DeCo: Frequency-Decoupled Pixel Diffusion for End-to-End Image Generation

Supplementary Material

A. Comparison with JiT

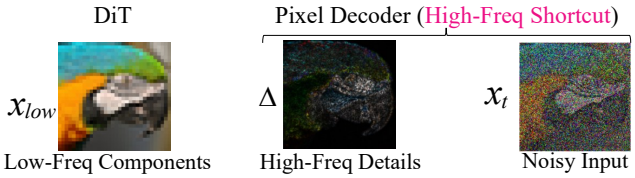


Figure 7. Comparison with JiT.

The key idea of DeCo is to provide a high-freq shortcut via the pixel decoder, enabling an implicit frequency decoupling. High-freq components include noisy input x_t and high-freq details Δ. In standard DiTs, high-freq signals tend to be suppressed after the patchification and deep transformer layers. Since pixel decoder directly models each pixel of the raw input, DeCo can automatically use this high-freq shortcut and offload high-freq modeling to pixel decoder, enabling DiT to focus on low-freq components x_{low}. Unlike JiT, which explicitly predicts $x_0 = x_{low} + \Delta$, our DiT implicitly models x_{low}, excluding hard high-freq details Δ. The pixel decoder, guided by x_{low}, only needs to fit a simple JiT-style formulation $v_\theta = \frac{x_{low}+(\Delta-x_t)}{t}$. This separation is learned end-to-end. Visualization in Fig. 2(c), spectrum analysis in Fig. 4, and superior performance over "JiT+REPA" in Tab. 1 validate our implicit decoupling.

B. Implementary Details

B.1. Baseline Comparisons

In this subsection, we summarize the settings used for all baseline comparisons. In the baseline comparisons, all diffusion models are trained on ImageNet at 256×256 resolution for 200k iterations using a large DiT variant. Following previous works [44, 66], we use a global batch size of 256 and the AdamW optimizer with a constant learning rate of 1e-4. Both baseline and DeCo adopt SwiGLU [59, 60], RoPE2d [53], and RMSNorm, and are trained with lognorm sampling and REPA [77]. The patch size of DiT's input is set to 16 for both baseline and our DeCo. The patch size of pixel decoder is set to 1. Our main architectural change on the baseline is to replace the final two DiT blocks of the baseline with our proposed pixel decoder.

For inference, we use 50 Euler steps without classifier-free guidance [18] (CFG) for all models except PixelFlow [6], which requires 100 steps. We also report results for the two-stage DiT-L/2 that requires a VAE and the recent pixel diffusion models PixelFlow [6] and PixNerd [66]. For a fair comparison, we further integrate DDT [67] into the pixel diffusion to form PixDDT, which has the similar parameter counts and computation FLOPs to our DeCo. Training memory and speed are measured with batch size of 256 on 8×A800 GPUs, while inference memory and time are measured on a single A800 with batch size of 1.

B.2. Class-to-Image Generation

For class-to-image generation experiments on ImageNet, we first train the model at a 256×256 resolution for 320 epochs. Subsequently, we fine-tune the model for additional 20 epochs at a 512×512 resolution. During inference, we use 100 Euler steps with CFG [18] and guidance interval [28]. Inference latency is measured on a single A800 GPU. The batch size and learning rate follow the default settings previously described. We use a global batch size of 256 and the AdamW optimizer with a constant learning rate of 1e-4. We set the CFG scale to 3.2 for the 256×256 resolution (320 epochs) and 5.0 for the 512×512 resolution (340 epochs). The CFG scale is set to 3.0 for the model of 800 epochs at a 256×256 resolution. The guidance interval [28] is set to 0.1 following previous work [66].

B.3. Text-to-Image Generation

For text-to-image generation, we trained our model on the BLIP3o [5] dataset, which contains approximately 36M pretraining images and 60k high-quality instruction-tuning data. We adopt Qwen3-1.7B [75] as the text encoder. The entire training takes about 6 days on 8× H800 GPUs. We adopt Qwen3-1.7B [75] as the text encoder. To improve the alignment of frozen text features [12], we jointly train several transformer layers on the frozen text features similar to Fluid [12]. The total batch size is 1536 for 256×256 resolution pretraining and 512 for 512×512 resolution pretraining. Following PixNerd [66], we pretrain DeCo on 256×256 resolution for 200K steps and pretrain on 512×512 resolution for 80K steps. We further fine-tune the pretrained DeCo on BLIP3o-60k with 40k steps at the 512×512 resolution following PixNerd. We adopt the gradient clip to stabilize training. *The whole training only takes about 6 days on 8× H800 GPUs.* We use the Adams-2nd solver with 25 steps as the default choice for sampling. The cfg scale is set to 4.0. We leave the native resolution [69] or native aspect training [13, 15, 34] as future works.

B.4. Experiment Configurations

Table 5 summarizes the experiment configurations for DeCo-L/16, DeCo-XL/16, and DeCo-XXL/16. In practice, we follow the training setups from previous works such as DiT [44], SiT [38], and PixNerd [66]. Besides, we sweep

	DeCo-L	DeCo-XL	DeCo-XXL
architecture			
DiT depth	22	28	16
hidden dim	1024	1152	1536
heads	16	16	24
params	426M	682M	1.1B
decoder depth		3	
decoder hidden dim		32	
patch size		16	
image size		256 (other settings: 512)	
training			
optimizer		AdamW [36], $\beta_1, \beta_2 = 0.9, 0.999$	
batch size		256	
learning rate		1e-4	
lr schedule		constant	
weight decay		0	
ema decay		0.9999	
time sampler		$\text{logit}(t) \sim \mathcal{N}(\mu, \sigma^2)$, $\mu = 0$, $\sigma = 1$	
noise scale		1.0	
sampling			
ODE solver		Euler	
ODE steps		100	
time steps		linear in [0.0, 1.0]	
CFG scale range		[3.0-3.2] (256×256), [4.5-5.0] (512×512)	
CFG interval [28]		[0.1, 1]	

Table 5. Configurations of experiments.

the CFG scale within the given ranges using an interval of 0.1.

C. Text-to-Image Prompts

Below, we list the prompts used for text-to-image generation in Fig. 1. These prompts cover a mix of animals, people, and scenes to evaluate semantic understanding and visual detail generation.

- A lovely horse stands in the bedroom.
- A baby cat stands on two legs, wearing a chothes.
- A cyberpunk woman with glowing tattoos and a mechanical arm beneath a holographic sky.
- A man sipping coffee on a sunny balcony filled with potted plants, wearing linen clothes and sunglasses, basking in the morning light.
- A beautiful woman.
- A cute panda is wielding a sword in realistic style.
- An extremely happy American Cocker Spaniel is smiling and looking up at the camera with his head tilted to one side.
- A raccoon wearing a detective's hat, observing something with a magnifying glass.
- Close-up of an aged man with weathered features and sharp blue eyes peering wisely from beneath a tweed flat cap.

D. Quantization Tables

In our DeCo, we use the normalized reciprocal of scaled JPEG quantization tables as adaptive weights to emphasize different frequency components. These tables are a core component of the JPEG compression standard and are designed based on properties of the human visual system (HVS) [45].

As shown in Sec. 3.3, a quantization table is an 8×8 matrix that determines the compression level for each frequency coefficient after the Discrete Cosine Transform (DCT). The JPEG standard uses two separate tables: one for the luminance (Y) component and another for the chrominance (Cb/Cr) components. This design is based on key characteristics of human perception. The core principle is that the human eye is not equally sensitive to all visual information. Specifically, two HVS properties are crucial. Firstly, the human eye is much more sensitive to low-frequency components than to high-frequency components. Secondly, the eye is more sensitive to changes in brightness (luminance) than in color (chrominance) [45].

Based on extensive experiments, the standard base quantization tables Q_{base} in Fig. 8 were developed to reflect these properties [45]. These tables have smaller values (finer quantization intervals) for low-frequency coefficients, which are perceptually more important. Conversely, these tables have larger values (coarser quantization intervals) for high-frequency coefficients, as the resulting information loss is less noticeable to the human eye. Similarly, the luminance table generally has smaller values than the chrominance table. These base tables can be scaled using a quality factor q to create new scaled quantization tables Q_{cur} for different compression levels.

Since a smaller quantization step implies that a frequency component is more significant to human perception, we use the normalized reciprocal of the scaled quantization tables as adaptive weights, i.e., $\frac{1}{Q_{\text{cur}}}$ with normalization. This allows us to assign a higher weight to the frequency components that are visually more important in our frequency-aware flow-matching loss $\mathcal{L}_{\text{FreqFM}}$.

E. Pseudocodes for DeCo

E.1. Training Step of DeCo

In Algorithm 1, we provide the pseudocodes for the training step of DeCo. DeCo utilizes the DiT to specialize in low-frequency semantic modeling with downsampled small-scale inputs $\bar{x}_t$. Semantic cues c are hence incorporated with a lightweight pixel decoder to reconstruct high-frequency signals. In other words, the pixel decoder takes the low-frequency semantics c from DiT as condition and predicts pixel velocities v_θ with a high-resolution input x_t. This new paradigm hence frees the DiT to specialize in

Base Luminance (Y)

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Base Chrominance (Cb/Cr)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Base Quantization Tables Q_{base}

⇩

Scaled Luminance (Y)

5	3	3	5	7	12	15	18
4	4	4	6	8	17	18	16
4	4	5	7	12	17	20	17
4	5	7	9	15	26	24	18
5	7	11	17	20	32	31	23
7	10	16	19	24	31	34	27
15	19	23	26	31	36	36	30
21	27	28	29	33	30	31	29

Scaled Chrominance (Cb/Cr)

5	5	7	14	30	30	30	30
5	6	8	20	30	30	30	30
7	8	17	30	30	30	30	30
14	20	30	30	30	30	30	30
30	30	30	30	30	30	30	30
30	30	30	30	30	30	30	30
30	30	30	30	30	30	30	30
30	30	30	30	30	30	30	30

Scaled Quantization Tables in Quality 85 Q_{cur}

Figure 8. Base and Scaled Quantization Tables.

Algorithm 1 Training step

```
# θ_DiT: DiT network
# θ_Dec: Pixel Decoder network
# x_0: training batch
# y: class label or textual prompt
# Q_cur: scaled quantization tables in quality 85.

# Prepare inputs
t = sample_t()
x_1 = randn_like(x_0)
x_t = (1-t)x_0 + tx_1 # original scale
x̄_t = patchify(x_t, patch_size=16) # small-scale

# Prepare ground-truth velocities
v_t = x_1 - x_0
𝕍_t = DCT2D(RGB2YCbCr(v_t))

# Generate low-frequency semantic condition
c = θ_DiT(x̄_t, t, y)
# Predict velocity conditioned on c
v_θ = θ_Dec(x_t, t, c)
𝕍_θ = DCT2D(RGB2YCbCr(v_θ))

# Compute Loss
FM_loss = mean(‖v_θ − v_t‖^2)
w = 1/Q_cur
w = w / w.mean() # normalized adaptive weights
FreqFM_loss = mean(w * ‖𝕍_θ − 𝕍_t‖^2)
loss = FM_loss + FreqFM_loss + REPA_loss
```

modeling semantics, and allows for more specialized details generation. To emphasize visually salient frequencies and suppress perceptually insignificant high-frequency components, we further introduce a frequency-aware Flow-Matching loss $\mathcal{L}_{\text{FreqFM}}$ inspired by the JPEG [23]. A REPA [77] loss is used in both our Baseline and DeCo.

Algorithm 2 K-Means Visualization in Fig. 2 (c)

```
# Feats: DiT outputs (T, H, W, C)
# I: generated images (T, H, W, 3)
# T: Sampling steps

for t in T:
    # Flatten spatial dimensions
    f = Feats[t].reshape(-1, C)

    # Cluster pixel-wise features
    labels = kmeans(f, n_clusters=8)

    # Map clusters to visualization
    vis = colormap(labels).reshape(H, W)

    plot(vis, I[t])
```

Algorithm 3 DCT Spectral Analysis

```
# V: Predicted velocity (B * T, ori_H, ori_W, C)
# Feats: DiT outputs (B * T, H, W, C)

# Pre-compute frequency scan order
# The block size of DCT is set to 8
idx = ZigZagIndices(block_size=8)

F_energy, V_energy = 0, 0
F_num_blocks, V_num_blocks =0, 0
for f, v in (Feats, V):
    # Split input into patches
    f_patches = Unfold(f, kernel_size=8)
    v_patches = Unfold(v, kernel_size=8)

    # 2D Discrete Cosine Transform
    f_freq = DCT2D(f_patches)
    f_energy = f_freq ** 2
    v_freq = DCT2D(v_patches)
    v_energy = v_freq ** 2

    # Accumulate energy following ZigZag order
    # Maps 2D (u,v) to 1D frequency index
    F_energy += Sum(f_energy.reorder(idx))
    F_num_blocks += len(f_patches)
    V_energy += Sum(v_energy.reorder(idx))
    V_num_blocks += len(v_patches)

# Log-scale Normalization
Feats_S = log(1 + F_energy / F_num_blocks)
Feats_S = Feats_S / max(Feats_S)
V_S = log(1 + V_energy / V_num_blocks)
V_S = V_S / max(V_S)

plot(Feats_S)
plot(V_S)
```

E.2. K-Means Visualization in Fig. 2 (c)

In Algorithm 2, we provide the pseudocodes for the K-Means visualization in Fig. 2 (c). The number of clusters in K-Means is set to 8. We uniformly select 4 timesteps from the sampling process and visualize the clustering results at

these timesteps.

E.3. DCT energy distribution in Fig. 4

In Algorithm 3, we provide the pseudocodes for DCT spectral analysis of Fig. 4. We apply an 8×8 DCT to transfrom the DiT outputs and pixel velocities into frequency domain. Each 8×8 patch yields 64 frequency coefficients, which are then converted into energy via a square operation. We reorder these energies from low to high frequency using standard zigzag indexing, where lower indices correspond to lower-frequency components. Finally, we apply log-scale normalization to rescale all energies to the range [0, 1] for comparison. As demonstrated in Fig. 4, compared with baseline, DeCo suppresses high-frequency signals in DiT outputs while preserving strong high-frequency energy in pixel velocity, confirming effective frequency decoupling. The distribution is computed on 10K images across all diffusion steps, i.e., B=10,000 and T=100.

F. More Visualizations

In this section, we provide more visualizations, including text-to-image generation in Fig. 9, class-to-image generation at a 256×256 resolution in Fig. 10, and class-to-image generation at a 512×512 resolution in Fig. 11. Our DeCo supports multiple languages with the Qwen3 text encoder after pretraining on the BLIP3o dataset [5], such as Chinese, Japanese, and English.

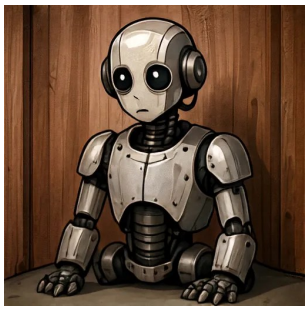
蒸汽朋克风格的长城，巨大齿轮与空中飞艇，黄昏史诗光影
(Steampunk Great Wall with huge gears and airships, epic dusk light)

破碎的青花瓷风少女面孔，细腻纹理，超现实主义
(Shattered blue-and-white porcelain girl's face, fine texture, surreal)

赛博朋克大熊猫，佩戴机械义肢，在霓虹街道漫步
(Cyberpunk panda with mechanical prosthetics walking through neon streets)

透明冰晶材质的龙，盘踞雪山之巅
(A transparent ice-crystal dragon coiled atop a snowy peak)

朝日を浴びる海辺の少女
(A girl on the seaside bathed in morning light)



悲しいロボット
(A sad robot)

海辺に置かれた木箱
(A wooden box placed on the seaside)

雨の草原を歩く馬
(A horse walking on a rainy grassland)

A beautiful girl with hair flowing like a cascading waterfall.

Portrait of a woman made entirely of porcelain and gold.

Old fisherman holding a galaxy inside a glass jar.

Viking warrior wearing a red sweater

A fox sleeping inside a large transparent lightbulb

A lion made of burning charcoal and glowing embers.

Pixel art sunset over a retro synthwave city.

A highway leading straight into a giant purple moon with a person.

Figure 9. More Qualitative results of text-to-image generation at a 512×512 resolution. Our DeCo supports multiple languages with the Qwen3 text encoder, such as Chinese, Japanese, and English.

Figure 10. More qualitative results of class-to-image generation at a 256×256 resolution.

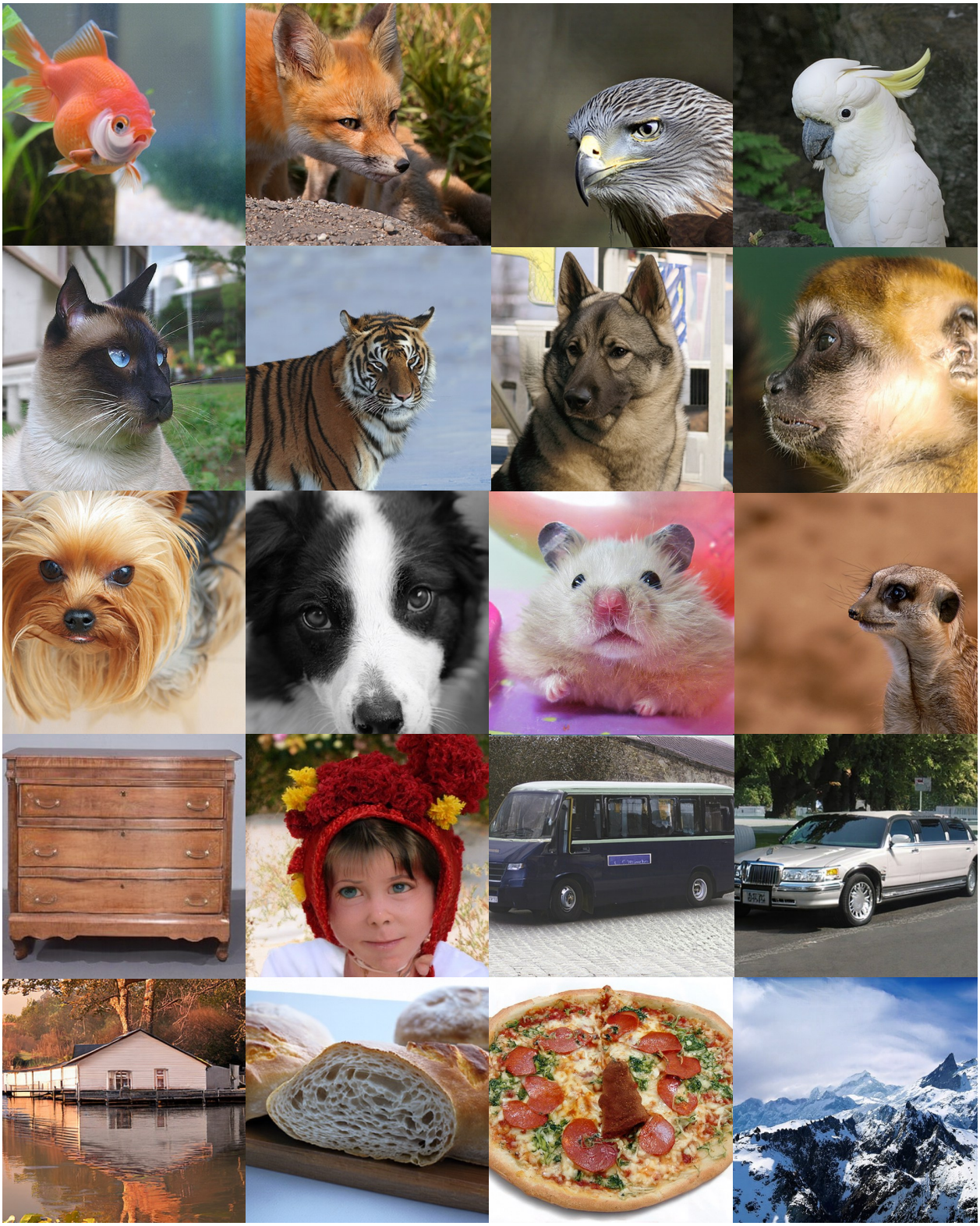


Figure 11. Qualitative results of class-to-image generation at a 512×512 resolution.