A TODDLERDIFFUSION METHOD (MORE DETAILS)

A.1 DERIVATIONS

From the forward process, we have:

$$x_t = (1-\alpha_t)x_0 + \alpha_t y + \sigma_t \epsilon_t \tag{9}$$
$$x_{t-1} = (1-\alpha_{t-1})x_0 + \alpha_{t-1} y + \sigma_{t-1} \epsilon_{t-1} \tag{10}$$

From equation 10, we get:

$$x_0 = \frac{1}{1-\alpha_{t-1}} x_{t-1} - \frac{\alpha_{t-1}}{1-\alpha_{t-1}} y - \frac{\sigma_{t-1}}{1-\alpha_{t-1}} \epsilon_{t-1} \tag{11}$$

Inserting equation 11 into equation 9, we get:

$$\begin{aligned} q(x_t \mid x_{t-1}, x_0) = \frac{1-\alpha_t}{1-\alpha_{t-1}} x_{t-1} - \frac{(1-\alpha_t)\alpha_{t-1}}{1-\alpha_{t-1}} y \\ + \alpha_t y - \frac{(1-\alpha_t)\sigma_{t-1}}{1-\alpha_{t-1}} \epsilon_{t-1} + \sigma_t \epsilon_t \end{aligned} \tag{12}$$

The reverse process follows:

$$q\left(x_{t-1} \mid x_t, x_0, y\right) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}\left(x_t, x_0, y\right), \tilde{\sigma}_t^2 \mathbf{I}\right) \tag{13}$$

Using Bayes' rule:

$$q\left(x_{t-1} \mid x_t, x_0, y\right) = q(x_t \mid x_{t-1}, x_0) \frac{q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)} \tag{14}$$

Plugging equation 9, equation 10, and equation 12 into equation 14, yields:

$$\begin{aligned} \tilde{\mu}_t(x_t, x_0, y) = \frac{\sigma_{t-1}^2}{\sigma_t^2} \frac{1-\alpha_t}{1-\alpha_{t-1}} x_t + (1-\alpha_{t-1}\left(1 - \frac{\sigma_{t-1}^2}{\sigma_t^2} \frac{(1-\alpha_t)^2}{(1-\alpha_{t-1})^2}\right)) x_0 \\ + (\alpha_{t-1} - \alpha_t \frac{(1-\alpha_t)}{(1-\alpha_{t-1})} \frac{\sigma_{t-1}^2}{\sigma_t^2}) y \end{aligned} \tag{15}$$

$$\tilde{\sigma}_t^2 = \sigma_{t-1}^2 - \frac{\sigma_{t-1}^4}{\sigma_t^2} \frac{(1-\alpha_t)^2}{(1-\alpha_{t-1})^2} \tag{16}$$

Finally, we can apply the DDIM non-Markovian forward processes.
In return, only the x_0 coefficient will be affected.

A.2 ARCHITECTURE DETAILS

Figure 11 depicts the details of our architecture. We adopt the LDM architecture, which consists of two main blocks: 1) VQGAN Encoder-Decoder to convert the image into low-cost latent space. 2) Unet that responsible for predicting x_{t-1}. Figure 11 shows that the condition y is encoded into x_t. However, it could be optionally concatenated to x_t. In addition, shown in orange and green are the σ^2 and α, respectively. Consequently, Figure 12 depicts an abstract flow for the sampling mode, where we show how our approach works starting from random noise till generating the final RGB image.

A.3 1^{st} STAGE: ABSTRACT STRUCTURE

Sketch FID. A discrepancy exists between the reported conventional FID (RGB-FID), trained on ImageNet (Deng et al., 2009), and qualitative results, as illustrated in Figure 13. This discrepancy (Ge et al., 2020) may arise from differences between the training data (RGB images) and the evaluation

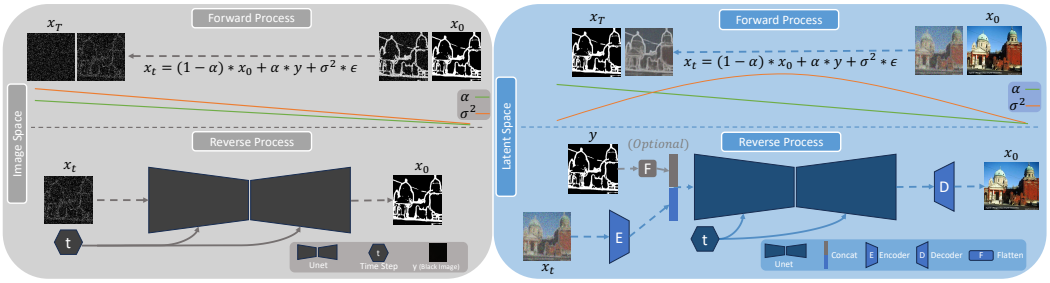


Figure 11: An overview of proposed architecture, dubbed ToddlerDiffusion. The left block demonstrates the first stage, which generates a sketch unconditionally. Due to our efficient formulation, this stage operates in the image space on 64×64 resolution. The right module depicts the third stage, which generates an RGB image given a sketch only or both sketch and palette.

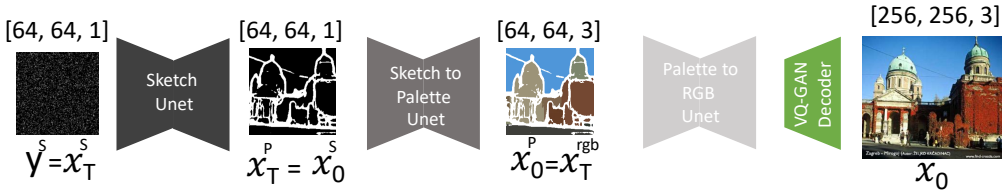
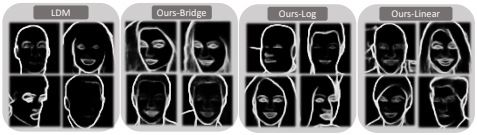


Figure 12: An abstract flow for the sampling mode, where we show how our approach works starting from random noise till generating the final RGB image.



	Method	Unet Parameters	Training Time (Min:Sec)↓	Sampling Time (FPS)↑	Noise Scheduler	Sketch FID ↓	RGB FID ↓
a)	LDM	187 M	4:05	2.2	Linear	126.4	23.5
b)	Ours	1.9 M	0:18	53	Bridge	110.5	35.2
c)	Ours	1.9 M	0:18	53	Log	61.2	15.6
d)	Ours	**1.9 M**	**0:18**	**53**	Linear	**58.8**	**15.1**

Figure 13: Benchmarking results for the $1^{st} stage$ on the CelebA-HQ dataset (Karras et al., 2017). The training time is calculated per epoch using 4 NVIDIA RTX A6000. The sampling time is calculated per frame using one NVIDIA RTX A6000 with a batch size equals 32.

data (binary images). To bridge this gap, we introduce Sketch-FID by re-training the inception model (Szegedy et al., 2015) on a sketch version of the ImageNet dataset. We generate sketches for ImageNet RGB images using PidiNet (Su et al., 2021) and train the inception model on the conventional classification task.

Noise Scheduler. In the 1^{st} stage, where our starting point y is a black image (Section 2.1), designing an appropriate noise scheduler is crucial. The bridge noise scheduler is intuitively unsuitable, as it eliminates randomness by adding no noise at both edges, fixing the starting point to a black image. This hypothesis is supported by empirical results in Figure 13, row b, where the model outputs random patterns. We explored linear and logarithmic schedulers, finding the linear schedule superior, yielding Sketch-FID scores of 15.19 and 18.47, respectively (Figure 13, rows c-d).

Ours vs. LDM. In Section 2.1, we propose an alternative way to generate sketches by leveraging LDM (Rombach et al., 2022), as shown in Figure 3, row A. However, this approach deviates from the nature of sketching. Our proposed formulation, aligned with the topology of sketches (Figure 3, row C), resulted in significant improvements over LDM in both model complexity and performance, as depicted in Figure 13. Our formulation (Section 2.1) allows direct operation on the image space (64×64) and compression of the Unet to a tiny variant without sacrificing performance. Despite the aggressive compression, our performance is significantly better than LDM, with respective Sketch-FID scores of 15.19 and 49, using a 41x smaller network.

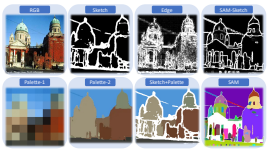
Sketch Intensity. Controlling the intensity is crucial. Thus, we conducted further analysis, as shown in Table 3, that probes for best performance, the intensity should follow the data distribution. We refer to the intensity as the ratio between the white pixels to the black ones in the sketch. The reported FID

Table 3: Sketch Intensity Analysis.

Dataset/Intensity	19%	24%	31%	37%	43%
CelebHQ	**7.4**	7.9	9.2	10.5	12.7
Lsun-Churches	7.8	**6.9**	8.3	8.9	10.1

Figure 14: Palette generation pipeline. First, we employ FastSAM (Zhao et al., 2023b) to segment the image. Then, we get the dominant color for each segment.

Figure 15: Ablation study for different input's types for the 2^{nd} stage on LSUN-Churches dataset (Yu et al., 2015).



	Input Type	FID ↓
a)	Edges	9.5
b)	Edges + Palette-2	8.4
c)	Sketch	8.6
d)	Sketch + Palette-1	8.3
e)	Sketch + Palette-2	**7.1**
f)	SAM (Colored)	13.1
g)	SAM (Edges)	8.1

Table 4: Comparison of the LDM concatenation mechanism against the schrödinger bridge (SB) using CelebHQ dataset.

2^{nd} Stage	FID ↓
Concat	10.04
SB	9.45
SB + Concat	**8.10**

Table 5: Training objective ablation study on CelebHQ dataset (Karras et al., 2017).

Epochs	LDM		Ours	
	ϵ	x_0	$(x_t - x_0)$	x_0
50	12.7	15.7	8.5	8.1
100	10.0	14.6	8.4	7.8
200	8.7	15.1	8.2	**7.7**

scores in Table 3 are the overall FID using all stages, and the intensity numbers are the initial intensity at $t = T$. The GT intensity for CelebHQ and Lsun-Churches are 19.2% and 23.5%, respectively.

A.4 2^{nd} Stage: Palette

As depicted in Figure 14, we introduce a more realistic way to have high-quality palettes for free by first generating the segments from the RGB image using FastSAM (Zhao et al., 2023b). Then, a simple color detector module is utilized to get the dominant color, e.g., the median color per segment.

A.5 Training and Sampling Algorithms

Algorithm 1 Training Pipeline

for $i = 1, \dots, N$ **do** ▷ Train each stage separately
  for $j = 1, \dots, E$ **do** ▷ Train for E epochs
    for $(x_0^i, y^i) \in \mathcal{D}$ **do** ▷ Loop on the dataset $\mathcal{D}$
      $\epsilon \in \mathcal{N}(0, I)$ ▷ Sampling the noise
      $t \in U(1, \dots, T)$ ▷ Sampling time-step
      Get x_0^i by applying Eq. 3 ▷ $\tilde{x}_0$ based on the stage
      $x_t = \alpha_t x_0^i + (1 - \alpha_t) y^i + \sigma_t^2 \epsilon_t$ ▷ Forward process
      $\nabla_\theta \left\| \tilde{x}_0^i - p_\theta(x_0^i | x_t, y^i) \right\|^2$ ▷ Simplified objective

Algorithm 2 Sampling Pipeline

$x_{inter} = []$ ▷ Initialize the list to save intermediate output for each stage
$y^1 = Zeros((H, W, 1))$ ▷ Initialize 1^{st} condition as a black image
$\epsilon \in \mathcal{N}(0, I)$ ▷ Sampling the noise once
for $i = 1, \dots, N$ **do** ▷ Sequential Sampling
  $x_T = y^i + \sigma_T^2 \epsilon_T$ ▷ Run forward process once
  for $t = T, \dots, 1$ **do** ▷ Progressive Sampling
    Get x_{t-1} by applying Eq. 6 $: x_0^i = p_\theta(x_0 | x_t)$ ▷ Reverse process
  $y^{i+1} = x_0^i$ ▷ Update the condition for the next stage
  $x_{inter}.append(x_0^i)$ ▷ Store the intermediate outputs for each stage
return x_{inter}

A.6 Ablations

1^{st} **Stage Representation.** In Section 2.2, we explore the versatility of the 3rd stage (detailed image) by examining six input modalities, detailed in Figure 15. Comparing different contours representations, namely Edges (using Laplacian edge detector (Wang, 2007)), Sketch (utilizing PidiNet (Su et al., 2021)), and SAM-Edges (generated by SAM followed by Laplacian edge detector), we find that Sketch outperforms Edges, as edges tend to be noisier. However, SAM-Edges provides more detailed contours, yielding superior results. Notably, feeding SAM-Colored leads to significant performance degradation, likely due to color discrepancies, as observed in SAM in Figure 15. While SAM-Edges achieves optimal results, its computational intensity renders it impractical. In contrast, Sketch and Edges are computationally inexpensive. Furthermore, Sketch's sparse and user-friendly

nature makes it more suitable for editing, facilitating interpretation and modification than dense, noisy edges. Consequently, we adopt Sketch as the input modality for subsequent experiments.

Palette Effect. Adding more guidance will offer more editing abilities to the pipeline and enhance the performance. As shown in Figure 15, rows b and d, when we incorporate the palette into the contours, i.e., edges and sketch, the performance improved by almost 1 and 1.5, respectively. In addition, Palette-2 outperforms Palette-1 by 1 FID score. A more detailed analysis of fusing the stages together and how to mitigate the error accumulation can be found in Appendix A.7.

Concatenation *v.s.* Schrödinger Bridge. We formulate the generation problem as image-to-image translation. For instance, generating a palette given sketch or generating the final RGB image given overlaid palette and sketch. One possible approach is to utilize the conditioning mechanisms offered by LDM (Rombach et al., 2022), e.g., cross-attention and concatenation. However, these mechanisms are inefficient as their SNR tends to 0 as $t \to T$. In contrast, our formulation follows the Schrödinger bridge, which directly incorporates the condition y into x_t during the forward process. Accordingly, the SNR$\gg 0$ at $t = T$, which allows reducing training steps from 1K to 50 (Figure 5) and achieving faster convergence (Figure. 4). Additionally, Table 4 demonstrates a quantitative comparison between Schrödinger Bridge and concatenation. As shown in Table, 4, the schrödinger bridge (SB) combined with the concatenation mechanism achieves the best results; however, SB alone is better than using the naive concatenation.

Training Objectives. Previous work (Rombach et al., 2022) showed that learning the noise ϵ directly enhances the performance compared to predicting the x_0. As, intuitively, predicting the difference $\epsilon_t \propto x_t - x_0$ is easier than predicting the original complicated x_0. However, this is not valid in our case due to the new term y, which makes predicting the difference much harder, as discussed in Section 2.1. Table 5 shows the same conclusion reported in (Rombach et al., 2022), that predicting the difference ϵ leads to better performance than x_0, where the FID score is 8.7 and 15.1, respectively. On the contrary, our method achieves almost the same performance regardless of the training objective, which is expected, as predicting the difference involves predicting x_0, Eq. 8.

A.7 How to Fuse different Stages Efficiently?

Table 6: Systematic search for the best stopping step s for the condition truncation.

Metric/Steps (s)	0	10	20	40	80	120	160	200
2^{nd} stage FID ↓	**6.1**	7.4	7.9	8.6	9.9	10.4	10.8	11.2
Overall FID ↓	11.6	10.7	**9.5**	10.9	11.2	13.5	15.7	18.9

Table 7: Ablation study of the sketch augmentation effect on the overall performance after fusing the abstract and the detailed stages.

	Cutout (DeVries & Taylor, 2017) Percentage	Dropout Percentage	2^{nd} Stage FID ↓	Overall FID ↓
a)	0	0	8.6	16.10
b)	5-10	5-20	9.89	13.94
c)	10-20	20-40	9.77	13.98
d)	20-30	50-70	9.80	**13.76**
e)	30-40	70-90	11.68	17.99

The reported FID scores for the 1^{st} and the 2^{nd} stages, in Figure 13 and Figure 15, respectively, are for each stage separately. In other words, in Figure 15, row c, the 2^{nd} stage achieves an 8.6 FID score when a GT sketch is fed, which is obtained from PidiNet (Su et al., 2021). However, when we fed the generated sketch from the 1^{st} stage, the performance drastically dropped from 8.6 to 16.1 (almost doubled), as shown in Table 7, row a, due to the domain gap between the generated and the GT sketches. To fill this gap, we explored two types of augmentations: 1) Cutout and Dropout augmentation. 2) Condition truncation augmentation.

Cutout and Dropout Augmentation. First, we explored the straightforward augmentation types, such as Cutout (DeVries & Taylor, 2017) and Dropout augmentations. For the Cutout, we apply a kernel to randomly blackout patches in the sketch. Additionally, regarding the dropout augmentation, we randomly convert white pixels to black pixels, interpreted as dropping some white points from the sketch. As shown in Table 7, generally, augmentation leads to a drop in the 2^{nd} stage performance while helping fill the gap in the overall performance, as the FID score decreased from 16 to almost 14. However, as shown in rows b-d, gradually increasing the amount of the applied augmentation does not help much, as the performance remains almost the same. However, the overall accuracy, i.e., FID, drops significantly from 14 to 18 when an aggressive augmentation is applied.

Condition Truncation Augmentation. As shown in Table 7, the conventional augmentation techniques do not help much. Accordingly, we explored another augmentation variant tailored for the

diffusion models, i.e., condition truncation (Ho et al., 2022). During training the 2^{nd} stage, we apply a random Gaussian noise to the fed condition, i.e., the sketch. So now the 2^{nd} stage is trained on noisy sketches instead of pure ones, which makes it more robust to the variations between the real sketches and the generated ones from the 1^{st} stage. Typically, we progressively generate the sketch in T time steps; $T \rightarrow 0$. However, This added noise could be interpreted as we stop the sketch generation (1^{st} stage) at a particular step s; $T \rightarrow s$. Consequently, we search for it during sampling to determine which step s works best for the overall performance, as shown in Table 6. In other words, the 2^{nd} stage is trained on a wide range of perturbed sketches, e.g., pure sketch ($s = 0$), noisy one ($0 < s < T$), and even pure noise ($s = T$). The 1^{st} stage is trained for 200 steps, so $s = 200$ means we omit the sketch and feed pure noise. In contrast, $s = 0$ indicates that we feed the generated sketch as it is to the $2^n d$ stage. As shown in Table 6, following this truncation mechanism, the fusion of the two stages is performed more efficiently, where the overall performance drops from 16.1 to 10.6.

A.8 Conditioning Mechanism Error Vulnerability

In this section, we will compare the two conditioning mechanisms, Concatenation (Concat) and Schrödinger Bridge (SB), in terms of the error vulnerability.

While SB+Concat shows superior performance in Table 4 for conditional generation—when the input condition y is the ground truth (GT)—its performance diminishes in the unconditional setup.

In the unconditional setup, y is generated from the previous stage rather than being a perfect GT. This introduces errors into the generated y, leading to error accumulation, as discussed in Appendix A.7. Consequently, the model's performance suffers when relying on concatenation for conditioning. This is intuitive as y is fed to the network in each denoising step t. Thus, the network emphasizes the y a lot, making it more vulnerable to error accumulation.

To address this, we use only the Schrödinger Bridge (SB) as the conditioning mechanism for unconditional generation because of its demonstrated robustness against accumulated errors.

To this end, we tested a compromise approach by applying concatenation for 50% of the steps and then dropping the condition from concatenation by feeding zeros. The experiment is conducted on the LSUN-Churches dataset. Table 9 shows that this trick improves performance compared to concatenation in all steps. However, it still performs worse than using SB alone on generated sketches.

Accordingly, the choice of y-feeding strategy depends on the setup:

- Unconditional generation: Use only the Schrödinger Bridge, as it exhibits superior robustness against noise in the input condition.
- Conditional generation: Use a combination of Concatenation and Schrödinger Bridge, as shown in Table 3, for optimal results.

Table 8: Human evaluation for the real image editing capabilities between our method and SEED-X on 100 randomly sampled images from CelebHQ dataset.

Method	FID
SEED-X	2.6
Ours	**4.3**

Table 9: Comparison between different conditioning mechanisms regarding error vulnerability on the LSUN dataset.

Method	FID
SB	**6.6**
SB+Concat (100% steps)	10.1
SB+Concat (50% steps)	8.9

A.9 Future Work

In this section, we will discuss some possible future work that shows the versatility of ToddlerDiffusion and its potential to drive innovation across image, video, and 3D tasks.

Distill the knowledge from substage features for perception tasks. The intermediate outputs from Toddler Diffusion (e.g., the sketch stage) can be reused as strong geometric priors for perception tasks by distilling the knowledge into perception networks to enforce geometric or perceptual constraints, making them structure-aware and geometry-guided aligned (Zhao et al., 2023a).

Reusing Diffusion Features as High- and Low-Frequency Representations. Toddler Diffusion's staged outputs inherently capture distinct frequency information, with the sketch stage emphasizing

low-frequency global structures and geometry and the palette and RGB stages capturing high-frequency details like textures. These features can be repurposed into a dual-branch network (Bi et al.; Wei et al., 2024): one branch focusing on low-frequency representations for tasks requiring global context (e.g., semantic segmentation, object localization) and the other specializing in high-frequency details for tasks like texture classification, instance segmentation, or boundary refinement. Integrating this knowledge into VLMs can improve performance and generalization for perception tasks.

Addressing Limitations in T2I Models. A significant challenge for text-to-image (T2I) models lies in handling complex prompts involving counting or spatial relations (Bakr et al., 2023; Ghosh et al., 2024). Fixing these issues in the sketch or abstract stage will be easier and simpler due to its lightweight. In addition, the subsequent stages (e.g., sketch-to-RGB) will remain unaffected and reusable. This approach allows targeted improvements without retraining/finetuning the entire pipeline, significantly reducing computational costs.

Modular Prompt Handling for T2I Models. Toddler Diffusion can simplify complex text prompts by decomposing them into modular components mapped to specific stages. For example, a detailed prompt can be split into geometry-focused instructions for the sketch stage and color-specific details for the sketch-to-palette stage. In addition, we can add more stages and specific prompts such as depth, e.g., obj_1 closer than obj_2. This enables a more focused and interpretable generation, particularly for long, detailed prompts with intricate requirements.

For instance: Given a complex prompt: "A bustling city street during sunset with a red double-decker bus on the right, a blue car parked on the left, people walking on the sidewalk, and sunlight casting long shadows on the ground." will be split into:

- Structure-Focused prompt: "A city street with a double-decker bus on the right, a car parked on the left, and people walking on the sidewalk."
- Palette-Focused prompt: "The double-decker bus is red, the car is blue, the sky shows a sunset with shadows on the ground."

Video Generation in Abstract Domains. Generating videos directly in the RGB domain is computationally expensive and prone to inconsistencies. Toddler Diffusion can split the problem into geometry consistency and stylistic consistency. Specifically, it first generates videos in an abstract domain (e.g., sketches), ensuring geometric consistency across frames. Then, these sketches can be refined into RGB frames using another stage, Sketch-to-RGB, conditioned on a reference frame for color and style consistency. This two-step process simplifies the complex task of maintaining temporal and stylistic coherence in video generation.

Spatial Decompositionality. An intriguing future direction is to adapt Toddler Diffusion to cascade generation across spatial dimensions instead of modalities:

- Object-Level Cascading: Generate foreground objects first, followed by backgrounds, enabling fine-grained control for game development or storyboarding applications.
- Part-Level Cascading: Decompose objects into parts (e.g., car wheels, body, interior) for industries like automotive design and industrial manufacturing, where precise control at the component level is critical.

Enhancing 3D Generation. Toddler Diffusion can streamline 3D generation by reusing the sketch stage for geometry, followed by a depth estimation stage, and finally, lifting 2D geometry into full 3D representations. This staged approach is particularly valuable for AR/VR scene creation, autonomous driving simulations, etc. Additionally, we can first generate the layout, followed by another stage that generates the meshes (Koo et al., 2023).

Integration with NeRFs. The sketch stage can provide strong structural priors for Neural Radiance Fields (NeRFs), improving 3D scene reconstruction from sparse views. This integration could accelerate NeRF optimization and enhance generalization in applications like scene editing and AR/VR reconstruction, where structural integrity is critical.

Intrinsic Image Decomposition. Toddler Diffusion's staged approach can be extended to tasks like intrinsic image decomposition, where images are disentangled into reflectance, illumination, and shading components. This decomposition enables applications like relighting, material editing, and realistic AR/VR rendering, where physically consistent outputs are crucial.

B MORE QUALITATIVE RESULTS

B.1 DEMO

Please refer to our *demo*, which probes our controllability capabilities in performing consistent edits.

Our demo demonstrates a lot of use cases, including:

- (0:00 –> 0:20) [1st stage] Starting from noise generates a sketch by running only the 1st stage, which is unconditional generation.
- (0:20 –> 0:24) [2nd stage] Starting from a generated sketch generates a palette. This one is considered an unconditional generation as noise generates the sketch unconditionally.
- (0:24 –> 0:32) [3rd stage] Starting from a generated palette generates an RGB image. This one is also considered an unconditional generation.
- (0:32 –> 0:44) [Conditional Generation] Starting from the GT sketch, we generate an RGB image.
- (0:44 –> 1:15) [Editing] We show our sketch-based editing capabilities.
- (1:15 –> 1:46) [Editing] We show our palette-based editing capabilities.

B.2 REAL IMAGE EDITING

As explained in Section 2.4, we ensure consistent edits for real images by storing the noise used during the generation process for later edits. The primary challenge with real image editing is that we don't have access to the noise values at each denoising step, as our model didn't generate the image. To address this, we employ DDPM-inversion (Huberman-Spiegelglas et al., 2024) to reconstruct the intermediate noise. As illustrated in Figure 16, we compare our model against SEED-X (Ge et al., 2024), and our model achieves consistent edits on real reference images better than SEED-X, despite being trained on much fewer data compared to SEED-X and using smaller architecture. Additionally, SEED-X fails to perform the edits correctly and keeps the rest of the image unedited, such as the rest of the face or the background. On the contrary, our method shows great faithfulness to the original image while performing the edits depicted in the sketch. We have created a small test-set for editing where we manually designed 100 edits on CelebHQ and generated the corresponding edits for both our method and SEED-X. Then, we asked annotators (4 annotators) to give a score based on the aestheticism of the generated image and the adherence to the edit command. As shown in Table 8, the annotators scores show great preferences to our method over SEED-X where our score is 4.3 while SEED-X is only 2.6

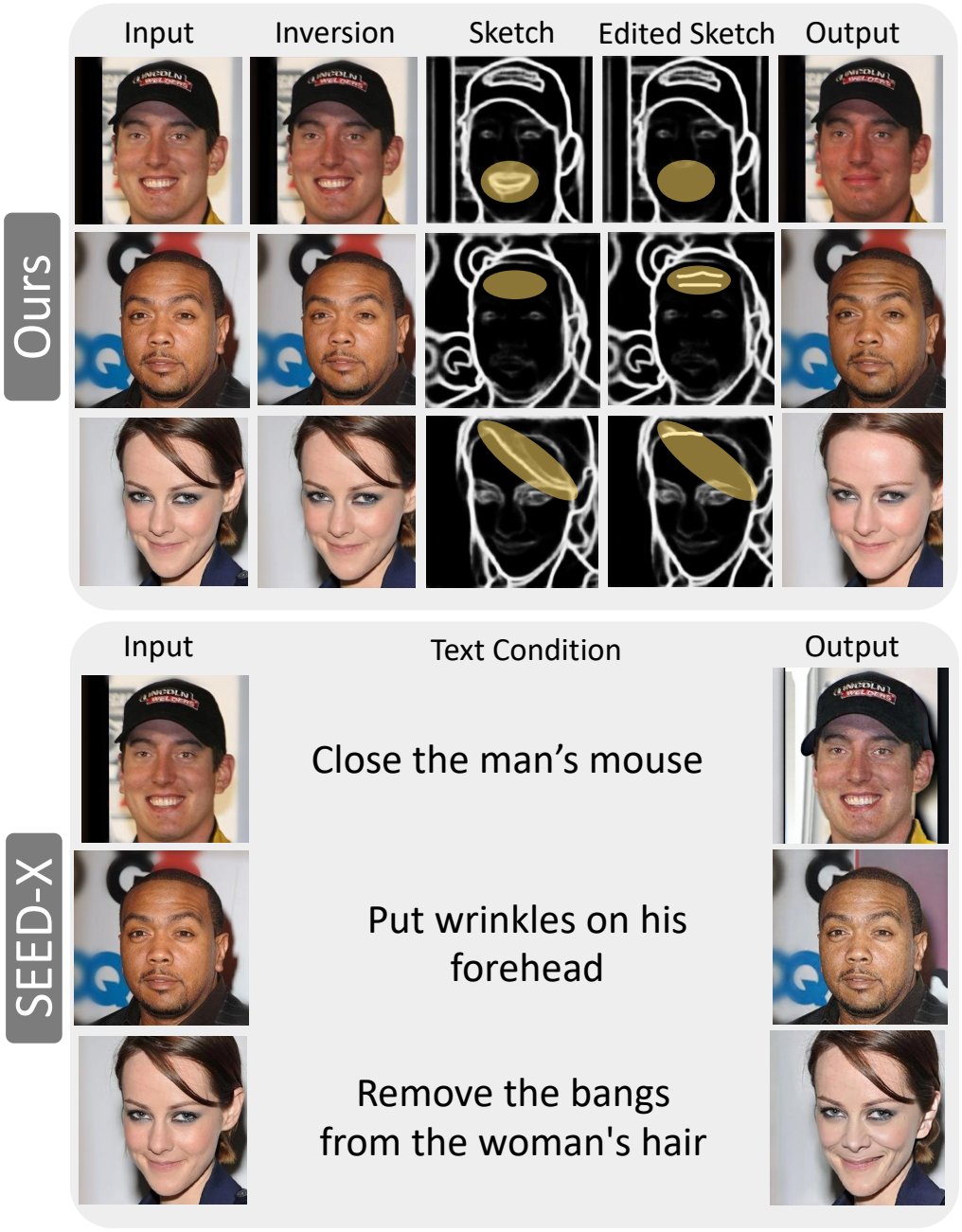


Figure 16: Comparison between our approach, ToddlerDiffusion and SEED-X (Ge et al., 2024) in terms of the real images capabilities.

B.3 Class-Label Robustness

Beyond these quality comparisons, we also conducted a robustness analysis to assess our model's behavior when faced with inconsistencies between conditions. Specifically, we deliberately provided incorrect class labels for the same sketch. This mismatched the sketch (geometry) and the desired class label (style). Despite the inconsistent conditions, as illustrated by the off-diagonal images in Figure 17, our model consistently adheres to the geometry defined by the sketch while attempting to match the style of the misassigned category, demonstrating the robustness of our approach. Successful

cases are highlighted in yellow, and even in failure cases (highlighted in red), our model avoids catastrophic errors, never hallucinating by completely ignoring either the geometry or the style.

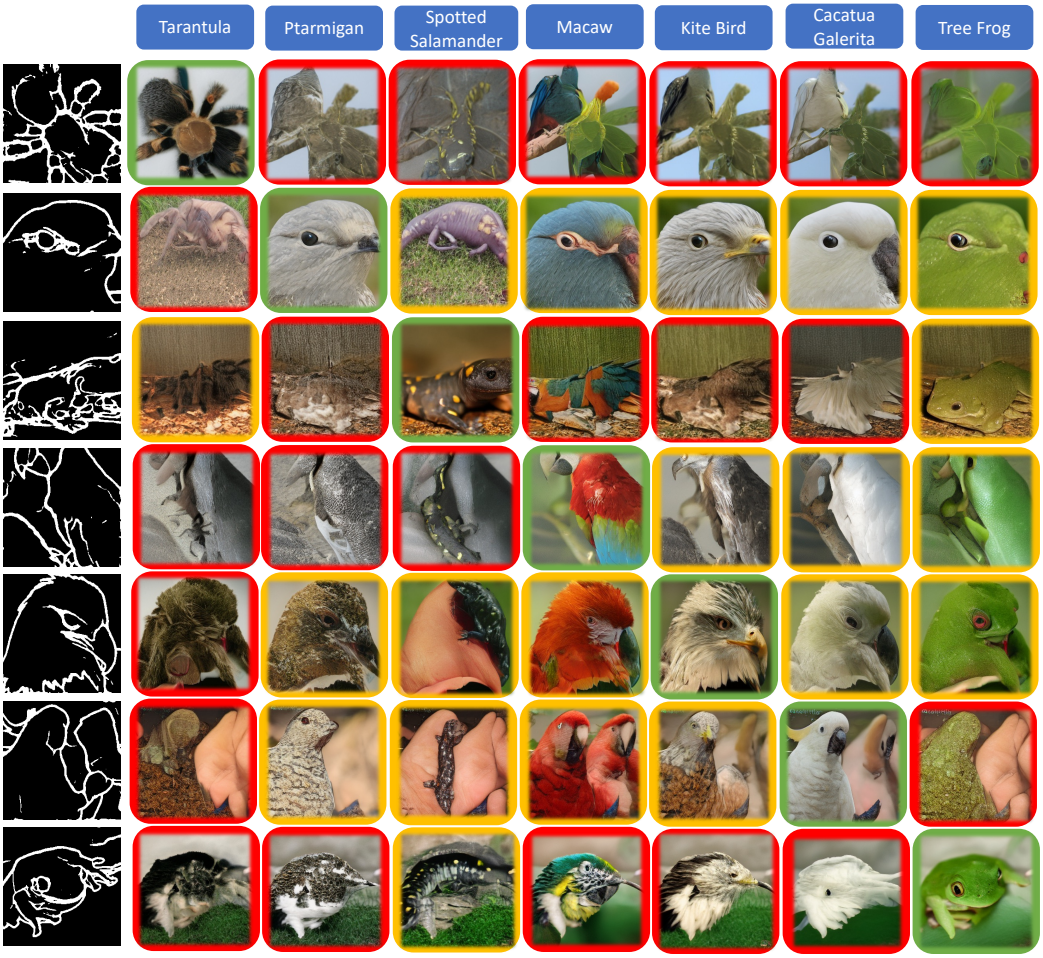


Figure 17: Our results on the ImageNet dataset. The first column depicts the input sketch. The diagonal (green) represents the generated output that follows the sketch and the GT label. The off-diagonal images show the output given the sketch and inconsistent label. These adversarial labels show our model's robustness, where the successful cases are in yellow, and the failure ones are in red.

B.4 Editing Capabilities

Our approach introduces an interpretable generation pipeline by decomposing the complex RGB generation task into a series of interpretable stages, inspired by the human generation system (Stevens, 2012; Marr, 2010; 1974). Unlike traditional models that generate the complete image in one complex stage, we break it into N simpler stages, starting with abstract contours, then an abstract palette, and concluding with the detailed RGB image. This decomposition not only enhances interpretability but also facilitates dynamic user interaction, offering unprecedented editing capabilities for unconditional generation, as shown in Figure 18, Figure 19, Figure 20, and Figure 21.

Figure 18: Controllability ability of our framework, ToddlerDiffusion. Starting from generated sketch and RGB image (A), we can remove artifacts or undesired parts, in red, (B), add a new content, in yellow, (C-D), and edit the existing content, in green, (E-F) by manipulating the sketch.

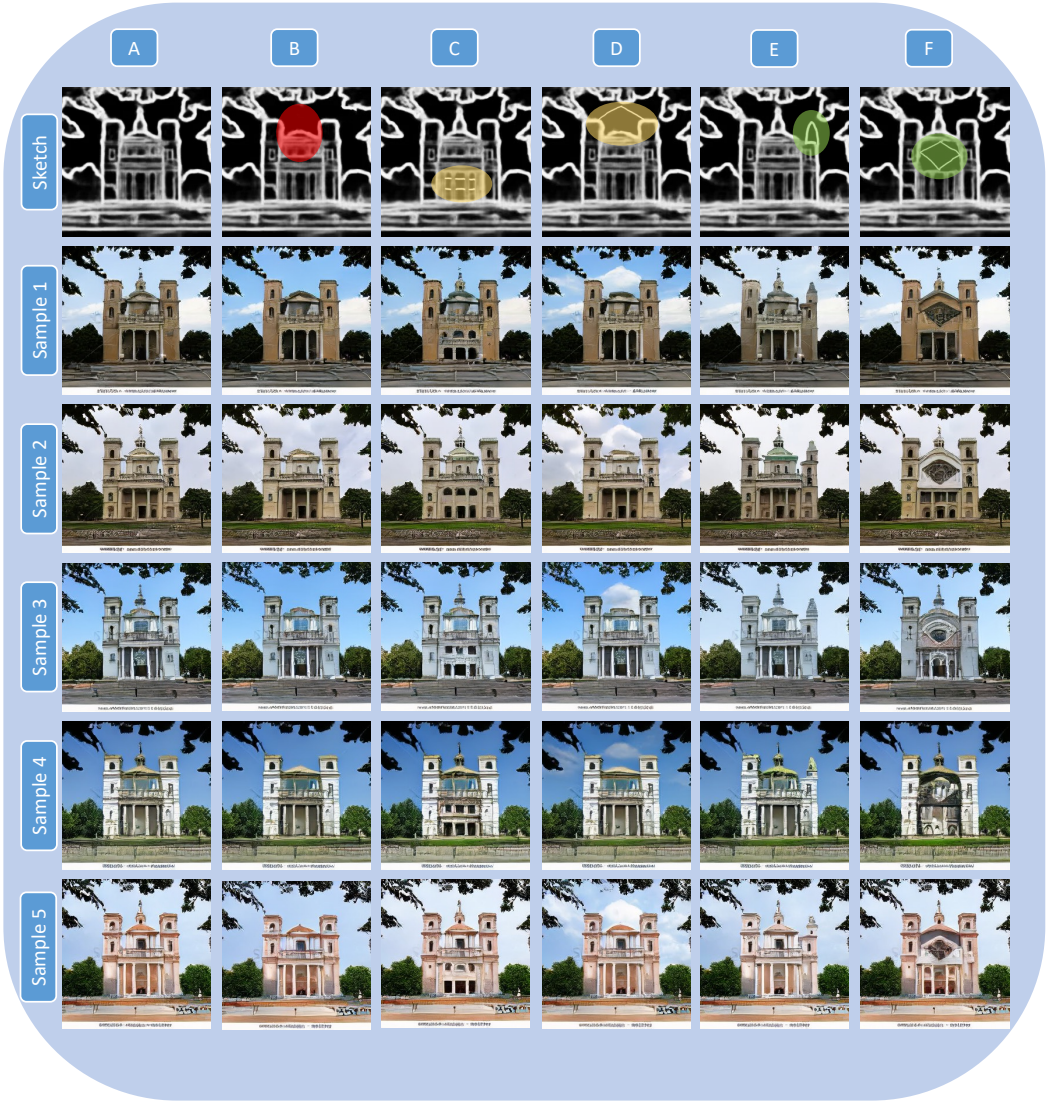


Figure 19: Controllability ability of our framework, ToddlerDiffusion. Starting from generated sketch and RGB image (A), we can remove artifacts or undesired parts, in red, (B), add a new content, in yellow, (C-D), and edit the existing content, in green, (E-F) by manipulating the sketch.

Figure 20: Controllability ability of our framework, ToddlerDiffusion. Starting from generated sketch and RGB image (A), we can remove artifacts or undesired parts, in red, (B), add a new content, in yellow, (C-D), and edit the existing content, in green, (E-F) by manipulating the sketch.

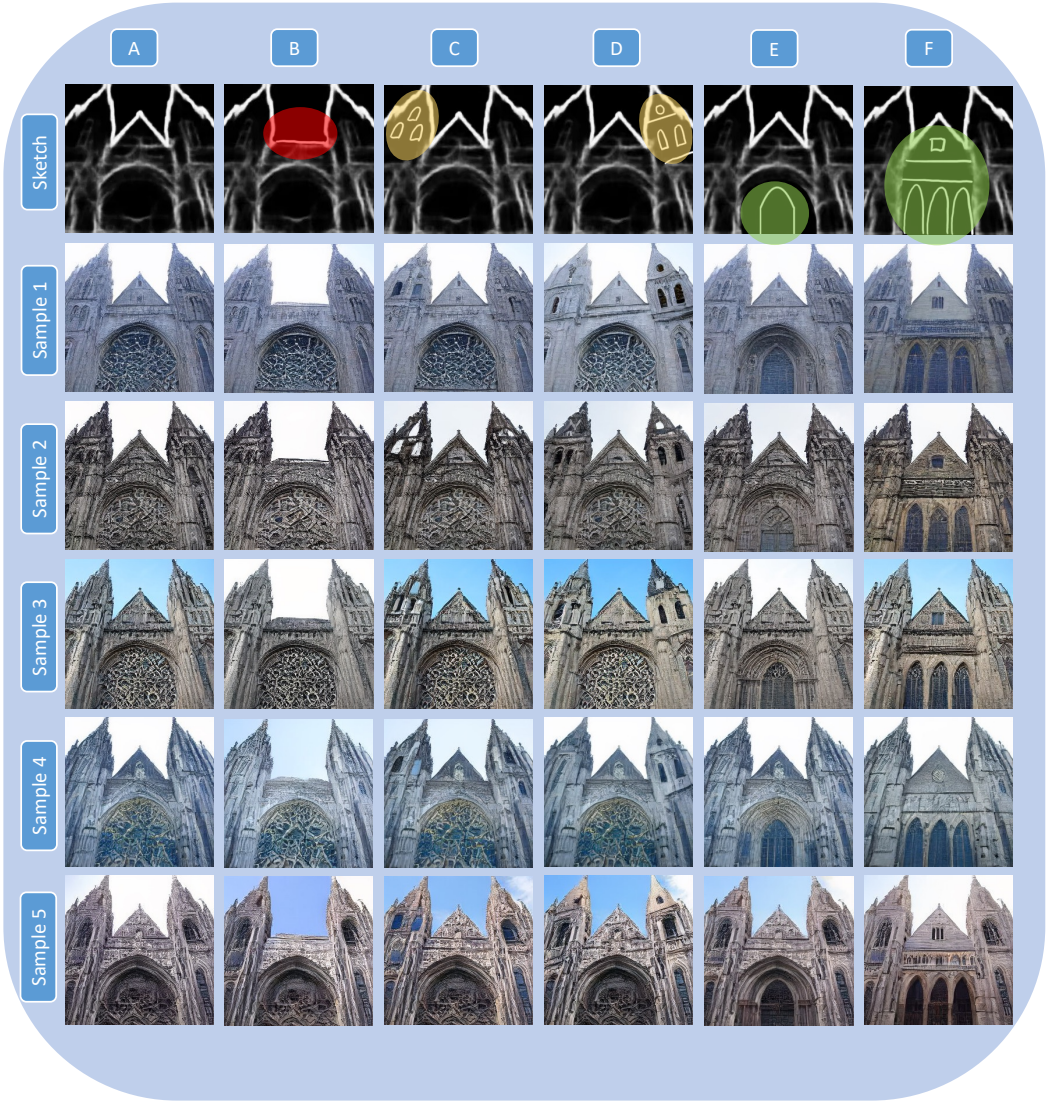


Figure 21: Controllability ability of our framework, ToddlerDiffusion. Starting from generated sketch and RGB image (A), we can remove artifacts or undesired parts, in red, (B), add a new content, in yellow, (C-D), and edit the existing content, in green, (E-F) by manipulating the sketch.

B.5 Sketch Conditioning (SD-v1.5)

We convert our method to a "sketch2RGB" by omitting the 1^{st} stage to show our controllability capabilities. By doing this, we can compare our model against the vast tailored sketch-based models built on top of SD-v1.5. However, these methods have been trained for a long time on large-scale datasets. Thus, for a fair comparison, we retrain them using CelebHQ datasets for 5 epochs starting from SD-v1.5 weights. As demonstrated in Figure 22, despite this is not our main focus, our method outperforms conditioning methods without adding additional modules (Zhang et al., 2023; Zhao et al., 2024) or adapters (Mou et al., 2024). We compare training-free (Tumanyan et al., 2023; Parmar et al., 2023) and training-based (Li et al., 2023b; Zhang et al., 2023; Zhao et al., 2024) methods. The performance is worse than expected for the trained-based methods due to two reasons: 1) We train all the models for only 5 epochs; thus, some of this method, in contrast to us, needs more epochs to converge. 2) We apply augmentations to the input sketch, which shows that some of these methods are vulnerable to sketch perturbations.

Figure 22: Comparison between our approach, ToddlerDiffusion and training-free (Tumanyan et al., 2023; Parmar et al., 2023) and training-based (Li et al., 2023b; Zhang et al., 2023; Zhao et al., 2024) sketch conditioning methods.

B.6 Text + Sketch Conditioning (SD-v1.5)

CelebHQ: We conducted another experiments to condition the model on both text and sketch. To this end, we have extended the CelebHQ dataset by adding a detailed text description for each image using Llava-Next v1.6. As Llava generates too long descriptions, we ask GPT-4o to summarize them. We train the model starting from the SDv1.5 weights for only five epochs. As shown in Figure 23, we feed the same sketch with different prompts, where the only difference between them is the gender, then the model generates the RGB images that follow both the sketch and the text conditioning.

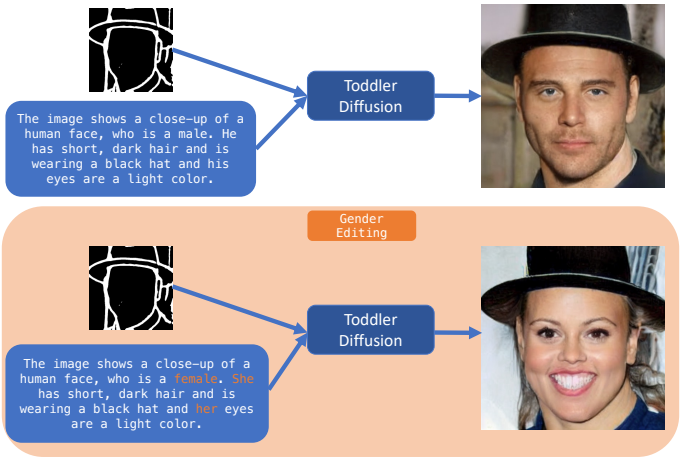


Figure 23: Text and Sketch conditioning capabilities of our proposed method.

LionArt: Furthermore, we conducted a more challenging experiment to condition the model on both text and sketch on a more challenging and diverse dataset, LionArt. To this end, we randomly sample 1 M images from LionArt and train our model for only 1 epoch starting from SDv1.5 weights. As shown in Figure 24, our model generates RGB images that adheres to both conditions; the text prompt and the input sketch.

Figure 24: Qualitative results for our Text-conditional generation on LionArt dataset.

B.7 TEXT CONDITIONING

To show the versatility of our approach, we conducted experiments to show our capabilities in generating sketches from text on two different datasets: CUB200 and Dress Code dataset. The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset is the most widely-used dataset for fine-grained visual categorization task. It contains 11,788 images of 200 subcategories belonging to birds. Due to its importance, DM-GAN extends the dataset to include a text description for each image. There is no GT paired sketches for each image, thus, we run PidiNet to generate sketch for each RGB image. Additionally, we show our capabilities on the Dress Code dataset, which is more than 3x larger than publicly available datasets for image-based virtual try-on and features high-resolution paired images (1024x768) with front-view, full-body reference models. Like CUB200, the Dress Code data is an unimodal dataset that only contains RGB images. Fortunately, Ti-MGD extends the dataset to be a multi-modal dataset by having image-text pairs.

In this setup we have two conditioning mechanisms: 1) The sketch y, which is handled by the forward process (Eq.2). 2) Cross-attention layer to handle the text condition.

We have trained the 1^{st} stage for 1K epochs as it is very small, only 5 M parameters, and the dataset scale is very small, thus the 1k epochs takes less than 12 hours using a single A100 GPU. Then, we train the 2^{nd} stage, 141 Million parameters, for only 200 epochs. In contrast, LDM takes longer to converges as shown in Figure 4, thus we train it for 400 epochs.

As shown in Table 10 and Table 11, we have achieved better accuracies on both datasets compared to LDM despite being training for half the epochs. For qualitative results, please refer to Figures 25 and 26.

Table 10: Comparison between our approach and LDM using text conditioning multi-modal dataset, Dress Code (Baldrati et al. 2024).

Method	FID
LDM	10.7
Ours	**9.1**

Table 11: Comparison between our approach and LDM using text conditioning multi-modal dataset, CUB200 (Zhu et al. 2019).

Method	FID
LDM	12.2
Ours	**10.8**

Figure 25: Qualitative results for our Text-conditional generation on Dress Code dataset (Baldrati et al., 2024). The first column is the input text prompt, second column is the generated sketch, last column is the generated RGB image.

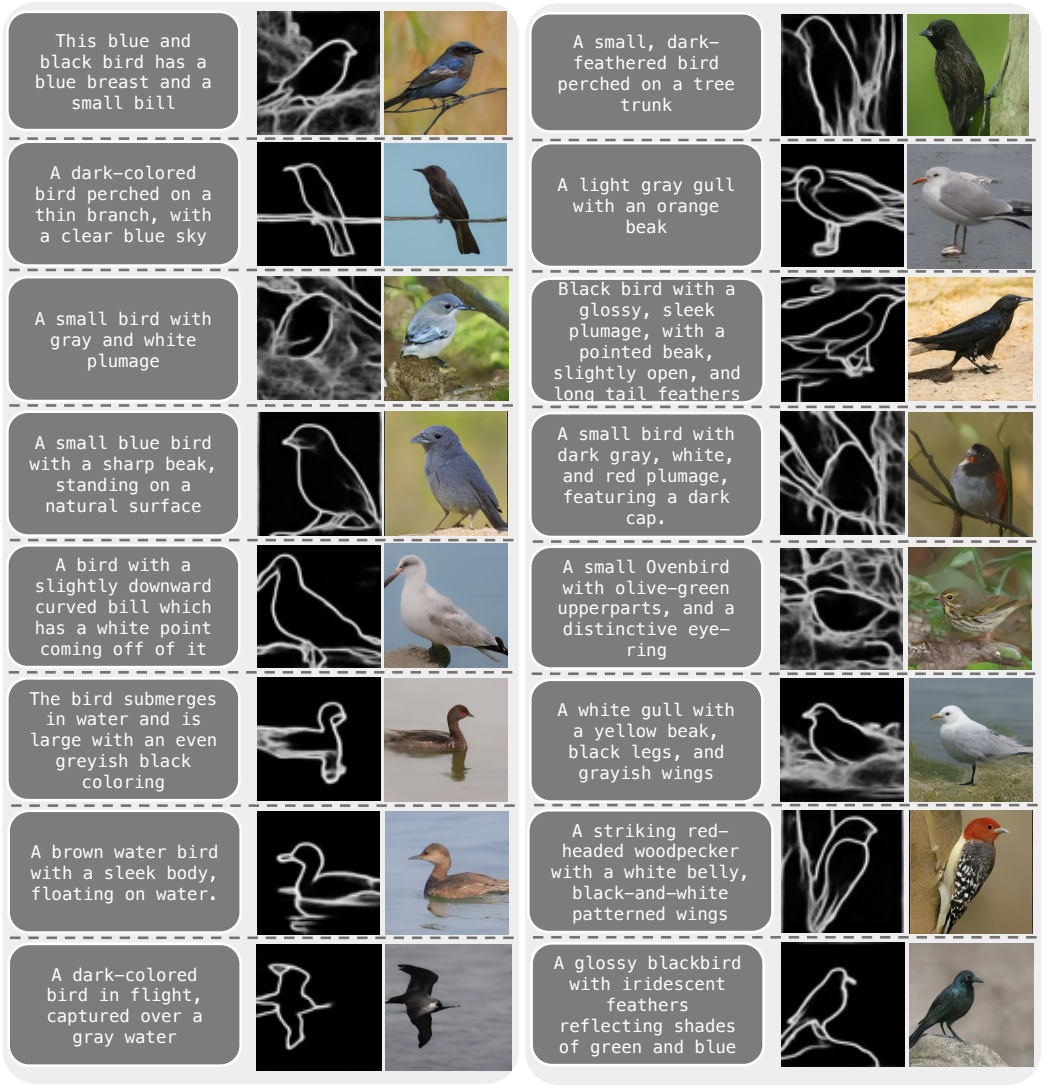


Figure 26: Qualitative results for our Text-conditional generation on CUB200 dataset (Zhu et al., 2019). The first column is the input text prompt, second column is the generated sketch, last column is the generated RGB image.

B.8 Free-Hand Sketch Conditional Generation Samples

Figure 27 depicts more qualitative results for our conditional generation pipeline, where we first generate the sketch using drawing tools and then use the generated sketch to generate the RGB image. Despite the sketch being too bad and sparse, our model still generates high-quality images.

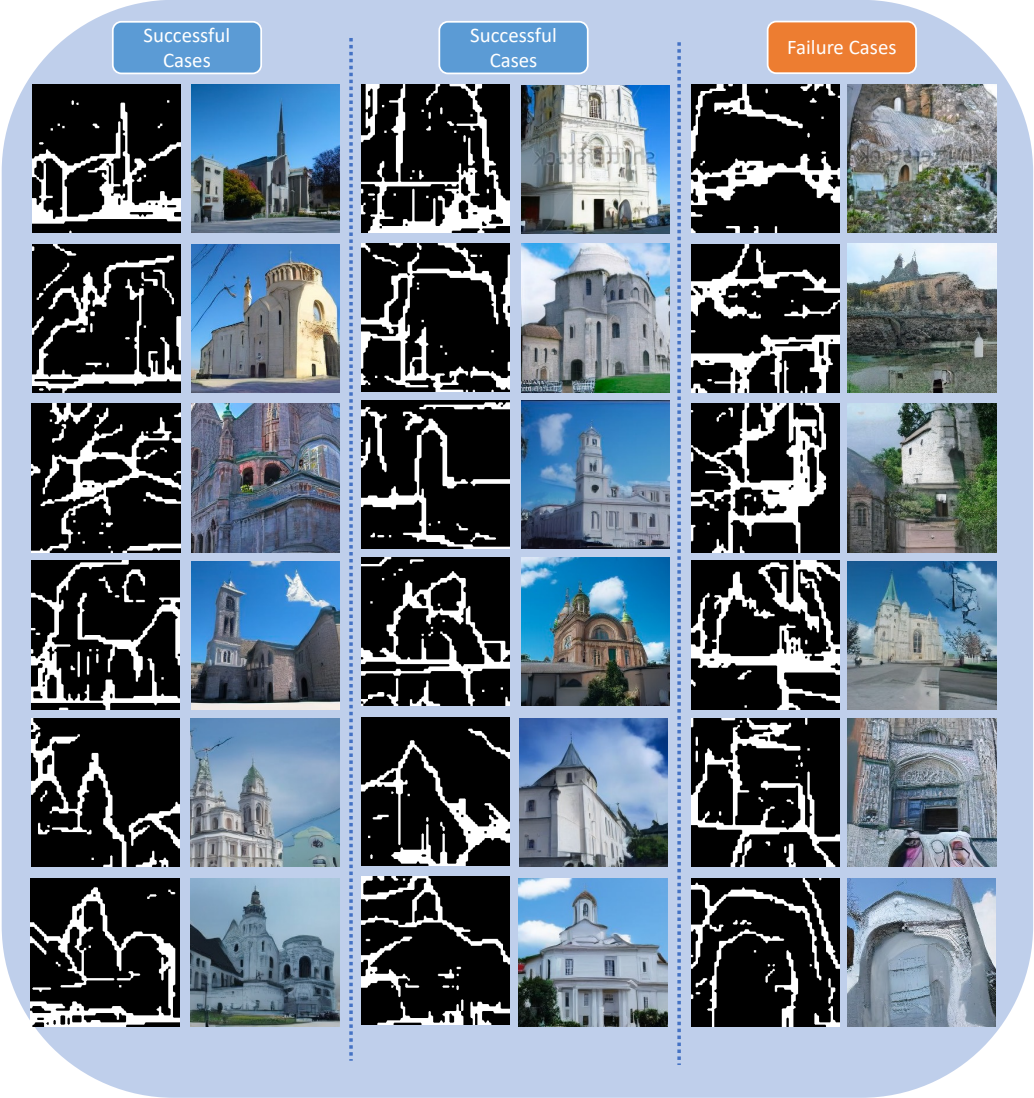


Figure 27: Qualitative results for our unconditional generation.

B.9 QUALITATIVE RESULTS FOR OUR SKETCH-CONDITIONAL GENERATION

In addition to our unconditional generation, our framework can be used as a conditional generation pipeline, where the user can omit the first stage and directly fed a reference sketch. As shown in Figure 28, given the reference sketch, we run only the second stage, which is responsible for generating the RGB image given a sketch. Compared to the generated images in Figure 27, the generated images in Figure 28 are much better as the reference sketches are much better than the generated ones. This observation is aliened with our discussion regarding the gap between the two stages. Therefore, improving the quality of the generated sketches will directly reflect on the overall images quality.

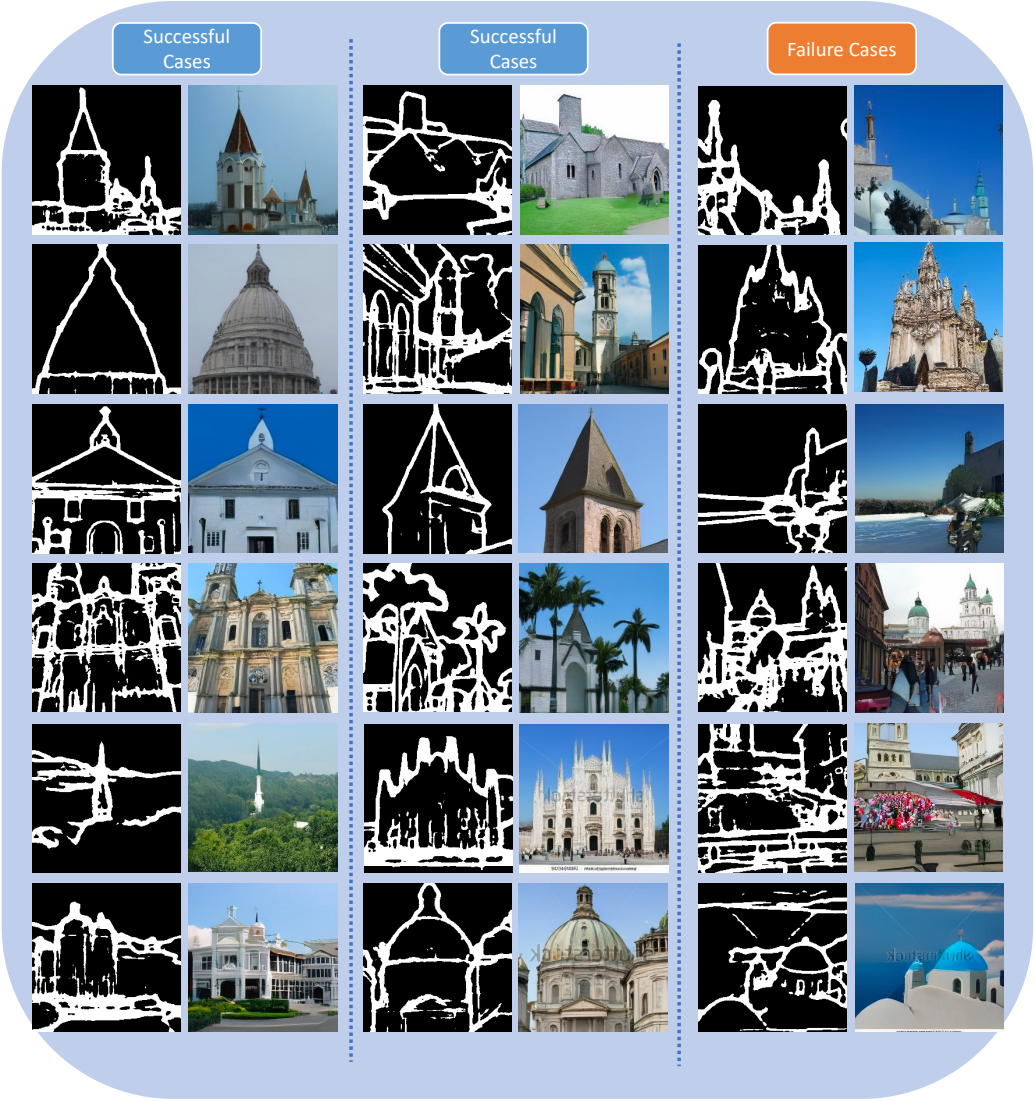


Figure 28: Qualitative results for our sketch-conditional generation.

C FAILURE CASES

The last column of Figure 27 and Figure 28 depicts failure cases, where the generated RGB is following the sketch but generates unrealistic image. Intuitively, the failure cases are more apparent in the unconditional generation due to the generated sketches quality.