Decoupling Zero-Shot Semantic Segmentation

Jian Ding\textsuperscript{1,2}, Nan Xue\textsuperscript{1}, Gui-Song Xia\textsuperscript{1}, Dengxin Dai\textsuperscript{2}
\textsuperscript{1}CAPTAIN, Wuhan University, China
\textsuperscript{2}MPI for Informatics, Germany
\{jian.ding, xuenan, guisong.xia\}@whu.edu.cn, ddai@mpi-inf.mpg.de

Abstract

Zero-shot semantic segmentation (ZS3) aims to segment the novel categories that have not been seen in the training. Existing works formulate ZS3 as a pixel-level zero-shot classification problem, and transfer semantic knowledge from seen classes to unseen ones with the help of language models pre-trained only with texts. While simple, the pixel-level ZS3 formulation shows the limited capability to integrate vision-language models that are often pre-trained with image-text pairs and currently demonstrate great potential for vision tasks.

Inspired by the observation that humans often perform segment-level semantic labeling, we propose to decouple the ZS3 into two sub-tasks: 1) a class-agnostic grouping task to group the pixels into segments. 2) a zero-shot classification task on segments. The former sub-task does not involve category information and can be directly transferred to group pixels for unseen classes. The latter sub-task performs at segment-level and provides a natural way to leverage large-scale vision-language models pre-trained with image-text pairs (e.g., CLIP) for ZS3. Based on the decoupling formulation, we propose a simple and effective zero-shot semantic segmentation model, called ZegFormer, which outperforms the previous methods on ZS3 standard benchmarks by large margins, e.g., \textbf{35 points} on the PASCAL VOC and \textbf{3 points} on the COCO-Stuff in terms of mIoU for unseen classes. Code will be released at \url{https://github.com/dingjiansw101/ZegFormer}.

1. Introduction

Semantic segmentation targets to group an image into segments with semantic categories. Although remarkable progress has been made \cite{10, 11, 36, 56, 57, 62}, current semantic segmentation models are mostly trained in a supervised manner with a fixed set of predetermined semantic categories, and often require hundreds of samples for each class. In contrast, humans can distinguish at least 30,000 basic categories \cite{6, 18}, and recognize novel categories merely from some \textit{high-level descriptions}. How to achieve human-level ability to recognize stuff and things in images is one of the ultimate goals in computer vision.

Recent investigations on zero-shot semantic segmentation (ZS3) \cite{7, 54} have actually moved towards that ultimate goal. Following the fully supervised semantic segmentation models \cite{10, 11, 36} and zero-shot classification models \cite{1, 24, 30, 47, 60}, these works formulate zero-shot semantic segmentation as a \textit{pixel-level zero-shot classification problem}. Although these studies have reported promising results, two main issues still need to be addressed:
They usually transfer knowledge from seen to unseen classes by language models [7, 27, 54] pre-trained only with texts, which limit their performance on vision tasks. Although large-scale pre-trained vision-language models (e.g., CLIP [46] and ALIGN [26]) have demonstrated potentials on image-level vision tasks, how to efficiently integrate them into the pixel-level ZS3 problem is still unknown.

(2) They usually build correlations between pixel-level visual features and semantic features for knowledge transfer, which is not natural since we humans often use words or texts to describe objects/segments instead of pixels in images. As illustrated in Fig. 1, it is unsurprising to observe that the pixel-level classification has poor accuracy on unseen classes, which in turn degrades the final segmentation quality. This phenomenon is particularly obvious when the number of unseen categories is large (see Fig. 6).

An intuitive observation is that, given an image for semantic segmentation, we humans can first group pixels into segments and then perform a segment-level semantic labeling process. For example, a child can easily group the pixels of an object, even though he/she does not know the name of the object. Therefore, we argue that a human-like zero-shot semantic segmentation procedure should be decoupled into two sub-tasks:

- A class-agnostic grouping to group pixels into segments. This task is actually a classical image partition/grouping problem [44, 50, 52], and can be renewed via deep learning based methods [12, 32, 59].
- A segment-level zero-shot classification to assign semantic labels either seen or unseen to segments.

As the grouping task does not involve the semantic categories, a grouping model learned from seen classes can be easily transferred to unseen classes. The segment-level zero-shot classification is robust on the unseen classes and provides a flexible way to integrate the pre-trained large-scale vision-language models [46] to the ZS3 problem. To instantiate the decoupling idea, we present a simple yet efficient zero-shot semantic segmentation model with transformer, named ZegFormer, which uses a transformer decoder to output segment-level embeddings, as shown in Fig. 2. It is then followed by a mask projection for class-agnostic grouping (CAG) and a semantic projection for segment-level zero-shot classification (s-ZSC). The mask projection maps each segment-level embedding to a mask embedding, which can be used to obtain a binary mask prediction via a dot product with a high-resolution feature map. The semantic projection establishes the correspondences between segment-level embedding and semantic features of a pre-trained text encoder for s-ZSC.

While the steps mentioned above can form a standalone approach for ZS3, the model trained on a small dataset is struggling to have strong generalization. Thanks to the decoupling formulation, it is also flexible to use an image encoder of a vision-language model to generate image embeddings for zero-shot segment classification. As we empirically find that the segment classification scores with image embeddings and s-ZSC are complementary. We fuse them to achieve the final classification scores for segments.

The proposed ZegFormer model has been extensively evaluated with experiments and demonstrated superiority on various commonly-used benchmarks for ZS3. It outperforms the state-of-the-art methods by 35 points in terms of mIoU for unseen classes on the PASCAL VOC [15], and 3 points on the COCO-Stuff [8]. Based on the challenging ADE20k-Full dataset [63], we also create a new ZS3 benchmark with 275 unseen classes, the number of unseen classes in which are much larger than those in PASCAL-VOC (5 unseen classes) and COCO-Stuff (15 unseen classes). On the ADE20k-Full ZS3 benchmark, our performance is comparable to MaskFormer [12], a fully supervised semantic segmentation model.

Our contributions in this paper are three-fold:

- We propose a new formulation for the task of ZS3, by decoupling it into two sub-tasks, a class-agnostic grouping and a segment-level zero-shot classification. The class-agnostic grouping has a strong generalization to unseen classes, while the segment-level zero-shot classification provides a more natural and flexible way to integrate the pre-trained large-scale vision-language models into ZS3.
- With the new formulation, we present a simple and effective ZegFormer model for ZS3, which uses a transformer decoder to generate segment-level embeddings for grouping and zero-shot classification. To the best of our knowledge, the proposed ZegFormer is the first model taking full advantage of the pre-trained large-scale vision-language model (e.g., CLIP [46]) for ZS3.
- We achieved state-of-the-art results on standard benchmarks for ZS3. The ablation and visualization analyses show that the decoupling formulation is superior to pixel-level zero-shot classification by a large margin.

2. Related Works

Zero-Shot Image Classification aims to classify images of unseen categories that have not been seen during training. The key idea in zero-shot learning is to transfer knowledge from seen classes to unseen classes via semantic representation, such as the semantic attributes [2, 24, 28, 30], concept ontology [16, 37, 39, 47, 48] and semantic word vectors [17, 41, 60]. Recently, there are some works that use large-scale vision-language pretraining [26, 46] via contrastive loss. For example, by training a vision-language model on 400 million image and text pairs collected by Google Engine without any human annotations, CLIP [46] has achieved impressive performances on more than 30 vision datasets even compared to the supervised models. It
has also shown the potential for zero-shot object detection [19]. However, there is a large gap between the pixel level features used in previous ZS3 models and image-level features used in CLIP. To bridge this gap, we build the correspondence between segment-level visual features and image-level vision-language features.

Zero-Shot Semantic Segmentation is a relatively new research topic [7, 25, 42, 54, 61], which aims to segment the classes that have not been seen during training. There have been two streams of work: discriminative methods [5, 42, 54] and generative methods [7, 20, 49]. SPNet [54] and ZS3Net [7] are considered the representative examples. In detail, SPNet [54] maps each pixel to a semantic word embedding space and projects each pixel feature into class probability via a fixed semantic word embedding [27, 38] projection matrix. ZS3Net [7] first train a generative model to generate pixel-wise features of unseen classes by word embeddings. With the synthetic features, the model can be trained in a supervised manner. Both of these two works formulate ZS3 as a pixel-level zero-shot classification problem. However, this formulation is not robust for ZS3, since the text embeddings are usually used to describe objects/segments instead of pixels. The later works [20, 21, 29, 31, 49, 51] all follow this formulation to address different issues in ZS3. Among them, SIGN [5] addressed the importance of learning a joint vision-language embedding space for ZS3. Our work also addresses the issue but solves it by using a large-scale pre-trained vision-language model. In a weaker assumption that the unlabelled pixels from unseen classes are available in the training images, self-training [7, 42] is widely used. Although promising performances are reported, self-training often needs to retrain a model whenever a new class appears. Different from the pixel-level zero-shot classification, we propose a new formulation for ZS3, by decoupling ZS3 into a class-agnostic learning problem on pixels and a segment-level zero-shot learning problem. Based on the formulation, we implement a ZegFormer for the ZS3 task, which does not have a complicated training scheme and is flexible to transfer to new classes without retraining.

Class-Agnostic Segmentation is a long-standing problem and has been extensively studied in computer vision [3, 4, 14, 44, 50, 52, 64]. There are evidences [22, 23, 43] that class-agnostic segmentation model learned from seen classes can be well transferred to unseen classes in the task of instance segmentation, under a partially supervised training paradigm. Recently, a class-agnostic segmentation task [45] called entity segmentation (ES) is proposed, which can predict segments for both thing and stuff classes. However, ES is an instance-aware task, which is different from semantic segmentation. In addition, entity segmentation [45] does not predict the detailed class names of unseen classes. Our work is inspired by the aforementioned findings, but we focus on the semantic segmentation of novel classes and also predict the detailed class names of unseen classes. With the formulation, our proposed method is simpler, flexible, and robust.

3. Methodology

3.1. Decoupling Formulation of ZS3

Given an image $I$ on the domain $\Omega = \{0, \ldots, H - 1\} \times \{0, \ldots, W - 1\}$, the semantic segmentation of $I$ can be defined as a process to find a pair of mappings $(R, L)$ for $I$, where $R$ groups the image domain into $N$ piece-wise “homogeneous” segments $\{R_i\}_{i=1}^N$, such that $\cup_{i=1}^N R_i = \Omega$ and $R_i \cap R_j = \emptyset$, if $i \neq j$, and $L$ associates every segment $R_i \in \Omega$ with a semantic label $c \in C$, where $C$ is a predefined set of categories.

A fully-supervised semantic segmentation suggests that, one can learn such a pair of mappings $(R, L)$ for $I$ from a large-scale semantic annotated dataset, i.e., $D = \{I_k, R_k, L_k\}_{k=1}^K$. This type of methods are often with an assumption that the category set $C$ are closed, i.e., the categories appearing in testing images are well contained by $C$, which, however are usually violated in real-application scenarios.

Actually, if we denote $S$ as the category set of the annotated dataset $D$, i.e., the seen classes, and $E$ as those appearing in the testing process, we have three types of settings for semantic segmentation,

- fully-supervised semantic segmentation: $E \subseteq S$,
- zero-shot semantic segmentation (ZS3): $S \cap E = \emptyset$,
- generalized ZS3 (GZS3) problem: $S \subset E$.

In this paper, we mainly address the problem with the GZS3 setting, and denote $U = E - E \cap S$ as the set of unseen classes.

Relations to Pixel-Level Zero-Shot Classification. Previous works [7, 20, 54] formulate ZS3 as a pixel-level zero-shot classification problem, where a model learned from pixel-level semantic labels of $S$ needs to be generalized to pixels of $U$. It can be considered as a special case of our decoupled formulation, where each pixel represents a segment $R_i$. Since the learning of $\mathcal{R}$ does not involve the semantic categories, our formulation separates a class-agnostic learning sub-task from ZS3. The class-agnostic task has a strong generalization to unseen categories, as demonstrated in [43, 45]. In addition, as humans often associate semantics to whole images or at least segments, establishing a connection from semantics to segment-level visual features is more natural than that to pixel-level visual features. Therefore, we argue that our formulation is more efficient than pixel-level zero-shot classification in transferring knowledge from $S$ to $U$. 


3.2. ZegFormer

Fig. 2 illustrates the pipeline of our proposed ZegFormer model. We first generate a set of segment-level embeddings and then project them for class-agnostic grouping and segment-level zero-shot classification by two parallel layers. A pre-trained image encoder is also used for segment classification. The two segment-level classification scores are finally fused to obtain the results.

Segment Embeddings. Recently, there are several segmentation models \cite{12, 32, 59} that can generate a set of segment-level embeddings. We choose the Maskformer \cite{12} as a basic semantic segmentation model for simplicity. By feeding \( N \) segment queries and a feature map to a transformer decoder, we can obtain \( N \) segment-level embeddings. Then we pass each segment embedding through a semantic projection layer and a mask projection layer to obtain a mask embedding and a semantic embedding for each segment. We denote segment-level semantic embedding (SSE) as \( G_q \in \mathbb{R}^d \) and the segment-level mask embedding as \( B_q \in \mathbb{R}^d \), where \( q \) indexes a query.

Class-Agnostic Grouping. Denote the feature maps out from pixel decoder as \( F(I) \in \mathbb{R}^{d \times H \times W} \). The binary mask prediction for each query can be calculated as \( m_q = \sigma(B_q \cdot F(I)) \in [0, 1]^{H \times W} \), where \( \sigma \) is the sigmoid function. Note that \( N \) is usually smaller than the number of classes.

Segment Classification with SSE. For training and inference, each “class name” in a set of class names \( C \) is put into a prompt template (e.g. “A photo of a [class name] in the scene”) and then fed to a text encoder. Then we can obtain \( |C| \) text embeddings, denoted as \( T = \{ T_c \in \mathbb{R}^d | c = 1, \ldots, |C| \} \), where \( C = S \) during training, while \( C = S \cup U \) during inference. In our pipeline, we also need a “no object” category, if the intersection over union (IoU) between a segment and any ground truths is low. For the “no object” category, it is unreasonable to be presented by a single class name. Therefore, we add an extra learnable embedding \( T_0 \in \mathbb{R}^d \) for “no object”. The predicted probability distribution over the seen classes and the “no object” for a segment query is calculated as:

\[
p_q(c) = \frac{\exp\left(\frac{1}{\tau} s_c(T_q, G_q)\right)}{\sum_{c=0}^{|C|} \exp\left(\frac{1}{\tau} s_c(T_q, G_q)\right)},
\]

where \( q \) indexes a query. \( s_c(e, e') = \frac{e \cdot e'}{|e| |e'|} \) is the cosine similarity between two embeddings. \( \tau \) is the temperature parameter.

Segment Classification with Image Embedding. While the aforementioned steps can already form a standalone approach for ZS3, it is also possible to use an image encoder of a pre-trained vision-language model (e.g. CLIP \cite{46}) to improve the classification accuracy on segments, owing to the flexibility of the decoupling formulation. In this module, we create a suitable sub-image for a segment. The process can be formulated as, given a mask prediction \( m_q \in [0, 1]^{H \times W} \) for a query \( q \), and the input image \( I \), create a sub-image \( I_q = f(m_q, I) \), where \( f \) is a preprocess function (e.g., a masked image or a cropped image according to \( m_q \)). Since the preprocess is very critical, we give detailed ablation studies in Sec. 4.5. We pass \( I_q \) to a pre-trained image encoder and obtain image embedding \( A_q \). Similar to Eq. 1, we can calculate a probability distribution, denoted as \( p_q(c) \).

Training. During the training of ZegFormer, only the pixel labels belonging to \( S \) are used. To compute the training loss, a bipartite matching \cite{9, 12} is performed between the predicted masks and the ground-truth masks. The loss of classification for each segment query is \(-\log p_q(c^q)\), where \( c^q \) belongs to \( S \) if the segment is matched with a ground truth mask and “no object” if the segment does not have a matched ground truth segment. For a segment matched with ground truth segment \( R^q \), there is a mask loss \( L_{mask}(m_q, R^q) \). In detail, we use the combination of a dice loss \cite{40} and a focal loss \cite{34}.

Inference. During inference, we integrate the predicted binary masks and class scores of segments to obtain the final results of semantic segmentation. According to the class probability scores, we have three variants of ZegFormer.

(1) ZegFormer-seg. This variant use the segment classification scores with segment queries for inference by calculate a per-pixel class probability \( \sum_{q=1}^N p_q(c) \cdot m_q[h, w] \), where \( (h, w) \) is a location in an image. Since there is an imbalanced data problem, which results in predictions being biased to seen classes. Following \cite{54}, we calibrate the prediction by decreasing the scores of seen classes. The final category prediction for each pixel is then calculated as:

\[
\arg \max_{c \in S+U} \sum_{q=1}^N p_q(c) \cdot m_q[h, w] - \gamma \cdot \mathbb{1}[c \in S],
\]

where \( \gamma \in [0, 1] \) is the calibration factor. The indicator function \( \mathbb{1} \) is equal to 1 when \( c \) belongs to the seen classes.

(2) ZegFormer-img. The inference process of this variant is similar to Eq. 2. The only difference is that the \( p_q(c) \) is replaced by \( p_q'(c) \).

(3) ZegFormer. This variant is our full model. We first fuse \( p_q(c) \) and \( p_q'(c) \) for each query as:

\[
p_{\text{fusion}}(c) = \begin{cases} p_q(c) \lambda \cdot p_q'(c) \lambda & \text{if } c \in S \\ p_q(c) \lambda \cdot p_q'(c) \lambda & \text{if } c \in U, \end{cases}
\]
balanced by $\lambda$. Since $p_q(c)$ is usually more accurate than $p_q'(c)$ if $c \in S$, we do not want $p_q'(c)$ contribute to the prediction of $S$. Therefore, we calculate a geometry mean of $p_q(c)$ and $p_q, \text{avg} = \sum_{j \in S} p_q(j)/|S|$ on seen classes. This way, the probabilities of seen classes and unseen classes can be adjusted to the same range, and only $p_q(c)$ contributes to distinguishing seen classes. The final results for semantic segmentation are obtained by a process similar to Eq. 2.

4. Experiments

Since most of the previous works [7,20,42] focus on the GZS3 setting, we evaluate our method on the GZS3 setting in the main paper. See our results on the ZS3 setting in the appendix. We introduce the datasets and evaluation metrics that we use in the following.

4.1. Datasets and Evaluation Metrics

COCO-Stuff is a large-scale dataset for semantic segmentation that contains 171 valid classes in total. We use 118,287 training images as our training set and 5,000 validation images as our testing set. We follow the class split in [54] to choose 156 valid classes as seen classes and 15 valid classes as unseen classes. We also use a subset of the training classes as a validation set for tuning hyperparameters, following the cross-validation procedure [54, 55].

PASCAL-VOC Dataset has been split into 15 seen classes and 5 unseen classes in previous works [20, 54]. There are 10582 images for training and 1,449 images for testing.

ADE20k-Full Dataset is annotated in an open-vocabulary setting with more than 3,000 categories. It contains 25k images for training and 2k images for validation. We are the first that evaluate GZS3 methods on the challenging ADE20k-Full. Following the supervised setting in [12], we choose 847 classes that are present in both train and validation sets for evaluation, so that we can compare our ZegFormer with supervised models. We split the classes into seen and unseen according to their frequency. The seen classes are present in more than 10 images, while the unseen classes are present in less than 10 images. In this way, we obtain 572 classes for seen and 275 classes for unseen.

Class-Related Segmentation Metric includes the mean intersection-over-union (mIoU) averaged on seen classes, unseen classes, and their harmonic mean, which has been widely used in previous works of GZS3 [7, 54].

Class-Agnostic Grouping Metric has been well studied in [44]. We use the well-known precision-recall for boundaries ($P_b$, $R_b$, and $F_b$) as the evaluation metric, and use the public available code\footnote{https://github.com/jponttuset/seism} for evaluation.

4.2. Implementation Details

Our implementation is based on Detectron2 [53]. For most of our ablation experiments, we use ResNet50 as our backbone and FPN [33] as the pixel decoder. When compared with the state-of-the-art methods, we use the ResNet-101 as our backbone. We use the text encoder and image en-
The widely used ft + w2v in GZS3 [54] is trained only by concatenation of fastText (ft) [27] and word2vec (w2v) [38]. This setting has a more realistic value. For simplicity, we still call it GZS3.

4.3. Zero-Shot Pixel Classification Baseline

To compare decoupling formulation with pixel-level zero-shot classification, we choose SPNet [54] as our pixel-level zero-shot classification baseline, since SPNet and ZegFormer both belong to the discriminative methods with neat designs. For fair comparison, we reimplement the SPNet [54] with FPN (which is also used by ZegFormer). We denote this variant of SPNet as SPNet-FPN. We implement SPNet-FPN in the same codebase (i.e., Detectron2 [53]) as our ZegFormer. The common settings are also the same as the ones used in ZegFormer).

4.4. Comparisons with the Baseline

Class-Related Segmentation Metric. We compare ZegFormer-seg with the baseline under two types of text embeddings (i.e., CLIP [46] text embeddings and the concatenation of fastText [27] and word2vec [38] (w2v)). The widely used ft + w2v in GZS3 [54] is trained only by language data. In contrast, CLIP text encoder is trained by multimodal contrastive learning of vision and language. From Tab. 1, we can conclude: 1) CLIP text embeddings is better than the concatenation of fastText [27] and word2vec [38]; 2) Our proposed decoupling formulation for ZS3 is better than the commonly-used zero-shot pixel classification ones, regardless of what class embedding methods are used; 3) ZegFormer-seg has a much large gain than the SPNet-FPN, when the class embedding method is changed from the ft + w2v to CLIP (10.6 points v.s. 4.1 points improvements). We argue that the segment-level visual features are aligned better to the features of CLIP than the pixel-level visual features.

Class-Agnostic Grouping Metric. We compare the image grouping quality of ZegFormer-seg and the baseline on COCO-Stuff [8]. The image grouping quality is a class-agnostic metric, which can provide us more insight. We can see in Tab. 2 that ZegFormer significantly outperforms the baseline on the $F_b$, $P_b$, and $R_b$, regardless of what class embedding methods are used. This verifies that the decoupled ZS3 formulation has much stronger generalization than pixel-level zero-shot classification to group the pixels of unseen classes.
Table 4. Comparison with the previous GZS3 methods on PASCAL VOC and COCO-Stuff. The “Seen”, “Unseen”, and “Harmonic” denote mIoU of seen classes, unseen classes, and their harmonic mean. The STRICT [42] proposed a self-training strategy and applied it to SPNet [54]. The numbers of STRICT and SPNet (w/ self-training) are from [42]. Other numbers are from their original papers.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Type</th>
<th>w/ Self-train.</th>
<th>Re-train. for new classes?</th>
<th>PASCAL VOC</th>
<th>COCO-Stuff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Seen</td>
<td>Unseen</td>
</tr>
<tr>
<td>SPNet [54]</td>
<td>discriminative</td>
<td>×</td>
<td>×</td>
<td>78.0</td>
<td>15.6</td>
</tr>
<tr>
<td>ZS3 [7]</td>
<td>generative</td>
<td>×</td>
<td>✓</td>
<td>77.3</td>
<td>17.7</td>
</tr>
<tr>
<td>CaGNet [20]</td>
<td>generative</td>
<td>×</td>
<td>✓</td>
<td>78.4</td>
<td>26.6</td>
</tr>
<tr>
<td>SIGN [13]</td>
<td>generative</td>
<td>×</td>
<td>✓</td>
<td>75.4</td>
<td>28.9</td>
</tr>
<tr>
<td>Joint [3]</td>
<td>discriminative</td>
<td>×</td>
<td>✓</td>
<td>77.7</td>
<td>32.5</td>
</tr>
<tr>
<td>ZS3 [7]</td>
<td>generative</td>
<td>✓</td>
<td>✓</td>
<td>78.0</td>
<td>21.2</td>
</tr>
<tr>
<td>CaGNet [20]</td>
<td>generative</td>
<td>✓</td>
<td>✓</td>
<td>78.6</td>
<td>30.3</td>
</tr>
<tr>
<td>SIGN [13]</td>
<td>generative</td>
<td>✓</td>
<td>✓</td>
<td>83.5</td>
<td>41.3</td>
</tr>
<tr>
<td>SPNet [42]</td>
<td>discriminative</td>
<td>✓</td>
<td>✓</td>
<td>77.8</td>
<td>25.8</td>
</tr>
<tr>
<td>STRICT [42]</td>
<td>discriminative</td>
<td>✓</td>
<td>✓</td>
<td>82.7</td>
<td>35.6</td>
</tr>
<tr>
<td>ZegFormer</td>
<td>discriminative</td>
<td>×</td>
<td>×</td>
<td>81.3</td>
<td>76.8</td>
</tr>
</tbody>
</table>

4.5. Ablation Study on ZegFormer

Preprocess for Sub-Images. We explored three preprocess to obtain sub images (i.e. “crop”, “mask”, and “crop and mask”) in the full model ZegFormer. The three ways to preprocess a segment for “tree” are shown in Fig. 3. We can see that when we merely crop a region from the original image, there may exist more than one categories, which will decrease the classification accuracy. In the masked image, the nuisances are removed, but there are many unnecessary pixels outside the segment. The combination of crop and mask can obtain a relatively suitable image for classification. We compare the influences of three ways for the ZegFormer in Tab. 3. We can see that the combination of crop and mask can get the best performance, while only using the crop preprocess is lower than the performance of ZegFormer-seg, which does not use an image embedding for segment classification.

ZegFormer-seg vs. ZegFormer-img. We compare their performance for unseen categories on COCO-Stuff in Fig. 4. We can see that the ZegFormer-img is better at things categories (such as “giraffe”, “suitcase”, and “carrot”, etc), while worse at stuff categories (such as “grass”, “playing-field”, and “river”, etc). Therefore, these two kinds of classification scores are complementary, which illustrates why their fusion will improve performance.

4.6. Comparison with the State-of-the-art

We compare our ZegFormer with the previous methods in Tab. 4. Specifically, ZegFormer outperforms Joint [5] by 31 points in the mIoU of unseen classes on PASCAL VOC [15], and outperforms SIGN [13] by 18 points in the mIoU of unseen classes on COCO-Stuff [8]. When compared to the results with self-training, ZegFormer outperforms SIGN [13] by 35 points in mIoU of unseen classes on PASCAL VOC [15], and STRICT [42] by 3 points on COCO-Stuff [8]. It is worth noting that the generative methods and self-training methods need a complicated multi-stage training scheme. They also need to be re-trained whenever new classes are incoming (although semantic category labels of unseen classes are not required). The self-training also needs to access the unlabelled pixels of unseen classes to generate pseudo labels. In contrast, our ZegFormer is a discriminative method, which is much simpler than those methods and can be applied to any unseen classes on-the-fly. Similar to our ZegFormer, SPNet [54] (without self-training) and Joint [5] are also discriminative methods that can be flexibly applied to unseen classes, but their performances are much worse than ours.

4.7. Results on ADE20k-Full

Since we are the first that report the results on the challenging ADE20k-Full [63] for GZS3, there are no other methods for comparison. We compared ZegFormer with SPNet-FPN, and a fully supervised Maskformer trained on both seen and unseen classes. From Tab. 5, we can see that our ZegFormer significantly surpasses the SPNet-FPN baseline by 4 points in mIoU unseen and is even comparable with the fully supervised model. We can also see that the dataset is very challenging that even the supervised model only achieved 5.6 points in the mIoU unseen, which indicates there is still much room for improvements.

5. Visualization

For the visualization analyses, we use a SPNet-FPN and a ZegFormer-seg trained with 156 seen classes on COCO-Stuff. The CLIP text encoder is used to generate text embeddings. Then we test the two models with different sets of class names.

Results with 171 classes from COCO-Stuff. From Fig. 5, we can see that when segmenting unseen classes, it is usu-
ally confused by similar classes. Since the pixels of a segment have large variations, these pixel-level classifications are not consistent. In contrast, decoupling formulation can obtain high-quality segmentation results. Based on the segmentation results, the segment-level zero-shot classification is more related to CLIP pretraining. Therefore, ZegFormer-seg can obtain more accurate classification results.

Results with 847 classes from ADE20k-Full. As shown in Fig. 6, the SPNet-FPN (pixel-level zero-shot classification) is much worse than the ZegFormer-seg (our decoupling formulation). The reason is that there is severe competition among classes at pixel-level classification when we use 847 classes for inference. In contrast, the decoupling model is not influenced by the number of classes. These results confirm that the decoupling formulation is a right way to achieve human-level ability to segment objects with a large number of unseen classes. We can also see that the set of 847 class names contains richer information to generate text embeddings for inference than the set of 171 class names. For example, the unannotated “light, light source” is segmented by the two ZS3 models (white box in the 1st row of Fig. 6.) The labeled “motorcycle” is predicted as “tricycle” by ZegFormer (red box in the 2nd row of Fig. 6). The predicted “garage” by ZegFormer is more accurate than the labeled “house” in the ground truth (yellow box in the 2nd row of Fig. 6).

6. Conclusion

We reformulate the ZS3 task by decoupling. Based on the new formulation, we propose a simple and effective ZegFormer for the task of ZS3, which demonstrates significant advantages to the previous works. The proposed ZegFormer provides a new manner to study the ZS3 problem and serves as a strong baseline. Beyond the ZS3 task, ZegFormer also has the potentials to be used for few-shot semantic segmentation. We leave it for further research.

Limitations. Although the full model of ZegFormer shows
superiority in all the situations, we empirically find that ZegFormer-seg does not perform well when the scale of training data is small. One possible reason is that the transformer structure needs a large number of data for training, which has been discussed in [35]. A more efficient training strategy for ZegFormer-seg or using other mask classification methods such as K-Net [59] may alleviate this issue, and can be studied in the future.

Acknowledgement. We would like to thank Di Hu, XingXing Weng, and Zeran Ke for insightful discussion.

A. More Experiments

A.1. Results with the ZS3 setting

Apart from the generalized ZS3 (GZS3), we also report the results achieved with the ZS3 setting, in the evaluation of which the models only predict unseen labels \( c \in U \) (see more details in the Sec. 3.1 of our main paper.), and the pixels belong to the seen classes are ignored. The results on COCO-Stuff [8] are reported in Tab. 6. Since most of the existing studies do not report the results with the ZS3 setting, we compared with the SPNet [54] and re-implemented it with FPN and CLIP [46] text embeddings for building a baseline, i.e., SPNet-FPN. The results on ADE20k-Full [63] are reported by Tab. 7.

Table 6. Results on COCO-Stuff with the ZS3 setting. The result of SPNet is directly taken from its original paper [54], and SPNet-FPN is our re-implementation of the SPNet [54] with FPN and CLIP [46] text embeddings, which can be considered as a baseline.

<table>
<thead>
<tr>
<th>methods</th>
<th>backbone</th>
<th>class embed.</th>
<th>mIoU unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPNet [54]</td>
<td>R-101</td>
<td>ft+w2v</td>
<td>35.2</td>
</tr>
<tr>
<td></td>
<td>R-50</td>
<td>clip text</td>
<td>41.3</td>
</tr>
<tr>
<td>ZegFormer-seg</td>
<td>R-50</td>
<td>clip text</td>
<td>48.8</td>
</tr>
<tr>
<td>ZegFormer</td>
<td>R-50</td>
<td>clip text</td>
<td>61.5</td>
</tr>
</tbody>
</table>

A.2. Speed and Accuracy Analyses

Analyses of the Computational Complexity. Given \( C \) as the number of channels in a feature map, \( K \) as the number of classes, \( H \times W \) as the size of feature maps that are used for pixel-wise classification, and \( N \) being the number of segments in an image, the complexity of the classification head in the pixel-level zero-shot classification is \( O(H \times W \times C \times K) \), while the complexity of the classification head of our decoupling formulation is \( O(N \times C \times K) \). \( N \) is usually much smaller than \( H \times W \). For instance, in our COCO-Stuff experiments, \( N \) is 100, but \( H \times W \) is larger than \( 160 \times 160 \). Therefore, when \( K \) is large, pixel-level zero-shot classification will be much slower than the proposed decoupling formulation of ZS3.

Table 7. Results on ADE20k-Full achieved with the ZS3 setting. All the models use R-50 as a backbone and CLIP [46] as text embeddings.

<table>
<thead>
<tr>
<th>methods</th>
<th>mIoU unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPNet-FPN</td>
<td>7.4</td>
</tr>
<tr>
<td>ZegFormer-seg</td>
<td>9.4</td>
</tr>
<tr>
<td>ZegFormer</td>
<td>18.7</td>
</tr>
</tbody>
</table>

Table 8. Results on COCO-Stuff (171 classes) achieved with the GZS3 setting. The FPS is tested on images with the short side of 640 with a single GeForce RTX 3090.

<table>
<thead>
<tr>
<th>methods</th>
<th>seen</th>
<th>unseen</th>
<th>harmonic</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPNet-FPN</td>
<td>32.3</td>
<td>11</td>
<td>16.4</td>
<td>17.0</td>
</tr>
<tr>
<td>ZegFormer-seg</td>
<td>37.4</td>
<td>21.4</td>
<td>27.2</td>
<td>25.5</td>
</tr>
<tr>
<td>ZegFormer</td>
<td>35.9</td>
<td>33.1</td>
<td>34.4</td>
<td>6.0</td>
</tr>
</tbody>
</table>

Table 9. Results on ADE20k-Full (847 classes) achieved with the GZS3 setting. The FPS is tested on images with the short side of 512 with a single GeForce RTX 3090.

<table>
<thead>
<tr>
<th>methods</th>
<th>seen</th>
<th>unseen</th>
<th>harmonic</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPNet-FPN</td>
<td>9.2</td>
<td>0.9</td>
<td>1.6</td>
<td>7.9</td>
</tr>
<tr>
<td>ZegFormer-seg</td>
<td>18.9</td>
<td>1.3</td>
<td>2.4</td>
<td>31.3</td>
</tr>
<tr>
<td>ZegFormer</td>
<td>19.7</td>
<td>5.6</td>
<td>8.7</td>
<td>6.3</td>
</tr>
</tbody>
</table>

Speed and Accuracy Experiments. We compare the speeds of ZegFormer, ZegFormer-seg, and the SPNet-FPN. All these three models use R-50 with FPN as a backbone. ZegFormer-seg is an implementation for the decoupling formulation of ZS3, while SPNet-FPN is our implementation for pixel-level zero-shot classification. ZegFormer is our full model, with a branch to generate image embeddings (see Sec. 3.2 of our main paper for details.) As shown in Tab. 8 and Tab. 9, the ZegFormer-seg performs better than SPNet-FPN in both speed and accuracy on COCO-Stuff and ADE20k-Full. The ZegFormer improves the ZegFormer-seg by 12 points in term of mIoU of unseen classes on COCO-Stuff, and still remains an acceptable FPS. We can also see that the speed of SPNet-FPN is slow on ADE20k-Full. This verifies that the speed of pixel-level zero-shot classification is largely influenced by \( K \) (number of classes), as we have discussed before.

A.3. Comparisons with Different Backbones.

Since the ZegFormer-seg with R-50 is better than SPNet-FPN with R-50 in the supervised semantic segmentation, we also report the results of SPNet-FPN with R-101. From Tab. 10, we can see that the SPNet-FPN with R-101 is comparable with ZegFormer-seg in the supervised evaluation, but much lower than ZegFormer-seg with the GZS3.
B. More Visualization Results

We visualize the results of ZegFormer-Seg with R-50 and SPNet-FPN with R-50. The two models are trained on COCO-Stuff with 156 classes, and required to segment 847 classes. The visualization results are shown in Fig. 7, and Fig. 8.

C. More Implementation Details

C.1. HyperParameters

Following [12], we use a FPN [33] structure as the pixel decoder of ZegFormer and SPNet-FPN. The output stride of the pixel decoder is 4. Following [9, 12], we use 6 Transformer decoder layers and apply the same loss after each layer. The mask projection layer in ZegFormer consists of 2 hidden layers of 256 channels. During training, we crop images from the original images. The sizes of cropped images are \(640 \times 640\) in COCO-Stuff, and \(512 \times 512\) in the ADE20k-Full [63] and PASCAL VOC [15]. During testing, we keep the aspect ratio and resize the short size of an image to \(512\) in COCO-Stuff, and \(512\) in the ADE20k-Full and PASCAL VOC.

C.2. Prompt Templates

Following the previous works [19, 46], for each category, we used multiple prompt templates to generate the text embeddings then ensemble these text embeddings by averaging. The following is the prompt templates that we used in ZegFormer:

- `a photo of a {}`,
- `This is a photo of a {}`,
- `This is a photo of a small {}`,
- `This is a photo of a medium {}`,
- `This is a photo of a large {}`,
- `This is a photo of a {} in the scene`,
- `a photo of a {} in the scene`.

ZegFormer: $X$

<table>
<thead>
<tr>
<th>method</th>
<th>prompt ensemble</th>
<th>seen</th>
<th>unseen</th>
<th>harmonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZegFormer</td>
<td>✔</td>
<td>35.9</td>
<td>33.1</td>
<td>34.4</td>
</tr>
<tr>
<td>ZegFormer-seg</td>
<td>✖</td>
<td>35.4</td>
<td>32.7</td>
<td>34.0</td>
</tr>
</tbody>
</table>

To see the influence of prompt templates ensemble, we set a baseline by using only one prompt template, (i.e., “A photo of the {} in the scene.”) The comparisons are shown in Tab. 11. We can see that the prompt ensemble will slightly improve the performance.

References


Figure 7. Results on COCO-Stuff. ZegFormer-seg is our proposed model as an implementation of decoupling formulation of ZS3, while the SPNet-FPN is a pixel-level zero-shot classification baseline. Both the two models are trained with only 156 classes on COCO-Stuff, and required to segment with 847 class names. We can see that the pixel-level zero-shot classification is totally failed when there is a large number of unseen classes. In the yellow box are the unannotated category in COCO-Stuff but segmented by our model.

separable convolution for semantic image segmentation. In ECCV, pages 801–818, 2018. 1


Figure 8. Results on COCO-Stuff. ZegFormer-seg is our proposed model as an implementation of decoupling formulation of ZS3, while the SPNet-FPN is a pixel-level zero-shot classification baseline. Both models are trained with only 156 classes on COCO-Stuff, and required to segment with 847 class names. In the yellow box are the unannotated category in COCO-Stuff but segmented by our model.


[18] Yanwei Fu, Tao Xiang, Yu-Gang Jiang, Xiangyang Xue, Leonid Sigal, and Shaogang Gong. Recent advances in zero-


[26] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. arXiv:2102.05918, 2021. 2


