

# Appendix to “GPSToken: Gaussian Parameterized Spatially-adaptive Tokenization for Image Representation and Generation”

1 In this appendix, we provide the following materials:

- 2 **A** Spatially-adaptive token initialization and Gaussian calibration algorithms (referring to Sec. 3.2  
3 and Sec. 3.3 in the main paper);
- 4 **B** Experimental settings for reconstruction and generation (referring to Sec. 4.1 in the main paper);
- 5 **C** Ablation studies on spatial adaptivity designs (referring to Sec. 4.2 in the main paper);
- 6 **D** More visual comparisons and results on reconstruction and generative tasks (referring to Sec. 4.2  
7 and Sec. 4.3 in the main paper);
- 8 **E** Broader impacts of GPSToken and its generator.

## 9 **A Spatially-adaptive Token Initialization and Gaussian Calibration**

10 The spatially-adaptive token initialization algorithm is described in Sec. 3.1 of the main paper. It  
11 outlines a procedure for segmenting the entire image based on regional complexity and for initializing  
12 the Gaussian parameters of the GPS-tokens accordingly. The algorithm is summarized in Algorithm 1.

---

### Algorithm 1: Spatially-adaptive Token Initialization Algorithm

---

**Input:** image  $I$ ; target token count  $l$ ; metric hyper-parameter  $\lambda$ ; minimal size of region  $s_{\min}$ .

**Output:** regions list  $L$ ; initialized Gaussian parameters  $\{\mathbf{g}_0^{init}, \dots, \mathbf{g}_{l-1}^{init}\}$ .

---

```

1 Initialize region candidate list  $L = \{I\}$ ;
2 while  $|L| < l$  do
3   Calculate complexities  $\{m_0, \dots, m_{|L|-1}\}$  for regions in  $L$  using Eq. 4 in the main paper;
4   Let  $\hat{L} = \{I_i \in L \mid \text{at least one side of } I_i \text{ is greater than } s_{\min}\}$ ;
5   Let  $I_{\max} = \arg \max_{I_i \in \hat{L}} m_i$ ;
6   Obtain size  $(w, h)$  of  $I_{\max}$ ;
7   if  $w \neq h$  then
8     Equally divide  $I_{\max}$  into two sub-regions  $I_1$  and  $I_2$  along the longer side;
9     Update  $L$  by replacing  $I_{\max}$  with  $\{I_1, I_2\}$ ;
10  else
11    // step1: width-wise division -> compute complexities
12    Divide  $I_{\max}$  into  $I_1, I_2$  along width;
13    Calculate complexities  $m_1, m_2$  for  $I_1, I_2$  using Eq. 4 in the main paper;
14    // step2: hidth-wise division -> compute complexities
15    Divide  $I_{\max}$  into  $I_3, I_4$  along height;
16    Calculate complexities  $m_3, m_4$  for  $I_3, I_4$  using Eq. 4 in the main paper;
17    // step3: compare the complexities
18    if  $\min(m_1, m_2) \leq \min(m_3, m_4)$  then
19      Update  $L$  by replacing  $I_{\max}$  with  $\{I_1, I_2\}$ ;
20    else
21      Update  $L$  by replacing  $I_{\max}$  with  $\{I_3, I_4\}$ ;
22    end
23  end
24 end
25 Obtain size  $(w_i, h_i)$  and center  $(x_i, y_i)$  for each region  $I_i \in L$ ;
26 Initialize  $\mathbf{g}_i^{init} = \{\sigma_x^{(i)}, \sigma_y^{(i)}, \rho^{(i)}, \mu_x^{(i)}, \mu_y^{(i)}\} = \{\frac{w_i}{6}, \frac{h_i}{6}, 0, x_i, y_i\}$ .
```

---

13 The Gaussian calibration algorithm is described in Sec. 3.2 of the main paper. It is used to refine the  
14 predicted Gaussians during the layout synthesis phase. The algorithm is summarized in Algorithm 2.

---

**Algorithm 2:** Gaussian Calibration Algorithm

---

**Input:** predicted Gaussian parameters  $\{\mathbf{g}_0, \dots, \mathbf{g}_{l-1}\}$ ; minimal size of region  $s_{min}$ ; image size  $W \times H$ .

**Output:** calibrated Gaussian parameters  $\{\mathbf{g}_0^{cal}, \dots, \mathbf{g}_{l-1}^{cal}\}$ .

```
1 // step 1: calibrate the means  $(\mu_x, \mu_y)$ 
2 Let  $\Delta x = \Delta y = s_{min}$ ;
3 Define grid  $\mathcal{G}$  as all points in  $[0, W-1] \times [0, H-1]$  spaced by  $(\Delta x, \Delta y)$ ;
4 Quantize each mean  $(\mu_x^{(i)}, \mu_y^{(i)}) \in \mathbf{g}_i$  to the nearest central point in  $\mathcal{G}$ , obtaining  $(\hat{\mu}_x^{(i)}, \hat{\mu}_y^{(i)})$ ;
5 // step 2: re-initialize sigmas via region partitioning
6 Initialize region list  $L = \{(0, W, 0, H)\}$ ;
7 while  $|L| < l$  do
8   Let  $m_i$  be the count of quantized means in region  $I_i \in L$ ;
9   Let  $\hat{L} = \{I_i \in L \mid \text{at least one side of } I_i \text{ is greater than } s_{min}\}$ ;
10  Let  $I_{max} = \arg \max_{I_i \in \hat{L}} m_i$ ;
11  Let  $x1, x2, y1, y2 \leftarrow I_{max}$  and  $(\hat{w}, \hat{h}) = (x2-x1, y2-y1)$ ;
12  if  $\hat{w} > \hat{h}$  then
13    | Replace  $I_{max}$  in  $L$  with its two equal sub-regions split vertically along the x-axis;
14  elif  $\hat{w} < \hat{h}$  then
15    | Replace  $I_{max}$  in  $L$  with its two equal sub-regions split horizontally along the y-axis;
16  else
17    | if no Gaussian falls exactly on the split line  $x = (x1 + x2)/2$  then
18      | Replace  $I_{max}$  in  $L$  with its two equal sub-regions split vertically along the x-axis;
19    else
20      | Replace  $I_{max}$  in  $L$  with its two equal sub-regions split horizontally along the y-axis;
21    end
22  end
23 end
24 Obtain size  $(w_i, h_i)$  and center  $(x_i, y_i)$  for each region  $I_i \in L$ ;
25 Compute  $\mathbf{g}_i^{cal} = \{\sigma_x^{cal-(i)}, \sigma_y^{cal-(i)}, \rho^{cal-(i)}, \mu_x^{cal-(i)}, \mu_y^{cal-(i)}\} = \{\frac{w_i}{6}, \frac{h_i}{6}, 0, x_i, y_i\}$ .
```

---

## B Experimental Settings

**Training and Inference Settings.** For the *image reconstruction task*, we train the encoder-decoder framework for 1M steps with a batch size of 96. The model is first trained using only the reconstruction loss  $L_{rec}$  for the initial 600K steps. Subsequently, the perceptual loss  $L_{perc}$  and adversarial loss  $L_{adv}$  [1] are incorporated for the remaining 400K steps to enhance texture details. We use the Adam optimizer [3] with a fixed learning rate of  $5 \times 10^{-5}$ . Additionally, we apply an exponential moving average (EMA) with a decay rate of 0.9999 to stabilize the training process. We set  $s = 5$  for Eq. 2 (in the main paper),  $\lambda = 2.5$  for Eq. 4 (in the main paper) and  $s_{min} = 4$  for Algorithm 1.

For the *image generation task*, we train the layout synthesis model for 500K iterations and the conditional layout-to-texture generation model for 1.5M iterations. Both models are trained with a batch size of 256 and a learning rate of  $1 \times 10^{-4}$  using the Adam optimizer. All experiments are conducted on eight A100 GPUs. During inference, we use a 5-step ODE sampler [4] to predict  $\mathbf{g}^{init}$ , followed by a 250-step SDE sampler [4], as used in SiT [4], for texture synthesis. We set the classifier-free guidance strength [2] to 1.5, following common practice. We set  $s_{min} = 4$  for Algorithm 2.

**Network Architecture.** In the *image reconstruction task*, the encoder architecture comprises two residual blocks for extracting image features, followed by 30 transformation blocks designed to process initial Gaussian parameters and extract textual features for each region. During the rendering stage, GPS-tokens are mapped into  $64 \times 64$  2D feature maps. The decoder adopts the same architecture as the last three stages of the SDXL-VAE [5] decoder but with double channels. GPSToken-M128 utilizes 128 tokens, each with 16 channels, whereas GPSToken-L256 employs 256 tokens, each with 32 channels, to match the capacity of VAAE [6]. GPSToken-S64 uses only 64 tokens, each with 16 channels, but incorporates 60 transformation blocks within the encoder.

Table 1: Ablation studies of our spatial adaptivity designs on the  $256 \times 256$  reconstruction task.  $\checkmark$  indicates that the component is used. “Init.” and “Refine.” denote the spatially-adaptive token initialization and spatially-adaptive token refinement, respectively.

Method	Components		sample-level			distribution-level			
	Init.	Refine.	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	rec. FID $\downarrow$	rec. sFID $\downarrow$	FID $\downarrow$	sFID $\downarrow$
<b>baseline</b>			23.52	0.638	0.110	1.02	4.07	2.59	4.34
<b>baseline+</b>		$\checkmark$	24.00	0.654	0.100	0.81	3.59	2.37	4.31
<b>GPSToken</b>	$\checkmark$	$\checkmark$	24.06	0.657	0.080	0.65	3.28	2.18	3.96

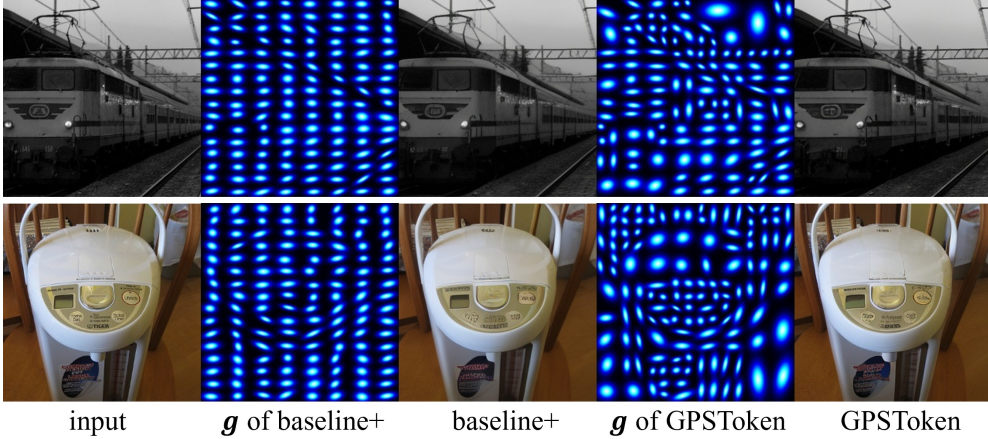


Figure 1: **Illustration of “baseline+” and GPSToken.** Left to right: the input image, visualization of Gaussians of “baseline+”, the reconstruction of “baseline+”, visualization of Gaussians of GPSToken, the reconstruction of GPSToken.

For the *image generation task*, we adopt the official SiT-S and SiT-XL architectures [4] as our backbone models. Specifically, SiT-S is utilized for layout synthesis, while SiT-XL is employed for conditional layout-to-texture generation. To support layout conditions, we use MLPs to transform these conditions before adding them into each attention block in SiT-XL.

## C Ablation Studies on the Spatial Adaptivity Designs

As described in Sec. 3.2 of the main paper, our GPSToken employs spatially-adaptive token initialization (“Init.”) followed by spatially-adaptive token refinement (“Refine.”) to progressively obtain coarse- and fine-grained spatial adaptations based on regional textures. We conduct ablation studies on GPSToken-M128 to validate the contribution of each component.

Table 1 presents the quantitative results. The baseline refers to the method that uses Gaussian-parameterized tokens without incorporating any spatially-adaptive components. The term “baseline+” denotes the method that additionally includes the “Refine.” component. In contrast, GPSToken integrates both the “Init.” and “Refine.” components. As shown in the table, “baseline+” yields improvements over the baseline across both sample-level and distribution-level metrics, with a decrease of 0.01 in LPIPS and a decrease of 0.21 in “rec. FID”. These enhancements indicate the general improvement achieved by adjusting Gaussians to match local textures. Compared to “baseline+”, GPSToken significantly improves distribution-level metrics, achieving reductions of 0.19 in FID and 0.35 in sFID, while showing slight improvements in sample-level metrics (an increase of 0.06 in PSNR and 0.003 in SSIM). This demonstrates the effectiveness of “Init.” component, which reallocates more Gaussians from simple regions to complex ones, thereby capturing finer semantic details in texture-rich regions without compromising reconstruction performance.

As shown in Fig. 1, without the “Init.” component, the Gaussian maps from “baseline+” still roughly align in a 2D grid, even after refinement. This limits their ability to fit complex textures, only capturing edges in simple regions. In contrast, with the “Init.” component, GPSToken aggregates



Figure 2: Visual comparisons on  $256 \times 256$  reconstruction task.

more Gaussians in texture-rich areas, making it better suited to fit complex structures. This highlights the importance of “Init.” component in achieving spatially adaptive representation of fine-grained visual contents.

## D More Visual Results

### D.1 Results for Reconstruction Task

**Visual Comparisons.** We provide visual comparisons among GPSToken and its competitors in Figs. 2, 3 and 4. It can be observed that our GPSToken achieves significantly more accurate and clearer textures in complex regions, without compromising the performance in simpler areas.

**More Visual Results.** Further visual results of our spatially adaptive designs are presented in Fig. 5. Fig. 6 illustrates the adjustment of the initial Gaussian parameters  $\mathbf{g}^{\text{init}}$  to better focus on the regions of interest. Fig. 7 shows the flexibility to adjust token counts during inference, demonstrating the adaptability of our approach under varying length.





Figure 3: **Visual comparisons on  $256 \times 256$  reconstruction task.**

## 74 D.2 Results for Generation Task

75 Fig. 8 shows a few images generated by our two-stage generator. One can see that our generator is  
 76 capable of synthesizing natural images depicting a wide variety of scenes. For instance, it successfully  
 77 generates fine details in objects such as beetles, eagles, trucks, bags, mailboxes, golf balls, dinosaur  
 78 fossils, and so on. Furthermore, the generated images exhibit high visual quality - characterized by  
 79 sharp details and realistic textures - demonstrating the generator’s strong ability to synthesize diverse  
 80 and photo-realistic images.



Figure 4: Visual comparisons on  $256 \times 256$  reconstruction task.





Figure 5: More visual results of spatial adaptivity.



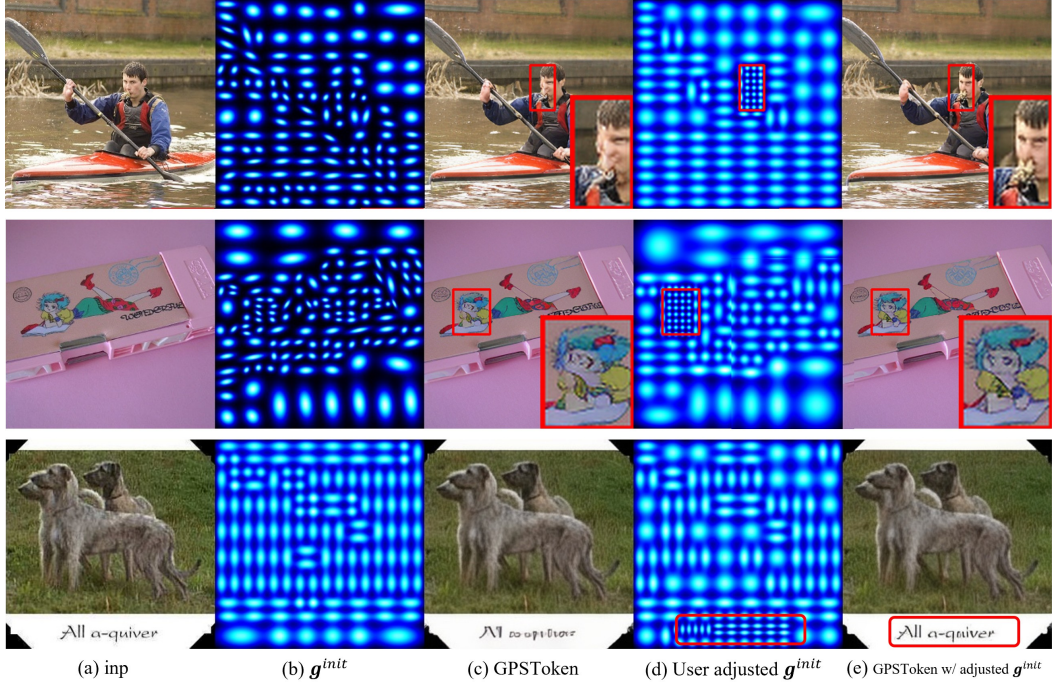


Figure 6: More visual results on User-Controllable Adjustment of  $g^{init}$ .

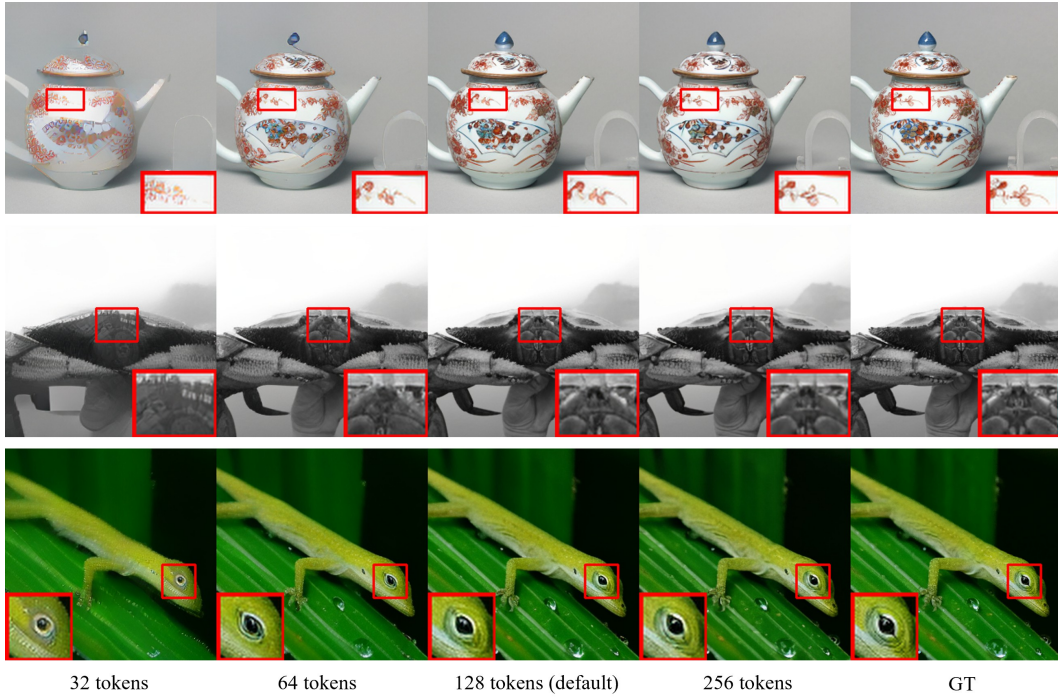


Figure 7: More visual results on Adjustment of Token Count at Inference.



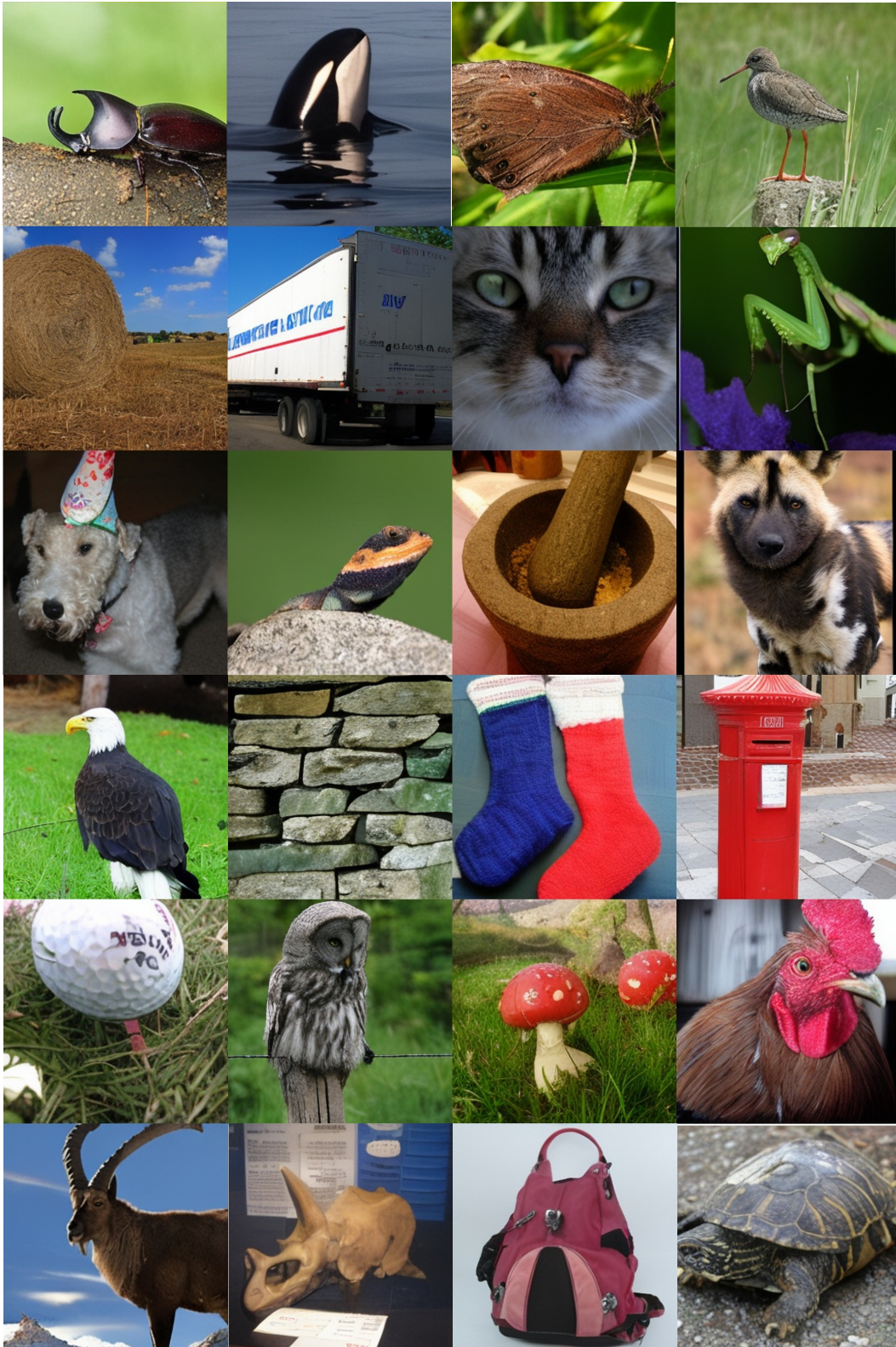


Figure 8: Visual result of  $256 \times 256$  generation.

## 81 E Broader Impacts

82 This work introduces GPSToken, a spatially-adaptive tokenization framework designed to enable  
83 efficient and content-aware image representation. By offering flexible feature modeling, GPSToken  
84 enhances representational capacity, benefiting both computer vision researchers and downstream  
85 applications in domains such as medical imaging and creative design. Furthermore, its two-stage  
86 layout-texture synthesis approach reduces computational barriers for generative tasks, making it  
87 accessible to individual users and small companies.

88 Despite its potential, the deployment of GPSToken also presents several risks. The ability to gen-  
89 erate high-quality synthetic media may be misused, potentially harming vulnerable populations  
90 through the spread of misinformation or deepfake technologies. Additionally, if trained on biased  
91 datasets, the model may encode disparities in texture and shape representation, which could com-  
92 promise fairness - particularly in sensitive applications such as facial recognition. In safety-critical  
93 domains like autonomous driving or medical diagnosis, failures in accurate tokenization could lead to  
94 misinterpretation of complex visual scenes, with potentially dangerous consequences.

## 95 References

- 96 [1] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution  
97 image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern*  
98 *recognition*, pages 12873–12883, 2021.
- 99 [2] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint*  
100 *arXiv:2207.12598*, 2022.
- 101 [3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*  
102 *arXiv:1412.6980*, 2014.
- 103 [4] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and  
104 Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant  
105 transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.
- 106 [5] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe  
107 Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image  
108 synthesis. *arXiv preprint arXiv:2307.01952*, 2023.
- 109 [6] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimiza-  
110 tion dilemma in latent diffusion models. *arXiv preprint arXiv:2501.01423*, 2025.