we concatenate the learnable prompt embeddings $V_{i1}, V_{i2}, ..., V_{iM}$ with the token embeddings $E_{i1}, E_{i2}, ..., E_{il}$ for

each class i. This concatenated vector is then processed by the CLIP text encoder to produce the output text embedding $T_i \in \mathbb{R}^d$, which is formulated as follows:

$$T_i = POM([V_{i1}\ V_{i2}\ ...\ V_{iM}\ E_{i1}\ E_{i2}\ ...\ E_{il}]) \tag{2}$$

During the training phase, while the parameters of the CLIP text encoder remain fixed, the learnable text embeddings T_i are optimized within the pre-trained domain of CLIP. This ensures that the enriched text embeddings continue to benefit from the rich textual knowledge inherent in the CLIP model, thus effectively improving our model's performance through the enhanced textual guidance.

3.2 Two-Stage Training Procedure

In our framework, we exploit a meticulously designed two-stage training procedure, which aims at efficiently transferring the rich textual knowledge encoded within the CLIP text encoder to the lightweight STDC backbone. This approach ensures that the semantic segmentation model not only benefits from the advanced capabilities of vision-language pre-training models but also maintains the required efficiency for real-time applications. The subsequent sections will outline the specific loss functions utilized throughout the training phases, followed by a detailed elaboration of each training stage.

3.2.1 Loss Functions

The LSM module, at the heart of our real-time semantic segmentation framework, generates the predicted semantic mask, with optimization achieved through the implementation of cross-entropy loss augmented through online hard example mining (OHEM) developed by Shrivastava *et al.* [25]. The cross-entropy loss (CE), a standard measure in segmentation tasks, is defined as:

$$CE(x, y) = \frac{1}{N} \sum_{n=1}^{N} -\log \left(\frac{\exp(x_{n,y_n})}{\sum_{k=1}^{K} \exp(x_{n,y_k})} \right), \tag{3}$$

where x represents the logits from the model, y is the ground-truth label, N is the total number of pixels, and K is the number of classes. OHEM strategically focuses the training effort on more difficult examples that contribute significantly to the loss, thus enhancing the learning efficiency.

In our framework, the loss for the segmentation task, referred to as L_{seg}, is based on the cross-entropy loss as defined in Equation (3). Additionally, the TGM module produces a score map acting as a low-resolution semantic mask. This output is first upscaled to the same dimensions as the ground-truth mask using bilinear upsampling. Then, we employ the same cross-entropy loss to

them, denoted as L_{text}, for auxiliary training. We use bilinear upsampling here to follow the convention of the previous works, especially Long *et al.* [15], which bilinearly upsampled the low-resolution feature maps to match the resolution of the ground truth labels before calculating the cross-entropy loss. The incorporation of L_{text} facilitates the utilization of the rich semantic information from the pre-trained CLIP model. The alignment with the textual space of the CLIP model offers a form of regularization that aids in developing a more effective segmentation model. Moreover, in line with the STDC-Seg methodology, a detail loss (L_{detail}) is incorporated to further refine our training process. Specifically, we adopt the combined loss function using binary cross-entropy loss (L_{bce}) and dice loss (L_{dice}) to jointly optimize detail learning, enhancing the overall segmentation performance. The detail loss is formulated as follows:

$$L_{detail}(p_e, g_e) = L_{dice}(p_e, g_e) + L_{bce}(p_e, g_e)$$

where p_e denotes the predicted edge map and g_e denotes the corresponding ground-truth edge map generated with Laplacian convolution, more information could be found in Fan *et al.* [7]. The dice loss L_{dice} is defined as:

$$L_{dice}(p_e, g_e) = 1 - \frac{2\sum_i p_e^i g_e^i + \epsilon}{\sum_i (p_e^i)^2 + \sum_i (g_e^i)^2 + \epsilon}$$

Here, i denotes the i-th pixel and ϵ is a smoothing term to avoid division by zero, with ϵ set to 1. The overall loss function, L, combines these components as follows:

$$L = \lambda_1 * L_{seg} + \lambda_2 * L_{text} + \lambda_3 * L_{detail}, \tag{4}$$

where the weighting coefficients λ_1, λ_2, and λ_3 are all set to 1, balancing the contribution of each loss component to the total loss.

3.2.2 First Training Stage

The initial stage of our training procedure focuses on establishing the alignment between the STDC backbone and the fixed text embeddings provided by the CLIP text encoder where during this phase, learnable prompts are not utilized as illustrated in Figure 2. In addition, this alignment allows the STDC backbone to leverage the rich textual knowledge of CLIP as additional information for semantic segmentation.

3.2.3 Second Training Stage

In the second stage, as depicted in Figure 3, we introduce the class-specific learnable prompt embeddings in the POM while the parameters of the STDC backbone are held fixed. This choice is based on the understanding that in

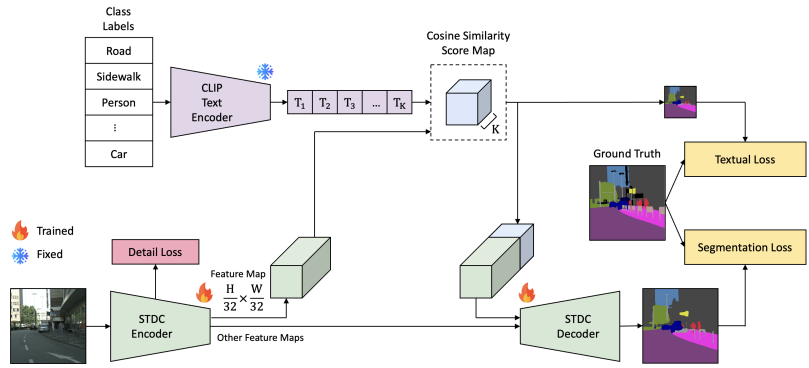

Figure 2: The illustration of our first training stage.

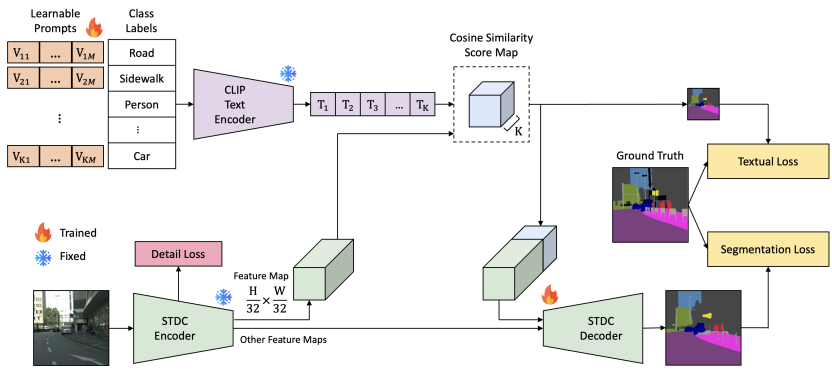

Figure 3: The illustration of our second training stage.

the first stage, the STDC backbone already assimilated textual knowledge from the CLIP text encoder, allowing it to adapt to changing text prompt embeddings. Consequently, the POM focuses on optimizing the prompts to provide precise descriptions for each class label, thereby augmenting the discriminative characteristics of the optimized text prompt embeddings for the STDC backbone.

The entire training procedure is summarized in Algorithm 1. Our framework achieves significant improvements after applying the two-stage training compared to the original model. The LSM can simultaneously handle semantic segmentation tasks with visual and textual knowledge. The POM provides optimized class-specific prompt embeddings with more discriminative information. The TGM connects these two modules and aligns the two embedding spaces, successfully transferring textual knowledge from the CLIP text encoder to the STDC backbone.

3.3 Inference

After applying the two-stage training procedure, the proposed model can still achieve real-time inference speed with introducing a minimal amount of additional computation time compared to the original STDC-Seg model during the inference phase. This achievement is accomplished by retaining only the final trained text embeddings, discarding the learnable prompt embeddings, and the CLIP text encoder, as illustrated in Figure 4. By adopting this approach, the additional computational cost is limited to that of calculating cosine similarity in the TGM and including a few additional channels to the encoded feature maps. Consequently, the overall increase in Floating Point Operations (FLOPs) is negligible, enabling efficient real-time inference while preserving the accuracy and performance gains of our training methodology.

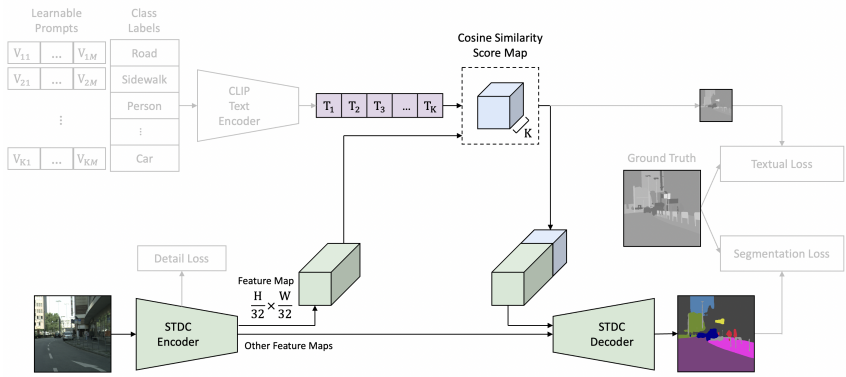

Figure 4: The colored area indicates the inference process of our framework.

4 Experiments

In this section, we present a comprehensive evaluation results and ablation studies of the proposed approach on the semantic segmentation datasets.

4.1 Experimental Settings

4.1.1 Datasets

To evaluate the performance of our proposed methods, we conduct experiments on two standard semantic segmentation datasets, including the Cityscapes dataset [5] and the CamVid dataset [1]. The Cityscapes dataset is a large-scale dataset for urban scene understanding with pixel-level annotations of 30 different classes, where 19 of them are generally used for semantic segmentation

Algorithm 1 The Proposed Two-Stage Training Process.

Input: Training data with image-class pairs $(X,\ Y)$
Input: Label texts $L_1,\ L_2,\ ...,\ L_K$
First Training Stage:
Initialize STDC backbone and CLIP text encoder with pretrained weights
Freeze CLIP text encoder parameters
for each i in K **do**
    $E_{i1},\ E_{i2},\ ...,\ E_{il} \leftarrow CLIPTokenEmbedder(CLIPTokenizer(L_i))$ ▷ l denotes the sequence length of the token embeddings
    $T_i \leftarrow CLIPTextEncoder([E_{i1}\ E_{i2}\ ...\ E_{il}])$
end for
for each training iteration **do**
    $F^{(1)},\ F^{(2)},\ F^{(3)},\ F^{(4)} \leftarrow STDCBackbone(X)$
    for each i in the height of $F^{(4)}$, j in the width of $F^{(4)}$, k in K **do**
        $S_{i,j,k} \leftarrow \frac{F^{(4)}_{i,j} \cdot T_k}{\| F^{(4)}_{i,j} \| \| T_k \|}$
    end for
    $F^{(4)'} \leftarrow [F^{(4)}\ S]$
    $Y' \leftarrow STDCDecoder(F^{(1)},\ F^{(2)},\ F^{(3)},\ F^{(4)'})$
    $Loss \leftarrow CrossEntropyOHEM(Y',\ Y)$ ▷ Loss of the main task only. Losses of auxiliary tasks are omitted here for simplicity.
    Backpropagate and update parameters
end for
Return: Trained model of the first training stage

Second Training Stage:
Initialize STDC backbone and decoder with trained weights from the first training stage
Initialize CLIP text encoder with pretrained weights
Freeze STDC backbone and CLIP text encoder parameters
for each i in K **do**
    $E_{i1},\ E_{i2},\ ...,\ E_{il} \leftarrow CLIPTokenEmbedder(CLIPTokenizer(L_i))$
end for
for each training iteration **do**
    for each i in K **do**
        $T_i \leftarrow CLIPTextEncoder([V_{i1}\ ...\ V_{iM}\ E_{i1}\ ...\ E_{il}])$ ▷ M denotes the number of learnable prompt embeddings
    end for
    $F^{(1)},\ F^{(2)},\ F^{(3)},\ F^{(4)} \leftarrow STDCBackbone(X)$
    for each i in the height of $F^{(4)}$, j in the width of $F^{(4)}$, k in K **do**
        $S_{i,j,k} \leftarrow \frac{F^{(4)}_{i,j} \cdot T_k}{\| F^{(4)}_{i,j} \| \| T_k \|}$
    end for
    $F^{(4)'} \leftarrow [F^{(4)}\ S]$
    $Y' \leftarrow STDCDecoder(F^{(1)},\ F^{(2)},\ F^{(3)},\ F^{(4)'})$
    $Loss \leftarrow CrossEntropyOHEM(Y',\ Y)$ ▷ Loss of the main task only. Losses of auxiliary tasks are omitted here for simplicity.
    Backpropagate and update parameters
end for
Return: Trained model of the second training stage

tasks. There are 5,000 carefully annotated images with 2,048 × 1,024 pixels in size, split into three subsets: training, validation, and testing sets, with 2,975, 500, and 1,525 images, respectively. Besides, the CamVid dataset, collected from vehicle-mounted cameras, has 11 semantic classes. It comprises 701 annotated images with 960 × 720 pixels in size, also split into three subsets: training, validation, and testing sets, with 367, 101, and 233 images, respectively. We train our models with a training set and evaluate them on a validation set of each dataset separately.

4.1.2 Implementation Details

In our experiments, we utilize stochastic gradient descent (SGD) as shown in Ruder [24] as our optimizer with a momentum of 0.9 and weight decay of 0.0005. For the Cityscapes dataset, the initial learning rate is set to 0.005 for the backbone and learnable prompts and 0.05 for other parameters. The batch size is set to 24, and we employ the "poly" learning rate policy, where the initial rate is multiplied by $(1 - \frac{\text{current\_iteration}}{\text{max\_iterations}})^{\gamma}$ with a total of 160,000 iterations and γ equals 0.9. Additionally, we use a linear warm-up strategy for the first 1,000 steps.

For the CamVid dataset, the initial learning rate is set to 0.01 for the backbone and learnable prompts and 0.1 for other parameters. The batch size is set to 16, and we also use the "poly" learning rate policy with the same parameters as before but with a total of 10,000 iterations. The warm-up strategy is applied for the first 200 steps.

We apply data augmentation techniques in both datasets, including color jittering, random horizontal flipping, random cropping, and random resizing, to enhance the training data. For the Cityscapes dataset, we train the model using a scale range of [0.125, 1.5] stepping by 0.125, and the cropped resolution is set to 1024 × 512, following the same settings as STDC1-Seg50. On the other hand, for the CamVid dataset, the scale range is [0.5, 2.5] stepping by 0.25, and the cropped resolution is set to 960 × 720, the same as STDC1-Seg.

During the training and evaluation stages, we use the MMSegmentation [17] library version 0.30.0 along with PyTorch 1.12.1 [18]. The training is performed on a system equipped with two NVIDIA GTX 3090 GPUs, utilizing CUDA 11.3 and cuDNN 8.2. For testing the inference speed, we conduct experiments on a different setup. We modify the code provided by STDC-Seg [7] and use PyTorch 1.2.0, CUDA 10.0, and cuDNN 7.5. Additionally, we employ TensorRT 5.1.5.0 to optimize the inference process as STDC-Seg did. These experiments are further carried out on an NVIDIA GTX 1080Ti GPU to benchmark the inference speed of our method.

4.2 Main Results

In our evaluation, we compare the performance of our proposed methods with the baseline STDC-Seg model on both the Cityscapes and the CamVid datasets. By doing so, we aim to demonstrate the effectiveness of our approach in improving the performance of STDC-Seg using textual knowledge.

4.2.1 Evaluation Results on Cityscapes

In Table 1, we present the segmentation accuracy and inference speed of the proposed method on the Cityscapes validation set. Notably, compared to the baseline STDC-Seg50 model, which achieves a mean Intersection over Union (mIoU) of 72.2% for the validation set, our proposed methods incorporating textual knowledge through a two-stage training procedure can enhance the performance to 72.9%. This improvement comes with only a slight sacrifice in inference speed. Additionally, we compared our model with other state-of-the-art methods, as shown in Figure 5. Methods further to the right indicate higher inference speed, while methods higher up indicate higher mIoU. Figure 5 demonstrates that our method strikes an excellent balance between quality and efficiency.

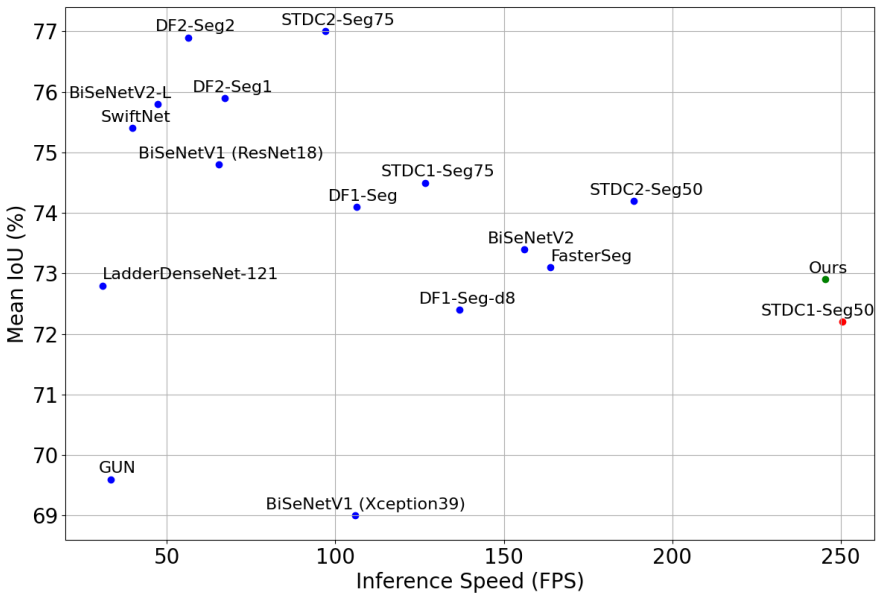

Figure 5: Speed-accuracy trade-off comparisons on the Cityscapes validation set. The green dot indicates our method, red dot indicates the STDC1-Seg50 method, and blue dots mean other methods.

Table 1: Quantitative comparisons of the baseline and our model with the proposed two-stage training on the Cityscapes validation set.

Model	#Params ↓	FLOPs ↓	mIoU (%) ↑	Inference Speed (FPS) ↑
STDC1-Seg50 (baseline)	8.57M	16.95G	72.2	250.4
Ours (stage 1)	8.60M	16.96G	72.7	245.3
Ours (stage 2)	8.60M	16.96G	**72.9**	245.3

4.2.2 Evaluation Results on CamVid

Table 2 presents the comparison results of our method and STDC-Seg on the CamVid dataset. We train the STDC-Seg model on the training set and evaluate its performance on the validation set. By applying our two-stage training procedure with text knowledge guidance, we are able to increase the mIoU of STDC-Seg from 75.3% to 78.0%. This significant performance improvement further demonstrates the effectiveness of our method in enhancing segmentation accuracy by leveraging textual knowledge.

Table 2: Quantitative comparisons of the baseline and our model with the proposed two-stage training on the CamVid validation set.

Model	#Params ↓	FLOPs ↓	mIoU (%) ↑	Inference Speed (FPS) ↑
STDC1-Seg (baseline)	8.57M	22.42G	75.3	197.6
Ours (stage 1)	8.59M	22.43G	76.6	196.8
Ours (stage 2)	8.59M	22.43G	**78.0**	196.8

4.2.3 Qualitative Results

Furthermore, we present qualitative results on the Cityscapes validation set, as depicted in Figure 6. Specifically, our method achieves more accurate segmentation of poles (colored in gray) compared to STDC-Seg. Poles are challenging to segment accurately due to their thin shape, which can often be overlooked by the model. However, our results demonstrate that the transfer of textual knowledge effectively enhances performance. In Figure 6(a), (b), and (c), the segmentation masks of our approach for the poles exhibit improved accuracy, addressing the issue of missing or incomplete segmentation in STDC-Seg. Furthermore, we observe similar improvements in segmenting fences (colored in Tuscan) as well. Fences are challenging to segment as they frequently overlap with other objects, such as buildings or walls. Our method demonstrates better results, suggesting that the transmitted textual knowledge

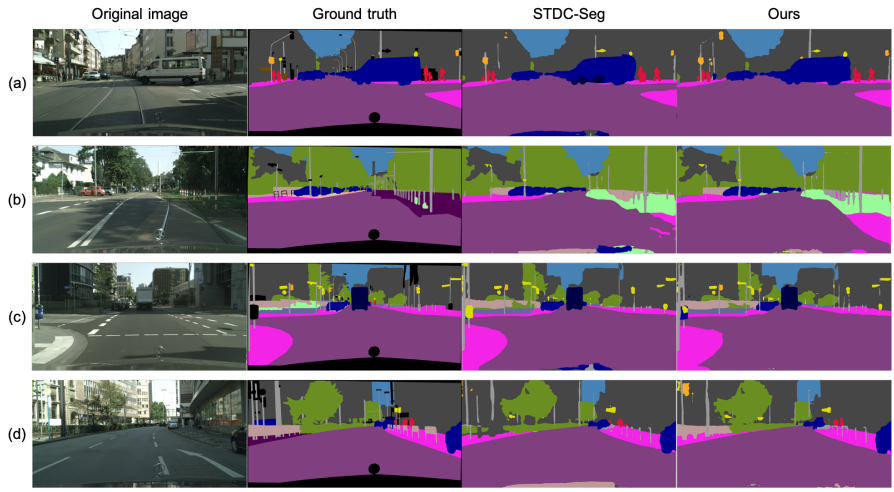

Figure 6: Comparisons of qualitative results between the proposed method and the SOTA (i.e., the STDC-Seg). Note that the region colored in black is marked as "ignore" in the standard evaluation. Best viewed in color and zoom in.

aids in discriminating the fence from other objects. In Figure 6(c) and (d), we can observe substantial improvements in fence segmentation compared to STDC-Seg.

These qualitative results demonstrate the effectiveness of our approach in enhancing the segmentation of challenging objects such as poles and fences. By leveraging the textual knowledge transmitted to the model, we address the limitations of the base STDC-Seg method and achieve more accurate and detailed segmentation results.

4.3 Ablation Studies

In this section, we conduct detailed studies to further demonstrate the contribution of each component to the final segmentation performance.

4.3.1 Effectiveness of Two-Stage Training Procedure

In the early stages of our experiments, we encountered challenges in directly transfer textual knowledge from the CLIP text encoder to the STDC backbone with learnable prompts. Training the learnable prompts and the STDC backbone at the same time results in worse performance than our two-stage settings. We believe this is because the STDC backbone lacks textual knowledge before our training since it is not pre-trained with a vision-language

training procedure. Consequently, when trained simultaneously with learnable prompts, the text embeddings from the CLIP text encoder constantly change, making it difficult for the STDC backbone to keep up and leading to unstable performance.

To address this issue, we adopt a two-stage training procedure. In the first stage, we fix the text embeddings and allow the STDC backbone to align itself with the textual knowledge. In the second stage, we fix the STDC backbone and adjust the learnable prompts to optimize the "prompt sentence" from the embedding space.

As shown in Table 3, on the Cityscapes dataset, training the model without the two-stage procedure yield an mIoU of 72.7%, whereas, with the two-stage training, the mIoU is improved to 72.9%. Similarly, as shown in Table 4, on the CamVid dataset, training without the two-stage procedure results in an mIoU of 76.4%, while training with the two-stage procedure achieves an mIoU of 78.0%. These results indicate a significant improvement in adopting of the two-stage training procedure.

Table 3: Accuracy comparisons between training Procedures without and with multiple stages (Cityscapes validation set).

Procedure	**mIoU (%) ↑**
end-to-end 1 stage	72.7
2 stages (1^{st} stage)	72.7
2 stages (2^{nd} stage)	**72.9**

Table 4: Accuracy comparisons between training procedures without and with multiple stages (CamVid validation set).

Procedure	**mIoU (%) ↑**
end-to-end 1 stage	76.4
2 stages (1^{st} stage)	76.6
2 stages (2^{nd} stage)	**78.0**

Based on these findings, we strongly suggest that the two-stage training procedure is crucial for effectively transmitting textual knowledge from the CLIP text encoder to the STDC backbone. It allows for better alignment and optimization, ultimately improving the segmentation performance on both the Cityscapes and CamVid datasets.

4.3.2 Number of Learnable Prompt Embeddings

During the second stage of training, we conduct experiments to determine the optimal number of learnable prompt embeddings, denoted as M. We explore different values of M, specifically $M = 8,\ 11,\ 14,$ and 17, and analyze their impact on performance. As shown in Table 5, we observe that the performance gain reaches a saturation point around $M = 11$. This suggests that increasing the number of embeddings beyond a certain threshold does not yield significant improvements in performance.

We conjecture that there is a limitation to the number of embeddings because excessively long prompts may contain unrelated or extraneous information for the task of classifying between labels in segmentation. Therefore, there is a balance to be struck between having enough prompt embeddings to capture relevant information and avoiding an excessive number that includes noise or irrelevant details.

Table 5: Comparisons of the effects of different Numbers of learnable prompt embeddings on the CamVid validation set.

M	**mIoU (%) ↑**
8	76.3
11	**76.4**
14	75.3
17	76.0

4.3.3 Regularization of Prompt Embedding Space

We make an interesting observation regarding the regularization of the learnable prompt embeddings and their impact on performance. Specifically, in the CamVid dataset, we notice that applying an L2 weight decay regularization of 0.0005 to the prompt embeddings results in a significant drop in performance, with the mIoU decreasing from 76.3% to 75.9%, as shown in Table 6. This trend is consistent across most of our experiments.

We believe the reason behind this phenomenon lies within the embedding space. If we imagine the embeddings space of all tokens as a normal distribution, with the centers representing frequently occurring words and the outer regions containing less frequent words, applying L2 weight decay regularization to the learnable prompt embeddings tends to push these embeddings towards the zero vector, which corresponds to the center of the distribution. Consequently, the prompts would predominantly consist of frequent words.

Table 6: Comparisons of the effects of using weight decay in learnable prompt embeddings on the CamVid validation set.

Weight decay (M=8)	**mIoU (%) ↑**
0.0005	75.9
No weight decay	**76.3**

However, we argue that a powerful prompt should incorporate specific words that help the model distinguish between different labels. Relying solely on frequent words may not provide the necessary discriminative power. From a higher-level perspective, let's consider the well-known prompt "A photo of a [CLASS]." In this example, words like "a" and "of" are likely close to the distribution's center, representing frequent words. On the other hand, the word "photo" is more specific and likely to be further away from the center. Restricting the prompt embeddings to only frequent words might hinder the ability of the prompts to convey specific meanings.

Based on these observations, we suggest that: to equip the learnable prompts with the ability to capture the specific meanings from the embedding space, one has better not to apply L2 weight decay regularization on them. By doing so, we provide the prompts with the opportunity to reach beyond frequent words and incorporate more specific and contextually relevant terms, ultimately improving the performance of the segmentation model.

4.4 Discussions

To gain a deeper understanding of our model's capabilities, we conducted a detailed analysis of the segmentation performance on the Cityscapes validation set, focusing on specific classes. The results, presented in Table 7, reveal significant improvements for key urban elements, including sidewalks, poles, and terrains. For example, sidewalks show a notable 2.48% improvement, poles exhibit a 2.38% gain, and terrains achieve a 2.15% increase in IoU. These enhancements underscore the efficacy of integrating textual knowledge through our two-stage training approach.

However, it is crucial to acknowledge certain performance variations in specific classes, particularly in the case of buses and trucks, which experience notable drops of 4.70% and 8.75%, respectively. Similarly, trains exhibit a decline with a performance drop of 4.42%. These variations highlight potential challenges in certain vehicle-related classes.

To further investigate these class-specific challenges, we examined our model's performance at a broader category level, as detailed in Table 8. Despite the limitations observed in specific vehicle classes, our method demonstrates

Table 7: Quantitative comparisons of different classes on the Cityscapes validation set. Best viewed in color.

Label	Number of Pixels	Portion of Pixels (%)	STDC1-Seg50 IoU (%)	Ours IoU (%)	Performance Gain (%)
road	345,221,953	37.66	97.6	97.8	+0.20
building	200,894,857	21.92	90.4	91.0	+0.66
vegetation	158,682,635	17.31	90.2	90.8	+0.67
car	59,759,316	6.52	93.1	93.6	+0.54
sidewalk	49,558,716	5.41	80.7	82.7	+2.48
sky	30,708,059	3.35	93.0	93.3	+0.32
pole	13,565,290	1.48	50.4	51.6	+2.38
person	11,890,232	1.30	73.7	74.1	+0.54
terrain	7,625,891	0.83	60.5	61.8	+2.15
fence	7,527,053	0.82	51.8	55.3	+6.76
wall	6,720,672	0.73	53.3	55.1	+3.38
bicycle	6,500,853	0.71	69.9	71.5	+2.29
traffic sign	6,110,650	0.67	69.9	71.4	+2.15
bus	3,564,222	0.39	82.9	79.0	-4.70
truck	2,760,469	0.30	77.7	70.9	-8.75
rider	1,970,537	0.21	53.8	55.5	+3.16
traffic light	1,813,814	0.20	56.7	60.3	+6.35
train	1,032,099	0.11	74.7	71.4	-4.42
motorcycle	728,923	0.08	51.9	57.9	+11.56

Table 8: Quantitative comparisons of different categories on the Cityscapes validation set. Best viewed in color.

Category	STDC1-Seg50 IoU (%)	Ours IoU (%)	Performance Gain (%)
construction	90.8	91.3	+0.57
flat	98.2	98.4	+0.14
human	75.1	75.7	+0.88
nature	90.6	91.0	+0.49
object	58.6	60.2	+2.74
sky	93.0	93.3	+0.33
vehicle	91.4	91.8	+0.43

superior performance in the overall vehicles category, with an overall positive performance gain of 0.43%. This suggests that although certain specific vehicle classes show a decline, the overall prediction for the vehicle category is improved with our method. Moreover, our model showcases superior performance across all categories, emphasizing its capability to outperform the baseline in capturing diverse urban scene characteristics.

Qualitative results in Figure 7 provide visual illustrationswhere our model encountered challenges, especially in distinguishing between buses, trains, and trucks. Figure 7(a) and (b) show buses in the center of the images, where our

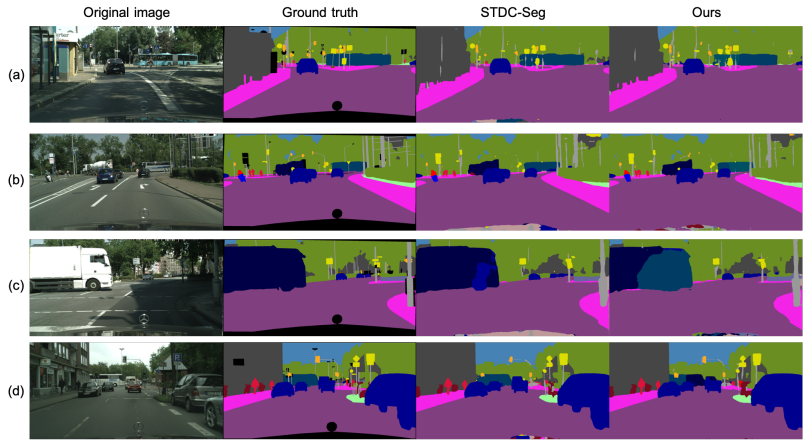

Figure 7: Illustrations of the limitations of the proposed method. Best viewed in color and zoom in.

model misclassifies part of the buses as trains. In Figure 7(c), a truck on the left of the image is not fully recognized, with part of it predicted as a train. Similarly, in Figure 7(d), a bus in the center of the input image is misclassified as part of a truck. These observations highlight the potential for confusion between similar-shaped vehicles, indicating a need for further refinement in capturing distinctive features.

Figure 8 reinforces our method's strength in handling challenging scenarios, such as occlusions. Figure 8(a), (b), (c), and (d) depict buses overlapping with cars, a situation where our model accurately segments buses even in the presence of occlusions. This capability is attributed to the leverage of textual knowledge, providing rich semantic information that aids in distinguishing between different classes, even under the challenging conditions.

In summary, while our model exhibits challenges in specific vehicle classes, the overall performance showcases the effectiveness of our approach in enhancing semantic segmentation in diverse urban scenes.

5 Conclusion

In conclusion, this research has contributed to the field of real-time semantic segmentation by leveraging the pre-trained vision-language model CLIP. The proposed framework enhances the STDC-Seg model by effectively incorporating textual knowledge from the CLIP text encoder. This integration enables the fusion of visual and textual information, leading to substantial performance improvements while maintaining minimal additional latency during

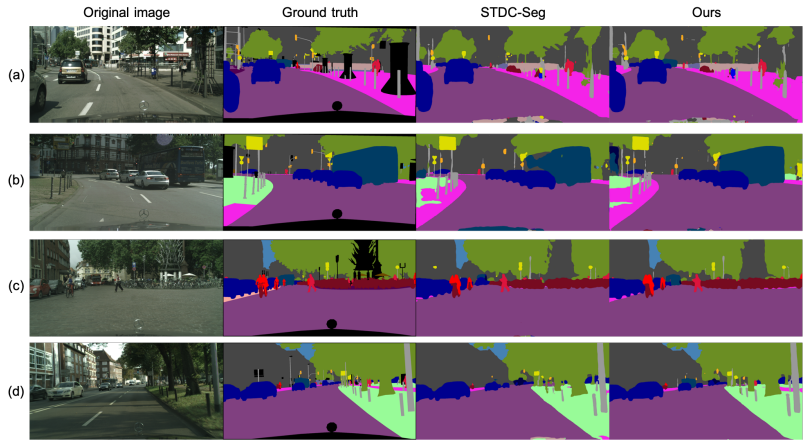

Figure 8: More comparison results on the category of vehicles between the proposed method and the the STDC-Seg. Best viewed in color and zoom in.

inference. To further improve the model's semantic understanding and capture class-specific details, learnable prompt embeddings are introduced. The two-stage training procedure effectively trains both the lightweight segmentation backbone and the learnable prompts, optimizing the model's performance in real-time semantic segmentation tasks.

Acknowledgments

The project is supported by the Featured Area Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (113L900901/113L900902/113L900903), and National Science and Technology Council (NSTC), Taiwan (R.O.C) under grant numbers of 111-2634-F-002-022, 111-2221-E-002-134-MY3, and 112-2634-F-002-006.

References

[1] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds", in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2008, 44–57.

[2] M. Bucher, T.-H. Vu, M. Cord, and P. Pérez, "Zero-shot semantic segmentation", *Advances in Neural Information Processing Systems*, 32, 2019.

[3] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation", *arXiv preprint arXiv:1912.10917*, 2019.

[4] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning", in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2020, 104–20.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 3213–23.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.

[7] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking bisenet for real-time semantic segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, 9716–25.

[8] P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao, "Clip-adapter: Better vision-language models with feature adapters", *arXiv preprint arXiv:2110.04544*, 2021.

[9] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners", *arXiv preprint arXiv:2012.15723*, 2020.

[10] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision", in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2021, 4904–16.

[11] M. U. Khattak, H. Rasheed, M. Maaz, S. Khan, and F. S. Khan, "Maple: Multi-modal prompt learning", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, 19113–22.

[12] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven semantic segmentation", *arXiv preprint arXiv:2201.03546*, 2022.

[13] H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep feature aggregation for real-time semantic segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, 9522–31.

[14] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too", *arXiv preprint arXiv:2103.10385*, 2021.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, 3431–40.

[16] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, 9190–200.

[17] MMSegmentation Contributors, "MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark", 2020, https://github.com/open-mmlab/mmsegmentation.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, 8024–35.

[19] J. Peng, Y. Liu, S. Tang, Y. Hao, L. Chu, G. Chen, Z. Wu, Z. Chen, Z. Yu, Y. Du, *et al.*, "Pp-liteseg: A superior real-time semantic segmentation model", *arXiv preprint arXiv:2204.02681*, 2022.

[20] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?", *arXiv preprint arXiv:1909.01066*, 2019.

[21] 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 *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2021, 8748–63.

[22] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", *The Journal of Machine Learning Research*, 21(1), 2020, 5485–551.

[23] Y. Rao, W. Zhao, G. Chen, Y. Tang, Z. Zhu, G. Huang, J. Zhou, and J. Lu, "Denseclip: Language-guided dense prediction with context-aware prompting", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, 18082–91.

[24] S. Ruder, "An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, 2016.

[25] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, 761–9.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need", *Advances in neural information processing systems*, 30, 2017.

[27] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A Real-Time Semantic Segmentation Network Inspired by PID Controllers", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, 19529–39.

[28] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation", *International Journal of Computer Vision*, 129, 2021, 3051–68.

[29] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, 325–41.

[30] P. Zhang, X. Li, X. Hu, J. Yang, L. Zhang, L. Wang, Y. Choi, and J. Gao, "Vinvl: Revisiting visual representations in vision-language models", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, 5579–88.

[31] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, 405–20.

[32] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models", *International Journal of Computer Vision*, 130(9), 2022, 2337–48.

[33] Z. Zhou, Y. Lei, B. Zhang, L. Liu, and Y. Liu, "Zegclip: Towards adapting clip for zero-shot semantic segmentation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, 11175–85.

[34] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning", *arXiv preprint arXiv:1611.01578*, 2016.