Enhancing Object Detection Training via Joint Image-Annotation Generation - Supplementary Material

Roy Uziel and Oded Bialer
General Motors, Technical Center Israel
uzielroy@gmail.com, oded.bialer8@gmail.com

1. Additional qualitative results

Figs. 6-16 provide additional qualitative examples from the COCO dataset, complementing those presented in the main paper. These figures compare the performance of *JointDiffuse*, which jointly generates images and segmentation maps, with *CondDiffuse*, a method that generates images conditioned on annotations. Both approaches were trained on the COCO dataset. The examples are arranged in rows. Each row begins with the original real image and its associated text caption on the right, followed by three images generated by *CondDiffuse*, and then three images generated by *JointDiffuse*. For each generated image, the corresponding class segmentation map is displayed below, with the instance segmentation map shown directly beneath it.

The examples in Figs. 6-13 illustrate the enhanced diversity achieved by *JointDiffuse* compared to *CondDiffuse*. *CondDiffuse* modifies the appearance of objects but preserves the original scene layout from the image annotations, whereas *JointDiffuse* generates a wider variety of scenarios based on the same text caption descriptions.

Figs. 14-16 demonstrates the superior alignment achieved by *JointDiffuse* between the generated annotations and the objects in the images. In Figure 14, two of the images produced by *CondDiffuse* include an additional ball that is missing from the segmentation annotations. In Figure 15, two images generated by *CondDiffuse* lack a motorcycle driver present in the segmentation annotations. Lastly, in Figure 16, the red ball appears in the image generated by *CondDiffuse* but not in the annotations. These misalignments between image and annotations are not observed in *JointDiffuse* outputs.

Figs. 17–21 present qualitative comparisons between *JointDiffuse* and *CondDiffuse*, both trained on the nuImages dataset. Each figure shows three images generated by *CondDiffuse*, followed by three from *JointDiffuse*. Below each image, the class segmentation map appears above the instance segmentation map. The results demonstrate that *JointDiffuse* produces images with varying object counts and layouts, resulting in a diverse range of automotive scenarios. In contrast, images generated by *CondDiffuse* depict the same scenario with only appearance variations.

2. Further details on *CondDiffuse*

To effectively evaluate the performance improvement of *JointDiffuse*, which simultaneously generates images and annotations, compared to image generation conditioned on annotations, we adapt the *JointDiffuse* framework into an annotation-conditioned generation variant called *CondDiffuse*. This modification involves minimal changes, ensuring that the primary distinction lies in simultaneous generation versus annotation-conditioned generation. *CondDiffuse* retains the same annotation representation as *JointDiffuse* and features a similar architecture, as illustrated in Figure 1.

CondDiffuse includes only an image denoiser that takes annotations as input, represented through instance and class segmentation maps. The training process for *CondDiffuse* is illustrated in Figure 1(a). The clean image and segmentation latents, $z_0^{\mathcal{I}}$, and $z_0^{\mathcal{S}}$, are identical to those in *JointDiffuse*. However, the forward diffusion process is applied exclusively to the image latent, while the segmentation latent remains clean and is integrated into the image denoiser via a segmentation encoder. This encoder is the same encoder network part that is used in the segmentation denoiser of *JointDiffuse*.

The architecture of the image denoiser and segmentation encoder is illustrated in Figure 2. Similar to the joint denoiser architecture in *JointDiffuse*, at each layer, the features from the segmentation encoder, $\gamma_i^{\mathcal{S}}$, are processed through three ResNet layers before being added to the corresponding features in the image denoiser, $\gamma_i^{\mathcal{I}}$. Notably, unlike *JointDiffuse*, there is no information flow from the image denoiser to the segmentation encoder. Both the segmentation encoder and the image denoiser are trained via backpropagation to minimize the image denoiser loss, defined as:

$$\mathcal{L} = \mathbb{E}\Big\{\|\epsilon^{\mathcal{I}} - \epsilon_\theta^{\mathcal{I}}(z_t^{\mathcal{I}}, \Gamma_t^{\mathcal{S}}, C, t)\|_2^2\Big\}, \quad (1)$$

using the same notations as described in the main paper.

Figure 1(b) illustrates the generation process of *CondDiffuse*. During this process, the image denoiser and segmentation encoder remain fixed. Generation begins with a random image latent $z_T^{\mathcal{I}}$ and a clean segmentation latent $z_0^{\mathcal{S}}$. The segmentation latent is incorporated into the image denoiser through the segmentation encoder, following the same method used during training (see Figure 2). The image denoiser estimates noise by utilizing the segmentation information and the text caption embedding, i.e., conditioned on them. A new image latent is then sampled from the estimated noise using Eqs. (3) and (5) in the main paper. This reverse diffusion process iteratively samples the image latent $z_t^{\mathcal{I}}$ from time step $t = T$ to $t = 0$, with the clean segmentation latent $z_0^{\mathcal{S}}$ fixed as input throughout all sampling steps. The final image latent $z_0^{\mathcal{I}}$ is passed through the VAE decoder to generate the output image.

The annotations corresponding to the generated image are identical to those used to create the latent segmentation $z_0^{\mathcal{S}}$. This noiseless latent was provided as input to the image denoiser, making it possible for the entire generation process to be effectively conditioned on these annotations.

3. Derivation of the *JointDiffuse* loss function

In this section, we derive the *JointDiffuse* loss function presented in Eq. (2) of the main paper, utilizing the notations provided therein. The derivation builds upon the independence of the additive noise terms applied to the image and segmentation during the forward diffusion process. Consequently, the reverse joint conditional distribution of the image and segmentation latents can be expressed as:

$$q(z_{t-1}^{\mathcal{I}}, z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{I}}, z_t^{\mathcal{S}}, z_0^{\mathcal{I}}, z_0^{\mathcal{S}}) = q(z_{t-1}^{\mathcal{I}} | z_t^{\mathcal{I}}, z_0^{\mathcal{I}}) q(z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{S}}, z_0^{\mathcal{S}}). \quad (2)$$

The denoiser network is employed to approximate the conditional joint probability distribution during the reverse diffusion process. Since the noise in the image and segmentation latents is independent, this joint distribution can be expressed as:

$$p_{\theta,\phi}(z_{t-1}^{\mathcal{I}}, z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{I}}, z_t^{\mathcal{S}}) = p_\theta(z_{t-1}^{\mathcal{I}} | z_t^{\mathcal{I}}) p_\phi(z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{S}}). \quad (3)$$

We minimize the negative log-likelihood of the joint distribution of the image and segmentation latents using the variational lower bound (VLB), formulated as:

$$VLB = \mathbb{E}_{q(z_{0:T}^{\mathcal{I}}, z_{0:T}^{\mathcal{S}})} \left[\log \left\{ \frac{q(z_{1:T}^{\mathcal{I}}, z_{1:T}^{\mathcal{S}} | z_0^{\mathcal{I}}, z_0^{\mathcal{S}})}{p_{\theta,\phi}(z_{0:T}^{\mathcal{I}}, z_{0:T}^{\mathcal{S}})} \right\} \right] \geq - \mathbb{E}_{q(z_0^{\mathcal{I}}, z_0^{\mathcal{S}})} \left[\log \left\{ p_{\theta,\phi}(z_0^{\mathcal{I}}, z_0^{\mathcal{S}}) \right\} \right], \quad (4)$$

where

$$q(z_{1:T}^{\mathcal{I}}, z_{1:T}^{\mathcal{S}} | z_0^{\mathcal{I}}, z_0^{\mathcal{S}}) = \prod_{t=1}^{T} q(z_t^{\mathcal{I}}, z_t^{\mathcal{S}} | z_{t-1}^{\mathcal{I}}, z_{t-1}^{\mathcal{S}}), \quad (5)$$

$$p_{\theta,\phi}(z_{0:T}^{\mathcal{I}}, z_{0:T}^{\mathcal{S}}) = p_{\theta,\phi}(z_T^{\mathcal{I}}, z_T^{\mathcal{S}}) \prod_{t=1}^{T} p_{\theta,\phi}(z_{t-1}^{\mathcal{I}}, z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{I}}, z_t^{\mathcal{S}}). \quad (6)$$

Following [2, 9], the VLB is approximated using the KL-divergence between the reverse distribution functions given in Eqs. (2) and (3), as follows:

$$\begin{aligned} & VLB \approx \\ & \mathcal{D}_{KL}\Big(q(z_{t-1}^{\mathcal{I}}, z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{I}}, z_t^{\mathcal{S}}, z_0^{\mathcal{I}}, z_0^{\mathcal{S}}) \| p_{\theta,\phi}(z_{t-1}^{\mathcal{I}}, z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{I}}, z_t^{\mathcal{S}})\Big) \\ & \quad = \mathcal{D}_{KL}\Big(q(z_{t-1}^{\mathcal{I}} | z_t^{\mathcal{I}}, z_0^{\mathcal{I}}) \| p_\theta(z_{t-1}^{\mathcal{I}} | z_t^{\mathcal{I}})\Big) + \\ & \quad \mathcal{D}_{KL}\Big(q(z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{S}}, z_0^{\mathcal{S}}) \| p_\phi(z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{S}})\Big). \end{aligned} \quad (7)$$

Using the derivation from [2, 9], we find:

$$\begin{aligned} & q(z_{t-1}^{*} | z_t^{*}, z_0^{*}) = \\ & \mathcal{N}\left(\alpha_t^{-0.5} \left(z_t^{*} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon^{*} \right), \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}(1-\alpha_t) \right), \end{aligned} \quad (8)$$

where $* \in \{\mathcal{I}, \mathcal{S}\}$, and

$$\begin{aligned} & p_\theta(z_{t-1}^{\mathcal{I}} | z_t^{\mathcal{I}}) = \\ & \mathcal{N}\left(\alpha_t^{-0.5} \left(z_t^{\mathcal{I}} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta^{\mathcal{I}}(z_t^{\mathcal{I}}, \Gamma_t^{\mathcal{S}}, C, t) \right), \Sigma_\theta(z_t^{\mathcal{I}}, t) \right), \end{aligned} \quad (9)$$

$$\begin{aligned} & p_\phi(z_{t-1}^{\mathcal{S}} | z_t^{\mathcal{S}})\Big) = \\ & \mathcal{N}\left(\alpha_t^{-0.5} \left(z_t^{\mathcal{S}} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\phi^{\mathcal{S}}(z_t^{\mathcal{S}}, \Gamma_t^{\mathcal{I}}(C), t) \right), \Sigma_\phi(z_t^{\mathcal{S}}, t) \right). \end{aligned} \quad (10)$$

Substituting (8), (9) and (10) into (7), and omitting the weighting terms dependent on $\Sigma_\theta(z_t^{\mathcal{I}}, t)$ and $\Sigma_\phi(z_t^{\mathcal{S}}, t)$, as in [2], leads to the following *JointDiffuse* loss function:

$$\mathcal{L} = \mathbb{E}\Big\{ \|\epsilon^{\mathcal{I}} - \epsilon_\theta^{\mathcal{I}}(z_t^{\mathcal{I}}, \Gamma_t^{\mathcal{S}}, C, t)\|_2^2 + \|\epsilon^{\mathcal{S}} - \epsilon_\phi^{\mathcal{S}}(z_t^{\mathcal{S}}, \Gamma_t^{\mathcal{I}}(C), t)\|_2^2 \Big\}. \quad (11)$$

4. Implementation details

Our approach leverages a pre-trained Stable Diffusion 2.1 model [7] for image denoising, along with its VAE encoder-decoder. For text conditioning, we used captions from the COCO dataset and generated scene descriptions for nuImages using LLaVA-NeXT 1.6 [6].

4.1. Segmentation annotation pipeline

Annotation segmentation maps were generated using the ViT-H SAM model [3]. Ground truth bounding box annotations were used as prompts for SAM to obtain precise object segmentation within each box.

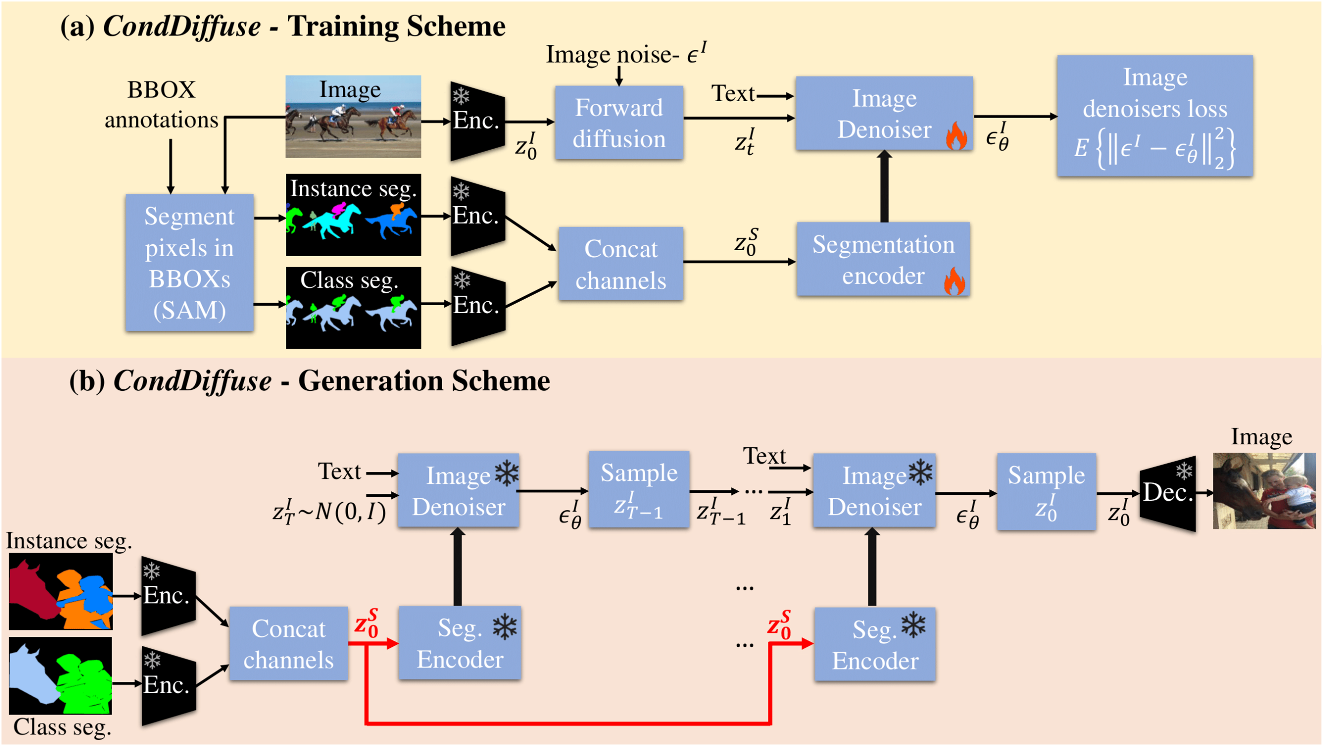


Figure 1. Overview of *CondDiffuse*. **(a) Training scheme**: The image and annotations (instance and class segmentations) are encoded using a VAE, producing $z_0^{\mathcal{I}}$ and $z_0^{\mathcal{S}}$, respectively, following the same approach as *JointDiffuse*. Forward diffusion is applied only to the image latent. Subsequently, the image denoiser estimates the noise in the image latent using the text caption and the segmentation latent, which is first processed through a segmentation encoder before being integrated into the image denoiser. The parameters of the image denoiser are optimized to minimize the MSE loss of the image noise $\epsilon^{\mathcal{I}}$. **(b) Generation scheme**: During generation, the image and segmentation denoisers are frozen. The image is generated iteratively; at each iteration, the image denoiser estimates the noise in the image latent using the text caption and the clean class and instance segmentation latents $z_0^{\mathcal{S}}$, which are processed through the segmentation encoder. This processes generates the image conditioned on the annotations, represented as instance and class segmentation maps. After the final iteration, the output image latent $z_0^{\mathcal{I}}$ is decoded using the VAE decoder, resulting in the generated image along with its corresponding input annotations.

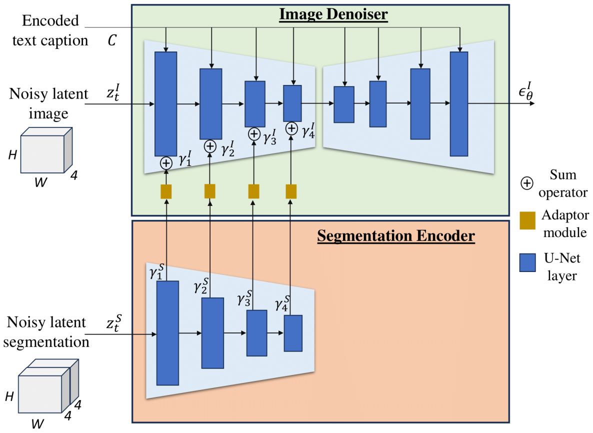


Figure 2. Architecture for integrating segmentation features into the image denoiser. The segmentation encoder has the same number of layers as the image denoiser encoder. Features from each layer of the segmentation encoder are processed through an adaptor module consisting of three ResNet blocks before being added to the corresponding layer in the image denoiser.

4.2. Obtaining bounding boxes from generated images and segmentation maps

Bounding box annotations for generated images were extracted using a DINO detector [11] with nine input channels: the image, instance segmentation map, and class segmentation map (three channels each). The detector was trained using its standard configuration.

4.3. Segmentation denoiser details

The segmentation denoiser is a UNet with the same number of layers as Stable Diffusion 2.1 [7] but with half the channels per layer and without image-text cross-attention layers. It contains only 12% of the image denoiser's parameters and is initialized with random weights. The adaptor module, used before merging image and segmentation denoiser features, consists of three ResNet layers, with input and output dimensions matching each denoiser.

4.4. Training details

The joint image and segmentation denoiser was trained for 61,400 iterations with a batch size of 32, a learning rate of 8×10^{-5}, and a weight decay of 0.01. We used cosine scheduling with zero terminal SNR at $t = T$ [5] and v-loss weighting [5, 8]. The model was trained on images and segmentation maps of size 800×456.

4.5. Sampling and guidance

Image generation was performed using 100 DDIM sampling steps [10]. We applied a classifier-free guidance scale of $\omega = 7.5$ [1].

5. Image and segmentation denoiser fusion

Table 6 in Section 4.4 of the main paper presents an ablation study on the image and segmentation denoiser fusion. Fig. 3 illustrates the fusion configurations tested in Table 6 of the main paper. Labels (a)–(f) in Fig. 3 correspond to rows 1–6 in Table 6 of the main paper.

6. Ablation study on noise scheduling in dual denoisers

We evaluate how different noise scheduling strategies affect the performance of the dual-denoiser architecture in *JointDiffuse*. Table 1 reports results of Faster-RCNN detector on the COCO validation set (image resolution 800×456) when training with the following three configurations:

1. Both denoisers use scale-linear noise scheduling [7].
2. Both denoisers use square-root noise scheduling [4].
3. Image denoiser uses scale-linear and segmentation denoiser uses square-root.

Among these, the third configuration yields the best performance. Although the first option performs better than the second, combining scale-linear for the image and square-root for the segmentation map results in the strongest gains.

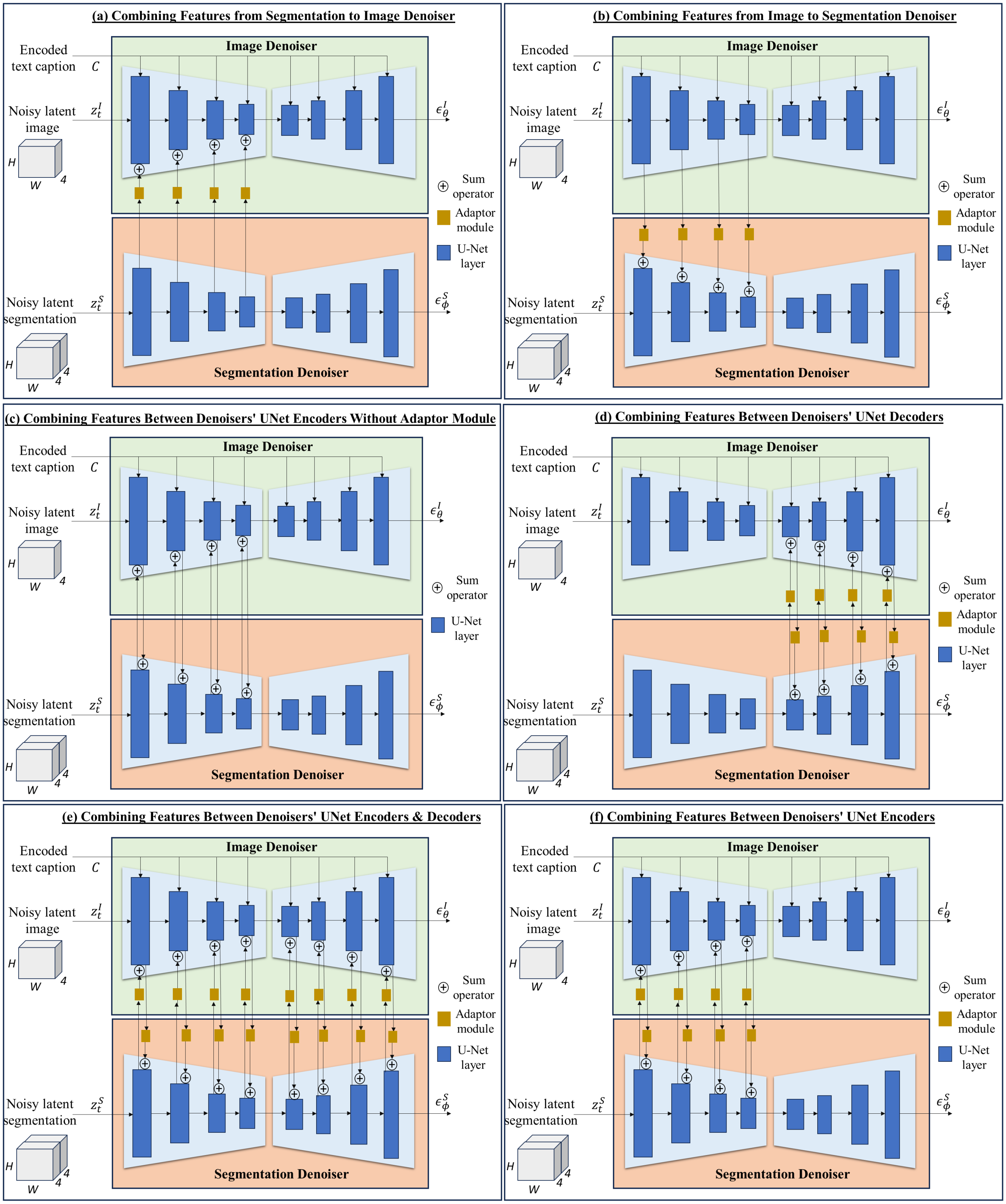
Figure Fig. 4 visualizes the noise variance over time for the scale-linear and square-root schedules. Compared to scale-linear, the square-root schedule maintains higher noise levels for a longer portion of the reverse diffusion process and only reduces noise significantly at later timesteps. We hypothesize that this difference improves coordination between the denoisers during inference (i.e., during the reverse denoising processes). Segmentation maps are typically easier to denoise and thus converge faster during inference. When both denoisers follow similar noise schedules, the image denoiser may over-rely on the already-converged segmentation, while the segmentation denoiser largely ignores the image, resulting in weak coupling. By maintaining higher noise in the segmentation denoiser for longer, its convergence is delayed, promoting mutual dependence: the segmentation denoiser benefits from the image features, and the image denoiser avoids over-reliance on the segmentation features. This tighter interaction improves joint generation quality, leading to better object detection when models are trained on the generated data.

Image Denoiser Scheduling	Segmentation Denoiser Scheduling	mAP
Scale-linear	Scale-linear	36.2
squre-root	squre-root	35.9
Scale-linear	squre-root	36.7

Table 1. Ablation on noise scheduling for the dual denoisers in *JointDiffuse*, evaluated on the COCO validation set using a Faster-RCNN detector at 800×456 resolution.

7. Runtime and Model Size Overhead of the Dual Denoiser

JointDiffuse employs a dual denoiser consisting of the Stable Diffusion 2.1 image denoiser and an additional lightweight segmentation denoiser, which uses half the channels of the image denoiser and omits text-image cross-attention layers. Adding the segmentation denoiser increases the parameter count by only 12% over the image denoiser. We further evaluate the runtime overhead of the dual denoiser relative to the standard Stable Diffusion 2.1 image denoiser. On an NVIDIA A100 (80GB), generating an 800×456 image with 100 DDIM steps (FP16) takes 3.52 seconds with standard Stable Diffusion 2.1 and 3.81 seconds with *JointDiffuse*, indicating that *JointDiffuse* introduces only a modest runtime overhead compared to baseline image generation.



(a) Combining Features from Segmentation to Image Denoiser

(b) Combining Features from Image to Segmentation Denoiser

(c) Combining Features Between Denoisers' UNet Encoders Without Adaptor Module

(d) Combining Features Between Denoisers' UNet Decoders

(e) Combining Features Between Denoisers' UNet Encoders & Decoders

(f) Combining Features Between Denoisers' UNet Encoders

Figure 3. Illustration of the image and segmentation denoiser fusion configurations tested in Table 6 of the main paper. Labels (a)–(f) in the figure correspond to rows 1–6 in Table 6 of the main paper.

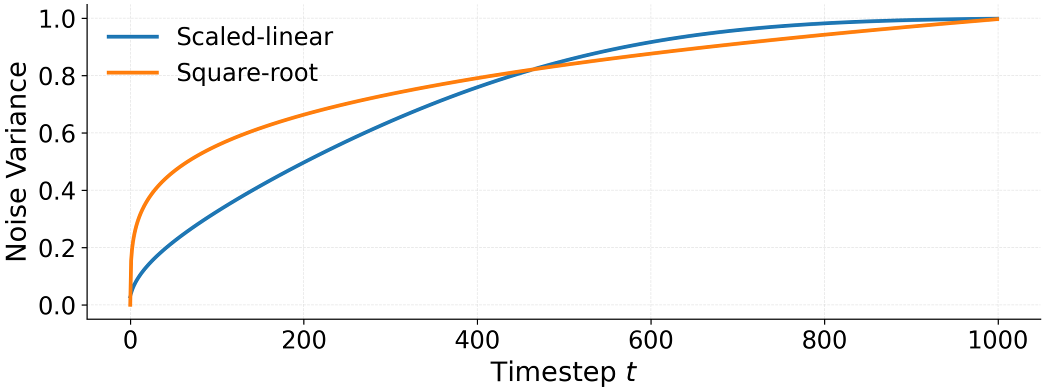


Figure 4. Noise-standard-deviation schedule $\sqrt{1-\bar{\alpha}_t}$ for the *scaled-linear* [7] and the *square-root* [4].

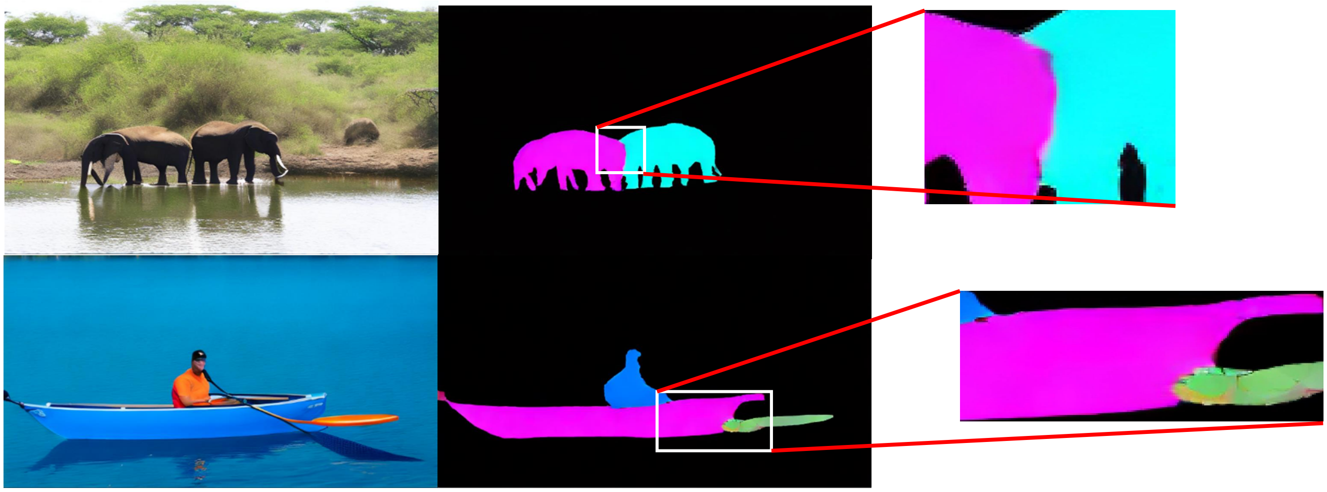


Figure 5. Qualitative examples illustrating imperfections in segmentation maps generated by *JointDiffuse*. Each row shows a generated image with its corresponding segmentation map, along with a zoom-in of the marked region (white rectangle). The upper example highlights soft boundary transitions around the elephants, while the lower one shows color variations in the green paddle. While these imperfections make the maps unsuitable for direct segmentation tasks, they remain sufficient for accurately extracting bounding boxes when combined with the generated images and a dedicated detector.

8. Imperfections in Generated Segmentation

Unlike conventional segmentation maps with discrete labels and sharp boundaries, the segmentation maps produced by *JointDiffusion* exhibit smooth color transitions, blurred edges, and occasional fluctuations in instance colors. Examples of these artifacts are highlighted in the zoomed-in regions of Figure 5: the upper example shows soft boundary transitions around the elephants, while the lower one illustrates variations in the green paddle's color.

These imperfections arise because the VAE encoder–decoder was pre-trained on natural images rather than segmentation maps, and the denoiser inherently generates continuous rather than discrete outputs.

Crucially, however, the generated maps still suppress background clutter and separate objects into distinct color regions. While not directly suitable for conventional segmentation tasks, they provide sufficient structure for accurate bounding box extraction. We therefore train a detection network that leverages the generated instance and class maps together with the image, enabling it to recover accurate bounding boxes. In practice, this approach achieves substantially higher detection accuracy compared to using the image alone as shown in Table 2 of Section 4 of the main paper.

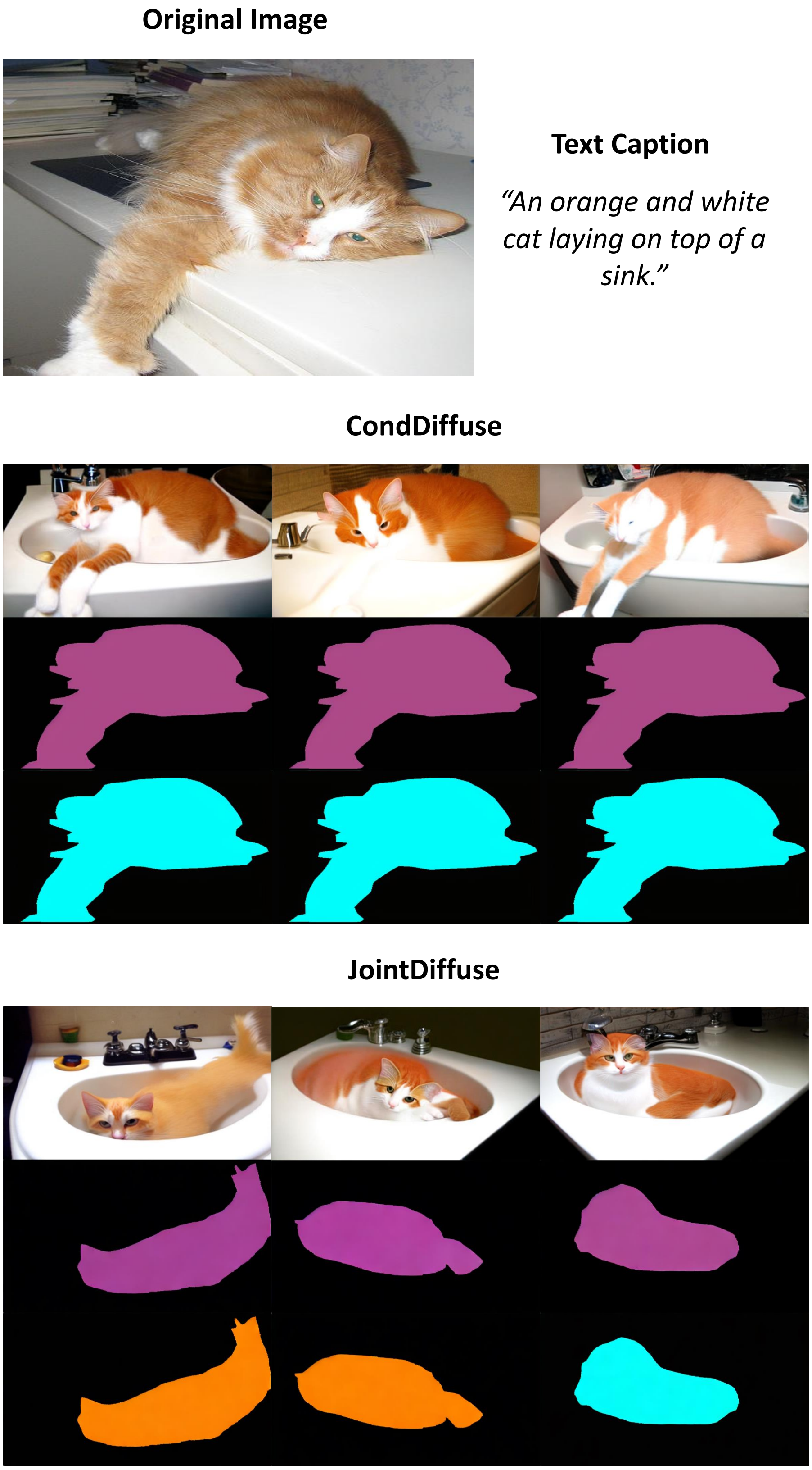


Figure 6. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

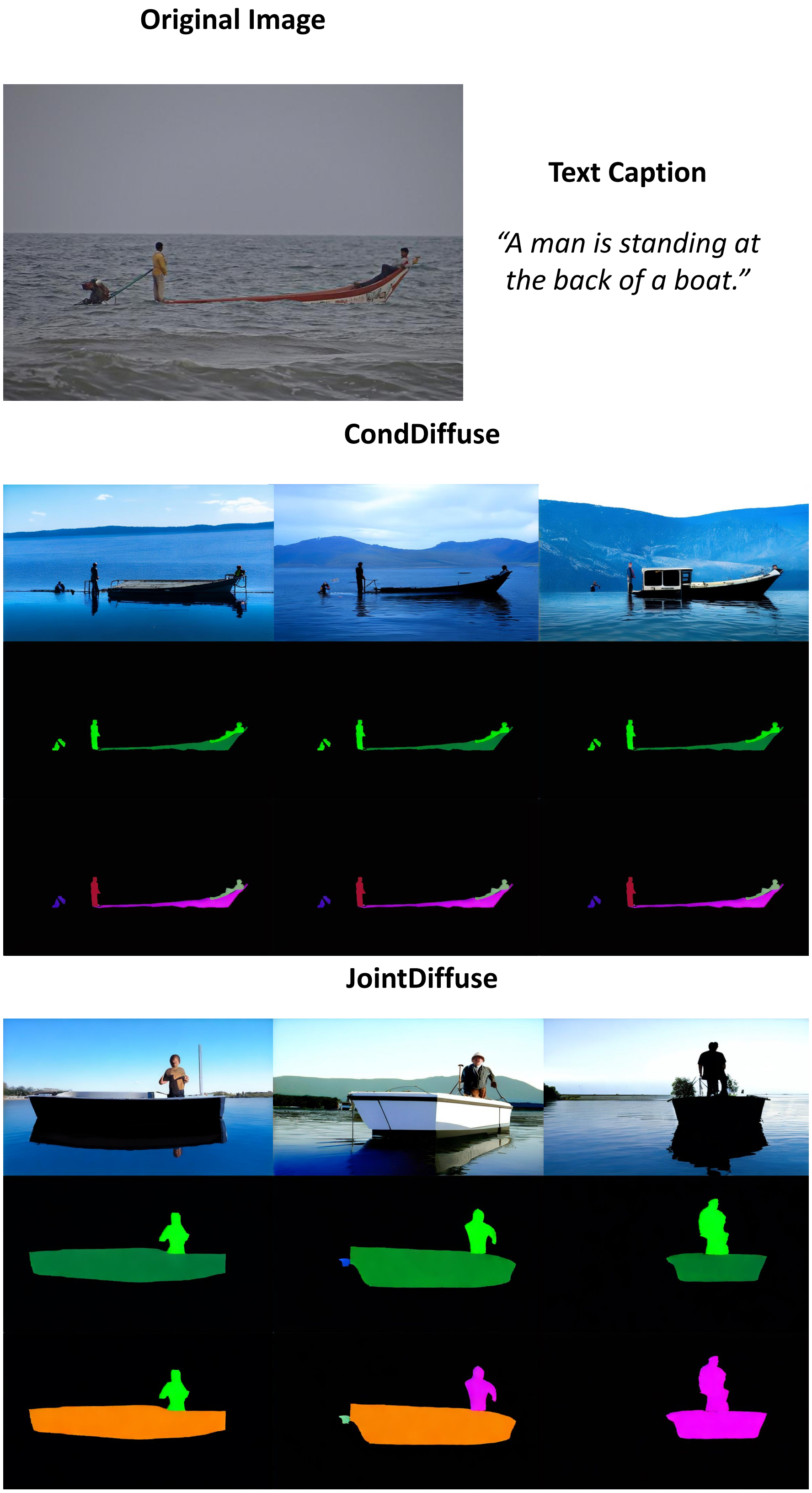


Figure 7. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

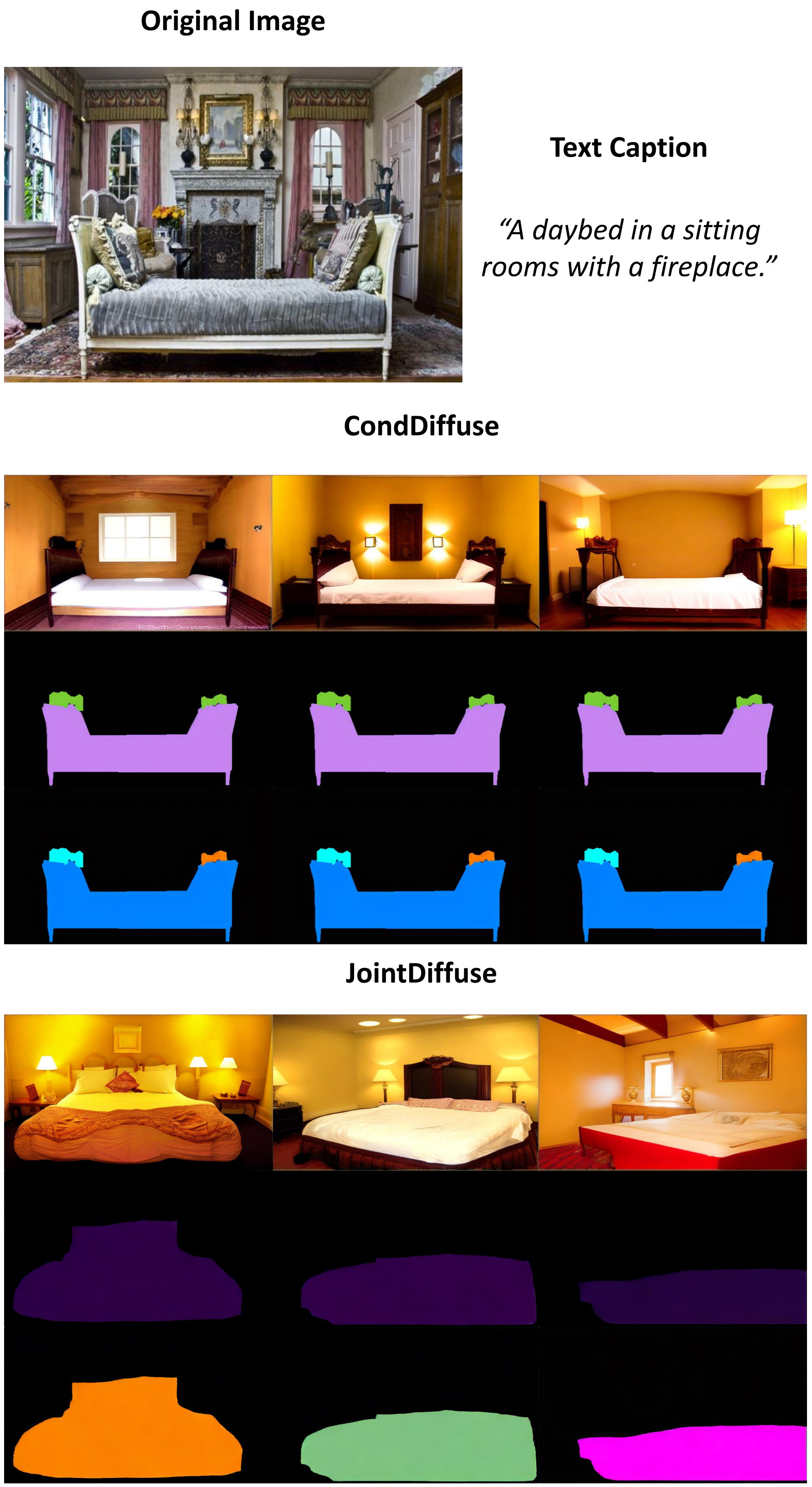


Figure 8. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

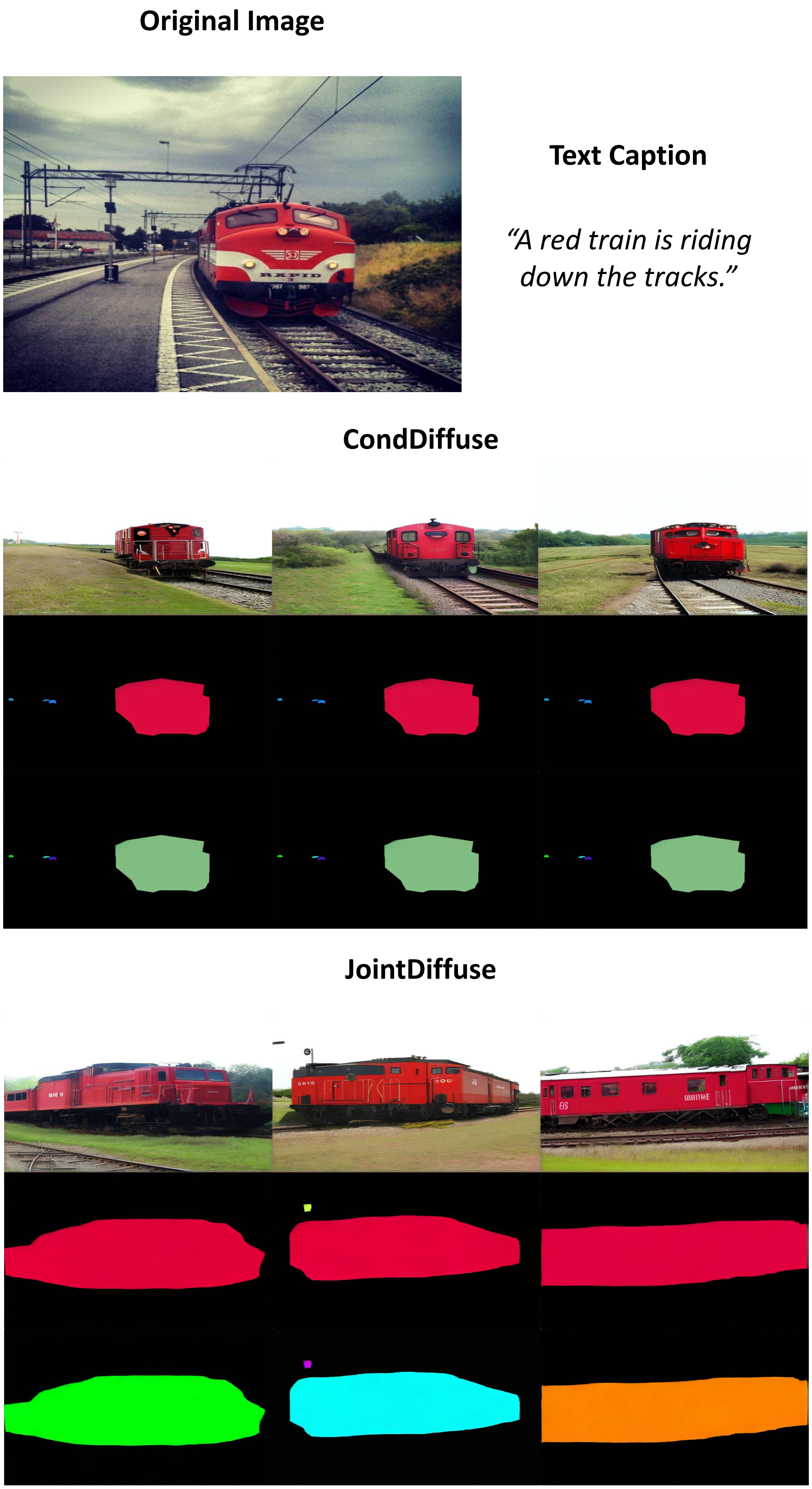


Figure 9. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

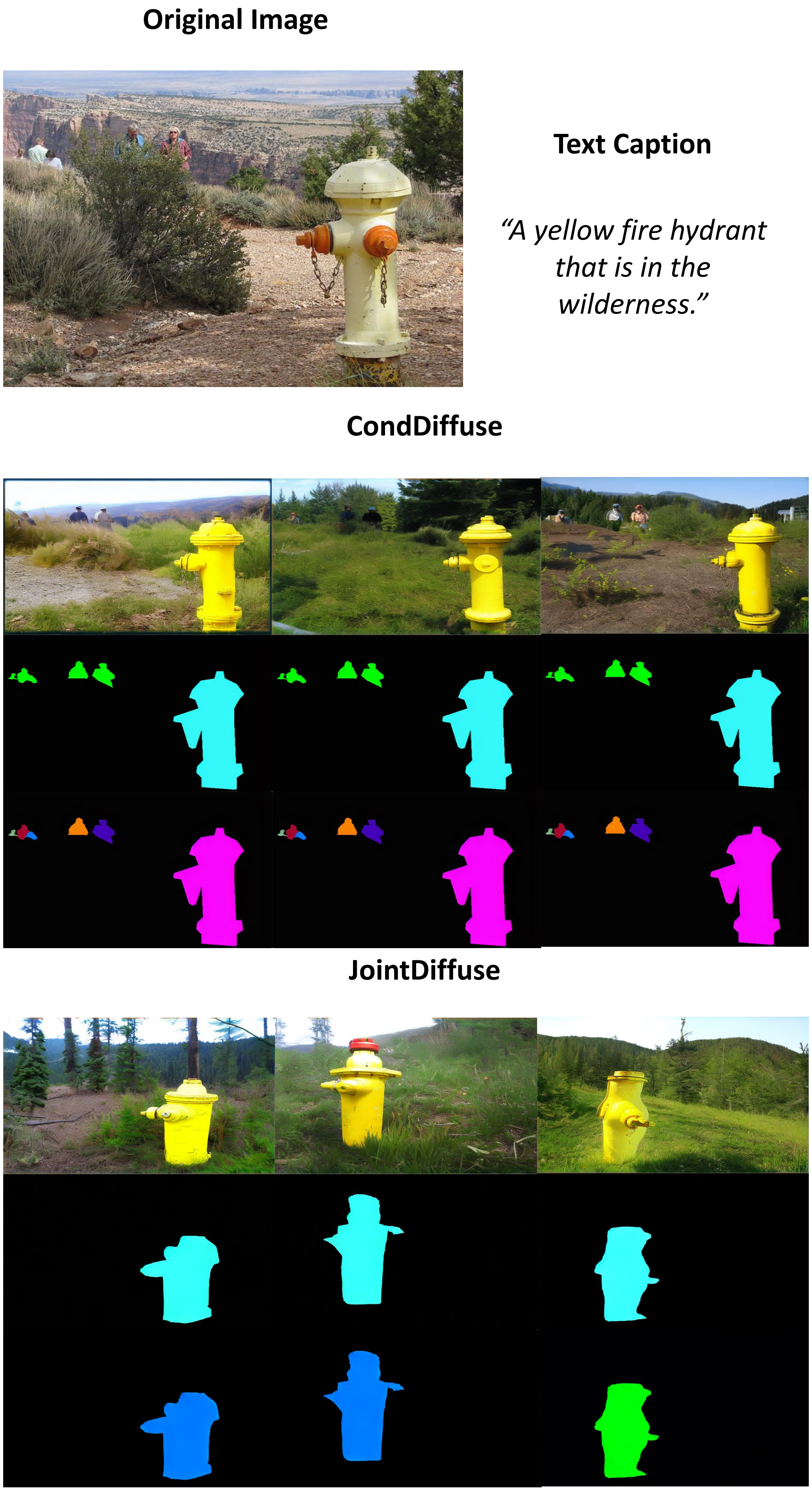
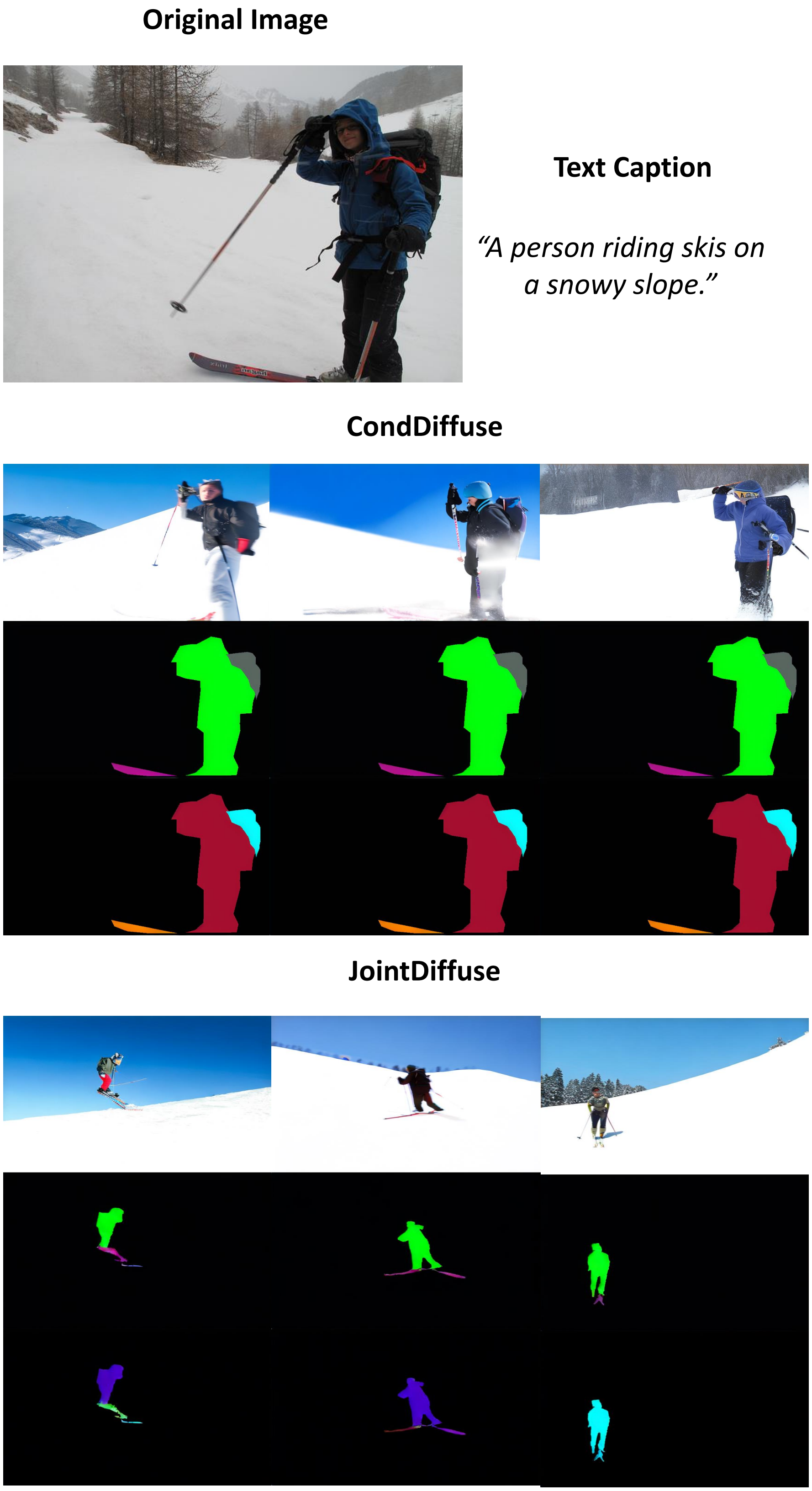


Figure 10. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.



Original Image

Text Caption

“A person riding skis on a snowy slope.”

CondDiffuse

JointDiffuse

Figure 11. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

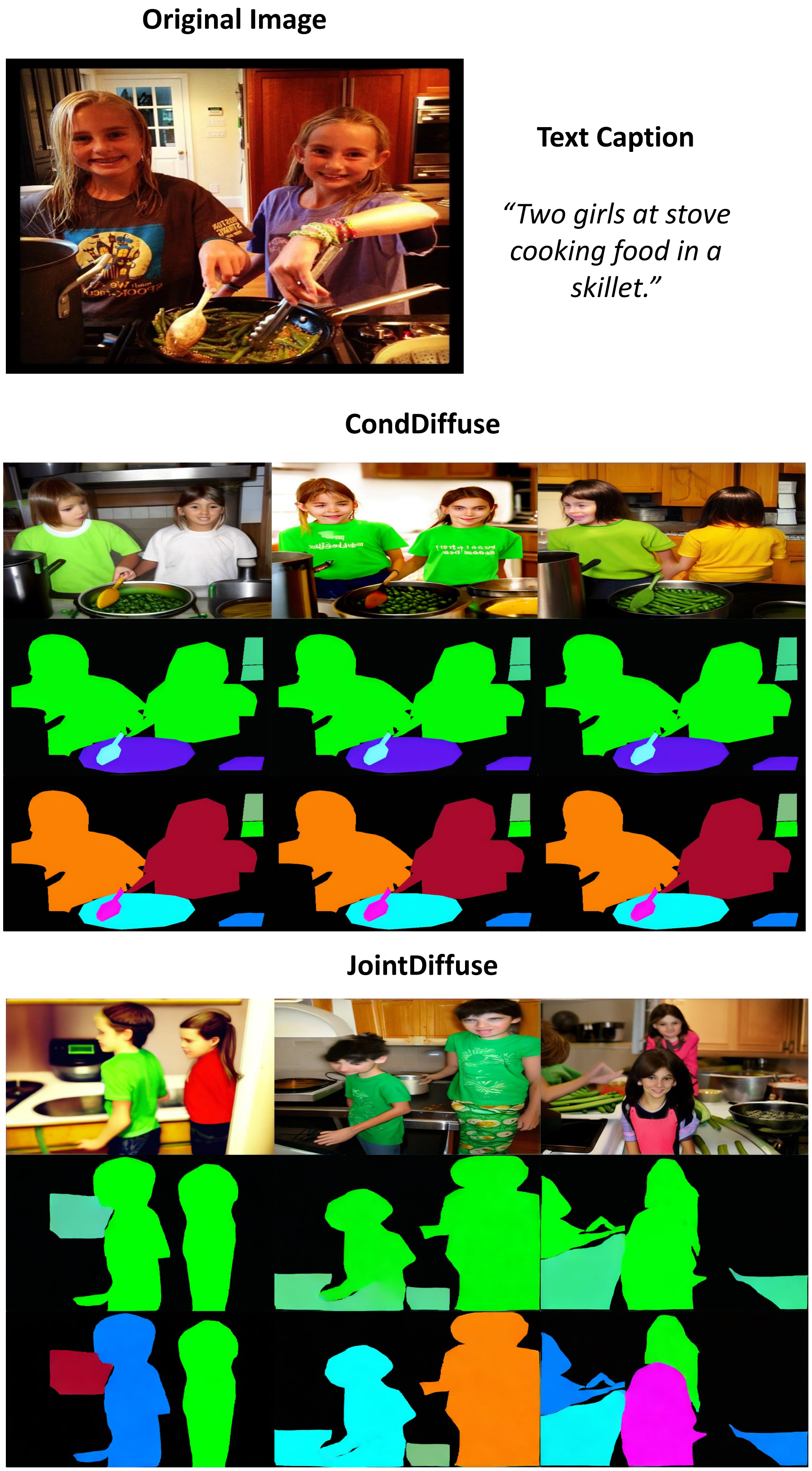


Figure 12. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

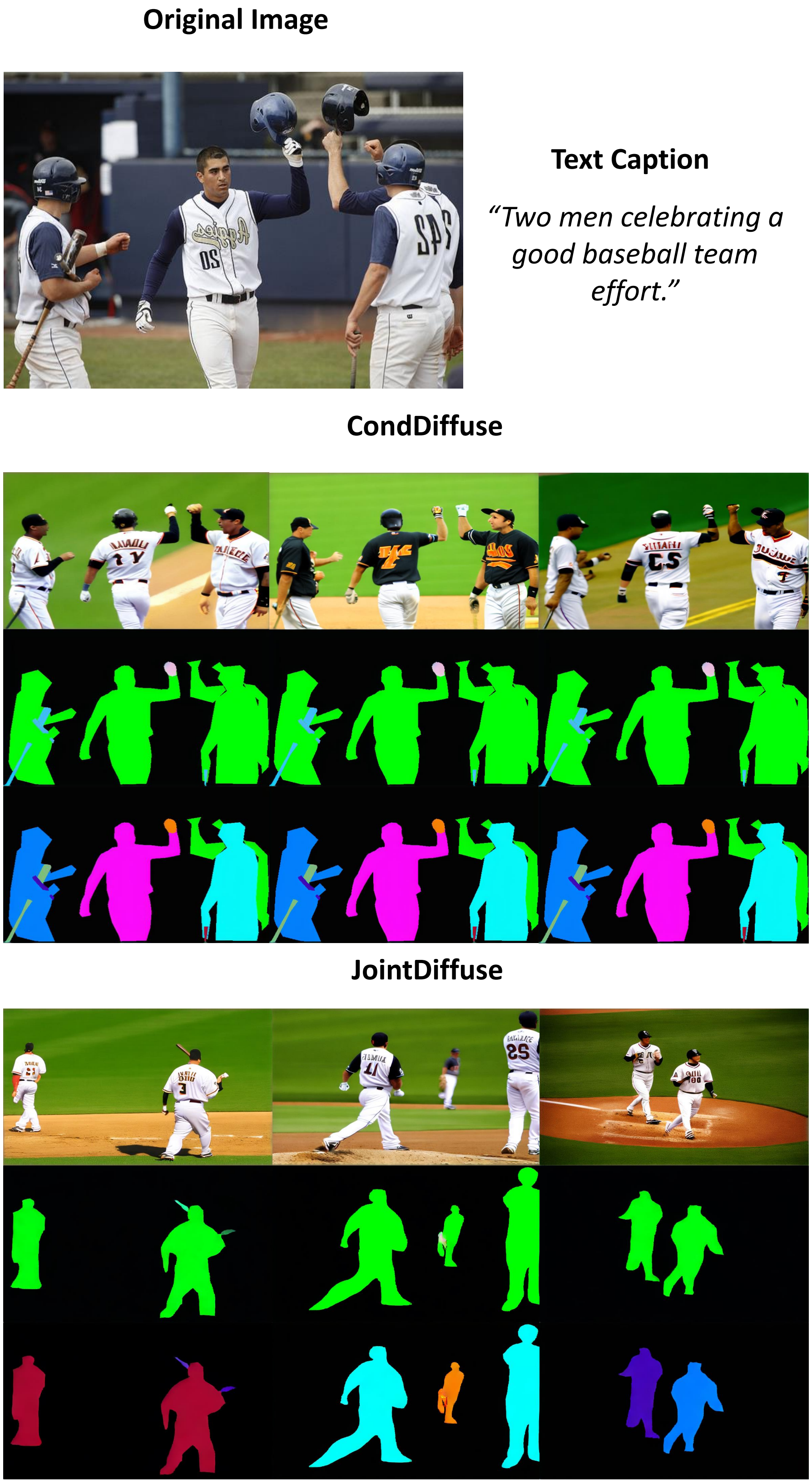


Figure 13. Qualitative example demonstrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

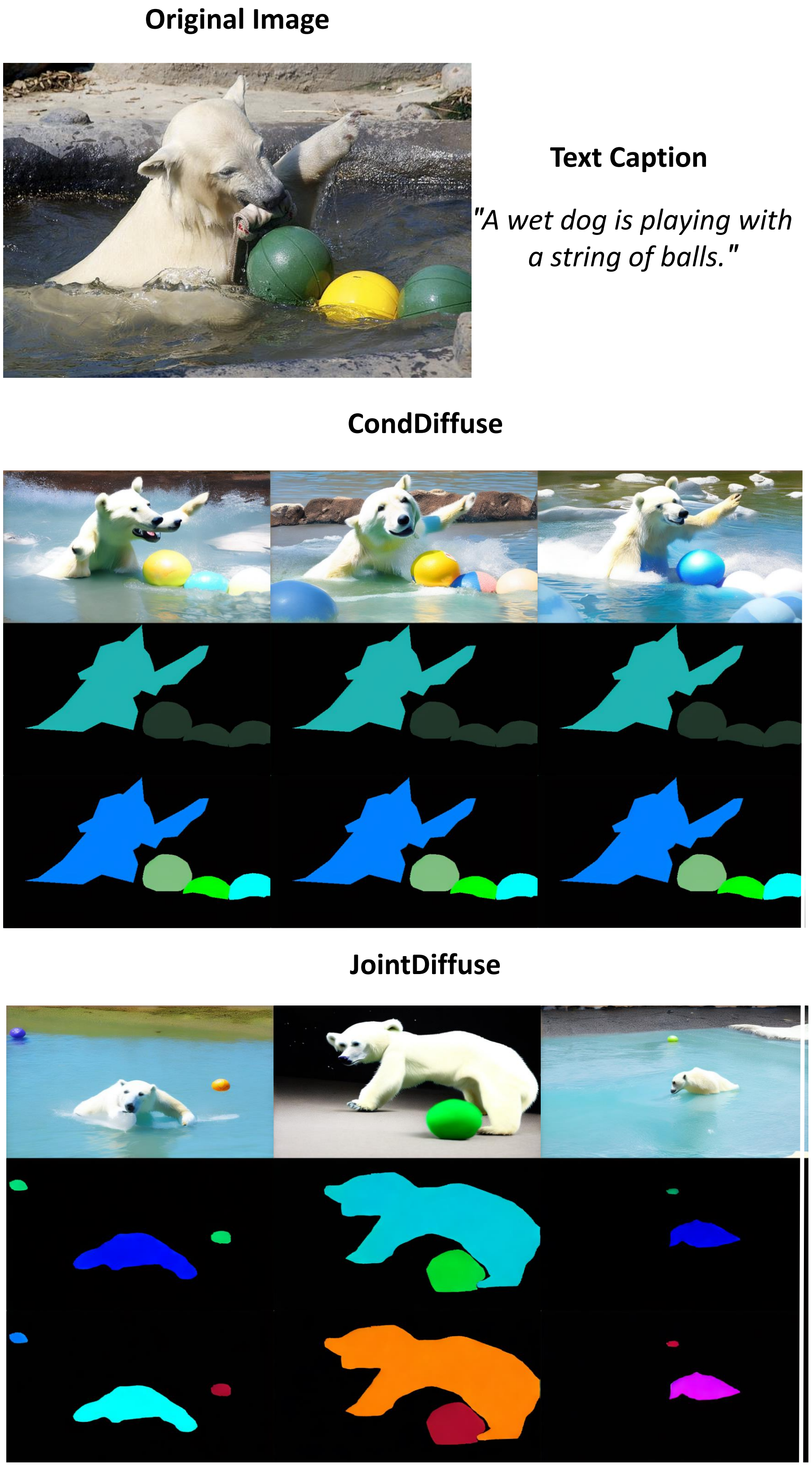


Figure 14. Qualitative example showing the superior alignment of images and annotations in *JointDiffuse* compared to *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

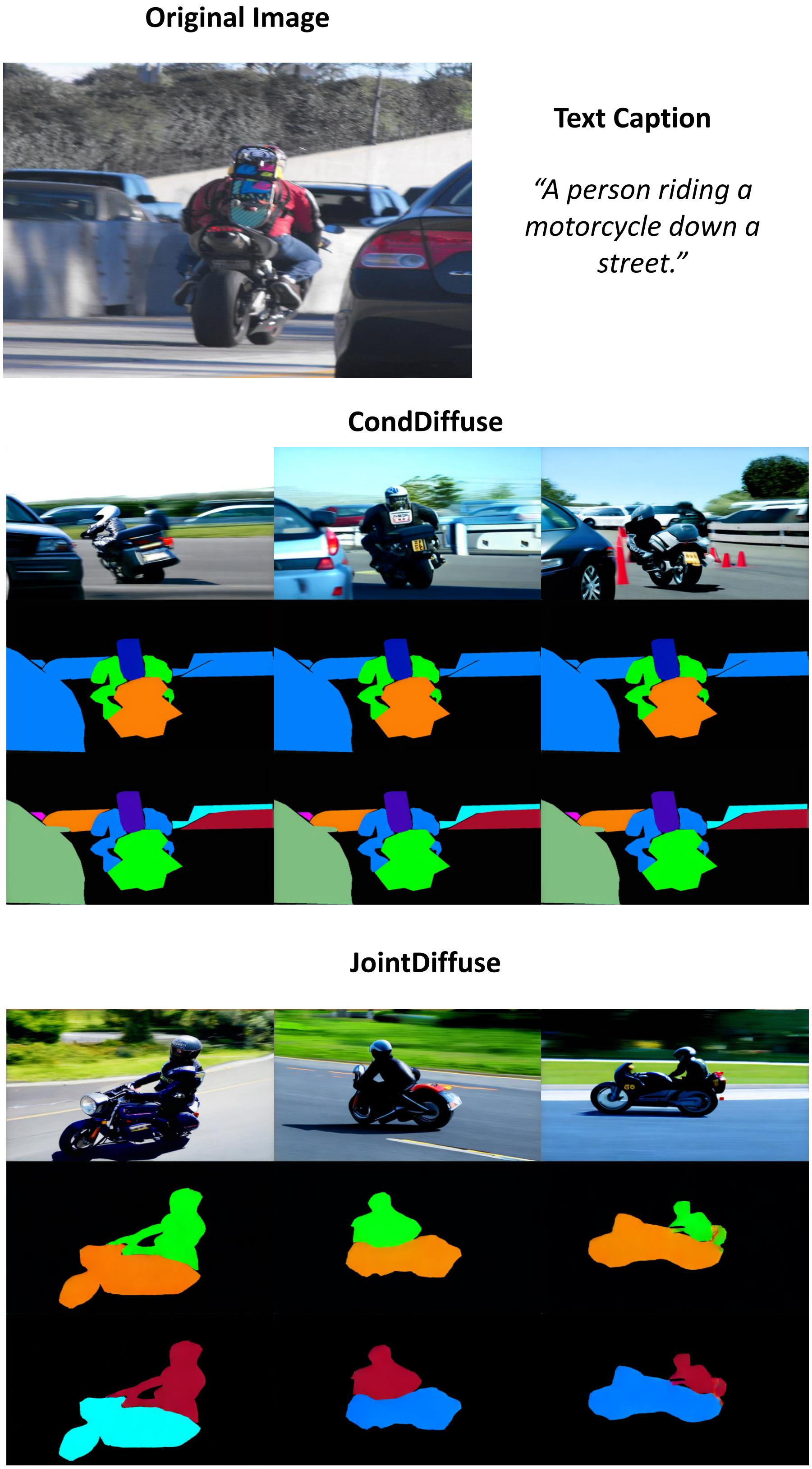


Figure 15. Qualitative example showing the superior alignment of images and annotations in *JointDiffuse* compared to *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

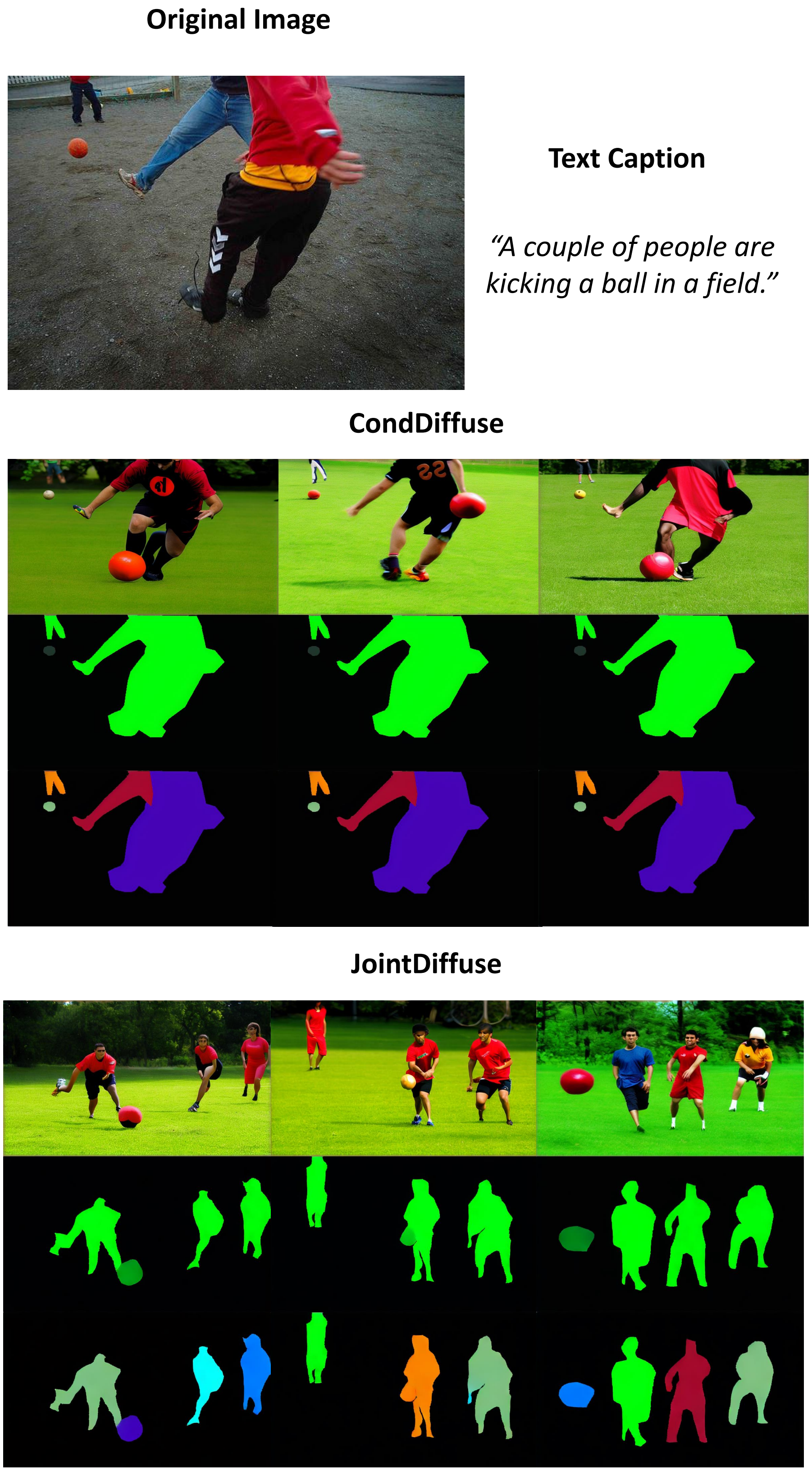


Figure 16. Qualitative example showing the superior alignment of images and annotations in *JointDiffuse* compared to *CondDiffuse*. The top row shows a COCO image with its text caption. Below, three *CondDiffuse* generated images are displayed, each followed by its class segmentation and instance segmentation (from top to bottom). The bottom section presents three *JointDiffuse* generated images with their class and instance segmentations in the same order.

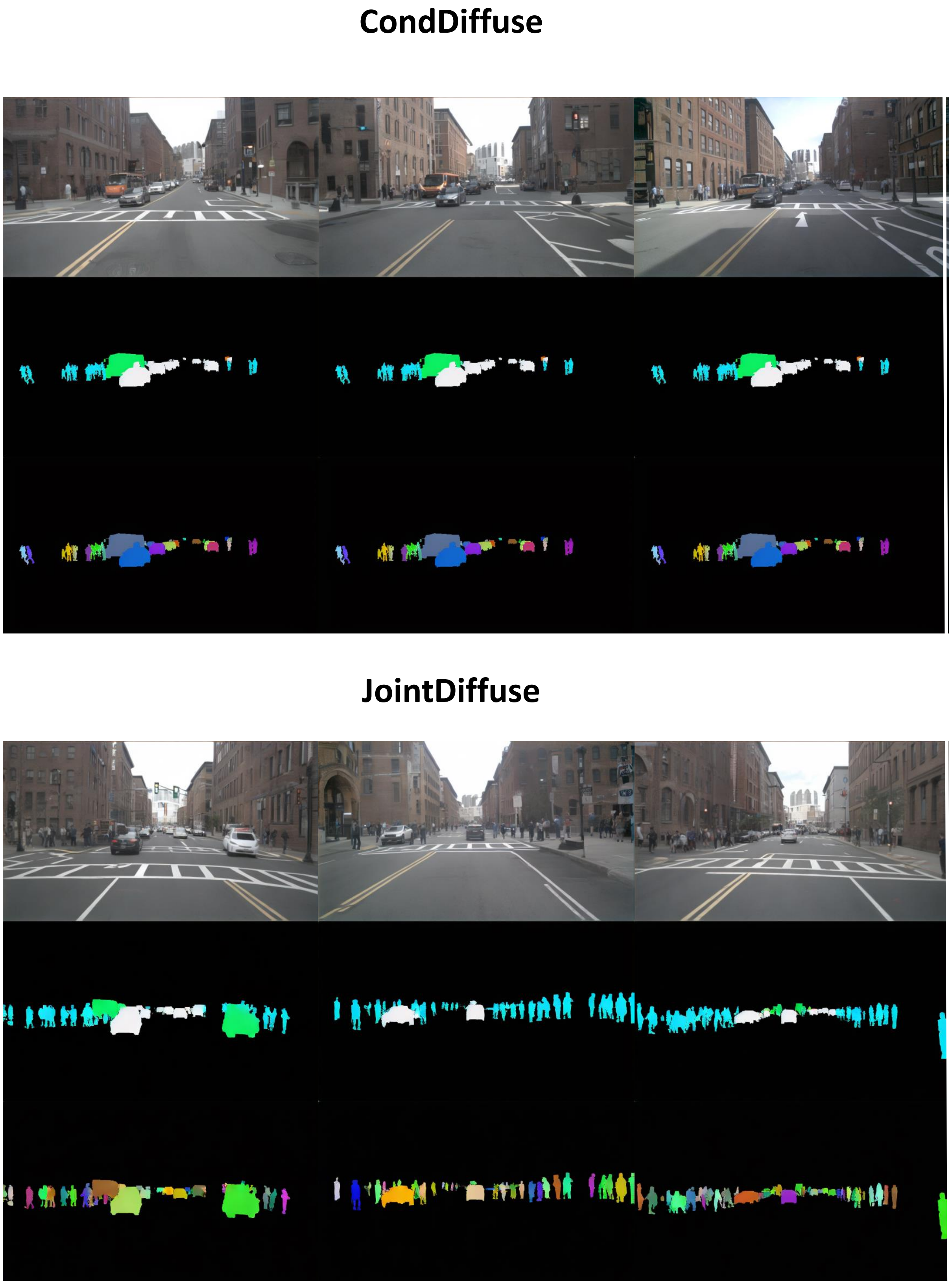


Figure 17. Qualitative example illustrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*, both trained on the nuImages dataset. The upper section displays three *CondDiffuse* generated images, each followed by its class and instance segmentations (top to bottom). The bottom section presents three *JointDiffuse* generated images with segmentations in the same order.

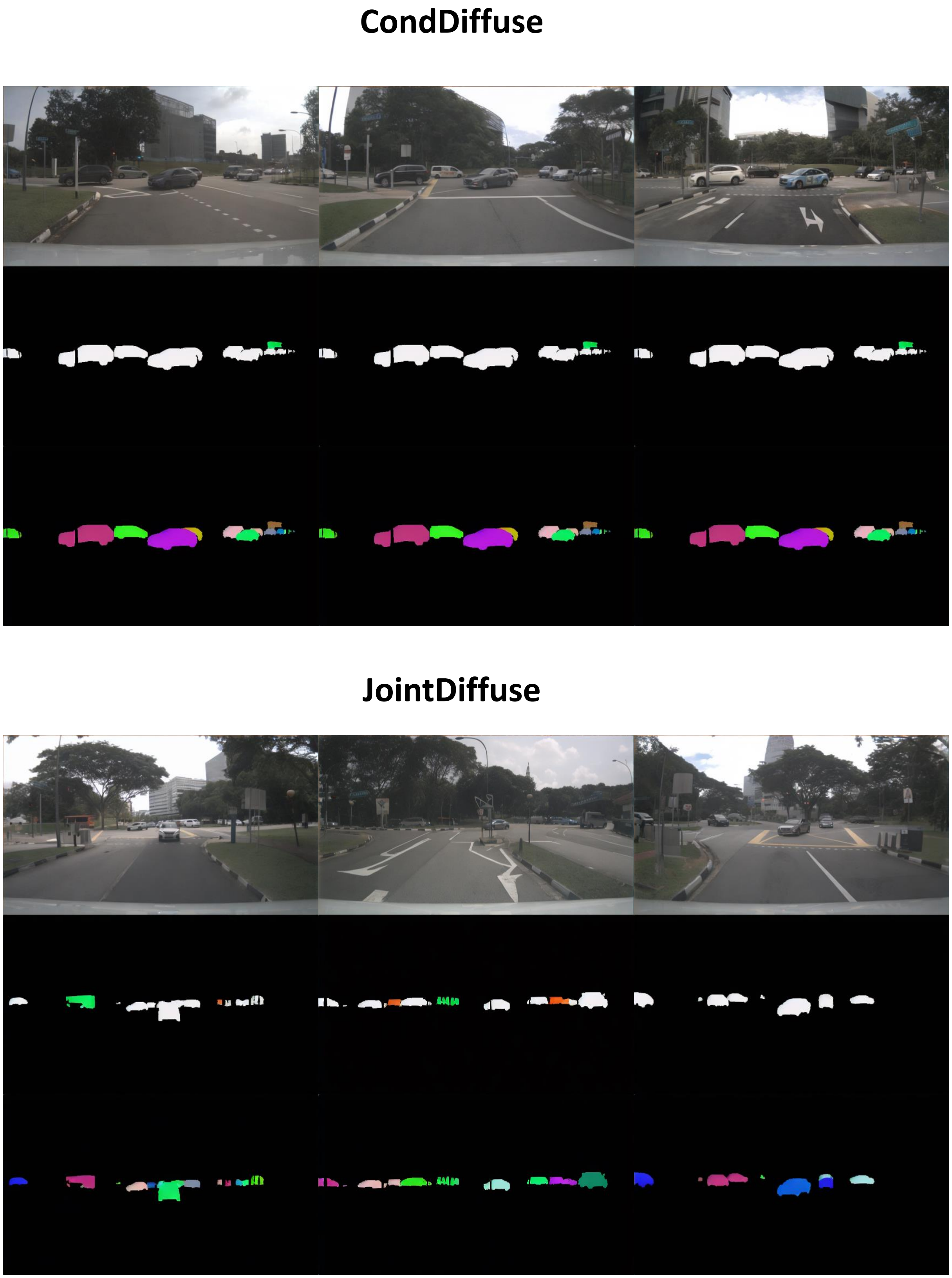


Figure 18. Qualitative example illustrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*, both trained on the nuImages dataset. The upper section displays three *CondDiffuse* generated images, each followed by its class and instance segmentations (top to bottom). The bottom section presents three *JointDiffuse* generated images with segmentations in the same order.

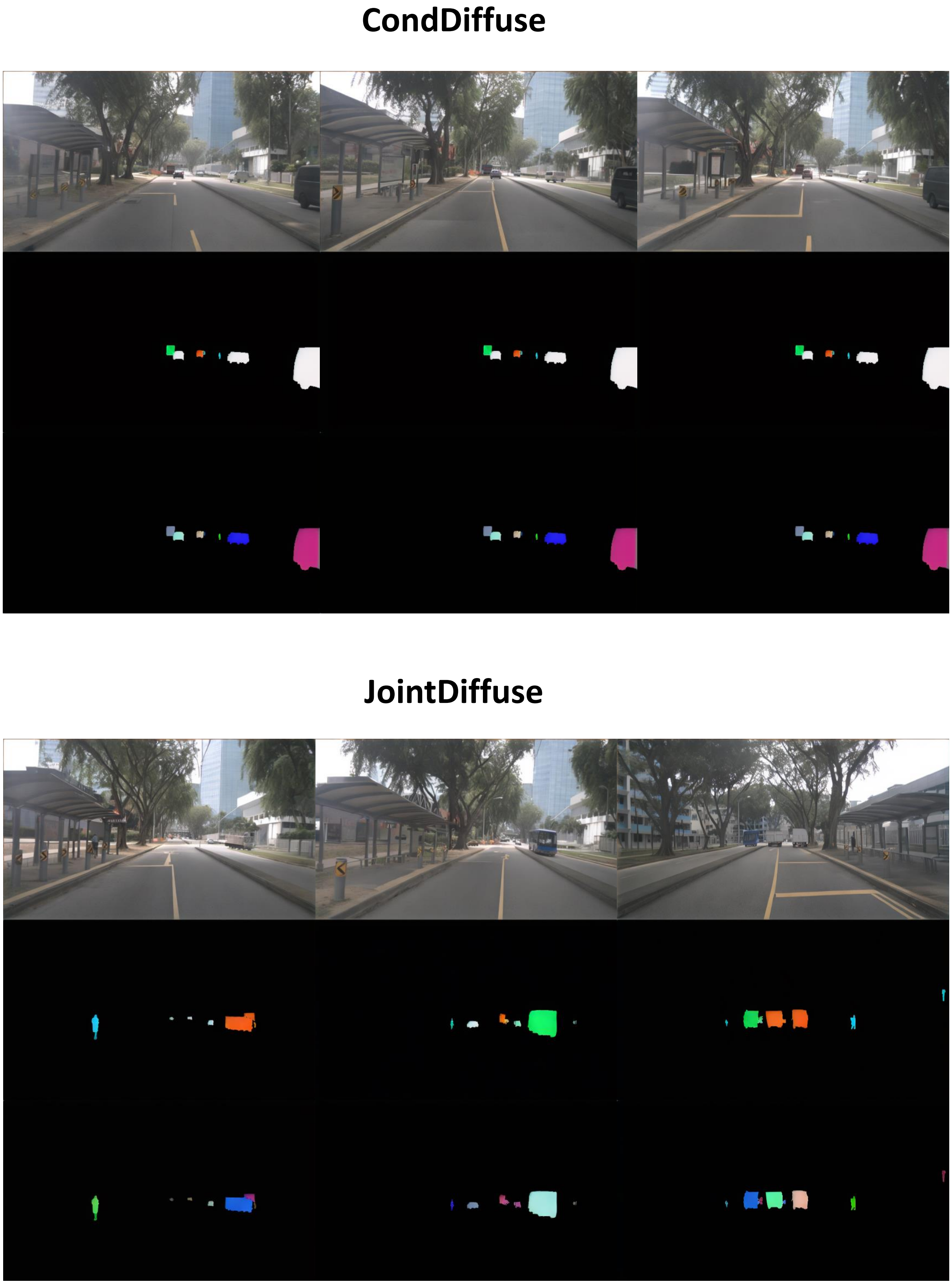


Figure 19. Qualitative example illustrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*, both trained on the nuImages dataset. The upper section displays three *CondDiffuse* generated images, each followed by its class and instance segmentations (top to bottom). The bottom section presents three *JointDiffuse* generated images with segmentations in the same order.

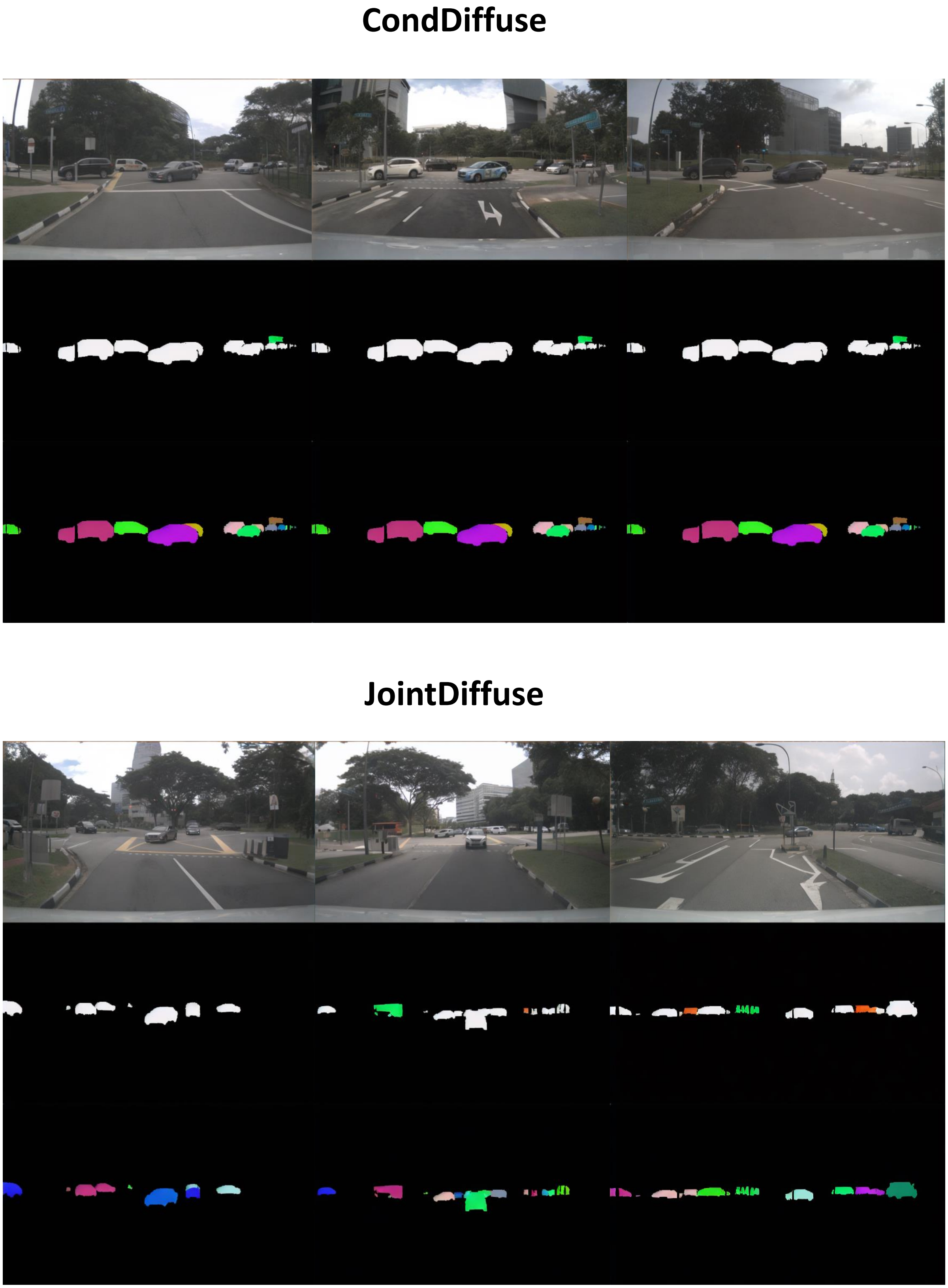


Figure 20. Qualitative example illustrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*, both trained on the nuImages dataset. The upper section displays three *CondDiffuse* generated images, each followed by its class and instance segmentations (top to bottom). The bottom section presents three *JointDiffuse* generated images with segmentations in the same order.

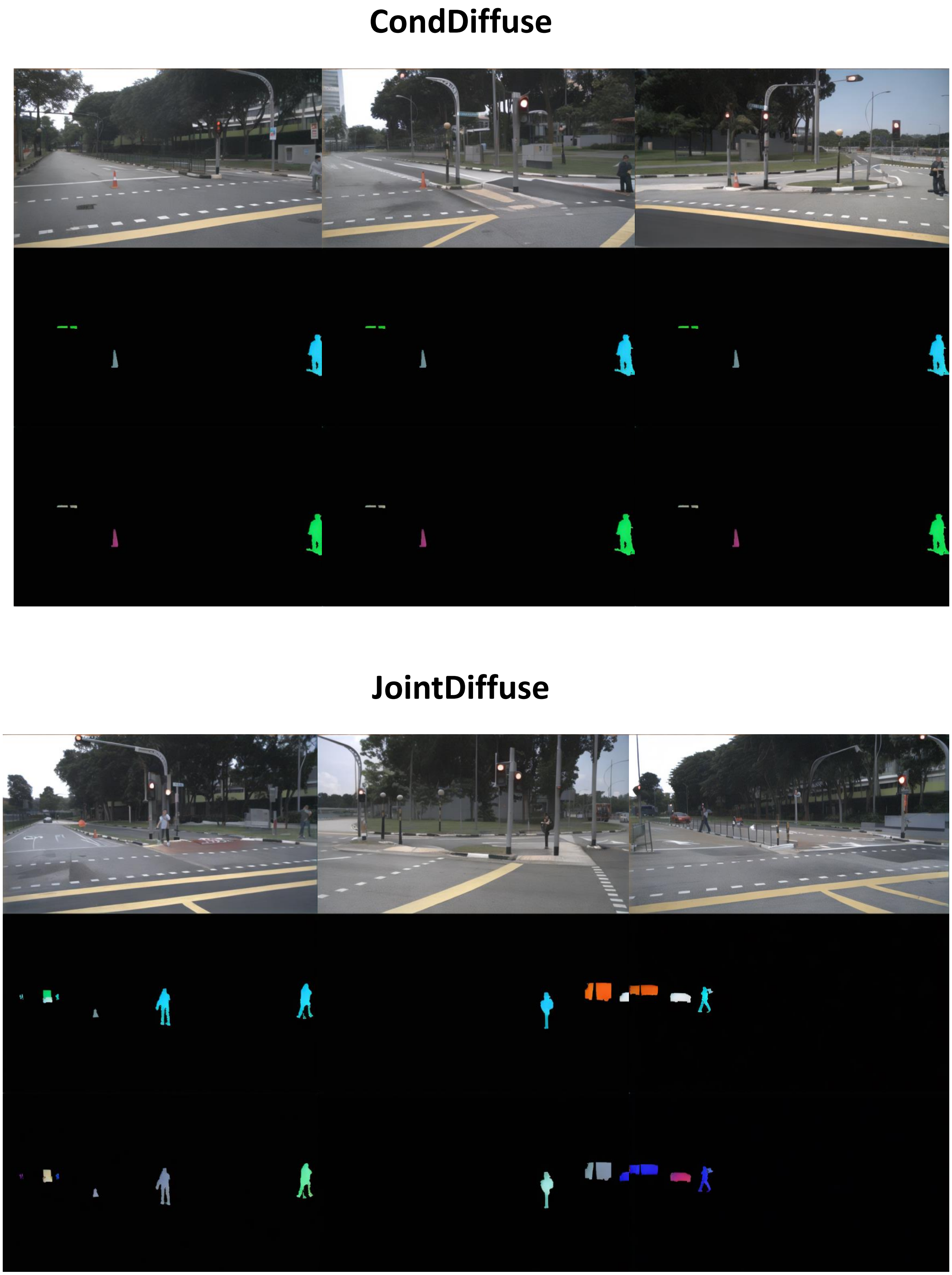


Figure 21. Qualitative example illustrating the greater scenario diversity of *JointDiffuse* over *CondDiffuse*, both trained on the nuImages dataset. The upper section displays three *CondDiffuse* generated images, each followed by its class and instance segmentations (top to bottom). The bottom section presents three *JointDiffuse* generated images with segmentations in the same order.

References

[1] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2

[4] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022. 4, 6

[5] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5404–5411, 2024. 4

[6] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024. 2

[7] 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. 2, 4, 6

[8] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 4

[9] 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

[10] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 4

[11] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022. 4