ZegCLIP: Towards Adapting CLIP for Zero-shot Semantic Segmentation

Ziqin Zhou[1]   Yinjie Lei[2]   Bowen Zhang[1]   Lingqiao Liu[1]*(Corr)   Yifan Liu[1]
[1]The University of Adelaide, Australia
[2]Sichuan University, China

Paper: *https://arxiv.org/abs/2212.03588*

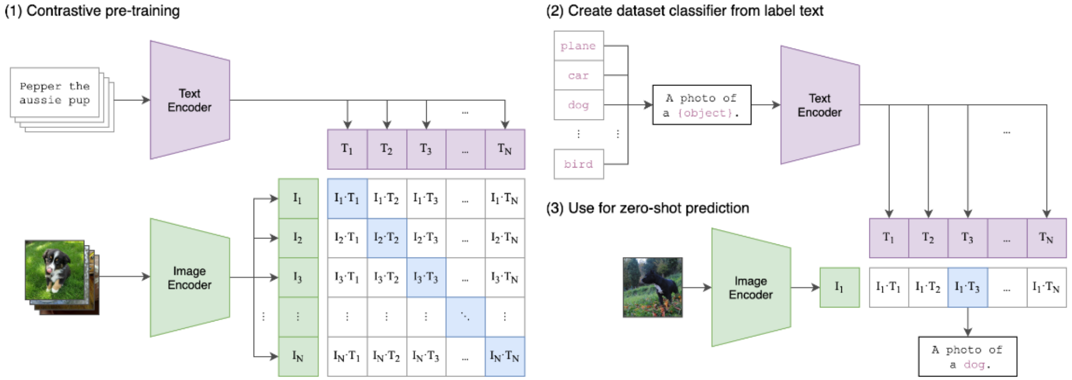
Github: *https://github.com/ZiqinZhou66/ZegCLIP*

Paper

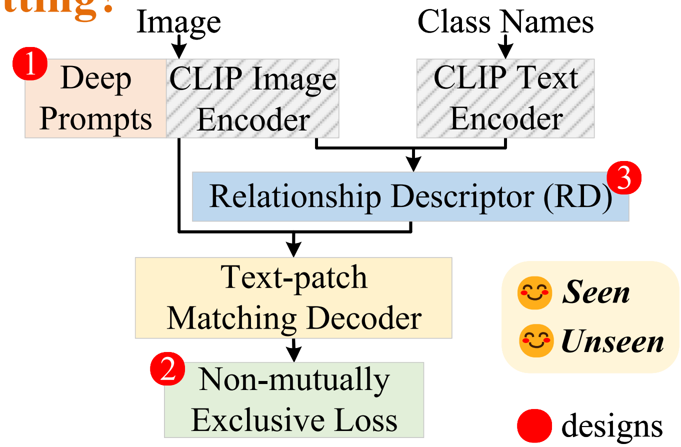
Github

Quick Preview

CLIP has powerful zero-shot classification ability!



How to directly extend CLIP from image to pixel-level in zero-shot setting?

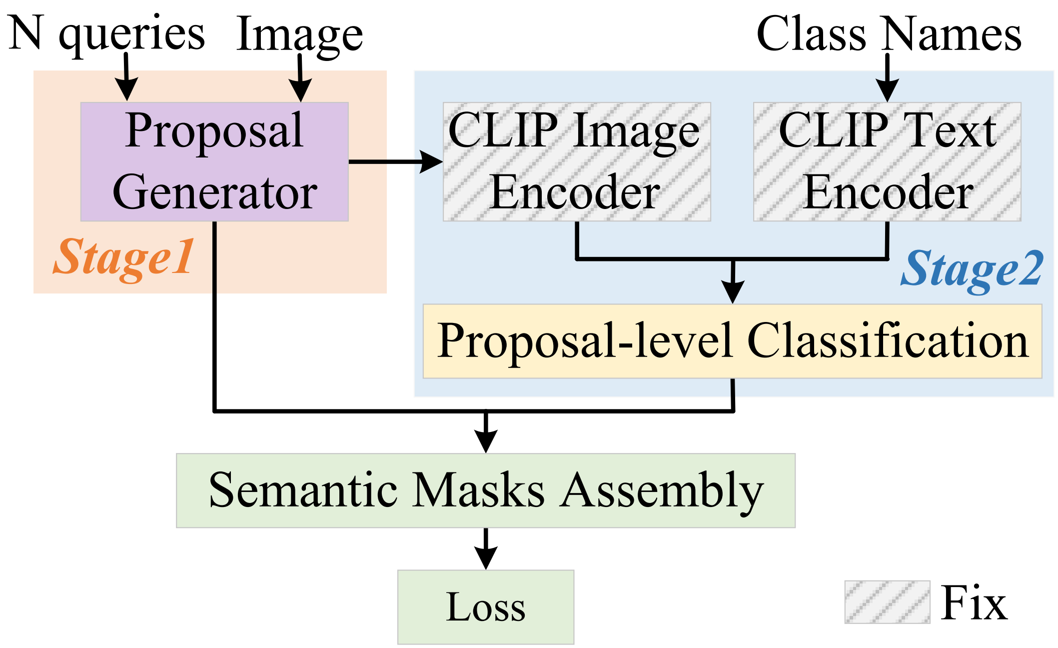


Contribution

- ✓ We propose an efficient **one-stage** straightforward paradigm based on CLIP;
- ✓ We transfer the CLIP's image-level classification ability to dense prediction tasks while maintaining the advanced zero-shot knowledge;
- ✓ We figure out three designs to achieve competitive results on seen classes while extremely improving the performance on novel classes;
- ✓ Our method demonstrates superior performance, outperforming the state-of-the-art methods by a large margin under both "inductive" and "transductive" zero-shot settings on three public benchmark datasets.
- ✓ Compared with the two-stage method, our method has achieved a speedup of about **5 times faster** during inference and shows competitive generalization ability.

Introduction

Review of previous zero-shot semantic segmentation methods based on CLIP



Two-stage Pipeline:

- Stage1: Generate class-agnostic proposals;
- Stage 2: Feed the cropped regions to CLIP for classification.

🎁 **Advantage:**
Inherent zero-shot ability of CLIP.

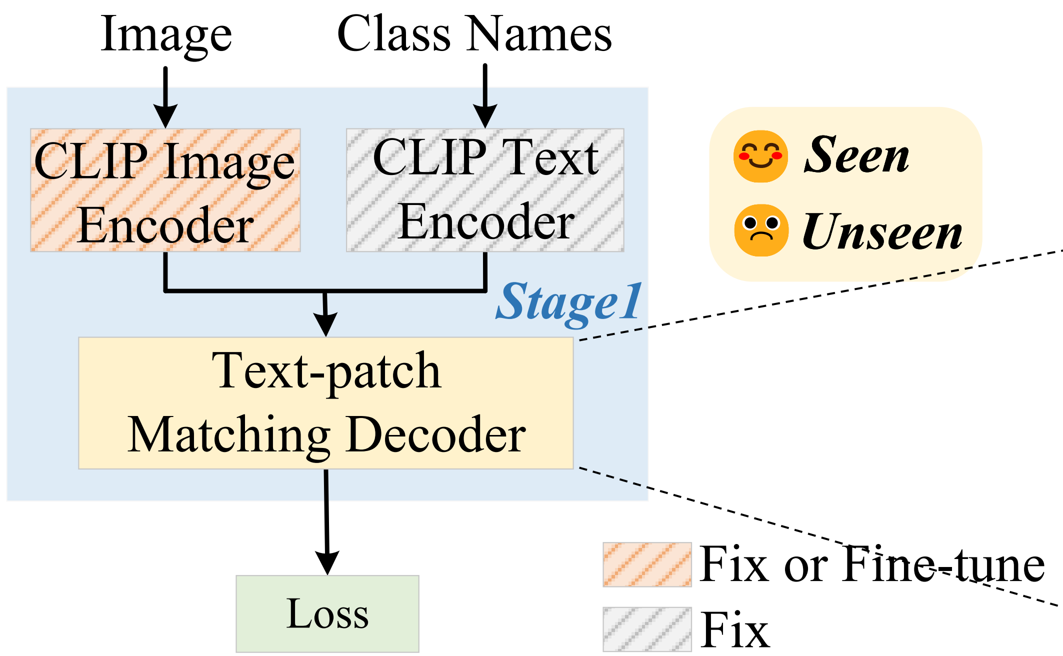
😳 **Disadvantage:**
Increase computational cost.

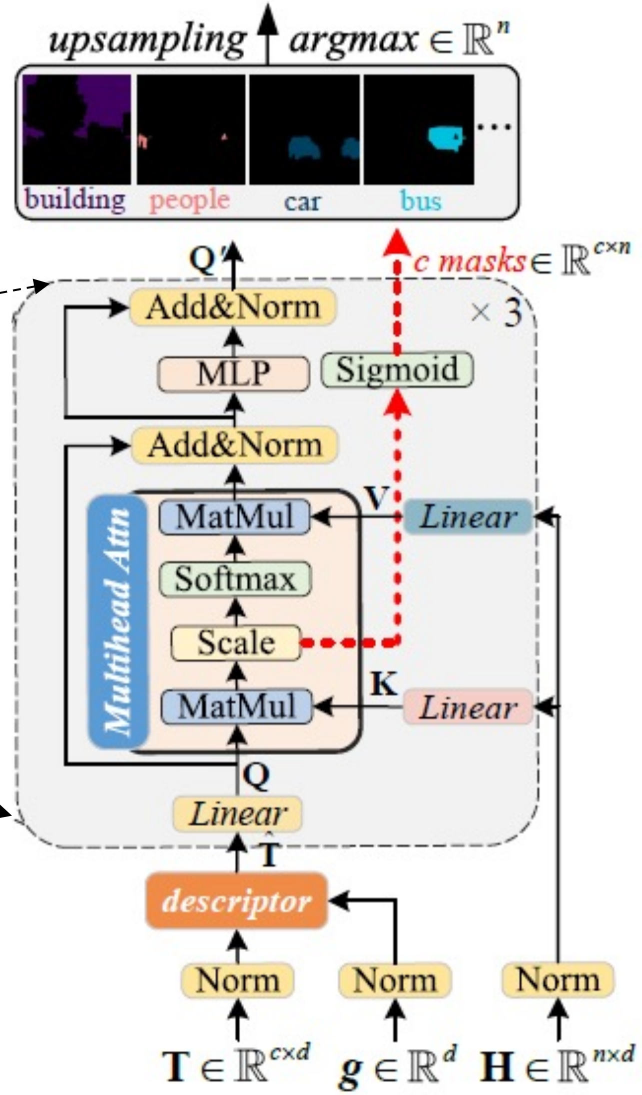
CLIP is still utilized for **image-level classification.**

🧐 How about directly extending CLIP for zero-shot dense prediction?

Motivation

Can we directly extend CLIP for zero-shot semantic segmentation?





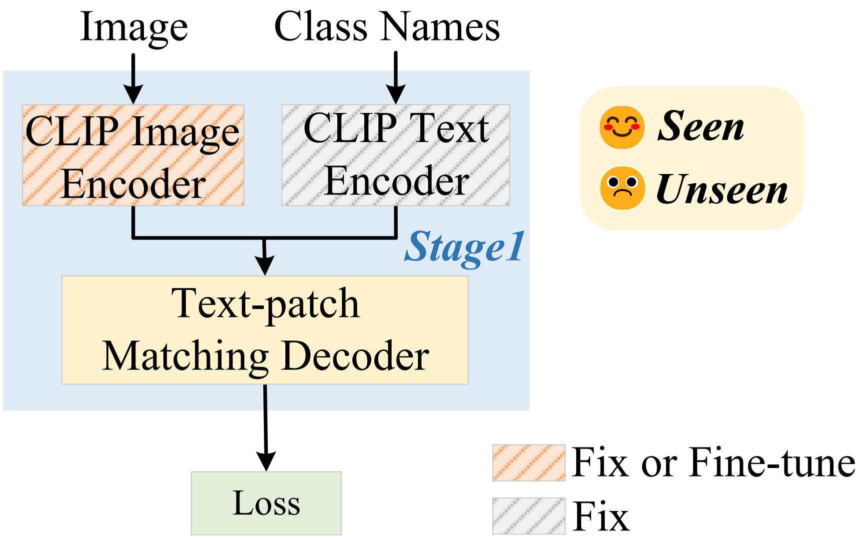
$$\text{Masks} = \frac{QK^T}{\sqrt{d_k}} \in \mathbb{R}^{c\times n},$$

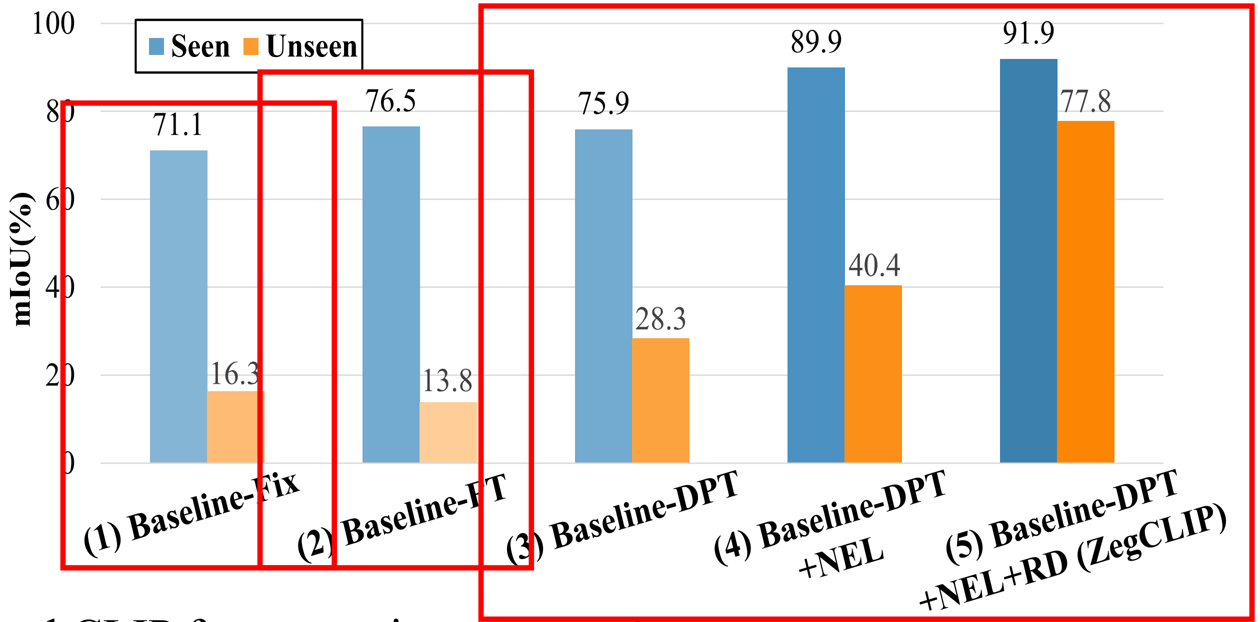
***One-stage Pipeline*:**

- Obtain text and patch embeddings;
- Generate semantic predictions by matching them in the decoder.

Motivation

Can we directly extend CLIP for zero-shot semantic segmentation?





Observation & Challenges:

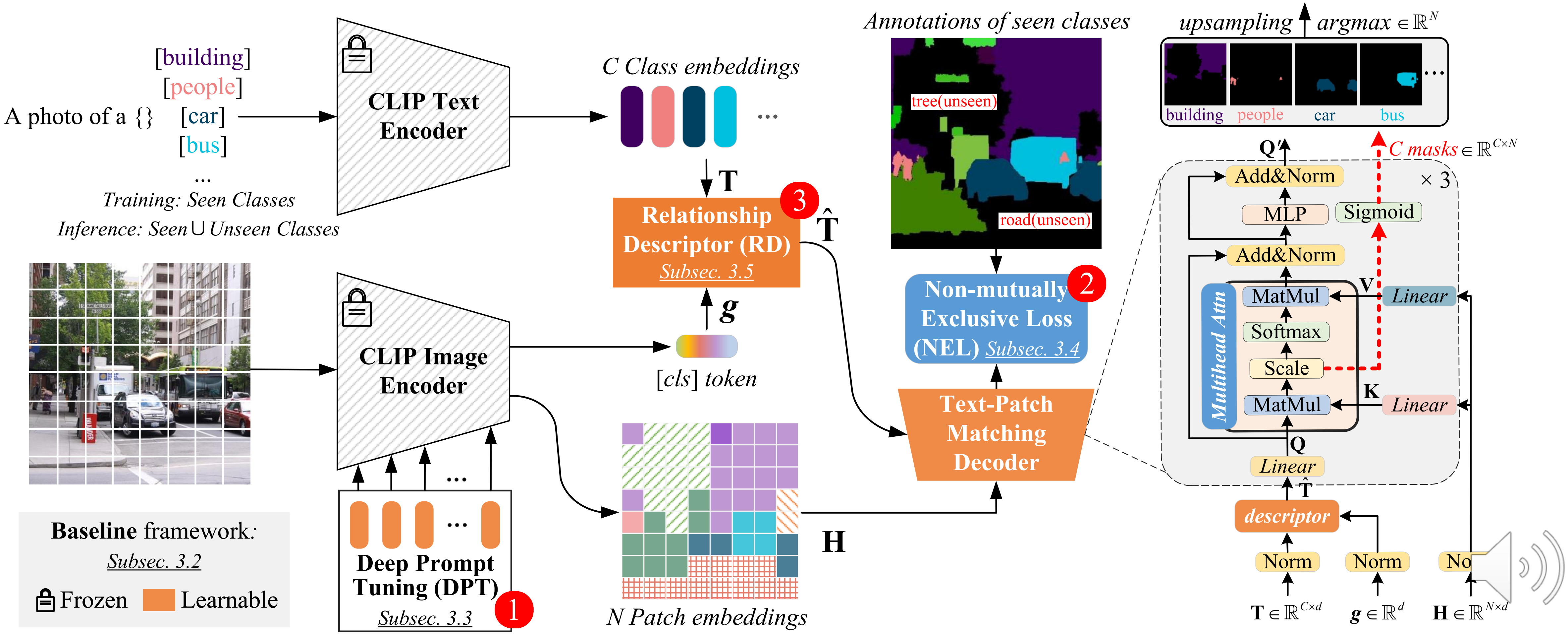
! Using the original CLIP for semantic segmentation:
Insufficient visual presentation due to CLIP is only pretrained on image-level;

! Finetuning CLIP image encoder on base dataset:
Better performance on seen classes but leads to overbias problem in zero-shot;

? How to extend CLIP into zero-shot segmentation?
We propose **efficient designs** to adapt CLIP's ability from image to pixel-level!

Method

- ***Design 1:*** **Deep Prompt Tuning (DPT)** instead of fine-tuning or fixing for the CLIP image encode.
- ***Design 2:*** Applying **Non-mutually Exclusive Loss (NEL)** instead of Mutually Exclusive Loss.
- ***Design 3:*** Introducing **Relationship Descriptor (RD)** to incorporate the image-level prior into text embedding before matching text-patch embeddings from CLIP in decoder:



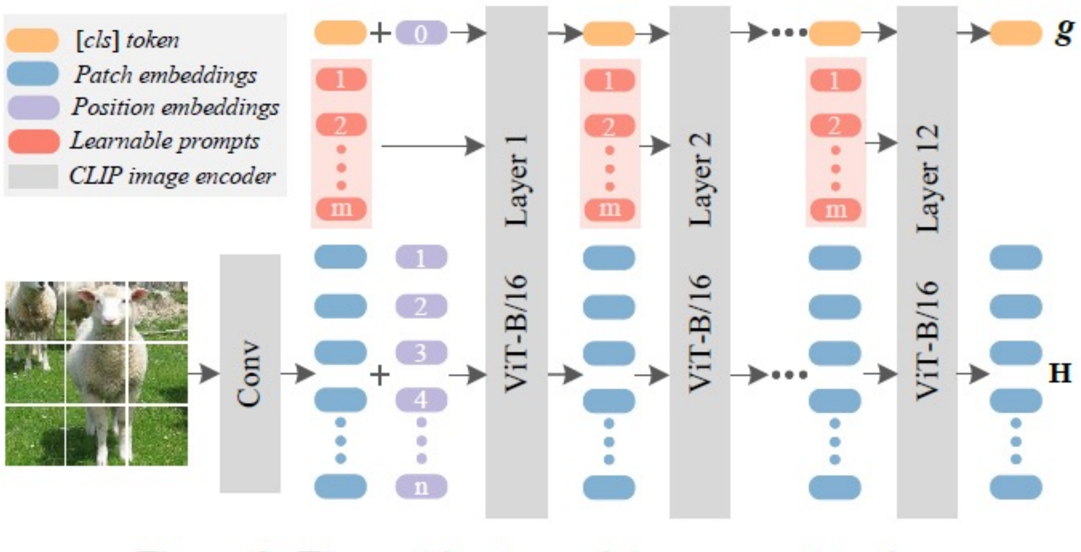
Method

➢ ***Design 1:*** **Deep Prompt Tuning (DPT)**

Fixing or Fine-tuning

V.S.

Deep Prompt Tuning



$$[\mathbf{g}^l, \_, \mathbf{H}^l] = \mathtt{Layer}^l([\mathbf{g}^{l-1}, \mathbf{P}^{l-1}, \mathbf{H}^{l-1}]) \quad (3)$$

➢ ***Design 2*: Non-mutually Exclusive Loss (NEL)**

CrossEntropy(*Softmax*(.))

V.S.

BinaryCrossEntropy(*Sigmoid*(.))

$$\mathcal{L}_{\mathtt{focal}} = -\frac{1}{\mathrm{hw}}\sum_{i=1}^{\mathrm{hw}}(1-y_i)^{\gamma}\times\hat{y}\log(y_i)+y_i^{\gamma}\times(1-\hat{y}_i)\log(1-y_i), \quad (4)$$

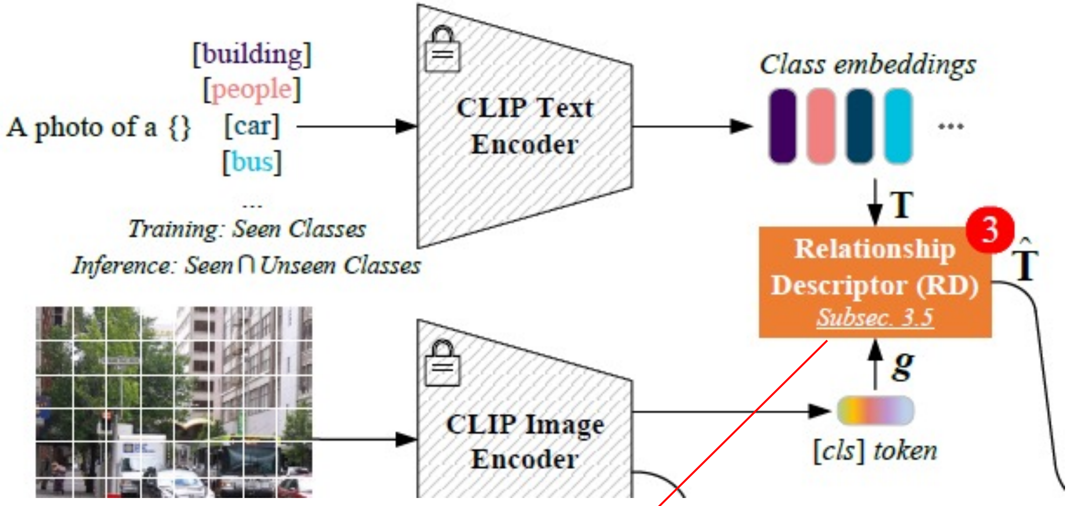
$$\mathcal{L}_{\mathtt{dice}} = 1 - \frac{2\sum_{i=1}^{\mathrm{hw}} y_i\hat{y}_i}{\sum_{i=1}^{\mathrm{hw}} y_i^2 + \sum_{i=1}^{\mathrm{hw}} \hat{y}_i^2}, \quad (5)$$

$$\mathcal{L} = \alpha\cdot\mathcal{L}_{\mathtt{focal}} + \beta\cdot\mathcal{L}_{\mathtt{dice}}, \quad (6)$$

where $\gamma = 2$ balances hard and easy samples and $\{\alpha, \beta\}$ are coefficients to combine focal loss and dice loss.

Method

➢ ***Design 3:*** **Relationship Descriptor**



$$\hat{\mathbf{t}} = concat[\mathbf{r}, \mathbf{t}] = concat[\mathbf{t} \odot \mathbf{g}, \mathbf{t}], \quad (8)$$

Table 4. Effect of different formats of text queries $\hat{\mathbf{t}}$.

dim	format of $\hat{\mathbf{t}}$	pAcc	mIoU(S)	mIoU(U)	hIoU
512	t	86.8	89.5	33.7	49.0
	t⊙g	93.1	90.2	68.4	77.8
	|t-g|	92.4	90.6	64.2	75.1
	t-g	88.7	87.9	46.5	60.8
	t+g	82.2	89.9	13.9	24.1
512*2	[t, g]	88.9	88.8	39.3	54.5
	[t⊙g, t]	**94.6**	**91.9**	**77.8**	**84.3**
	[|t-g|, t]	90.9	91.5	54.2	68.1
	[t⊙g, t+g]	88.3	90.0	38.0	53.4
	[t+g, t]	82.8	89.4	20.7	33.6
	[t⊙g, |t-g|]	94.1	91.2	73.9	81.6
512*3	[t⊙g, |t-g|, t]	93.4	91.6	67.3	77.6

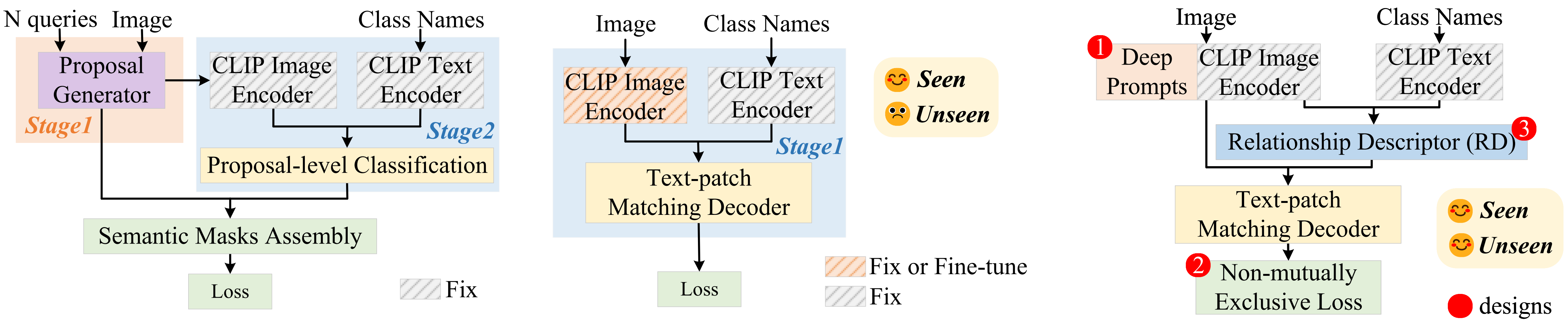
✔: dot product and absolute difference

✘: sum and concatenate operation

Method

Differences between our approach and related zero-shot methods based on CLIP.

Methods	Stages	Need extra image encoder?	CLIP as classifier?	Can do inductive?
SimBase		✓	✓	✓
ZegFormer	two	✓	✓	✓
MaskCLIP+		✓	✗	✗
ZegCLIP	**one**	✗	✗	✓



(a) Related **Two-Stage** methods.   (b) Our **Baseline**: **One-Stage** method.   (c) **ZegCLIP:** Baseline with designs.

Experiments

Benchmarks:

- **PASCAL VOC 2012** contains 10,582 augmented images for training and 1,449 for validation. We also ignore the ``background" category and use 15 classes as the seen part and 5 classes as the unseen part.

- **COCO-Stuff 164K** is a large-scale dataset that contains 171 categories with 118,287 images for training and 5,000 for testing. The whole dataset is divided into 156 seen classes and 15 unseen classes.

- **PASCAL Context** includes 60 classes with 4,996 for training and 5,104 for testing. The dataset is divided into 50 known classes (including ``background") and the rest 10 classes as used as unseen classes in the test set.

Seen classes C^S Unseen classes C^U $C^S \cap C^U = \emptyset$

Training:

- *Inductive*: name of unseen classes are unavailable

  Training images and ground truth of seen classes C^S

- *Transductive*: name of unseen classes are available

  Training images, ground truth of seen classes C^S and name of unseen classes C^U

Inference:

Per-pixel classicization on $C^S \cup C^U$

Evaluation Metrics:

- **pAcc, mIoU** on both seen and unseen classes
- **hIoU** among seen and unseen classes

$$hIoU = \frac{2 * mIoU(S) * mIoU(U)}{mIoU(S) + mIoU(U)}.$$

Experiments

Comparison with the state-of-the-art methods on three public benchmark datasets:

Table 2. Comparison with the state-of-the-art methods on PASCAL VOC 2012, COCO-Stuff 164K, and PASCAL Context datasets. "ST" represents applying self-training via generating pseudo labels on all unlabeled pixels, while "†"+"ST" denotes that pseudo labels are merely annotated on unseen pixels excluding the ignore part.

Methods	PASCAL VOC 2012				COCO-Stuff 164K				PASCAL Context			
	pAcc	**mIoU(S)**	**mIoU(U)**	**hIoU**	**pAcc**	**mIoU(S)**	**mIoU(U)**	**hIoU**	**pAcc**	**mIoU(S)**	**mIoU(U)**	**hIoU**
Inductive												
SPNet [44]	-	78.0	15.6	26.1	-	35.2	8.7	14.0	-	-	-	-
ZS3 [3]	-	77.3	17.7	28.7	-	34.7	9.5	15.0	52.8	20.8	12.7	15.8
CaGNet [17]	80.7	78.4	26.6	39.7	56.6	33.5	12.2	18.2	-	24.1	18.5	21.2
SIGN [10]	-	75.4	28.9	41.7	-	32.3	15.5	20.9	-	-	-	-
Joint [1]	-	77.7	32.5	45.9	-	-	-	-	-	33.0	14.9	20.5
ZegFormer [12]	-	86.4	63.6	73.3	-	36.6	33.2	34.8	-	-	-	-
zsseg [49]	90.0	83.5	72.5	77.5	60.3	39.3	36.3	37.8	-	-	-	-
ZegCLIP (Ours)	**94.6**	**91.9**	**77.8**	**84.3**	**62.0**	**40.2**	**41.4**	**40.8**	**76.2**	**46.0**	**54.6**	**49.9**
Transductive												
SPNet+ST [44]	-	77.8	25.8	38.8	-	34.6	26.9	30.3	-	-	-	-
ZS5 [3]	-	78.0	21.2	33.3	-	34.9	10.6	16.2	49.5	27.0	20.7	23.4
CaGNet+ST [17]	81.6	78.6	30.3	43.7	56.8	35.6	13.4	19.5	-	-	-	-
STRICT [34]	-	82.7	35.6	49.8	-	35.3	30.3	34.8	-	-	-	-
zsseg+ST [49]	88.7	79.2	78.1	79.3	63.8	39.6	43.6	41.5	-	-	-	-
ZegCLIP+ST (Ours)	**95.1**	**91.8**	**82.2**	**86.7**	**68.8**	**40.6**	**54.8**	**46.6**	**77.2**	**46.6**	**65.4**	**54.4**
†MaskCLIP+ [56]	-	88.8	86.1	87.4	-	38.1	54.7	45.0	-	44.4	66.7	53.3
†ZegCLIP+ST (Ours)	**96.2**	**92.3**	**89.9**	**91.1**	**69.2**	**40.7**	**59.9**	**48.5**	**77.3**	**46.8**	**68.5**	**55.6**
Fully Supervised												
ZegCLIP (Ours)	**96.3**	**92.4**	**90.9**	**91.6**	**69.9**	**40.7**	**63.2**	**49.6**	**77.5**	**46.5**	**78.7**	**56.9**

Experiments

Qualitative results on COCO-Stuff 164K:

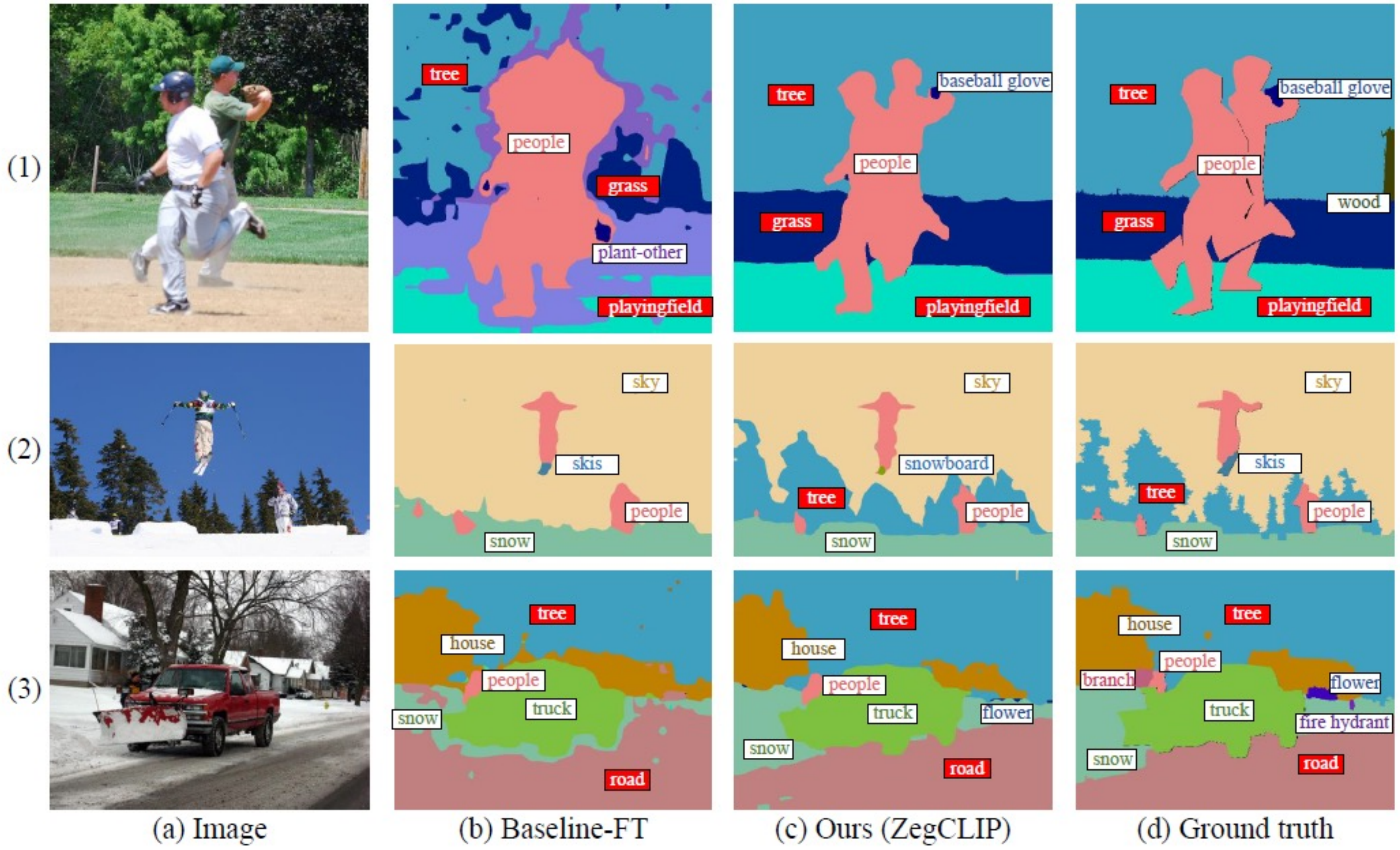


Figure 4. Qualitative results on COCO-Stuff 164K. (a) are the original testing images; (b) represent the performance of our proposed one-stage baseline (fine-tuning the image encoder); (c) are the visualization results of our proposed ZegCLIP; (d) are the ground truths of each image. Note that the white and red tags represent seen and unseen classes separately.

Ablation Study

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Effectiveness of our proposed designs:

Table 5. Quantitative results on VOC and COCO dataset to demonstrate the effectiveness of our proposed three designs.

method	PASCAL VOC 2012				COCO-Stuff 164K			
	pAcc	mIoU(S)	mIoU(U)	hIoU	pAcc	mIoU(S)	mIoU(U)	hIoU
Baseline-Fix	69.3	71.1	16.3	26.5	33.3	17.1	15.4	16.2
Baseline-Fix + NEL	85.5	85.2	36.6	51.2	52.4	31.7	20.8	25.1
Baseline-Fix + RD	86.0	82.5	46.6	59.6	41.0	23.3	23.4	23.3
Baseline-Fix + NEL + RD	89.6	83.3	66.4	73.9	53.7	32.3	32.5	32.4
Baseline-FT	77.3	76.5	13.8	23.4	48.4	32.4	17.5	22.7
Baseline-FT + NEL	83.8	84.1	27.5	41.4	56.5	39.9	25.4	31.0
Baseline-FT + RD	79.4	77.8	20.7	32.7	54.0	39.6	22.4	28.6
Baseline-FT + NEL + RD	89.6	90.2	42.4	57.7	60.2	**42.7**	22.3	29.3
Baseline-DPT	76.2	75.9	28.3	41.2	39.0	22.5	17.5	19.7
Baseline-DPT + NEL	89.2	89.9	40.4	55.7	58.5	38.0	27.4	31.8
Baseline-DPT + RD	85.5	81.0	55.2	65.7	46.4	28.4	27.8	28.1
Baseline-DPT + NEL + RD (ZegCLIP)	**94.6**	**91.9**	**77.8**	**84.3**	**62.0**	40.2	**41.4**	**40.8**

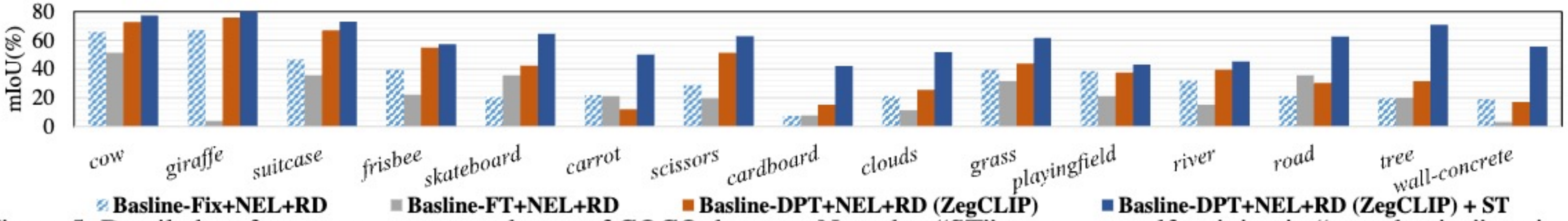


Figure 5. Detailed performance on unseen classes of COCO datasets. Note that "ST" represents self-training in "transductive" setting.

Efficiency comparison:

Table 3. Efficiency comparison with different metrics. All models are evaluated on a single 1080Ti GPU. #Params represents the number of learnable parameters in the whole framework.

Datasets	Methods	#Params(M) ↓	Flops(G) ↓	FPS ↑
VOC	ZegFormer [12]	60.3	1829.3	1.7
	ZegCLIP	**13.8**	**110.4**	**9.0**
COCO	ZegFormer [12]	60.3	1875.1	1.5
	ZegCLIP	**14.6**	**123.9**	**6.7**

Generalization ability:

Table 7. Generalization ability to other datasets.

source	target	method	pAcc	mIoU	mAcc
COCO	Context	Zegformer [12]	56.8	36.1	64.0
		ZegCLIP	60.9	41.2	68.4
		†ZegCLIP+ST	**68.4**	**45.8**	**70.9**
	VOC	Zegformer [12]	92.8	85.6	92.7
		ZegCLIP	96.9	93.6	96.4
		†ZegCLIP+ST	**97.2**	**94.1**	**96.7**

Ablation Study

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Effect of using advanced loss function:

Table 6. Comparison of introducing advanced loss function. Note that "plain" represents merely Binary Cross Entropy (BCE), while "plus" means adding focal loss on BCE and dice loss

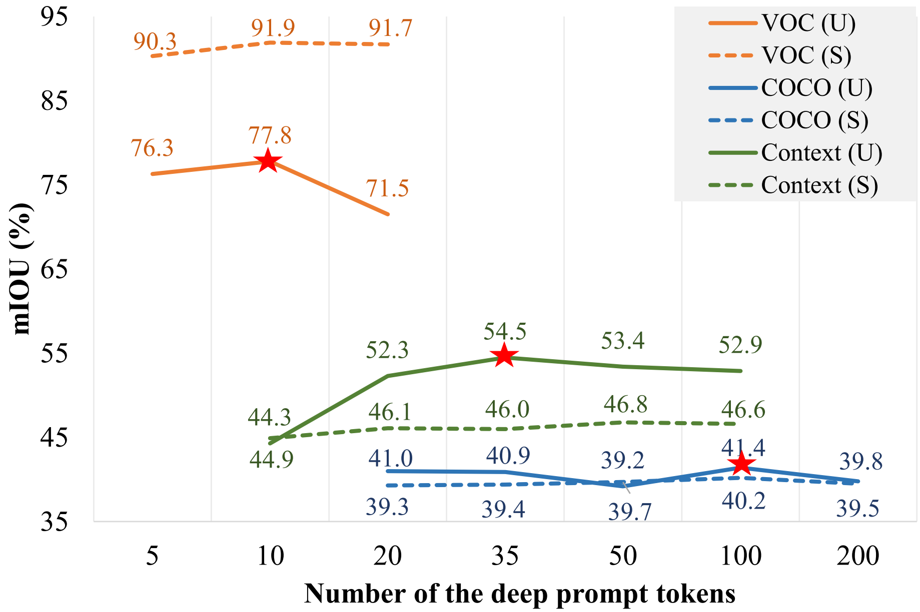
dataset	loss	pAcc	mIoU(S)	mIoU(U)	hIoU
VOC	plain	93.4	89.7	73.6	80.9
	plus	**94.6**	**91.9**	**77.8**	**84.3**
COCO	plain	59.8	38.8	39.0	38.9
	plus	**62.0**	**40.2**	**41.4**	**40.8**
Context	plain	75.3	43.5	50.0	46.5
	plus	**76.2**	**46.0**	**54.6**	**49.9**

Effect of single and multiple text templates:

Table 9. Comparison of using single and multiple templates on COCO-Stuff 164K and PASCAL Context datasets.

dataset	template	pAcc	mIoU(S)	mIoU(U)	hIoU
COCO	single	61.4	39.5	40.6	40.0
	multiple	**62.0**	**40.2**	**41.4**	**40.8**
Context	single	75.8	45.1	52.1	48.3
	multiple	**76.2**	**46.0**	**54.6**	**49.9**

Effect of number of deep prompt tokens:



Effect of depth of deep prompt tokens:

Table 8. Effect of the depth of deep prompt tuning on VOC.

layer	pAcc	mIoU(S)	mIoU(U)	hIoU
1	91.4	87.5	67.8	76.4
1→3	91.7	86.7	70.2	77.6
1→6	92.7	87.8	75.3	81.1
1→9	93.3	88.9	72.4	79.8
1→12	**94.6**	**91.9**	**77.8**	**84.3**
10→12	92.5	88.3	70.9	78.6
7→12	92.5	89.0	68.0	77.1
4→12	93.6	91.5	66.9	77.3

Visualization Results

$$\texttt{Masks} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \in \mathbb{R}^{c \times n}, \qquad (2)$$

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Conclusion

- ✓ Successfully extending CLIP into zero-shot semantic segmentation with **one-stage** straight-forward paradigm.
- ✓ Three **simple-but-effective designs** to achieve competitive results on seen classes while extremely improving performance on novel classes.
- ✓ Flexible text queries to handle both “**inductive**” and “**transductive**” settings.
- ✓ **5 times faster inference** compared with two-stage methods.

Thank You

Paper

Github