Diffusion Models for Open-Vocabulary Segmentation

Laurynas Karazija Iro Laina Andrea Vedaldi Christian Rupprecht
Visual Geometry Group, University of Oxford
https://www.robots.ox.ac.uk/~vgg/research/ovdiff/

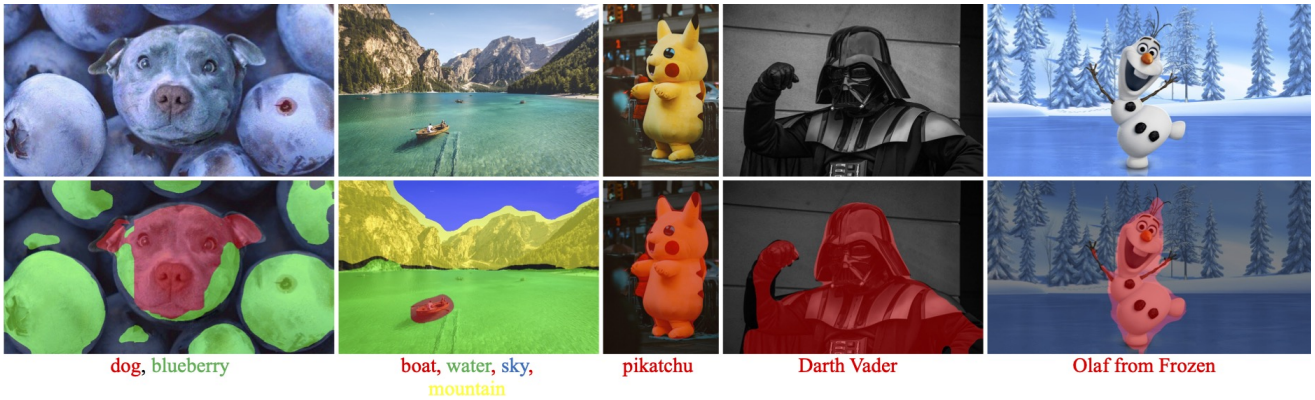


Figure 1. OVDiff is an open-vocabulary segmentation method that, given an image and a free-form set of class names, can segment any user-defined classes. It is fully automatic and does not require any further training.

Abstract

Open-vocabulary segmentation is the task of segmenting anything that can be named in an image. Recently, large-scale vision-language modelling has led to significant advances in open-vocabulary segmentation, but at the cost of gargantuan and increasing training and annotation efforts. Hence, we ask if it is possible to use existing *foundation models to synthesise on-demand efficient segmentation algorithms for specific class sets, making them applicable in an open-vocabulary setting without the need to collect further data, annotations or perform training. To that end, we present OVDiff, a novel method that leverages generative text-to-image diffusion models for unsupervised open-vocabulary segmentation. OVDiff synthesises support image sets for arbitrary textual categories, creating for each a set of prototypes representative of both the category and its surrounding context (background). It relies solely on pre-trained components and outputs the synthesised segmenter directly, without training.*

1. Introduction

Open-vocabulary semantic segmentation is the task of segmenting images into regions matching several free-form textual categories. As the field of Computer Vision moves towards large-scale general-purpose models, open-vocabulary "foundation" models have similarly emerged. Yet, the development of ones suitable for dense localisation tasks such as semantic segmentation incurs both enormous training costs and requires expensive mask annotations. Instead, we show that the open-vocabulary segmentation task can be effectively tackled starting from a set of frozen foundation models, without requiring additional data or even fine-tuning.

In order to do so, we introduce OVDiff, a method that turns existing foundation models into a "factory" of image segmenters, *i.e.*, using foundation models to synthesise on-demand a segmenter for any new concepts specified in natural language. Thus, OVDiff can be used for open-vocabulary segmentation, where it achieves state-of-the-art results in standard benchmarks. Moreover, once synthesised, the segmenters can be efficiently applied to any number of images and easily extended to new categories.

Specifically, segmenting an image using OVDiff can be done in three steps: *generation*, *representation*, and *matching*. Given a textual prompt, OVDiff uses an off-the-shelf text-to-image generator like StableDiffusion [35] to *generate* a support set of images. In the representation step, we use a feature extractor (that can be the same network as in the generation step) to extract feature prototypes that represent the

textual category. Finally, we use simple nearest-neighbour *matching* scheme to segment the target image using the prototypes computed in the previous step.

This approach differs from prior work that largely approaches the problem in either of two ways. Starting from multi-modal representations (*e.g.*, CLIP [32]) to bridge vision and language, the first way relies on costly dense annotations for some known categories to fine-tune image-level representations for the segmentation task. The second category of prior work [5, 27, 30, 34, 46, 47] extend large-scale vision-language models such as CLIP with additional grouping mechanisms for better localisation using only image-level captions, but no mask supervision. This, however, requires expensive additional contrastive training at scale. Additionally, most methods resort to heuristics, such as thresholding, to segment the background (*i.e.*, leave some pixels unlabelled), as it often cannot be described as a textual category. Finding an appropriate threshold, however, can be challenging and may vary depending on the image, often resulting in imprecise object boundaries. Effectively handling the background remains an open issue.

Our three-step approach departs substantially from both of these schemes. We show that large-scale text-to-image generative models, such as StableDiffusion [35], can help bridge the vision-and-language gap without the need for annotations or costly training. Furthermore, diffusion models also produce latent spaces that are semantically meaningful and well-localised. This solves a second problem: multi-modal embeddings are difficult to learn and often suffer from ambiguities and differences in detail between modalities. Instead, our approach can use unimodal features for open-vocabulary segmentation, which offers several advantages. Firstly, as text-to-image generators encode a distribution of possible images, this offers a means to deal with intra-class variation and captures the ambiguity in textual descriptions. Secondly, the generative image models encode not only the visual appearance of objects but also provide contextual priors, which we use for direct background segmentation.

2. Related work

Open-vocabulary segmentation. Open-vocabulary semantic segmentation is a relatively new problem and is typically approached in two ways. The first line of work poses the problem as "zero-shot", *i.e.*, segmenting unseen classes after training on a set of observed classes with dense annotations. Early approaches [2, 6, 12, 21] explore generative networks to sample features using conditional language embeddings for classes. Follow-up works [7, 11, 20, 23, 45, 49, 50] approach the problem in two steps, predicting class-agnostic masks and aligning the embeddings of masks with language. Different from our approach, all these works rely on mask supervision for a set of known classes.

The second line of work eliminates the need for mask annotations and instead aims to align image regions with language using only image-text pairs. Some methods introduce internal grouping mechanisms [25, 27, 34, 46, 47]. Another line of work [5, 30, 33, 52] aims to learn dense features that are better localised when correlated with language embeddings at pixel level. A closely related approach to ours is ReCO [38], where CLIP is used for image retrieval compiling a set of exemplar images from ImageNet for a given language query, which is then used for co-segmentation.

Diffusion models. Diffusion models [17, 39, 40] are a class of generative methods that have seen tremendous success. Dense annotations associate diffusion features with the desired output for discriminative tasks [1, 22, 48]. Annotation-free segmentation approaches are either closed-set [43] or not open-vocabulary [28].

3. Method

Our goal is to devise an algorithm which, given a new vocabulary of categories $c_i \in \mathcal{C}$ formulated as natural language queries, can segment any image against it. Let $I \in \mathbb{R}^{H \times W \times 3}$ be an image to be segmented. Let $\Phi_v : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{H'W' \times D}$ be an off-the-shelf visual feature extractor and $\Phi_t : \mathbb{R}^{d_t} \to \mathbb{R}^D$ a text encoder. Assuming that image and text encoders are aligned, one can achieve segmentation by simply computing a similarity function, for example, the cosine similarity between the densely encoded image $\Phi_v(I)$ and an encoding of a class label c_i.

We propose two modifications to this approach. First, we observe that comparing representations of the *same* modality is better than across vision and language modalities. We thus replace $\Phi_t(c_i)$ with a D-dimensional *visual* representation $\bar{P}$ of class c_i, which we refer to as a *prototype*. In this case, the same feature extractor can be used for both prototypes and target images; thus, their comparison becomes straightforward and does not necessitate further training. Second, we propose utilising *multiple* prototypes per category instead of a single class embedding. This enables us to accommodate intra-class variations in appearance, and, as we explain later, it also allows us to exploit contextual priors, which in turn help to segment the background.

Our approach, thus, proceeds in three steps: (1) a set of support images is sampled based on vocabulary $\mathcal{C}$, (2) a set of prototypes $\mathcal{P}$ is calculated, and (3) a set of images $\{I_1, I_2 \dots\}$ is segmented against these prototypes. We observe that in practical applications, whole image collections are processed using the same vocabulary, as altering the set of target classes for individual images in an informed way would already require some knowledge of their contents. Steps (1) and (2) are, thus, performed very infrequently, and their cost is heavily amortised. Next, we detail each step.

Support set generation. To construct a set of prototypes, for each category c_i, we define a prompt "`A good picture of a` $\langle c_i \rangle$" and generate a small batch of N

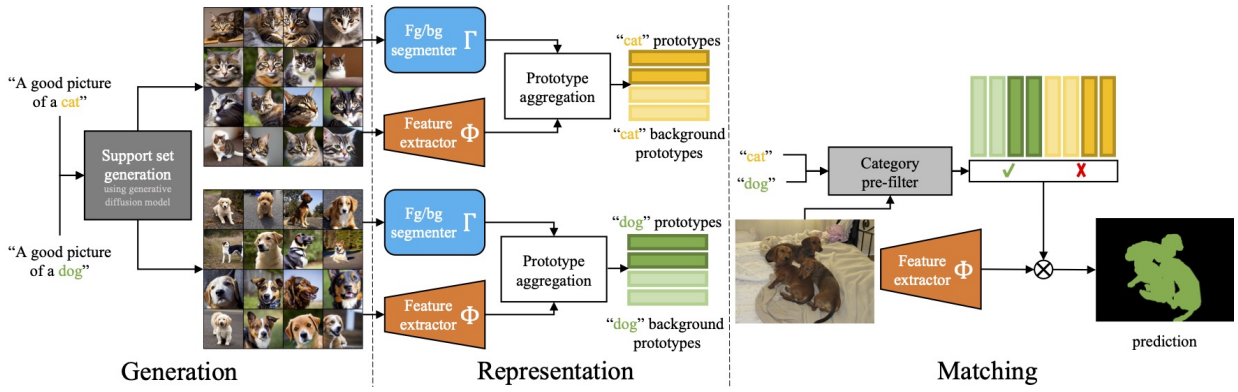


Figure 2. OVDiff overview. Prototype sampling: text queries are used to sample a set of support images which are further processed by a feature extractor and a segmenter forming positive and negative (background) prototypes. Segmentation: image features are compared against prototypes.The CLIP filter removes irrelevant prototypes based on global image contents.

support images $\mathcal{S}$ using Stable Diffusion [35].

Representing categories. Naïvely, prototypes $\bar{P}_{c_i}$ could be constructed by averaging all features across all images for class c_i. This is unlikely to result in good prototypes because not all pixels in the sampled images correspond to the class specified by c_i. Instead, we propose to extract the class prototypes as follows.

Class prototypes. Our approach generates two sets of prototypes, positive and negative, for each class. Positive prototypes are extracted from image regions that are associated with $\langle c_i \rangle$, while negative prototypes represent "background" regions. Thus, to obtain prototypes, the first step is segmenting the sampled images into foreground and background. To identify regions most associated with c_i, we use the fact that the layout of a generated image is largely dependent on the cross-attention maps of the diffusion model [15], *i.e.*, pixels attend more strongly to words that describe them. For a given word or description (in our case c_i), one can generate a set of attribution maps $\mathcal{A} = \{A_1, A_2, \ldots, A_N \mid A_n \in \mathbb{R}^{hw}\}$, corresponding to the support set $\mathcal{S}$, by summing the cross-attention maps across all layers, heads, and denoising steps of the network [41].

Yet, thresholding these attribution maps may not be optimal, as they are often coarse or incomplete, and sometimes only parts of objects receive high activation. To improve segmentation, we propose to optionally leverage an unsupervised instance segmentation method Γ. Unsupervised segmenters are not vocabulary-aware and may produce multiple binary object proposals. We denote these as $\mathcal{M}_n = \{M_{nr} \mid M_{nr} \in \{0,1\}^{hw}\}$, where n indexes the support images and r indexes the object masks (including a mask for the background). We thus construct a promptable extension of Γ segmenter to select appropriate proposals for foreground and background: for each image, we select from $\mathcal{M}_n$ the mask with the highest (lowest) average attribution as the foreground (background), calling them M_n^{fg} (M_n^{bg})

Prototype aggregation. We can compute prototypes P_n^{g} for foreground and background regions ($\mathrm{g} \in \{\mathrm{fg}, \mathrm{bg}\}$) as

$$P_n^{\mathrm{g}} = \frac{(\hat{M}_n^{\mathrm{g}})^\top \Phi_v(S_n)}{(\hat{M}_n^{\mathrm{g}})^\top \hat{M}_n^{\mathrm{g}}} \in \mathbb{R}^D, \tag{1}$$

where $\hat{M}_n^{\mathrm{g}}$ denotes a resized version of M_n^{g} that matches the spatial dimensions of $\Phi_v(S_n)$. Thus, prototypes are obtained by means of an off-the-shelf pretrained feature extractor and computed as the average feature within each mask.

We refer to these as *instance* prototypes because they are computed from each image individually, and each image in the support set can be viewed as an instance of class c_i.

In addition to instance prototypes, we found it helpful to also compute *class-level* prototypes $\bar{P}^{\mathrm{g}}$ by averaging the instance prototypes weighted by their mask sizes as $\bar{P}^{\mathrm{g}} = \sum_{n=1}^{N} m_n^{\mathrm{g}} P_n^{\mathrm{g}} / \sum_{n=1}^{N} (\hat{M}_n^{\mathrm{g}})^\top \hat{M}_n^{\mathrm{g}}$.

Finally, we propose to augment the set of class and instance prototypes using K-Means clustering of the masked features to obtain *part-level* prototypes. We perform spatial clustering separately on foreground and background regions and take each cluster centroid as a prototype P_k^{g} with $1 \leq k \leq K$. The intuition behind this is to enable segmentation at the level of parts, support greater intra-class variability, and a wider range of feature extractors that might not be scale invariant.

We consider the union of all these feature prototypes, for both foreground and background, for each $c_i \in \mathcal{C}$.

Segmentation via prototype matching. To perform segmentation of any target image I given a vocabulary $\mathcal{C}$, we first extract image features using the same visual encoder Φ_v used for the prototypes. The vocabulary is expanded with an additional background class for which a prototype is a union of all *background* prototypes in the vocabulary. For other classes we consider foreground prototypes. Then, a segmentation map can simply be obtained by matching dense image features to prototypes using cosine similarity.

Table 1. Open-vocabulary segmentation. Comparison of our approach, OVDiff, to the state of the art (under the mIoU metric). Our results are an average of 5 seeds $\pm\sigma$. *results from [5].

Method	Support Set	Further Training	VOC	Context	Object
ReCo* [38]	Real	✗	25.1	19.9	15.7
ViL-Seg [25]	✗	✓	37.3	18.9	-
MaskCLIP* [52]	✗	✗	38.8	23.6	20.6
TCL [5]	✗	✓	51.2	24.3	30.4
CLIPpy [33]	✗	✓	52.2	-	<u>32.0</u>
GroupViT [46]	✗	✓	52.3	22.4	-
ViewCo [34]	✗	✓	52.4	23.0	23.5
SegCLIP [27]	✗	✓	52.6	<u>24.7</u>	26.5
OVSegmentor [47]	✗	✓	53.8	20.4	25.1
CLIP-DIY [44]	✗	✗	<u>59.9</u>	–	31.0
OVDiff (-CutLER)	Synth.	✗	62.8	28.6	34.9
OVDiff	Synth.	✗	**66.3** $\pm$ **0.2**	**29.7** $\pm$ **0.3**	**34.6** $\pm$ **0.3**

Table 2. Segmentation performance of OVDiff based on different feature extractors.

Feature Extractor	MAE	DINO	CLIP (token)	CLIP (keys)	SD	SD + DINO + CLIP
VOC	54.9	59.1	51.4	61.8	64.4	66.4

Category pre-filtering. To limit the impact of spurious correlations that might exist in the feature space of the visual encoder, we introduce a pre-filtering process for the target vocabulary given image I. Specifically, we propose to leverage CLIP [32] in a multi-label fashion to constrain the segmentation to the subset of categories $\mathcal{C}' \subseteq \mathcal{C}$ that appear in the target image. First, we encode the target image and each category using CLIP. Any categories that do not score higher than $1/|\mathcal{C}|$ are removed from consideration. If more than η categories are present, then the top-η are selected. We then form "multi-label" prompts as "$\langle c_a \rangle$ `and` $\langle c_b \rangle$ `and ...`" where the categories are selected among the top scoring ones taking into account all 2^η combinations. The best-scoring multi-label prompt determines the final list of categories and, thus, prototypes for segmentation.

"Stuff" filtering. Occasionally, c_i might not describe a countable object category but an identifiable region in the image, *e.g.*, `sky`, often referred to as a "stuff" class. "Stuff" classes warrant additional consideration as they might appear as background in images of other categories. As a result, the process outlined above might sample background prototypes for one class that coincide with the foreground prototypes of another. To mitigate this issue, we introduce an additional filtering step to detect and reject such prototypes, when the full vocabulary, *i.e.*, the set of classes under consideration, is known. First, we only consider foreground prototypes for "stuff" classes. Additionally, any negative prototypes of "thing" classes with high cosine similarity with any of the "stuff" class prototypes are simply removed. In our experiments, we use ChatGPT [31] to automatically categorise a set of classes as "thing" or "stuff".

4. Experiments

We evaluate OVDiff on the open-vocabulary semantic segmentation task on validation splits of PASCAL VOC (VOC) [10], PASCAL Context (Context) [29] and COCO-Object (Object) [3] datasets reporting mean Intersection-over-Union. First, we consider different feature extractors and investigate how they can be grounded by our approach. We then compare our method with prior work, and conclude with a qualitative results on in-the-wild images. We leave specification of implementation details, ablations and further experiments for the Appendix.

Grounding feature extractors. Our method can be combined with *any* pretrained visual feature extractor for constructing prototypes and extracting image features. To verify this quantitatively, we experiment with various self-supervised ViT feature extractors (Tab. 2): DINO [4], MAE [14], and CLIP [32]. We also use SD features.

We find that SD performs the best, though CLIP and DINO also show strong performance based on our experiments on VOC. As feature extractors have different training objectives, we hypothesise that their feature spaces might be complementary. Thus, we also consider a combination of SD, DINO, and CLIP, which performs the best. We adopt this formulation for the main set of experiments.

Comparison to existing methods. In Tab. 1, we compare our method with prior work that does not rely on manual mask annotation on three datasets: VOC, Context, Object. We include a brief overview of the methods in the supplement. We find that our method compares favourably, outperforming other methods in all settings. In particular, results on VOC show the largest margin, with more than 5% improvement over prior work.

We also consider a version of our method, OVDiff (-CutLER), that does not rely on an additional unsupervised segmenter Γ. Instead, the attention masks are thresholded. We observe that such a version of OVDiff has strong performance, outperforming prior work as well. CutLER is helpful, but not a critical component, and OVDiff performs strongly without it. In Fig. 1, we investigate OVDiff on challenging in-the-wild images with simple and complex backgrounds.

5. Conclusion

We introduce OVDiff, an open-vocabulary segmentation method that operates in two stages. First, given queries, support images are sampled and their features are extracted to create class prototypes. These prototypes are then compared to features from an inference image. This approach offers multiple advantages: diverse prototypes accommodating various visual appearances and negative prototypes for background localisation. OVDiff outperforms prior work on benchmarks, exhibiting fewer errors, effectively separating objects from background.

References

[1] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *International Conference on Learning Representations*, 2022. 2

[2] Maxime Bucher, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez. Zero-shot semantic segmentation. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[3] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE, 2018. 4, 15

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 4, 13

[5] Junbum Cha, Jonghwan Mun, and Byungseok Roh. Learning to generate text-grounded mask for open-world semantic segmentation from only image-text pairs. *arXiv preprint arXiv:2212.00785*, 2022. 2, 4, 7, 10, 15

[6] Jiaxin Cheng, Soumyaroop Nandi, Prem Natarajan, and Wael Abd-Almageed. Sign: Spatial-information incorporated generative network for generalized zero-shot semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9556–9566, 2021. 2

[7] Jian Ding, Nan Xue, Gui-Song Xia, and Dengxin Dai. Decoupling zero-shot semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11583–11592, 2022. 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 13

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2): 303–338, 2010. 15

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 4, 15

[11] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Scaling open-vocabulary image segmentation with image-level labels. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*, pages 540–557. Springer, 2022. 2

[12] Zhangxuan Gu, Siyuan Zhou, Li Niu, Zihan Zhao, and Liqing Zhang. Context-aware feature generation for zero-shot semantic segmentation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1921–1929, 2020. 2

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 13

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 4

[15] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 3

[16] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 15

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 2

[18] Ronghang Hu, Shoubhik Debnath, Saining Xie, and Xinlei Chen. Exploring long-sequence masked autoencoders. *arXiv preprint arXiv:2210.07224*, 2022. 13

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2014. 13

[20] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2021. 2

[21] Peike Li, Yunchao Wei, and Yi Yang. Consistent structural relation learning for zero-shot segmentation. *Advances in Neural Information Processing Systems*, 33:10317–10327, 2020. 2

[22] Ziyi Li, Qinye Zhou, Xiaoyun Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. Guiding text-to-image diffusion model towards grounded generation. *arXiv:2301.05221*, 2023. 2

[23] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. *arXiv preprint arXiv:2210.04150*, 2022. 2

[24] Ping-Sung Liao, Tse-Sheng Chen, Pau-Choo Chung, et al. A fast algorithm for multilevel thresholding. *J. Inf. Sci. Eng.*, 17 (5):713–727, 2001. 15

[25] Quande Liu, Youpeng Wen, Jianhua Han, Chunjing Xu, Hang Xu, and Xiaodan Liang. Open-world semantic segmentation via contrasting and clustering vision-language embedding. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XX*, pages 275–292. Springer, 2022. 2, 4, 15

[26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 15

[27] Huaishao Luo, Junwei Bao, Youzheng Wu, Xiaodong He, and Tianrui Li. SegCLIP: Patch aggregation with learnable centers for open-vocabulary semantic segmentation. *arXiv preprint arXiv:2211.14813*, 2022. 2, 4, 15

[28] Chaofan Ma, Yuhuan Yang, Chen Ju, Fei Zhang, Jinxiang Liu, Yu Wang, Ya Zhang, and Yanfeng Wang. Diffusionseg: Adapting diffusion towards unsupervised object discovery. *arXiv preprint arXiv:2303.09813*, 2023. 2

[29] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014. 4, 15

[30] Jishnu Mukhoti, Tsung-Yu Lin, Omid Poursaeed, Rui Wang, Ashish Shah, Philip HS Torr, and Ser-Nam Lim. Open vocabulary semantic segmentation with patch aligned contrastive learning. *arXiv preprint arXiv:2212.04994*, 2022. 2

[31] OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt, 2023. 4

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2, 4, 14

[33] Kanchana Ranasinghe, Brandon McKinzie, Sachin Ravi, Yinfei Yang, Alexander Toshev, and Jonathon Shlens. Perceptual grouping in vision-language models. *arXiv preprint arXiv:2210.09996*, 2022. 2, 4, 7, 15

[34] Pengzhen Ren, Changlin Li, Hang Xu, Yi Zhu, Guangrun Wang, Jianzhuang Liu, Xiaojun Chang, and Xiaodan Liang. Viewco: Discovering text-supervised segmentation masks via multi-view semantic consistency. *arXiv preprint arXiv:2302.10307*, 2023. 2, 4, 15

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 1, 2, 3, 13

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 13

[37] Patrick Schramowski, Manuel Brack, Björn Deiseroth, and Kristian Kersting. Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22522–22531, 2023. 11

[38] Gyungin Shin, Weidi Xie, and Samuel Albanie. Reco: Retrieve and co-segment for zero-shot transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 4, 15

[39] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 2

[40] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. 2

[41] Raphael Tang, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Jimmy Lin, and Ferhan Ture. What the daam: Interpreting stable diffusion using cross attention. *arXiv preprint arXiv:2210.04885*, 2022. 3

[42] Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. *arXiv preprint arXiv:2301.11320*, 2023. 7, 15

[43] Weijia Wu, Yuzhong Zhao, Mike Zheng Shou, Hong Zhou, and Chunhua Shen. Diffumask: Synthesizing images with pixel-level annotations for semantic segmentation using diffusion models. *arXiv preprint arXiv:2303.11681*, 2023. 2

[44] Monika Wysoczańska, Michaël Ramamonjisoa, Tomasz Trzciński, and Oriane Siméoni. Clip-diy: Clip dense inference yields open-vocabulary semantic segmentation for-free. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1403–1413, 2024. 4, 15

[45] Yongqin Xian, Subhabrata Choudhury, Yang He, Bernt Schiele, and Zeynep Akata. Semantic projection network for zero-and few-label semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8256–8265, 2019. 2

[46] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18134–18144, 2022. 2, 4, 15

[47] Jilan Xu, Junlin Hou, Yuejie Zhang, Rui Feng, Yi Wang, Yu Qiao, and Weidi Xie. Learning open-vocabulary semantic segmentation models from natural language supervision. *arXiv preprint arXiv:2301.09121*, 2023. 2, 4, 7, 15

[48] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Open-vocabulary panoptic segmentation with text-to-image diffusion models. *arXiv preprint arXiv:2303.04803*, 2023. 2

[49] Mengde Xu, Zheng Zhang, Fangyun Wei, Yutong Lin, Yue Cao, Han Hu, and Xiang Bai. A simple baseline for open-vocabulary semantic segmentation with pre-trained vision-language model. In *European Conference on Computer Vision*, pages 736–753, 2022. 2

[50] Sukmin Yun, Seong Hyeon Park, Paul Hongsuck Seo, and Jinwoo Shin. Ifseg: Image-free semantic segmentation via vision-language model. *arXiv preprint arXiv:2303.14396*, 2023. 2

[51] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 10

[52] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 696–712. Springer, 2022. 2, 4, 15

Supplementary Material

In this supplementary material, we provide additional experimental results, including further ablations and qualitative comparisons (Appendix A), consider the limitations and broader impacts of our work (Appendix B), and conclude with additional details concerning the implementation (Appendix C).

A. Additional experiments

This section provides additional experimental results of OVDiff.

A.1. Additional Comparisons

Category filter. To ensure that the category pre-filtering does not give our approach an unfair advantage, we augment two methods (TCL [5] and OVSegmentor [47], which are the closest baselines with code and checkpoints available) with our category pre-filtering. We evaluate on the Pascal VOC dataset (where the category filter shows a significant impact; see Table 3) and report the results in Tab. A.2. We observe that TCL improves by 0.6, while the performance of OVSegmentor drops by 0.1. On the contrary, our method benefits substantially from this component, but it still shows stronger performance without the filter than baselines with.

Influence of Γ segmentation method. We also further investigate the use of CutLER [42] to obtain segmentation masks. We also provide example results of segmentation in Fig. C.8. In Tab. A.3, we devise a baseline where CutLER-predicted masks are used to average the CLIP image encoder's final spatial tokens after projection. Averaged tokens are compared with CLIP text embeddings to assign a class. While relying on pre-trained components (like ours), this avoids support set generation. In the same table, we also consider whether the objectness prior provided by CutLER could be beneficial to other methods as well. We consider a version of TCL [5] and OVSegmentor [47] which we augment with CutLER. That is, after methods assign class probabilities to each pixel/patch, a majority voting for a class is performed in every region predicted by CutLER. This combines CutLER's understanding of objects and their boundaries, aspects where prior methods struggle, with open-vocabulary segmentation. However, we observe that this negatively impacts the performance of these methods, which we attribute to only a limited performance of CutLER in complex scenes present in the datasets. Finally, we also include a version of OVDiff that does not rely on CutLER for mask extractions, instead using thresholded masks. We observe that such a version of our method also has strong performance.

We additionally experiment with stronger segmenters to understand the influence of FG/BG mask quality. We replace our FG/BG segmentation approach with strong supervised models: with SAM, we achieve 67.1 on VOC, and with Grounded SAM, 68.5. This slightly improves results from 66.3 of our configuration with CutLER, but the performance gain is not large and thus not critical.

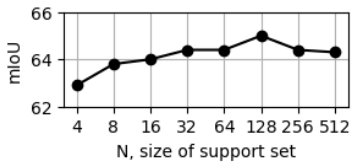


Figure A.1. PascalVOC results with increasing support size N.

Class prompts. We additionally consider whether corrections introduced to class prompts might have similarly provided additional benefits to our approach (see Appendix C.3 for details). To that end, we also evaluate TCL and OVSegmenter (methods that do not rely on additional prompt curation) with our corrected prompts and consider a version of our method without such corrections in Tab. A.4. We observe only marginal to no impact on the performance.

Prompt template Finally, we consider the prompt template employed when sampling support image set: "`A good picture of a` $\langle c_i \rangle$" for class prompt c_i. This template is generic and broadly applicable to virtually any natural language specification of a target class. While prior work adopts prompt expansion by considering a list of synonyms and subcategories, it is not entirely clear how such a strategy could be systematically performed for any in-the-wild prompts, such as a "chocolate glazed donut". We experiment with a list of synonyms and subclasses, as employed by [33], on VOC datasets measuring 66.4 mIoU, which is similar to our single prompt performance 66.3 ± 0.2. Curating such lists automatically is an interesting future scaling direction.

Computation cost. We focus on a construction of a method to show that existing foundational diffusion models can be used for segmentation with great efficacy without further training. OVDiff requires computing prototypes instead. With our unoptimized implementation, we measure around 110 ± 10s to calculate prototypes using SD for a single class, or around 1.14 TFLOP/s-hours of compute. While the focus of this study is not computational efficiency, we can compare prototype sampling to the cost of additional training of other methods: TCL requires 2688, GroupViT 10752, and OVSegmentor 624 TFLOP/s-hours.[1] While training has an upfront compute cost and requires special infrastructure (*e.g.* OVSegmentor uses 16×A100s), OVDiff's prototype set can be grown progressively as needed, while showing better performance.

A.2. Ablations

Next, we ablate the components of OVDiff on VOC and Context datasets. For these experiments, only SD is employed as a feature extractor. We remove individual components

[1] Estimated as training time × num. GPUs × theoretical peak TFLOP/s for GPU type.

Figure A.2. Qualitative comparison on challenging in-the-wild images with TCL, which struggles with object boundaries, missing parts of objects, or including surroundings. Our method has more appropriate boundaries and makes fever errors overall, but does produce a small halo effect around objects due to the upscaling of feature extractors.

and measure the change in segmentation performance, summarising the results in Tab. A.1. Our first observation is that background prototypes have a major impact on performance. When removing them from consideration, we instead threshold the similarity scores of the images with the foreground prototypes (set to 0.72, determined via grid search); in this case, the performance drops significantly, which again highlights the importance of leveraging contextual priors. On Context, the impact is less significant, likely due to the fact that the dataset contains "stuff" categories. Removing the *instance-* and *part-level* prototypes also negatively affects performance. Additionally, removing the category pre-filtering has a major impact. We hypothesize that this introduces spurious correlations between prototypes of different classes. On Context, "stuff" filtering is also important. We again consider the importance of using an unsupervised segmenter, CutLER, for prototype mask extractions, using thresholding instead. We find this slightly reduces performance in this setting as well. Overall, background prototypes and pre-filtering contribute the most.

Finally, we measure the effect of varying the size of the support set N in Fig. A.1. We find that OVDiff already shows strong performance even at a low number of samples for each query. With increasing the number of samples, the performance improves, saturating at around $N = 32$. which we use in our main experiments.

Prototype combinations. In Tab. A.7, we consider the three different types of prototypes described in Section 3 and test their performance individually and in various combinations. We find that the "part" prototypes obtained by K-means clustering show strong performance when considered individually on VOC. Instance prototypes show strong individual performance on Context, as well as in combination with the average category prototype. The combination of all three types shows the strongest results across the two datasets, which is what we adopt in our main set of experiments.

Table A.1. Ablation of different components. Each component is removed in isolation, measuring the drop (Δ) in mIoU on VOC and Context datasets. Using SD features.

Configuration	VOC	Δ	Context	Δ
Full	64.4		29.4	
w/o bg prototypes	53.2	-11.2	28.9	-0.5
w/o category filter	54.4	-10.0	25.2	-4.2
w/o "stuff" filter	n/a		26.9	-2.5
w/o CutLER	60.4	-4.0	27.6	-1.8
w/o sliding window	62.2	-2.2	28.6	-0.8
only average $\bar{P}$	62.5	-1.9	28.4	-1.0

We also consider the treatment of prototypes under the stuff filter. We investigate the impact of not excluding background prototypes for "stuff" classes. In this setting, we measure 29.1 on Context, which is a slight reduction in performance. We also investigate the benefit of categorisation into "things" and "stuff" used in the stuff filter component. Instead, we filter all background prototypes using all foreground prototypes. In this configuration, we measure 27.6 on Context. Both configurations show a reduction from 29.4, measuring using the stuff filter with categorisation in "stuff" and "things", as used in our main experiments. Finally, we experiment by removing part-level prototypes for "stuff" classes, which also results in a performance drop to 28.0.

K - number of clusters. In Tab. A.5, we investigate the sensitivity of the method to the choice of K for the number of "part" prototypes extracted using K-means clustering. Although our setting $K = 32$ obtains slightly better results on Context and VOC, other values result in comparable segmentation performance suggesting that OVDiff is not

Table A.2. Use of category filter component. OVDiff without category filter outperforms prior work with cat. filter.

Model	Category filter ✗	Category filter ✓
OVSegmentor	53.8	53.7
TCL	51.2	51.8
TCL (+PAMR)	55.0	56.0
OVDiff	**56.2**	**66.4**

Table A.3. Application of CutLER. Prior work does not benefit from using CutLER during inference, while OVDiff shows strong results without it.

Model	CutLER	VOC	Context	Object
CLIP	✓	33.0	11.6	11.1
OVSegmentor		53.8	20.4	25.1
OVSegmentor	✓	38.7	14.4	16.8
TCL		51.2	24.3	30.4
TCL	✓	43.1	20.5	22.7
OVDiff		62.8	28.6	34.9
OVDiff	✓	**66.3** $\pm$ **0.2**	**29.7** $\pm$ **0.3**	**34.6** $\pm$ **0.3**

Table A.4. Using corrected prompts. We consider if corrected class names benefit prior work. We observe negligible to no effect.

Model	Correction	VOC	Context	Object
OVSegmentor		53.8	20.4	25.1
OVSegmentor	✓	53.9	20.4	25.1
TCL		51.2	24.3	30.4
TCL	✓	50.6	24.3	30.4
OVDiff		66.1	29.5	34.9
OVDiff	✓	**66.3** $\pm$ **0.2**	**29.7** $\pm$ **0.3**	**34.6** $\pm$ **0.3**

Table A.5. Choice of K for number of centroids.

K	VOC	Context
8	63.8	29.2
16	64.0	29.3
32	64.4	29.4
64	64.3	28.0

sensitive to the choice of K and a range of values is viable.

SD features. When using Stable Diffusion as a feature extractor, we consider various combinations of layers/blocks in the UNet architecture. We follow the nomenclature used in the Stable Diffusion implementation where consecutive layers of Unet are organised into *blocks*. There are 3 down-sampling blocks with 2 cross-attention layers each, a mid-block with a single cross-attention, and 3 up-sampling blocks with 3 cross-attention layers each. We report our findings in Tab. A.6. Including the first and last cross-attention layers in the feature extraction process has a small positive impact on segmentation performance, which we attribute to the high feature resolution. We also consider excluding features from the middle block of the network due to small 8×8 resolution but observe a small negative impact on performance on the Context dataset. We also investigate whether including the first (Up-1) and the second upsampling (Up-2) blocks are necessary. Without them, the performance drops the most out of the configurations considered. Thus, we use a concatenation of features from the middle, first and second upsampling blocks and the first and last layers in our main experiments.

Table A.6. Ablation of different SD feature configurations. Removing first and last cross attention *layers*, mid, 1^{st} and 2^{nd} upsampling *blocks* (all layers in the block) has a negative effect.

1st layer	Mid block	Up-1 block	Up-2 block	Last layer	Context
✓	✓	✓	✓	✓	29.4
	✓	✓	✓	✓	29.4
✓		✓	✓	✓	29.2
✓	✓		✓	✓	27.3
✓	✓	✓		✓	28.9
✓	✓	✓	✓		29.3

Table A.7. Ablation of various configurations for prototypes. We consider average $\bar{P}$, instance P_n, and part P_k prototypes individually and in various combinations on VOC and Context datasets. Combination of all three types of prototypes shows strongest results.

$\bar{\mathbf{P}}$	$\mathbf{P_n}$	$\mathbf{P_k}$	VOC	Context
✓	✓	✓	64.4	29.4
✓		✓	61.7	29.3
✓	✓		63.5	29.4
	✓	✓	62.5	28.4
		✓	63.7	28.8
	✓		60.0	29.0
✓			62.5	28.4

A.3. Evaluation without background

One of the notable advantages of our approach is the ability to represent background regions via (negative) prototypes, leading to improved segmentation performance. Nevertheless, we hereby also evaluate our method under a different evaluation protocol adopted in prior work, which excludes the *background* class from the evaluation. We note that prior work often requires additional considerations to handle background, such as thresholding. In this setting, however, the background class is *not* predicted, and the set of categories, thus, must be exhaustive. As in practice, this is not the case, and datasets contain unlabelled pixels (or simply a background label), such image areas are removed from consideration. Consequently, less emphasis

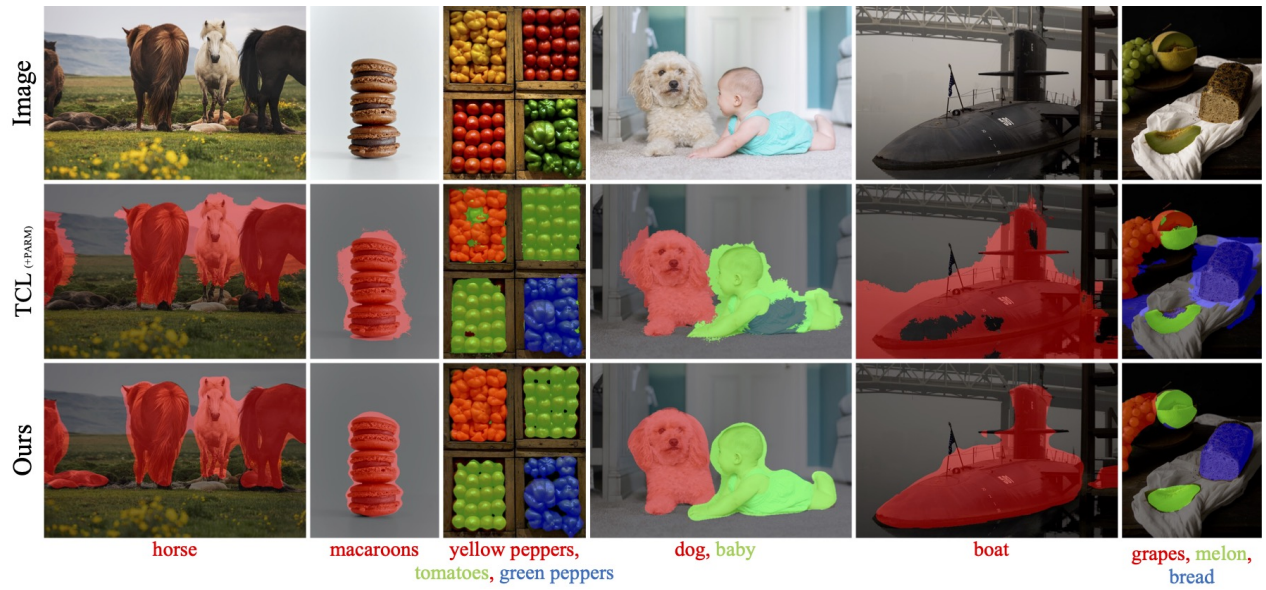


Figure A.3. Qualitative comparison on in-the-wild images. OVDiff performs significantly better than prior state-of-the-art, TCL, on wildlife images containing multiple instances, studio photos with simple backgrounds, images containing multiple categories and an image containing a rare instance of a class.

Table A.8. Comparison with methods when background is excluded (decided by ground truth). OVDiff shows comparable performance to prior works despite only relying on pretrained feature extractors. * result from [5].

Method	VOC-20	Context-59	ADE	Stuff	City
CLIPpy	–	–	13.5	–	–
OVSegmentor	–	–	5.6	–	–
GroupViT*	79.7	23.4	9.2	15.3	11.1
MaskCLIP*	74.9	26.4	9.8	16.4	12.6
ReCo*	57.5	22.3	11.2	14.8	21.1
TCL	77.5	30.3	**14.9**	19.6	23.1
OVDiff	**80.9**	**32.9**	14.1	**20.3**	**23.4**

is placed on object boundaries in this setting. As in this setting the background prediction is invalid, we do not consider negative prototypes. For this setting, we benchmark on 5 datasets following [5]: PascalVOC without background, termed VOC-20, Pascal Context without background, termed Context-59, and ADE20k [51], which contains 150 foreground classes, termed ADE-150, COCO-Stuff, termed Stuff, and Cityscapes, termed City. This setting tests the ability of various methods to discriminate between different classes, which for OVDiff is inherent to the choice of feature extractors. Despite this, our method shows competitive performance accross wide range of benchmarks Tab. A.8.

A.4. In-the-wild

In Fig. A.2 and Fig. A.3, we investigate OVDiff on challenging in-the-wild images with simple and complex backgrounds. We compare with TCL+PAMR. In the first three images, both methods correctly detect the objects identified by the queries. OVDiff has small false positive "corgi" patches. TCL however misses large parts of the objects, such as most of the person, and parts of animal bodies. The distinction between the house and the bridge in the second image is also better with OVDiff. We also note that our segmentations sometimes have halos around objects. This is caused by upscaling the low-resolution feature extractor (SD in this case). The last two images contain challenging scenarios where both approaches struggle. The fourth image only contains similar objects of the same type. Both methods incorrectly identify plain donuts as either of the specified queries. OVDiff however correctly identifies chocolate donuts with varied sprinkles and separates all donuts from the background. In the final picture, the query “red car” is added, although no such object is present. The extra query causes TCL to incorrectly identify parts of the red bus as a car. Both methods incorrectly segment the gray car in the distance. However, overall, our method is more robust and delineates objects better despite the lack of specialized training or post-processing.

A.5. Explaining segmentations

We inspect how our method segments certain regions by considering which prototype from $\mathcal{P}_c^{\text{fg}}$ was used to assign

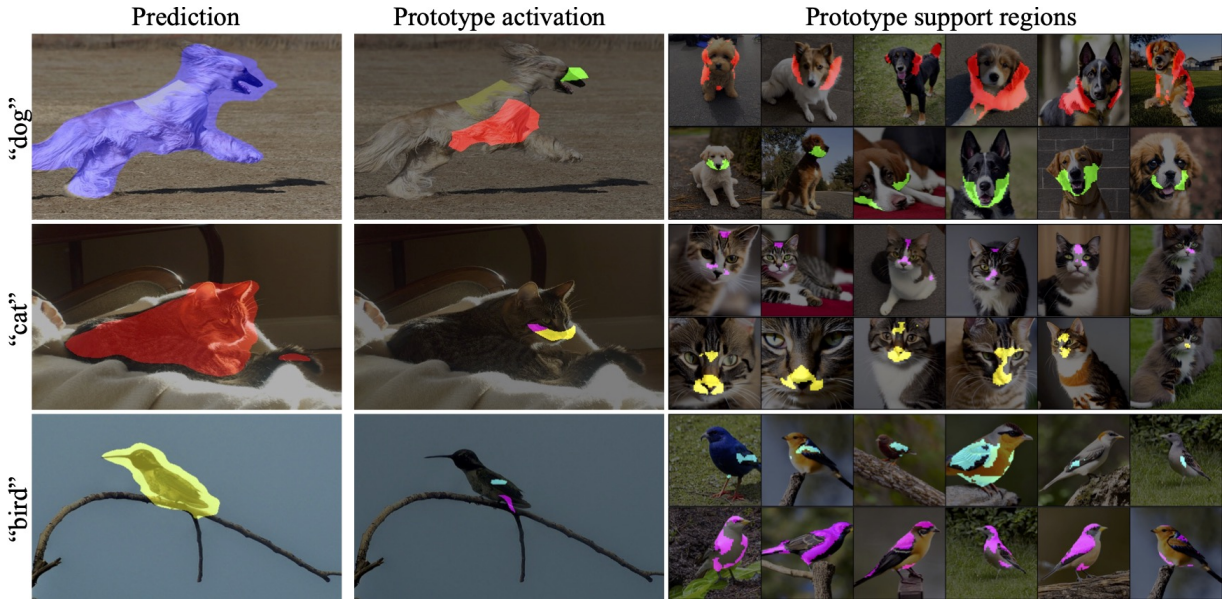


Figure A.4. Analysis of the segmentation output by linking regions to samples in the support set. Left: our results for different classes. Middle: select color-coded regions "activated" by different prototypes for the class. Right: regions in the support set images corresponding to these (part-level) prototypes.

a class c to a pixel. Prototypes map to regions in the support set from where they were aggregated, *e.g.*, instances prototypes are associated with foreground masks M_n^{fg} and part prototypes with centroids/clusters. By following these mappings, a set of support image regions can be retrieved for each segmentation decision, providing a degree of explainability. Fig. A.4 illustrates this for examples of `dog`, `cat`, and `bird` classes. For visualisation purposes, selected prototypes and corresponding regions are shown. On the left, we show the full segmentation result of each image. In the middle, we select regions that correlate best with certain class prototypes. On the right, we retrieve images from the support set and highlight where each prototype emerged. We find that meaningful part segmentation merges due to clustering the support image features, and similar regions are segmented by corresponding prototypes. However, sometimes region covered in the input image will not fully align with the whole prototype (*e.g.* `cat`'s face around the eyes or lower belly/tail of `bird`). Each segmentation is explained by precise regions in a small support set.

A.6. Qualitative results

We include additional qualitative results from the benchmark datasets in Fig. A.5. Qualitative results are shown in Fig. A.6 contain comparison with TCL. This figure highlights a key benefit of our approach: the ability to exploit contextual priors through the use of background prototypes, which in turn allows for the direct assignment of pixels to a background class. This improves segmentation quality because it makes it easier to differentiate objects from the background and to delineate their boundaries. In comparison, TCL predictions are very coarse and contain more noise. Our method achieves high-quality segmentation across all examples without any post-processing or refinement steps.

Our method achieves high-quality segmentation across all examples without any post-processing or refinement steps. In Fig. A.7, we show examples of support images sampled for some things, and stuff categories. In Fig. C.9, we show examples of support set images sampled for rare *pikachu* class.

B. Broader impact

Semantic segmentation is a component in a vast and diverse spectrum of applications in healthcare, image processing, computer graphics, surveillance and more. As for any foundational technology, applications can be good or bad. OVDiff is similarly widely applicable. It also makes it easier to use semantic segmentation in new applications by leveraging existing and new pre-trained models. This is a bonus for inclusivity, affordability, and, potentially, environmental impact (as it requires no additional training, which is usually computationally intensive); however, these features also mean that it is easier for bad actors to use the technology.

Because OVDiff does not require further training, it is more versatile but also inherits the weaknesses of the components it is built on. For example, it might contain the biases (e.g., gender bias) of its components, in particular Stable Diffusion [37], which is used for generating support images for any given category/description. Thus, it should not be exposed without further filtering and detection of, e.g., NSFW material in the sampled support set. Finally, OVDiff is also bound by the licenses of its components.

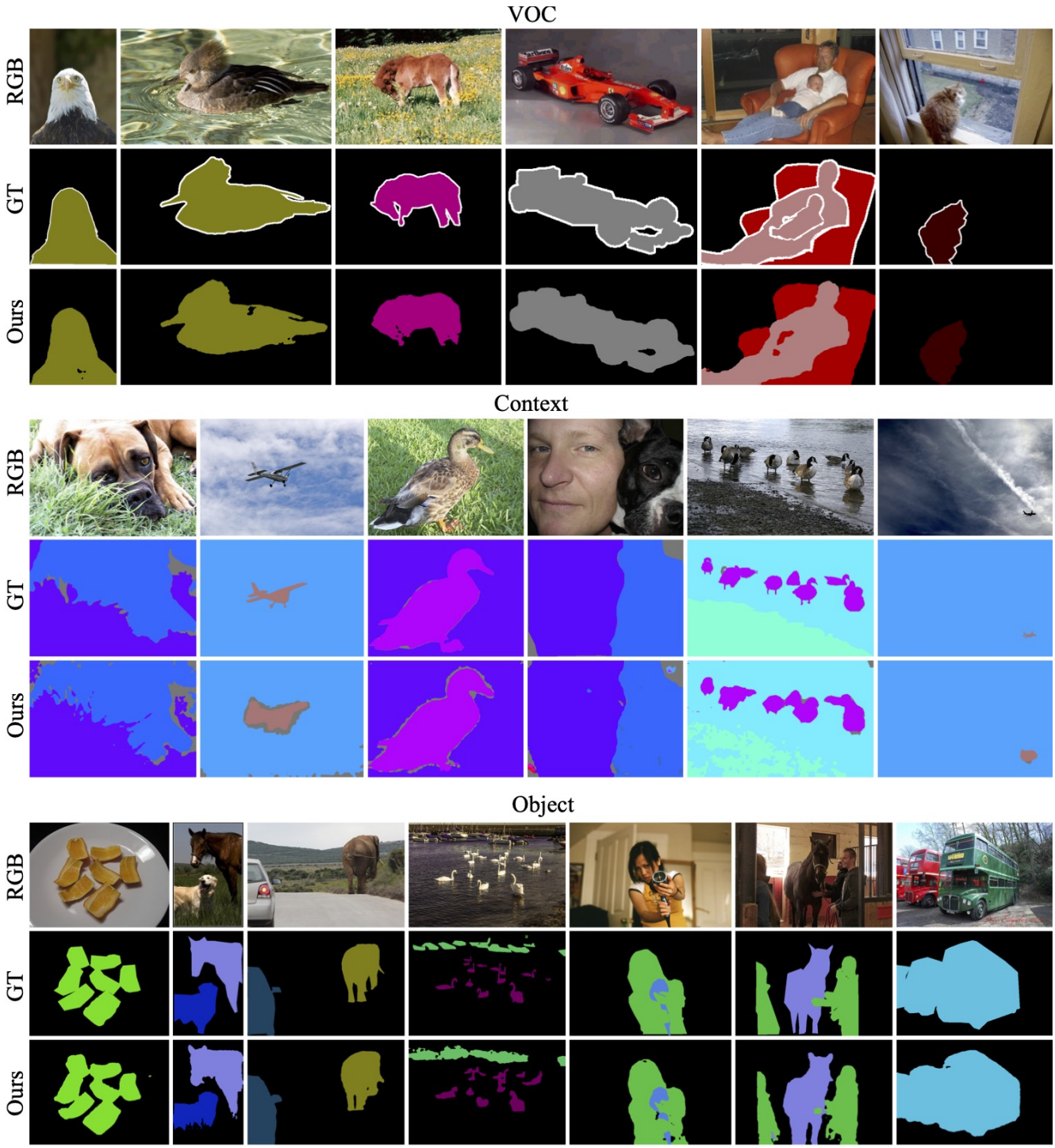


Figure A.5. Additional qualitative results. Images from Pascal VOC (top), Pascal Context (middle), and COCO Object (bottom).

B.1. Limitations

As OVDiff relies on pretrained components, it inherits some of their limitations. OVDiff works with the limited resolution of feature extractors, due to which it might occasionally miss tiny objects. Furthermore, OVDiff cannot segment what the generator cannot generate. For example, current diffusion models struggle with producing legible text, which can make it difficult to segment specific words. Furthermore, applications in domains far from the generator's training data

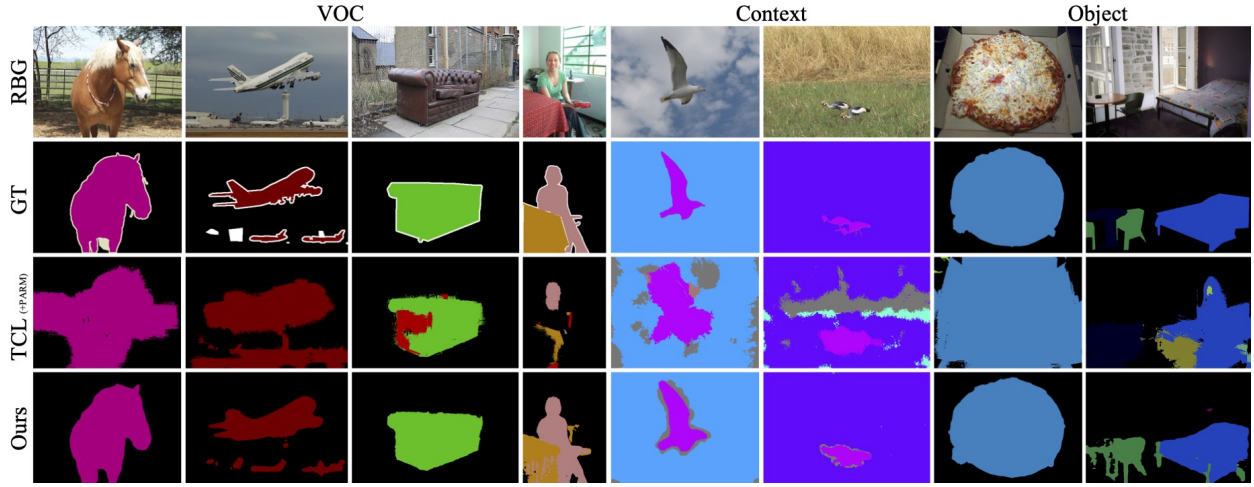


Figure A.6. Qualitative results. OVDiff in comparison to TCL (+ PAMR). OVDiff provides more accurate segmentations across a range objects and stuff classes with well defined object boundaries that separate from the background well.

(*e.g.* medical imaging) are unlikely to work out of the box.

C. OVDiff: Further details

In this section, we provide additional details concerning the implementation of OVDiff. We begin with a brief overview of the attention mechanism and diffusion models central to extracting features and sampling images. We review different feature extractors used. We specify the hyperparameter setting for all our experiments and provide an overview of the exchange with ChatGPT used to categorise classes into "thing" and "stuff".

C.1. Preliminaries

Attention. In this work, we make use of pre-trained ViT [8] networks as feature extractors, which repeatedly apply multi-headed attention layers. In an attention layer, input sequences $X \in \mathbb{R}^{l_x \times d}$ and $Y \in \mathbb{R}^{l_y \times d}$ are linearly project to forms *keys*, *queries*, and *values*: $K = W_k Y$, $Q = W_q X$, $V = W_v X$. In self-attention, $X = Y$. Attention is calculated as $A = \text{softmax}(\frac{1}{\sqrt{d}} Q K^\top)$, and softmax is applied along the sequence dimension l_y. The layer outputs an update $Z = X + A \cdot V$. ViTs use multiple heads, replicating the above process in parallel with different projection matrices W_k, W_q, W_v. In this work, we consider *queries* and *keys* of attention layers as points where useful features that form meaningful inner products can be extracted. As we detail later (Appendix C.2), we use the *keys* from attention layers of ViT feature extractors (DINO/MAE/CLIP), concatenating multiple heads if present.

Text-to-image diffusion models. Diffusion models are a class of generative models that form samples starting with noise and gradually denoising it. We focus on latent diffusion models [35] which operate in the latent space of an image VAE [19] forming powerful conditional image generators. During training, an image is encoded into VAE latent space, forming a latent vector z_0. A noise is injected forming a sample $z_\tau \sim \mathcal{N}(z_\tau; \sqrt{1-\alpha_\tau} z_0, \alpha_\tau I)$ for timestep $\tau \in \{1 \dots T\}$, where α_τ are variance values that define a noise schedule such that the resulting z_T is approximately unit normal. A conditional UNet [36], $\epsilon_\theta(z_t, t, c)$, is trained to predict the injected noise, minimising the mean squared error $\mathbb{E}_t \left(\alpha_t \| \epsilon_\theta(z_t, t, c) - z_0 \|_2\right)$ for some caption c and additional constants a_t. The network forms new samples by reversing the noise-injecting chain. Starting from $\hat{z}_T \sim \mathcal{N}(\hat{z}_T; 0, I)$, one iterates $\hat{z}_{t-1} = \frac{1}{\sqrt{1-\alpha_t}}(\hat{z}_t + \alpha_t \epsilon_\theta(\hat{z}_t, t, c)) + \sqrt{\alpha_t}\hat{z}_t$ until $\hat{z}_0$ is formed and decoded into image space using the VAE decoder. The conditional UNet uses cross-attention layers between image patches and language (CLIP) embeddings to condition on text c and achieve text-to-image generation.

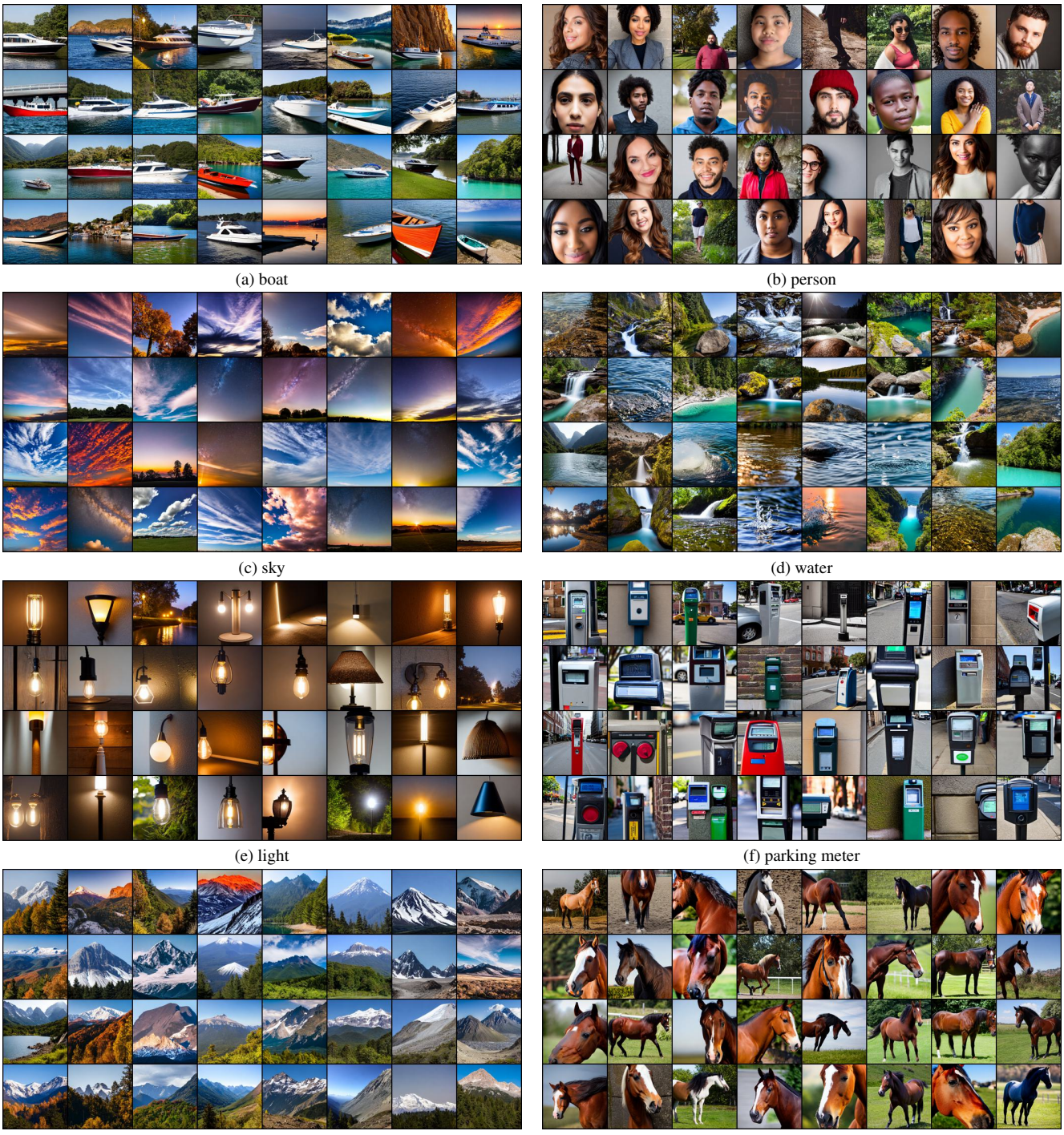
C.2. Feature extractors

OVDiff is buildable on top of any pre-trained feature extractor. In our experiments, we have considered several networks as feature extractors with various self-supervised training regimes:

- **DINO** [4] is a self-supervised method that trains networks by exploring alignment between multiple views using an exponential moving average teacher network. We use the ViT-B/8 model pre-trained on ImageNet[2] and extract features from the *keys* of the last attention layer.
- **MAE** [13] is a self-supervised method that uses masked image inpainting as a learning objective, where a portion of image patches are dropped, and the network seeks to reconstruct the full input. We use the ViT-L/16 model pre-trained on ImageNet at a resolution of 448 [18].[3] The

[2]Model and code available at https://github.com/facebookresearch/dino.

[3]Model and code from https://github.com/facebookresearch/long_seq_mae.



(a) boat (b) person (c) sky (d) water (e) light (f) parking meter (g) mountain (h) horse

Figure A.7. Images sampled for a support set of some categories.

keys of the last layer of the *encoder* network are used. No masking is performed.

- **CLIP** [32] is trained using image-text pairs on an internal dataset WIT-400M. We use ViT-B/16 model[4]. We consider two locations to obtain dense features: *keys* from a self-attention layer of the image encoder and *tokens* which are

[4]Model and code from https://github.com/openai/CLIP.

the outputs of transformer layers. We find that *keys* of the second-to-last layer give better performance.

- We also consider **Stable Diffusion**[5] (v1.5) itself as a feature extractor. To that end, we use the *queries* from the cross-attention layers in the UNet denoiser, which correspond to the image modality. Its UNet is organised into three downsampling blocks, a middle block, and three upsampling blocks. We observe that the middle layers have the most semantic content, so we consider the middle block, 1st and 2nd upsampling blocks and aggregate features from all three cross-attention layers in each block. As the features are quite low in resolution, we include the first downsampling cross-attention layer and the last upsampling cross-attention layer as well. The feature maps are bilinearly upsampled to resolution 64×64 and concatenated. A noise appropriate for $\tau = 200$ timesteps is added to the input. For feature extraction, we run SD in *unconditional* mode, supplying an empty string for text caption.

Figure C.8. FG/BG segmentation of classes of *water*, *snow* and *grass*. The foreground is in red, while the background is shown in blue.

Figure C.9. Example images from the support set of a rare *pikachu* class.

C.3. Datasets

We evaluate on validation splits of PASCAL VOC (VOC), Pascal Context (Context) and COCO-Object (Object) datasets. PASCAL VOC [9, 10] has 21 classes: 20 foreground plus a background class. For Pascal Context [29], we use the common variant with 59 foreground classes and 1 background class. It contains both "things" and "stuff" classes. The COCO-Object is a variant of COCO-Stuff [3] with 80 "thing" classes and one class for the background. Textual class names are used as natural language specifications of names. We renamed or specified certain class names to fix errors (*e.g.* `pottedplant` → `potted plant`), resolve ambiguity better (*e.g.* `mouse` → `computer mouse`) or change to more common spelling/word (*e.g.* `aeroplane` → `airplane`), resulting in 14 fixes. We experiment and measure the impact of this in Appendix A.1 for our and prior work.

[5]We use implementation from https://github.com/huggingface/diffusers.

C.4. Comparative baselines

We briefly review the prior work in used in our experiments, mainly in Table 1. We consider baselines that do not rely on mask annotations and have code and checkpoints available or detail their evaluation protocol that matches that used in other prior works [5, 46, 47].Most prior work [5, 25, 27, 34, 46, 47] trains image and text encoders on large image-text datasets with a contrastive loss. The methods mainly differ in their architecture and use of grouping mechanisms to ground image-level text on regions. ViL-Seg [25] uses online clustering, GroupViT [46] and ViewCo [34] employ group tokens. OVSegmentor [47] uses slot-attention and SegCLIP [27] a grouping mechanism with learnable centers. CLIPPy [33], TCL [5], and MaskCLIP [52] predict classes for each image patch: [33] use max-pooling aggregation, [5] self-masking, and [52] modify CLIP for dense predictions. To assign a background label [5, 25, 27, 34, 46] use thresholding while [33] uses dataset-specific prompts. CLIP-DIY [44] leverages CLIP as a zero-shot classifier and applies it on multiple scales to form a dense segmentation. ReCO [38] is closer in spirit to our approach as it uses a support set for each prompt; this set, however, is CLIP-retrieved from curated image collections, which may not be applicable for any category in-the-wild.

We also note that prior work builds on top of similar pre-trained components such as CLIP in [5, 27, 38, 52], OpenCLIP in [44], DINO + T5/RoBERTa in [33, 47]. We additionally make use of StableDiffusion, which is trained on a larger dataset (3B, compared to 400M of CLIP or 2B or OpenCLIP). OVDiff is, however, fundamentally different to all prior work, as (a) it generates a support set of synthetic images given a class description, and (b) it does not rely on additional training data and further training for learning to segment.

C.5. Hyperparameters

OVDiff has relatively few hyperparameters and we use the same set in all experiments. Similar to [5, 46, 47], we employ a sliding window approach. We use two scales to aid with the limited resolution of off-the-shelf feature extractors with square window sizes of 448 and 336 and a stride of 224 pixels. We set the size of the support set to $N = 32$. For the diffusion model, we use Stable Diffusion v1.5; for unsupervised segmenter Γ, we employ CutLER [42]. Unless otherwise specified, $N = 32$ images are sampled using classifier-free guidance scale [16] of 8.0 and 30 denoising steps. We employ `DPM-Solver` scheduler [26]. When sampling images for the support sets, we also use a negative prompt "*text, low quality, blurry, cartoon, meme, low resolution, bad, poor, faded*". If/when segmenter Γ fails to extract any components in a sampled image, a fallback of adaptive thresholding of A_n is used, following [24]. During inference, we set $\eta = 10$, which results in 1024 text prompts

processed in parallel, a choice made mainly due to computational constraints. We set the thresholds for the "stuff" filter between background prototypes for "things" classes and the foreground of "stuff" at 0.85 for all feature extractors. When sampling, a seed is set for each category individually to aid reproducibility. With our unoptimized implementation, we measure around 110 ± 10s to calculate prototypes (sample images, extract features and aggregate) for a single category or 50.2 ± 2s without clustering using SD. Using CLIP, we measure 49.2 ± 0.2s with clustering and 47.7 ± 0.2s without. We note that sampling time grows linearly: we measure 55s for 16, 110s for 32, and 213s for 64 images per class. The prototype storage requirements are 0.39MB using CLIP/DINO for each class.

We additionally measure the speed of inference at 0.6s per image, which is slightly slower but comparable to 0.2s for TCL and 0.08s for OVSegmentor. We performed inference measurements using SD on the same machine with a 2080Ti GPU using 21 classes and the same resolution/sliding window settings for all methods.

C.6. Interaction with ChatGPT

We interact with ChatGPT to categorise classes into "stuff" and "things" for the stuff filter component. Due to input limits, the categories are processed in blocks. Specifically, we input "*In semantic segmentation, there are "stuff" or "thing" classes. Please indicate whether the following class prompts should be considered "stuff" or "things":*". We show the output in Tab. C.9. Note there are several errors in the response, *e.g.* `glass`, `blanket`, and `trade name` are actually instances of tableware, bedding and signage, respectively, so should more appropriately be treated as "things". Similarly, `land` and `sand` might be more appropriately handled as "stuff", same as `snow` and `ground`. Despite this, We find ChatGPT contains sufficient knowledge when prompted with "in semantic segmentation". We have estimated the accuracy of ChatGPT in thing/stuff classification using the categories of COCO-Stuff, which are defined as 80 "things" and 91 "stuff" categories. ChatGPT achieves an accuracy rate of 88.9% in this case. We also measure the impact the potential errors have on our performance by providing "oracle" answers on the Context dataset. We measure 29.6 mIoU, which is similar to 29.7 ± 0.3 of using ChatGPT, showing that small errors do not drastically affect the method, however, enable using "stuff" filter component, which improves performance (see Table 3).

Table C.9. **Response from interaction with ChatGPT.** We used ChatGPT model to automatically categorise classes in "stuff" or "things".

airplane:	thing	window:	thing	awning:	thing
bag:	thing	wood:	stuff	streetlight:	thing
bed:	thing	windowpane:	thing	booth:	thing
bedclothes:	stuff	earth:	thing	television receiver:	thing
bench:	thing	painting:	thing	dirt track:	thing
bicycle:	thing	shelf:	thing	apparel:	thing
bird:	thing	house:	thing	pole:	thing
boat:	thing	sea:	thing	land:	thing
book:	thing	mirror:	thing	bannister:	thing
bottle:	thing	rug:	thing	escalator:	thing
building:	thing	field:	thing	ottoman:	thing
bus:	thing	armchair:	thing	buffet:	thing
cabinet:	thing	seat:	thing	poster:	thing
car:	thing	desk:	thing	stage:	thing
cat:	thing	wardrobe:	thing	van:	thing
ceiling:	stuff	lamp:	thing	ship:	thing
chair:	thing	bathtub:	thing	fountain:	thing
cloth:	stuff	railing:	thing	conveyer belt:	thing
computer:	thing	cushion:	thing	canopy:	thing
cow:	thing	base:	thing	washer:	thing
cup:	thing	box:	thing	plaything:	thing
curtain:	stuff	column:	thing	swimming pool:	thing
dog:	thing	signboard:	thing	stool:	thing
door:	thing	chest of drawers:	thing	barrel:	thing
fence:	stuff	counter:	thing	basket:	thing
floor:	stuff	sand:	thing	waterfall:	thing
flower:	thing	sink:	thing	tent:	thing
food:	thing	skyscraper:	thing	minibike:	thing
grass:	stuff	fireplace:	thing	cradle:	thing
ground:	stuff	refrigerator:	thing	oven:	thing
horse:	thing	grandstand:	thing	ball:	thing
keyboard:	thing	path:	thing	step:	stuff
light:	thing	stairs:	thing	tank:	thing
motorbike:	thing	runway:	thing	trade name:	stuff
mountain:	stuff	case:	thing	microwave:	thing
mouse:	thing	pool table:	thing	pot:	thing
person:	thing	pillow:	thing	animal:	thing
plate:	thing	screen door:	thing	lake:	stuff
platform:	stuff	stairway:	thing	dishwasher:	thing
plant:	thing	river:	thing	screen:	thing
road:	stuff	bridge:	thing	blanket:	stuff
rock:	stuff	bookcase:	thing	sculpture:	thing
sheep:	thing	blind:	thing	hood:	thing
shelves:	thing	coffee table:	thing	sconce:	thing
sidewalk:	stuff	toilet:	thing	vase:	thing
sign:	thing	hill:	thing	traffic light:	thing
sky:	stuff	countertop:	thing	tray:	stuff
snow:	stuff	stove:	thing	ashcan:	thing
sofa:	thing	palm:	thing	fan:	thing
table:	thing	kitchen island:	thing	pier:	thing
track:	stuff	swivel chair:	thing	crt screen:	thing
train:	thing	bar:	thing	bulletin board:	thing
tree:	thing	arcade machine:	thing	shower:	thing
truck:	thing	hovel:	thing	radiator:	thing
monitor:	thing	towel:	thing	glass:	stuff
wall:	stuff	tower:	thing	clock:	thing
water:	stuff	chandelier:	thing	flag:	thing