9. Appendix

9.1. Stable Diffusion Architecture

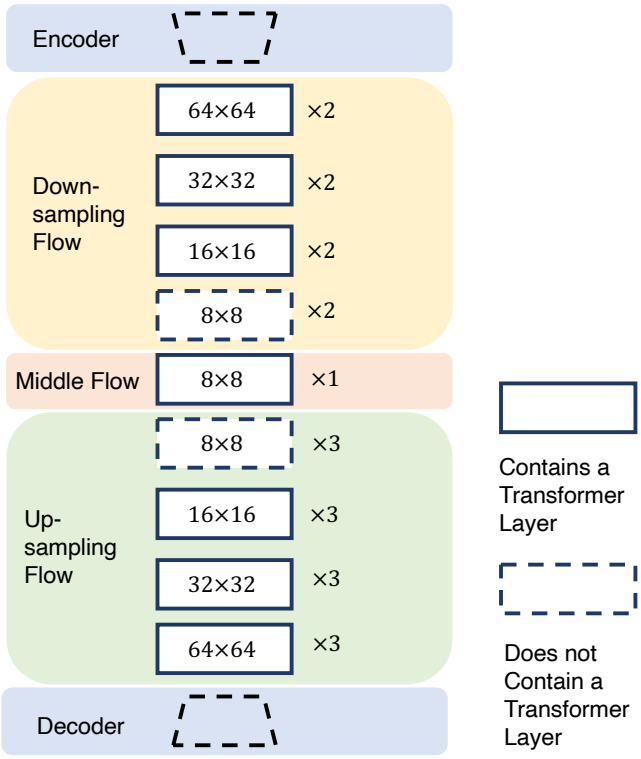


Figure 10. Stable Diffusion Model Schematics. There are in total 16 blocks having Transformer Layers. Each generates a 4D self-attention tensor of different resolutions.

9.2. Discussion on Averaging Multi-head Attentions

In Sec. 3.1, we mentioned that DiffSeg averages the multi-head attention along the multi-head axis, reducing a 5D tensor to 4D. This is motivated by the fact that different attention heads capture very similar correlations. To show this, we calculate the average KL distance across the multi-head axis for all attention tensors of different resolutions used for segmentation. We use 180 training data to calculate the following statistics in Tab. 4. The difference across

	COCO	Cityscapes
Avg. multi-heads KL	0.46	0.45
Merge Threshold KL (τ)	1.10	0.90

Table 4. Average KL distance across the multi-head axis and the merge threshold used in the paper.

the multi-head channels is much smaller than the KL merge threshold used in the paper. This means that the attentions are very similar across the channels and they would have been merged together if not averaged at first because their distance is well below the merge threshold.

9.3. Iterative Attention Aggregation

Algorithm 1 Iterative Attention Merging

Require: $\mathcal{L}_a, \mathcal{A}_f, N, \tau$

$\mathcal{L}_p = \left\{ \frac{1}{|\mathcal{V}|} \sum_{(i,j)\in\mathcal{V}} \mathcal{A}_f[i,j] | v = 1, ..., M^2 \right\}$
where $\mathcal{V} = \{(i,j) | \mathcal{D}\left(\mathcal{L}_a[v], \mathcal{A}_f[i,j]\right) < \tau\}$

for $N-1$ iterations **do**
  Initialize $\tilde{\mathcal{L}}_p = []$
  for $\mathcal{A}$ in $\mathcal{L}_p$ **do**
    $V = \frac{1}{|\mathcal{V}|} \sum_{v\in\mathcal{V}} \mathcal{L}_p[v]$ ▷ Merge attention maps
    where $\mathcal{V} = \{v | \mathcal{D}\left(\mathcal{A}, \mathcal{L}_p[v]\right) < \tau\}$
    Add V to $\tilde{\mathcal{L}}_p$
    Remove $\mathcal{L}_p[v] \quad \forall v \in \mathcal{V}$ from $\mathcal{L}_p$
  end for
  $\mathcal{L}_p \leftarrow \tilde{\mathcal{L}}_p$
end for

9.4. Comparisons to K-Means and DBSCAN Baselines

DiffSeg uses an iterative merging process to generate the segmentation mask to avoid two major limitations of popular k-means clustering-based algorithms: 1) K-means needs specification of the number of clusters and 2) K-means is stochastic depending on initialization of cluster initialization. In this section, we present a comparison between DiffSeg and a K-means baseline. Specifically, after the *Attention Aggregation* stage, we obtain a 4D attention tensor $\mathcal{A}_f \in \mathbb{R}^{64^4}$. Instead of using iterative merging as in DiffSeg, we directly apply the K-means algorithm on tensor $\mathcal{A}_f$. To do this, we reshape the tensor to 4096×4096, which represents 4096 vectors of dimension 4096. The goal is to cluster the 4096 vectors into N clusters.

We use the Sklearn k-means implementation [2] with k-means++ initialization [5]. We present results for the number of clusters using a constant number averaged over all evaluated images and a specific number for each image. The average number and specific number are obtained from the ground truth segmentation masks. We use the COCO-Stuff dataset as the benchmark.

DBSCAN [13] is a classic density-based clustering algorithm that does not require the number of clusters as input. We use the Sklearn DBSCAN implementation [1] with the default Euclidean metric. We sweep eps, which determines the maximum distance for a pair of samples to be considered as neighbors. The same as the K-means implementation, we directly apply DBSCAN to $\mathcal{A}_f$.

9.5. DinoSeg

DiffSeg is a generic unsupervised clustering algorithm theoretically applicable to any Transfer features backbone. In this section, we adopt DiffSeg to a DINO backbone [9]. Specifically, we use a DINO-Base backbone with a patch size of 8. DINO takes images at a resolution of 224.

The pipeline is similar to that of DiffSeg, the only difference being the attention aggregation. DINO produces attention tensors of the *same* spatial size 28×28 for all its layers. Therefore, aggregation simply averages over all 12 layers and 12 multi-head channels.

As shown in Tab. 1, the performance of DinoSeg is worse than that of DiffSeg. There can be multiple reasons. 1) DINO is pre-trained on ImageNet1K, which is much smaller than LAION5B [35]. 2) DINO's input images are of lower resolution. 3) The spatial dimension of DINO's attention outputs is only 28×28, which is smaller than the $64\times$ resolution found in Stable Diffusion.

9.6. Additional Ablation study

Most of them have a reasonable range that works well for general settings. Therefore, we do not tune them for each dataset and model. One exception is the KL threshold parameter τ. It is the most sensitive parameter as it directly controls the attention-merging process. We tune this parameter in our experiments on a small subset (180 images) from the training set from the respective datasets. All sensitivity study experiments are also conducted on the respective subsets.

Time step (t)**.** The stable diffusion model requires a time step t to indicate the current stage of the diffusion process. Because DiffSeg only runs a single pass through the diffusion process, the time step becomes a hyper-parameter. In Fig. 11a, we demonstrate the effects of setting this parameter to different numbers $t \in \{1, 100, 200, 300, 400, 500\}$ while keeping the other hyper-parameters constant ($R =$ propto., $M^2 = 256, N = 3, \tau = 1.0$). In the figure, we observe a general upward trend for accuracy and mIoU when increasing the time step, which peaked around $t = 300$. Therefore, we use $t = 300$ for our main results.

Number of anchors (M^2)**.** DiffSeg generates a sampling grid of M^2 anchors to start off the attention-merging process. In Fig. 11b, we show the number of proposals and accuracy with different numbers of anchor points $M \in \{4, 8, 16, 32\}$ while keeping the other hyper-parameters constant ($R =$ propto., $t = 100, N = 3, \tau = 1.0$). We observe that the number of anchor points does not significantly affect the performance of COCO-Stuff-27. Therefore, we keep the default $M = 16$.

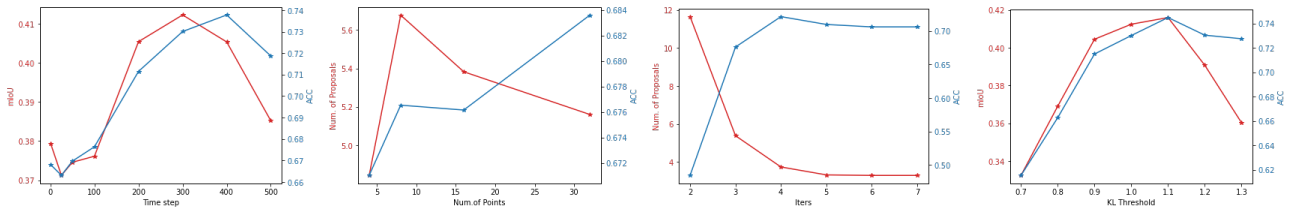
Number of merging iterations (N)**.** The iterative attention merging process runs for N iterations. Intuitively, the more iterations, the more proposals will be merged. In Fig. 11c, we show the effects of increasing the number of iterations $N \in \{2, 3, 4, 5, 6, 7\}$ in terms of the number of final objects and accuracy while keeping the other hyper-parameters constant ($R =$ propto., $M^2 = 256, t = 100, \tau = 1.0$). We observe that at the 3rd iteration, the number of proposals drops to a reasonable amount and the accuracy remains similar afterward. Therefore, we use $K = 3$ for a better system latency and performance trade-off.

KL threshold (τ)**.** The iterative attention merging process also requires specifying the KL threshold τ. It is arguably the most sensitive hyper-parameter and should be tuned preferably separately for each dataset. Too small a threshold leads to too many proposals and too large leads to too few proposals. In Fig. 11d, we show the effect of $\tau \in \{0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$ while using the validated values for the other hyper-parameters ($R =$ propto., $M^2 = 256, t = 100, N = 3$). We observe that a range $\tau \in [0.9, 1.1]$ should yield reasonable performance. We select $\tau = 1.1$ for COCO-Stuff-27 and $\tau = 0.9$ for Cityscapes, identified using the same procedure.

A Note on the Hyper-Parameters. We found that the same set of (R, t, M, N) works generally well for different settings. The only more sensitive parameter is τ. A reasonable range for τ is between 0.9 and 1.1. We would suggest using the default $\tau = 1.0$ for the segmentation of images in the wild. For the best benchmark results, proper hyper-parameter selection is preferred.

9.7. Limitations

While the zero-shot capability of DiffSeg enables it to segment almost any image, thanks to the generalization capability of the stable diffusion backbone, its performance on more specialized datasets such as self-driving datasets, e.g., Cityscapes, is far from satisfaction. There are several potential reasons. 1) The resolution of the largest attention map is 64×64, which can be too small to segment small objects in a self-driving dataset. 2) Stable diffusion models have limited exposure to self-driving scenes. The performance of zero-shot methods largely relies on the generalization capability of the pre-trained model. If during the pre-training stage, the stable diffusion model has not been exposed to a vehicle-centric self-driving scene, it could negatively affect the downstream performance. 3) While DiffSeg is much simpler than competing methods such as ReCo [36], which requires image retrieval and co-segmentation of a batch of images, it is still not a real-time algorithm. This is because the current stable diffusion model is very large, even though DiffSeg only runs the diffusion step once. Also, the attention aggregation and merging process are iterative, which incurs higher computation costs for CPU and GPU.



(a) Time Step t (b) Number of Anchors M^2 (c) Number of Merging Iters. (N) (d) KL threshold τ

Figure 11. Effects of the Time Step t, the Number of Anchors M^2, the Number of Merging Iterations (N) and the KL Threshold τ on COCO-Suff-27. The same set of (R, t, M, N) works generally well for different settings (Tab. 3). A reasonable range for τ is $0.9 \sim 1.1$.

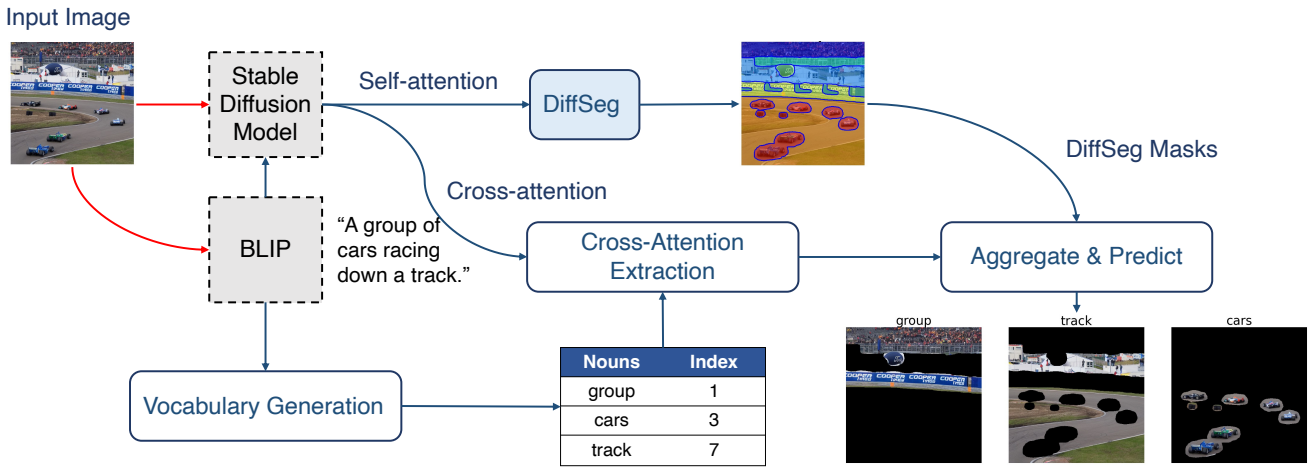


Figure 12. Overview of Semantic DiffSeg. Semantic DiffSeg extends DiffSeg to add labels to generated masks. It has three major additional components Vocabulary Generation, Cross-Attention Extraction and Aggregate & Predict.

9.8. Adding Semantics

While DiffSeg generates high-quality segmentation masks, like SAM [20], it does not provide labels for each mask. Inspired by recent open-vocabulary semantic segmentation methods [25, 41], which use an off-the-shelf image captioning model to generate vocabularies for an image and DiffuMask [39], which demonstrates the grounding capability of the *cross-attention* layers in a diffusion model, we propose a simple extension to DiffSeg to produce labeled segmentation masks. We refer to the extended version as Semantic DiffSeg. Specifically, Semantic DiffSeg has three additional components: vocabulary generation, cross-attention extraction, and aggregation & prediction as shown in Fig. 12.

Vocabulary Generation. Following existing works in open-vocabulary [25, 41] segmentation, we use an off-the-shelf image captioning model, e.g., BLIP [23] to generate a caption for a given image. Denote $\mathcal{W} = \{w_1, w_2, ..., w_K\}$ as the caption where each w represents a word in the caption. To construct the output label space, we extract all *nouns* from the caption using the NLTK toolkit [28]. Importantly, we also keep track of the *relative position* of the noun word in the caption for indexing purposes. Formally, let $\mathcal{N} = \{n_1, n_2, ...n_L\}$ denote the set of extracted nouns from $\mathcal{W}$ and $\mathcal{I} = \{i_1, i_2, ..., i_L\}$ denote the set of indices of the relative position corresponding to each noun in the original caption $\mathcal{W}$.

Cross-Attention Extraction. Unlike the vanilla DiffSeg, which uses the *unconditional* generation capability of stable diffusion models for segmentation without a prompt input, Semantic DiffSeg adds the generated caption $\mathcal{W}$ as another input to extract meaningful grounded cross-attention maps. DiffuMask [39] shows that cross-attention maps corresponding to noun tokens provide grounding for their respective concepts. Inspired by this finding, we also extract the corresponding cross-attention maps from different resolutions using the indices $\mathcal{I}$. Similar to the self-attention formulation in Eq. 1, there are a total 16-cross attention layers in a stable diffusion model, giving 16 *cross-attention tensor*:

$$\mathcal{A}_\mathcal{C} \in \{\mathcal{A}_c \in \mathbb{R}^{h_c \times w_c \times Q} | c = 1, ..., 16\}. \quad (7)$$

where Q is the number of tokenized words in the input caption sequence, e.g., $Q = 77$ for stable diffusion

models. Unlike the self-attention tensor, the cross-attention tensor is a 3-dimensional tensor, where each $\mathcal{A}_c[:,:,q] \in \mathbb{R}^{h_c \times w_c}, \forall q \in \{1,..,Q\}$ is the *un-normalized* attention map w.r.t the token q. We now extract cross-attention maps corresponding to only the nouns in our caption using our index set $\mathcal{I}$. Note that the caption length K is usually smaller than the token sequence length Q. Stable diffusion models use BOS and EOS paddings to unify all input lengths to Q. Therefore, in our case, the correct index of a noun word w.r.t the padded tokens is its index $i+1$ resulting from the BOS token offset. Finally, we obtain a cross-attention tensor $\mathcal{A}_\mathcal{N}$ corresponding only to the noun tokens.

$$\mathcal{A}_\mathcal{N} \in \{\mathcal{A}_n \in \mathbb{R}^{h_n \times w_n \times L} | n = 1, ..., 16\}. \tag{8}$$

Aggregation and Prediction. Similar to the self-attention maps, the cross-attention maps in $\mathcal{A}_\mathcal{N}$ are of different resolutions. To aggregate them, we upsample the *first* 2 dimensions of each map to 512.

$$\tilde{\mathcal{A}}_n = \text{Bilinear-upsample}(\mathcal{A}_n) \in \mathbb{R}^{512 \times 512 \times L}. \tag{9}$$

Note this is slightly different from the upsampling of the self-attention maps in Eq. 10, which upsamples the *last* 2 dimensions. Finally, we *sum and normalize* all cross-attention maps to obtain the aggregated cross-attention tensor $\mathcal{A}_{nf} \in \mathbb{R}^{512 \times 512 \times L}$. Specifically, the aggregated cross-attention map $\mathcal{A}^l_{nf} \in \mathbb{R}^{512 \times 512}$ corresponding to token l is

$$\mathcal{A}^l_{nf} = \frac{\sum_{n=1}^{16} \tilde{\mathcal{A}}_n[:,:,l]}{\sum_{w=1}^{512}\sum_{h=1}^{512}\sum_{n=1}^{16} \tilde{\mathcal{A}}_n[w,h,l]}. \tag{10}$$

To obtain the final labeled masks, we combine $\mathcal{A}_{nf}$ with the output from DiffSeg $\bar{\mathcal{L}}_p \in \mathbb{R}^{N_p \times 512 \times 512}$ in Eq. 6, where N_p denotes the number of generated masks and each $\bar{\mathcal{L}}_p[n_p,:,:] \in \mathbb{R}^{512 \times 512}$ is a binary mask. Specifically, for each mask $\bar{\mathcal{L}}_p[n_p,:,:], n_p \in \{1,...,N_p\}$, we calculate the prediction vector $l_{n_p} \in \mathbb{R}^L$. Each element $l^i_{n_p}$ in l_{n_p} is calculated as

$$l^i_{n_p} = \bar{\mathcal{L}}_p[n_p,:,:] \circledast \mathcal{A}_{nf}[:,:,i], \tag{11}$$

where $\circledast$ denotes the element-wise product. Finally, the label for mask n_p is obtained by taking the maximum element in the prediction vector l_{n_p}. As a post-processing step, we merge all masks with the same label into a single mask.

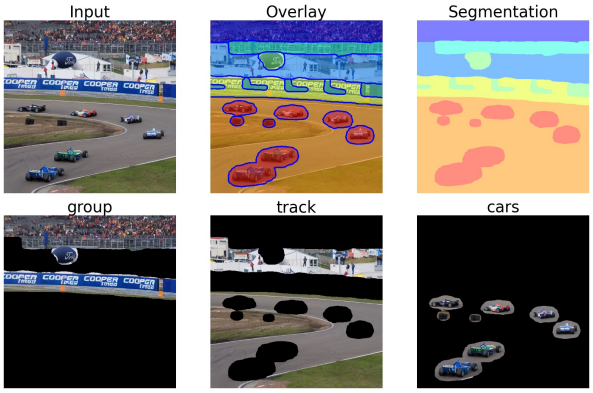


Figure 13. Semantic DiffSeg Example. First row: DiffSeg output. Second row: semantic segmentation with mask merging. Generated caption: A group of cars racing down a track.

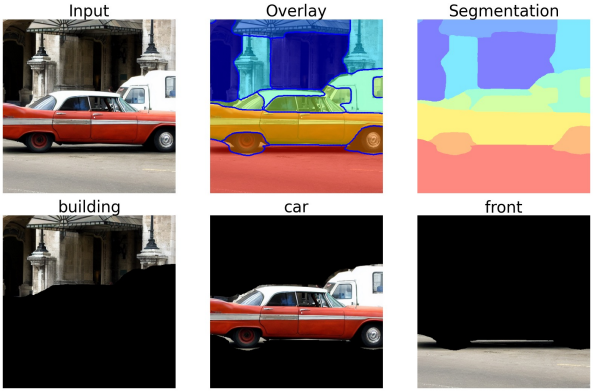


Figure 14. Semantic DiffSeg Example. First row: DiffSeg output. Second row: semantic segmentation with mask merging. Generated caption: A red car parked in front of a building.

9.9. Additional Visualization

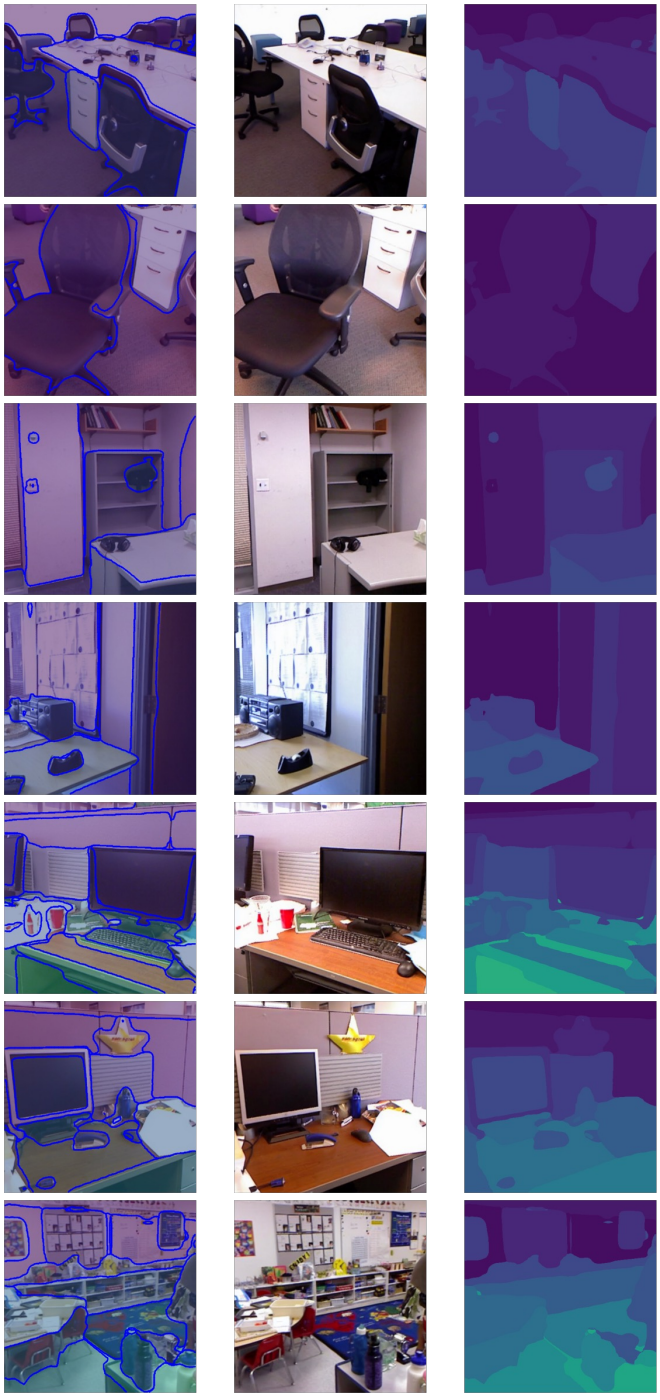


Figure 15. Examples of Segmentation on SUN-RGBD Images. Overlay (left), Input (middle), and segmentation (right)

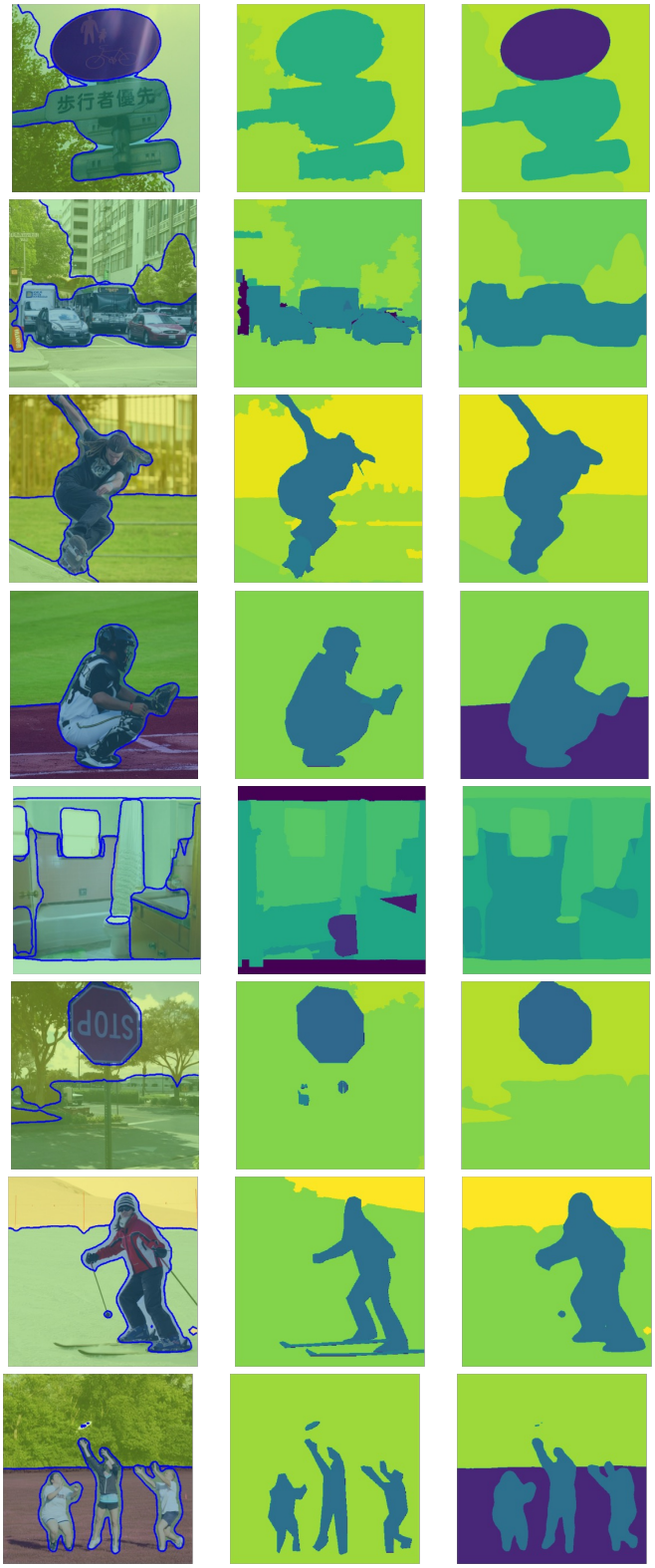


Figure 16. Examples of Segmentation on COCO-Stuff-27. Overlay (left), ground truth (middle), and segmentation (right)

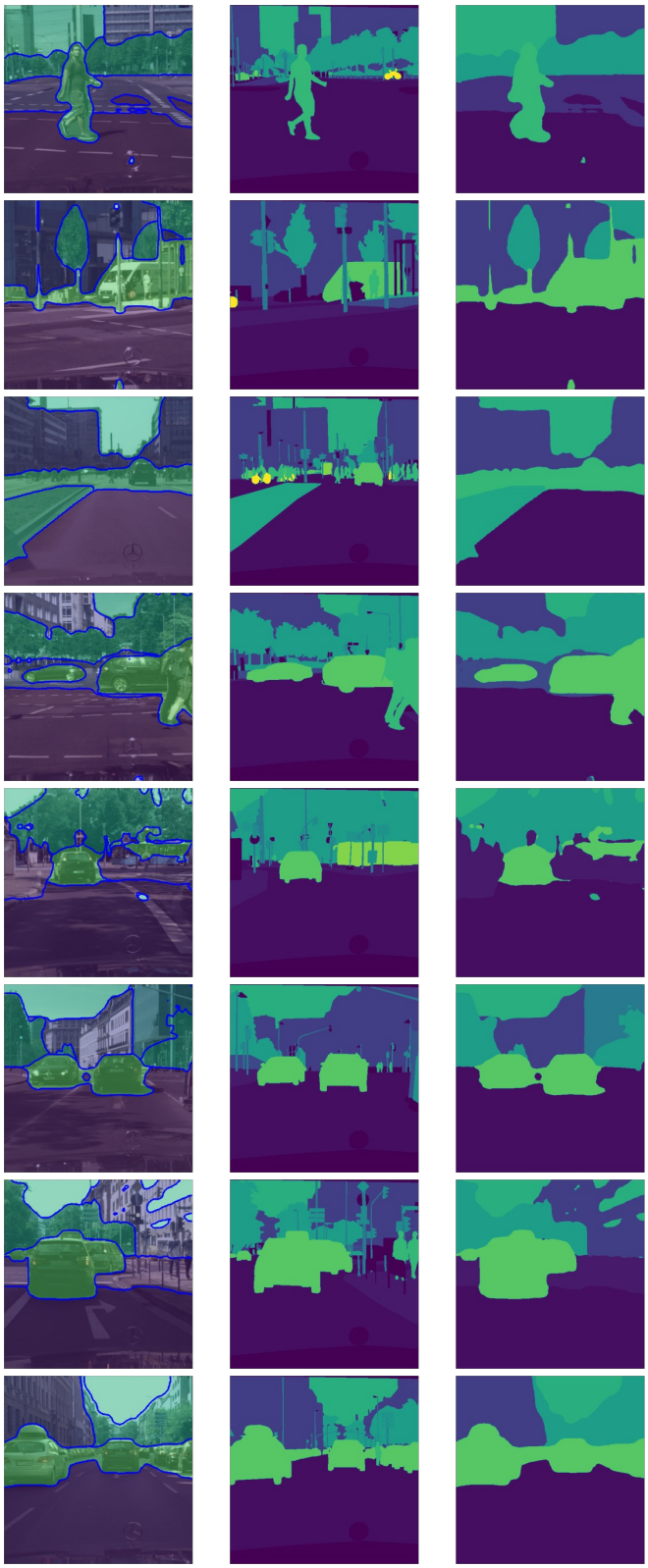


Figure 17. Examples of Segmentation on Cityscapes. Overlay (left), ground truth (middle), and segmentation (right)

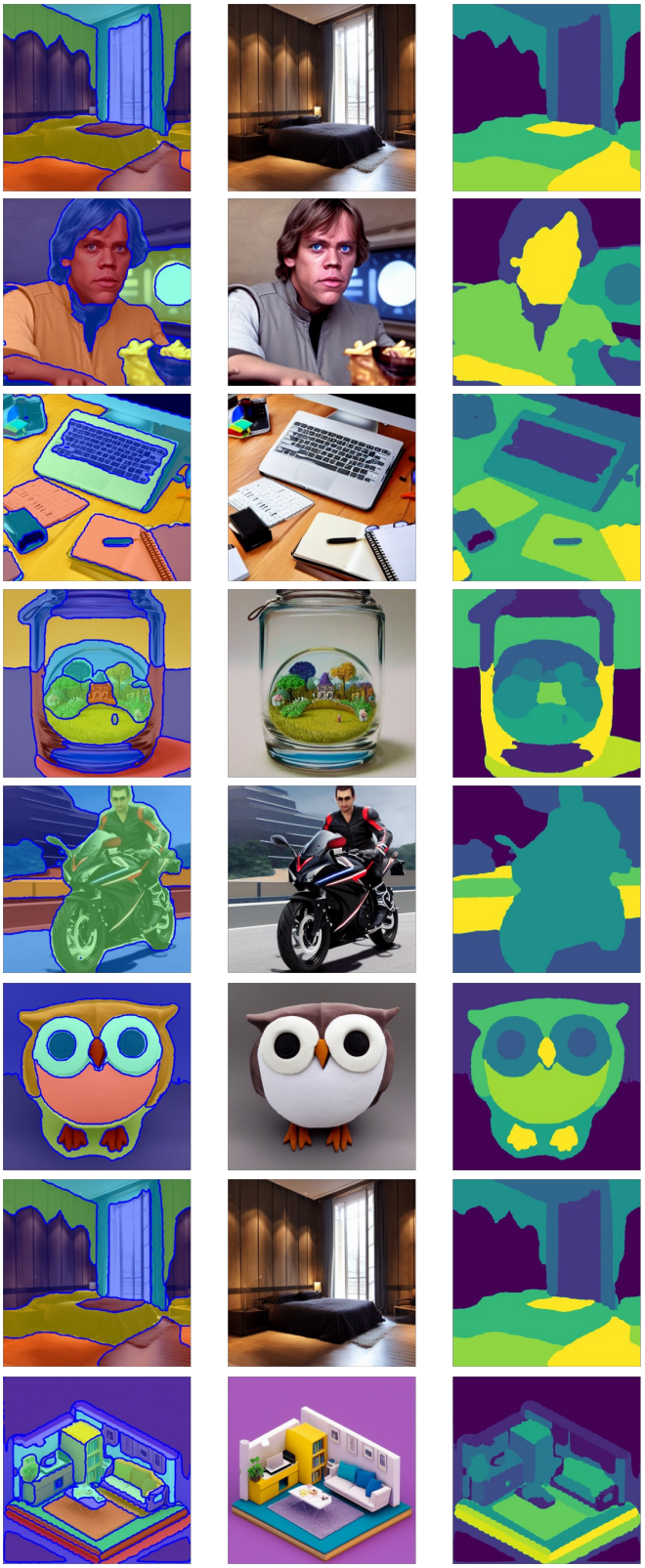


Figure 18. Examples of Segmentation on Synthetic Images (generated by a stable diffusion model). Overlay (left), Input (middle), and segmentation (right)