Composing Parts for Expressive Object Generation

Supplementary Material

Table of Contents

A. Additional Qualitative Results

Comparison with Recent Baselines. We compare PartComposer against existing baselines of StableDiffusion 3.5 [17] and Inpainting through Stable Diffusion. For the StableDiffusion (SD) 3.5 model, we use the SD3.5 Large variant for which the web-UI is available online at the following link[1]. We provide the details of the PartComposer by adding those meaningfully into the base prompt. For the swan example in the last row of Fig. 8, we provide the following prompt to the model *'A photo of a white swan with a peacock crown, 8k, full hd'*. We observe in Fig. 8 that although the model tries to adhere to the prompt in ways, it fails to modify the correct object part. Hence, it fails to capture the required details of the part the user expects. For the Inpainting baseline, we use the same StableDiffusion (SD) 2.1 model as a base and use the SAM2 Hiera [39] model for obtaining the segmentation masks. We use the following inpainting repo[2] from GitHub to obtain the results. After obtaining the masks, we provide an inpainting prompt to the SD model to generate the final result. We observe that the SD Inpaint models sometimes, due to incorrect masks for the painting domain, don't produce results (third row in Fig. 8) and often can't follow the multiple-part instruction correctly as in Fig. 8 (first row). Further we would like to highlight that inpainting still requires manual ground of masks to parts by user, which makes it a different setting making the quantitative results incomparable. In comparison, PartComposer can produce correct and coherent results by producing the part attributes.

Comparison with Editing Methods. We provide a comparison of our approach with SotA editing methods: PlugNPlay (PnP) [47], TurboEdit [15], ZONE [26], MagicBrush [54] in Fig. 9. We observe that approaches either do style edits (PnP) or structure edits at full object level (ZONE, MagicBrush). Only PartComposer (ours) can perform the specified localized changes at the semantic part level, demonstrating its novelty and value.

Additional Comparison of PartComposer to Baselines (Fig. 1). In Fig. 11, we demonstrate a comparison of the StableDiffusion, InstructPix2Pix, and Rich-Text on the prompts demonstrated in the teaser figure. For Rich-Text, we add the description in PartText to the part token if it's present in the base prompt; if not, we add the part description as a footnote in the object token in the base prompt. For InstructPix2Pix, we make instructions regarding part modification one by one on the base generation. We find that the baselines significantly change the entire composition of the object instead of just specified parts.

We also provide additional results for comparison with baselines in Fig. 12, where the Stable Diffusion baseline with part details added often ignores them or generates artifacts. In comparison, PartComposer can produce the desired details for the parts as specified by the user. As in Merida's example, the Stable Diffusion method doesn't produce the brown skirt and over uses the green color specified for hair. The Stable Diffusion ignores the desired part detail for the hornbill and cardinal examples. The blue jay example produces an artifact of the dual beak. In comparison, PartComposer generates aesthetic compositions following the part prompt details mentioned by the user.

B. Generalization to SDXL

We implement the PartComposer Method for SDXL, and generate the results for prompts supplied in Fig. 1. Using the SD-XL implementation from Rich-Text Gen [18] as our base code, we implement PartComposer. We present our results in Fig. 10, where we observe that PartComposer can generate high-resolution images with specific attribute details.

C. Attribute Variations

We provide further results for attribute variations for prompts of natural domain in Fig. 13.

[1] https://huggingface.co/spaces/stabilityai/stable-diffusion-3.5-large

[2] https://github.com/Uminosachi/inpaint-anything

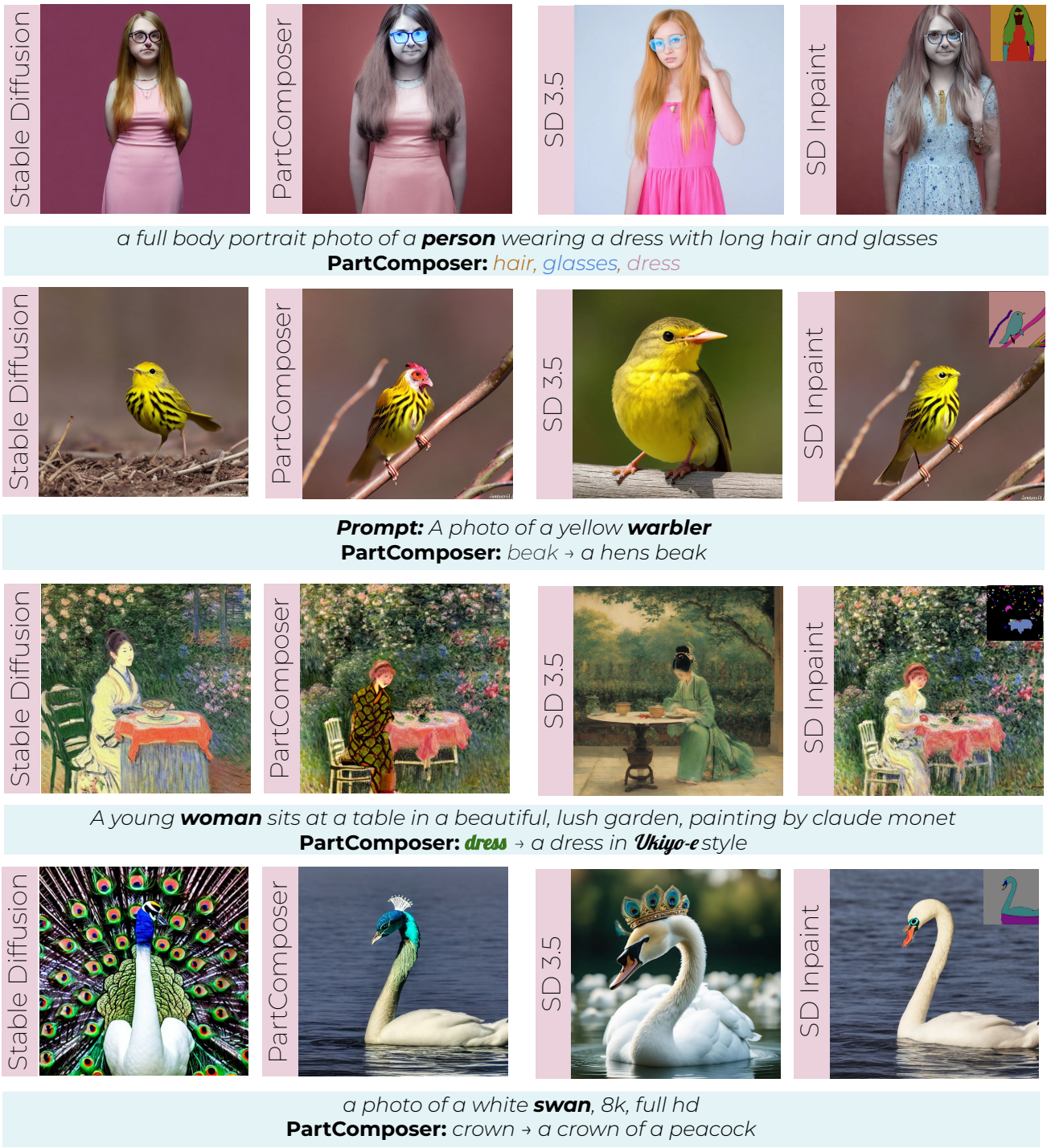


Figure 8. **Qualitative Comparison** of our proposed method with recent baselines of Stable Diffusion (SD) 3.5 and Stable Diffusion 2.1 Inpainting with SAM2 masks (top right). PartComposer leads to coherent part modification in comparison to advanced baselines.

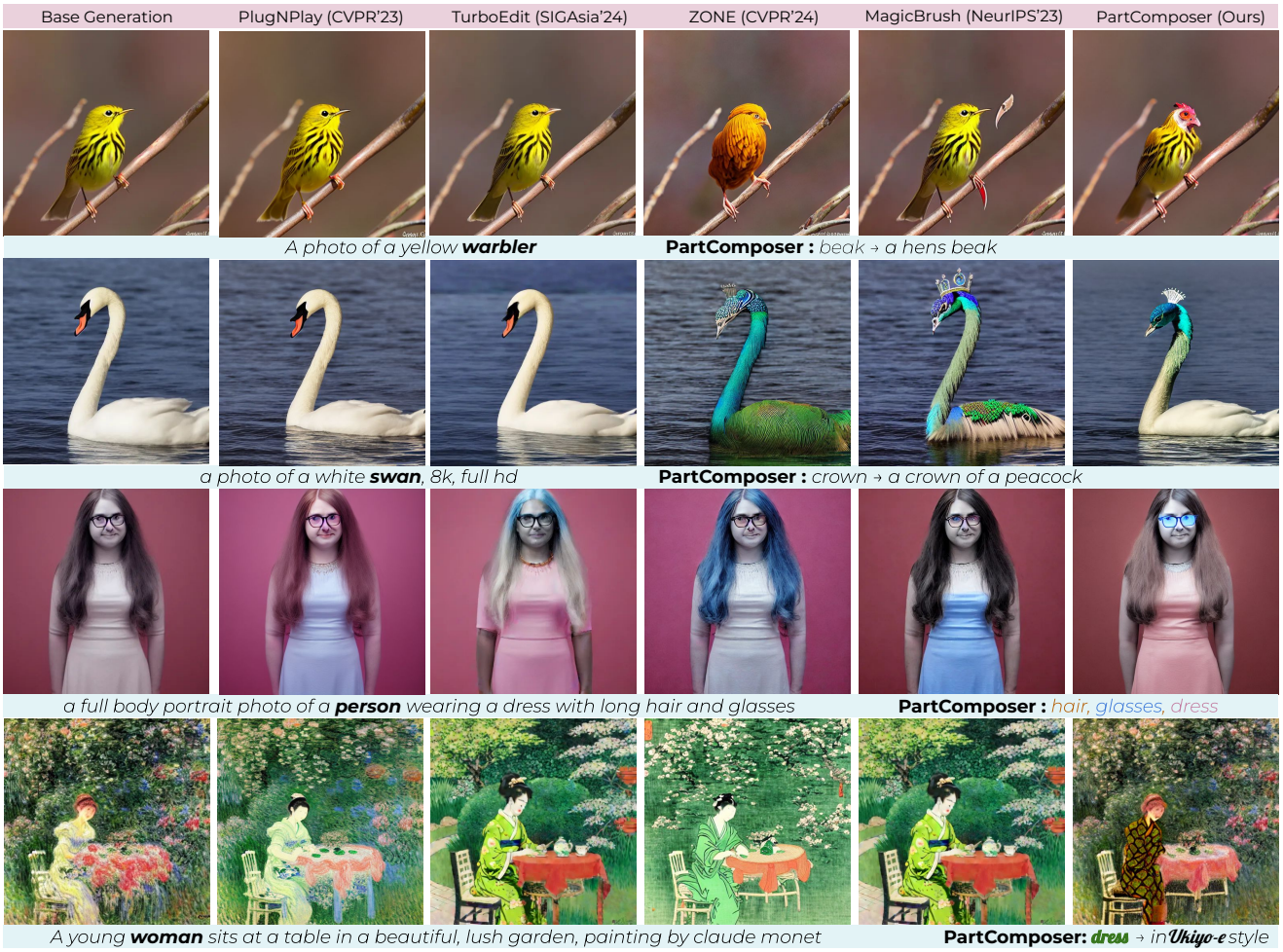


Figure 9. **Qualitative Comparison** of PartComposer with SotA editing methods applied on the generated base prompt image.

Figure 10. **High-Resolution** PartComposer results using SD-XL as base model, demonstrating its generalization.

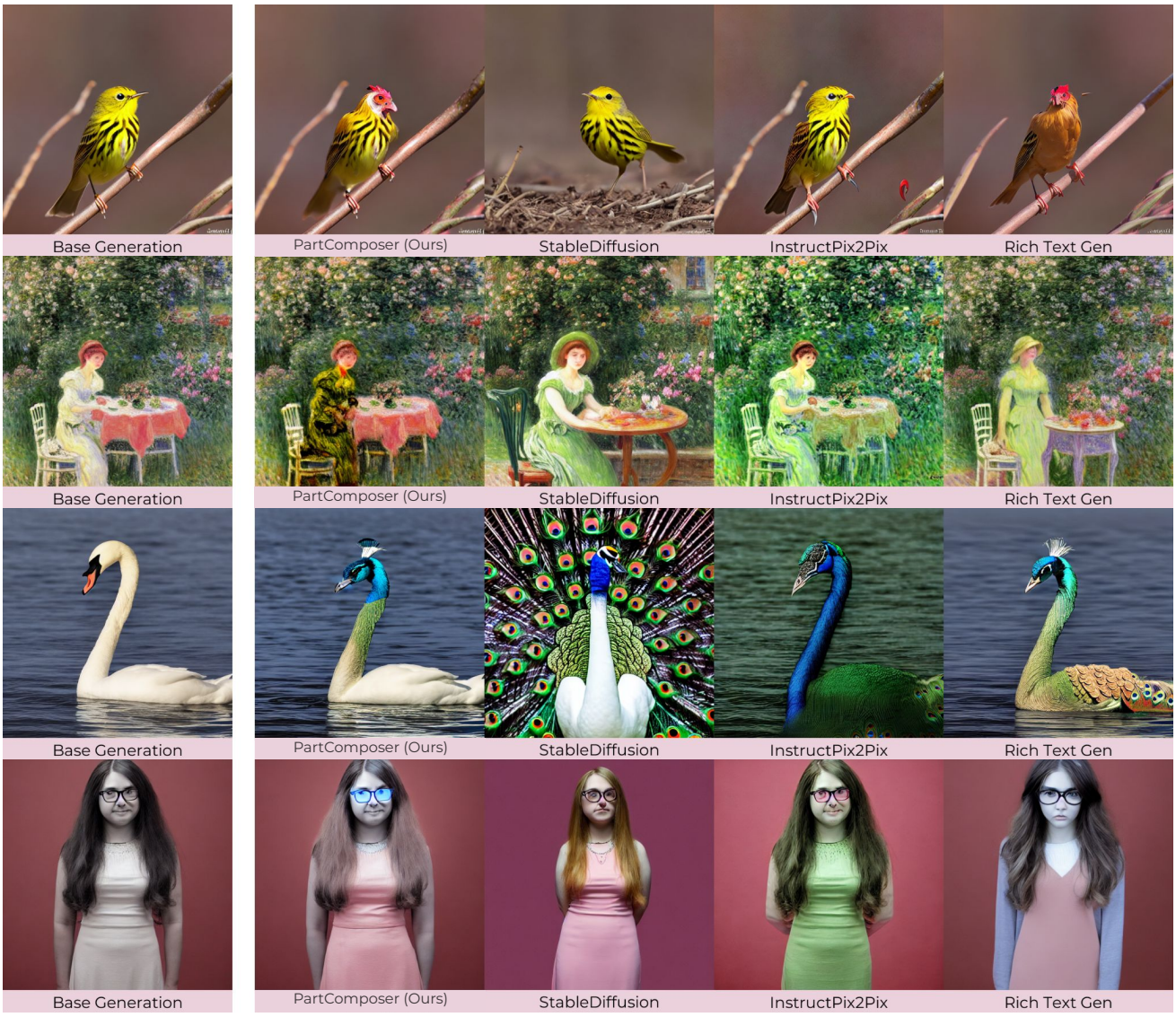


Figure 11. **Comparison of PartComposer to Other Approaches**, for generating using the prompts specified in Fig. 1 of paper. The PartComposer approach can modify base generations in the specified parts and generate aesthetic images compared to the state-of-the-art.

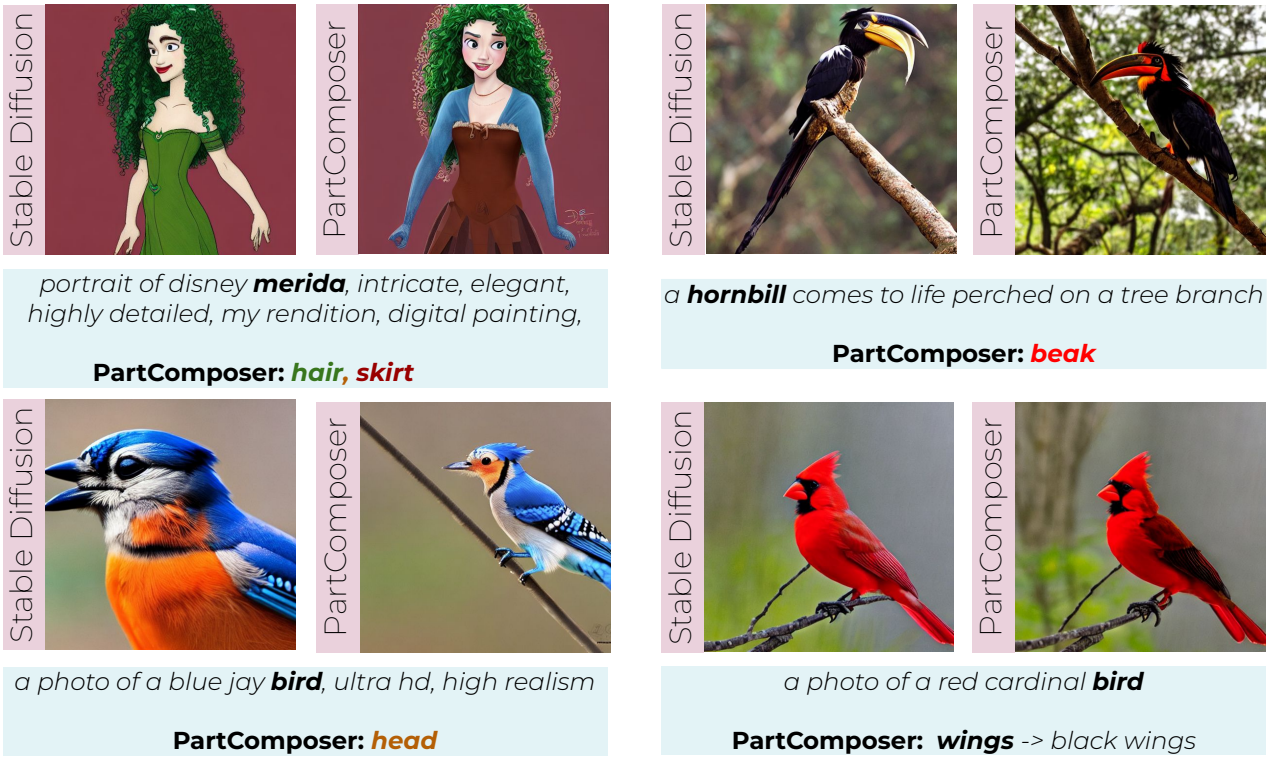


Figure 12. **Qualitative Results.** We provide a qualitative comparison of additional prompts with Stable Diffusion. We add the part attribute details to the prompt during generation for Stable Diffusion baseline. The PartComposer can generate the image following the specified part attribute details with better coherence and fewer artifacts compared to the baseline.

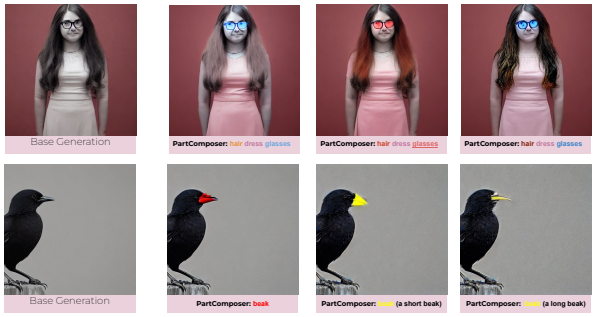


Figure 13. **Attribute Variations** for the base generated images are provided for the person portrait, where we change color of hair and eyeglasses, increase weight of eye glasses to change its shape. In the bird example, we generate various colors of beaks and generate its part variations by specifying part description in bracket.

D. Notations

We provide notations used in the paper in Table 3.

E. Potential Negative Impact

Text-to-image generative models have shown great promise in image generation and can be utilized in content and media creation. However, ensuring the created content is unbiased, harmless, and free of misinformation is essential. Our work gives users more control over text-to-image generation, which should be used responsibly to avoid misinformation.

F. Implementation Details

We discuss implementing the two parts of the PartComposer process, the Part Localization and Part Generation steps.

Part Localization. In Part Localization, we first discuss the implementation on StableDiffusion 2.1, where we use 50 steps of DDIM Scheduler [43] for denoising. As we aim to evaluate our method on CUB-200 and DeepFashion datasets, we first perform inversion using Null-Text Inversion [32] with a guidance scale of 0.05 (other hyper-parameters are kept default). We keep the α value linearly scaled with time steps from 0 to 0.5 (Eq. 1). To obtain the mask for each segment, we keep a low $\epsilon = 0.05$ threshold (Eq. 3), as here, we want to segment the region in all possible parts as the

Table 3. **Notation Table** for the paper.

Symbol	Meaning
$\mathcal{M}_o$	Mask of Object $\mathbf{o}$
$\hat{\mathbf{b}}, \hat{b}$	Set of Base Prompt Tokens
$\hat{\mathbf{p}}$	Set of Part Prompt Tokens
T_{th}	Threshold After Which The Part Diffusion Starts
$\hat{\mathbf{M}}_j$	Self Attention Map (j^{th} index)
$\hat{\mathbf{m}}_k$	Cross Attention Maps (for kth token)
$x \cdot y$	Dot Product Between x and y, viewed as vectors
$A \odot B$	Elementwise (Hadamard) product between A and B
$\mathbf{M}_{\mathbf{p}_i}$	Part Mask for the part $\mathbf{p}_i$
$\mathbf{a}_i$	Attribute Description of part $\mathbf{p}_i$
$f(\mathbf{p}_i, \mathbf{a}_i)$	Text Description of part $\mathbf{p}_i$ with attributes $\mathbf{a}_i$
x_t	Partial Denoised Image at time t
ϵ_t	Noise output for estimated noise at time t
D	Denoising U-Net
$\mathbf{M}_b$	Background (Other) Token Mask
α	Hyperparameter for the Part Diffusion Contribution
δ	Hyperparameter for Part Selection
ϵ	Hyperparameter for the Object Categorization to Background

DeepFashion and CUB-200 generate localization masks for the 14 part regions. However, for comparison with baselines producing segments in 4 parts, we perform the following clustering of parts in the four cluster regions (Table 4). We use (K=9) to cluster the different regions into 4 clusters. We use the foreground masks for objects provided by Choudhury *et al*. [12][3] for evaluating the FG-NMI and FG-ARI metrics in Table 4.

We use StableDiffusion (SD) 1.5 to compare the generation results to the Rich-Text [18][4] baseline. In this case, we want to start the localization process at a later stage of denoising, as we aimed the part diffusion not to alter the base generation too much; hence, start part denoising from step $T_{th} = 24$. We do merging of base with initial diffusion process for 0.5 number of steps (Eq. 5) and $\delta = 0.3$ for max part localization (in Eq. 2) respectively. We use a PNDM scheduler with 41 steps and an 8.5 guidance scale, as done by default in Rich-Text [18] generation.

Part Generation. We use the same scheduler and guidance scale for the part generation. We blend the base x_t^{base} and part generations x_t for 0.2 fraction of the time steps in the denoising process.

Baselines. For the Rich-Text baseline, we use the attribute properties $\mathbf{a}_i$ for the part token and add it to the base token if it is in the base prompt. In other cases, we add the Part information as the footnote of the object token in the base prompt. For the InstructPix2Pix [8] baseline, we iteratively add the instructions for each part on the base generation through. For the StableDiffusion baseline, we add the PartComposer instructions as the object's description in the prompt. We use the same StableDiffusion model with the same seed and guidance scale across baselines. We run it on the same Nvidia A100 40GB to ensure the sanity of comparison across methods. PartCraft [33], a recent method, also aims to generate objects based on part composition. However, if the text-to-image model is PartCraft trained for CUB-200, it can only generate variations of CUB-200. Further, it also requires annotations of the parts regarding key points, etc. Hence, it cannot generate parts based on text descriptions like our work. Hence, due to additional supervision and focus on particular datasets, it cannot be used as a general baseline for comparison with our work.

G. Analysis of PartComposer

In this section, we provide additional analysis regarding the design choices made in PartComposer. In particular, we ablate the components we have introduced in the PartDiffusion process of PartComposer (Sec. 3.2). We provide an analysis of the effect of using **a)** independent text embeddings, **b)** usage of dot product-based protocol in the part assignment, and **c)** the visualization of attention maps for the localized and unlocalized parts. We have used a subset of CUB-200 images to perform all evaluations, which is kept fixed across ablations. We tabulate the ablations in Table 5 (also including ablations from the main text Table 2) and provide visual results in Fig. 14. We observe that all components introduced in PartComposer significantly contribute to the overall segmentation performance of the part localization module. Further, in Fig. 14, we find that normalizing parts that do not satisfy localization conditions (forehead and back) leads to high attention values in most regions. This demonstrates the effectiveness of the max-value-based selection (Eq. 4) of parts proposed in PartComposer.

For comparison to the Stable Diffusion baseline, in addition to the results provided in Table 4, we provide a qualitative comparison in Fig. 15 and 16. The Stable Diffusion baseline assigns arbitrary masks to the wrong parts. On the contrary, if PartComposer localizes parts, they are often cor-

[3] https://github.com/subhc/unsup-parts
[4] https://github.com/SongweiGe/rich-text-to-image

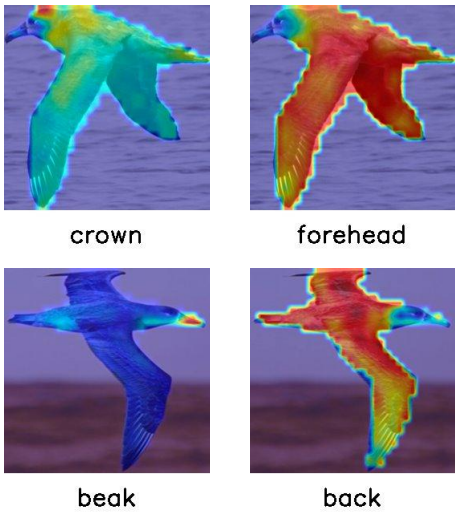
Table 4. **Part Names** in Clusters for CUB and DeepFashion Datasets

Cluster	CUB Part Names	DeepFashion Part Names
0	background	background
1	beak, forehead, left eye, right eye	cap, hair
2	breast, crown, nape, throat	dress, shirt (top) , accessories, outer
3	belly, left leg, right leg, tail	glasses, face, body
4	back, left wing, right wing	pants, footwear, leggings

Table 5. **Ablation Analysis** of PartComposer for Part Localization.

PartComposer (Ablations)		
Method	FG-NMI	FG-ARI
PartComposer	35.4	11.0
w/o Null-Text Inversion	23.1	5.2
w/o Max Localization	21.3	2.8
w/o Dot Product Localization	23.7	5.0
w/o Independent Text	31.2	8.9
PartComposer (Clustering)		
PartComposer (K = 9)	35.4	11.0
PartComposer (K = 4)	20.4	2.8
PartComposer (K = 14)	35.2	10.3

Figure 14. **Normalized Attention Maps** for parts which are localized (left) and non-localized (right).



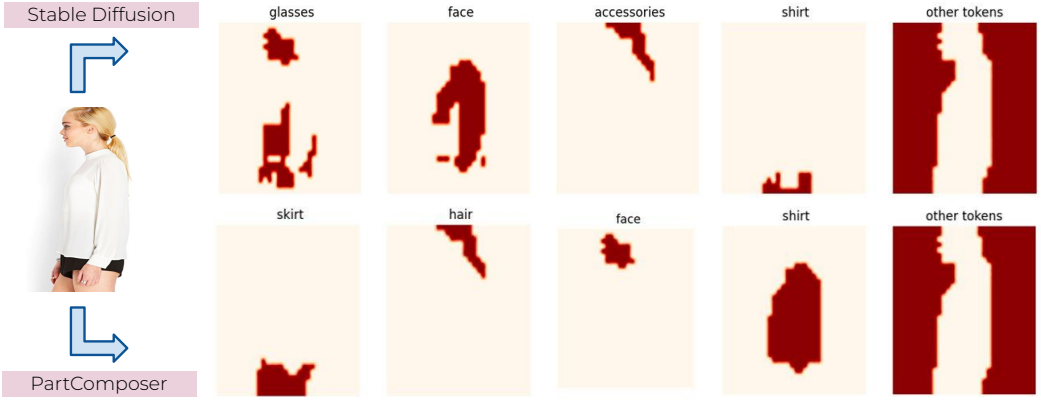


Figure 15. **Qualitative Comparison** of part masks (on DeepFashion) generated from the Stable Diffusion Baseline to PartComposer (Ours). The Stable Diffusion Baseline assigns arbitrary part masks to segments, whereas our part marks are consistent to part.

rectly associated with the right region of the object. This shows the advantage of using part diffusion rather than inducing additional tokens in the base prompt of StableDiffusion.

Robustnes of PartComposer w.r.t. Masks. PartComposer uses masked diffusion process to generate and compose the object parts. Hence, we first see the effect of the mask region, where we observe (Fig. 17) that in cases where the mask occupies more region than the desired part, the part diffusion process mostly modifies the requested part. This demonstrates that PartComposer can still generate desired aesthetic outputs in case the masks are not very accurate.

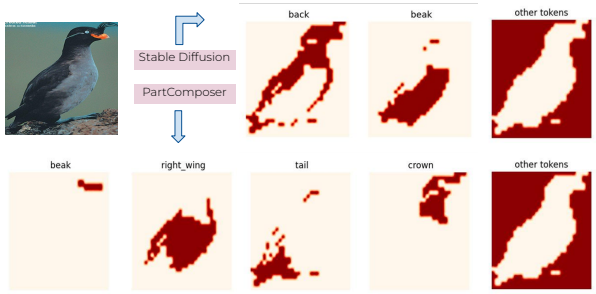


Figure 16. **Qualitative Comparison** of part masks (on CUB200) generated from the Stable Diffusion Baseline to PartComposer (Ours). The Stable Diffusion Baseline assigns arbitrary part masks to segments, whereas our part marks are consistent to part.

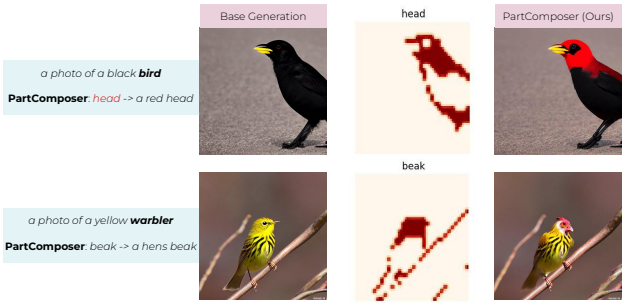


Figure 17. **PartComposer is robust when the masks don't exactly fit** the specified parts. This is because the text description for the masked region contains a description only of specified parts (*e.g.* beak and head above). Hence, the other regions, despite being in the masked region, remain similar to the base generation.

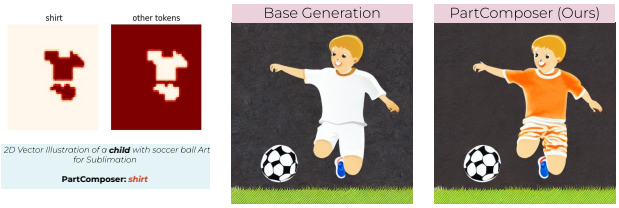


Figure 18. **Limiting Case.** When only color is specified in PartComposer, and the masks cover extraneous regions. In such cases, when only color guidance is applied, it might leak to some other parts instead of the specified part ('shirt'). As color is added through gradient guidance in the entire masked region.

H. Limitations of PartComposer Generations

We find that the color guidance applies to the entire masked region. Hence, in some cases, the font color specification $\mathbf{a}_c$, can alter other areas in minor ways besides the specified part. This is due to color gradient loss, also used in Rich-Text [18] to specify the specific color of the region. We highlight that in the example in Fig. 18, where some part of the orange lines appear in the shorts worn by the child as it was also the part of the mask. We find that mask localization is the bottleneck for performance. Hence, improving the part understanding and segmentation capability of the Text-to-Image models is an important direction to be pursued across future works, including explorations on new models like Flux and SD3.5.