Janus: Decoupling Visual Encoding for Unified Multimodal Understanding and Generation

Supplementary Material

A. Details of Semantic Tokenizer Mentioned in Ablation Study

A.1. Architecture of Semantic Tokenizer

We build the semantic tokenizer based on the LlamaGen tokenizer, which has a downsample rate of 16. In addition to the original CNN pixel decoder, we add an additional semantic decoder branch after Vector Quantization, as shown in Figure 1 (a). The semantic decoder is a 12-layer ViT, with 12 attention heads and a hidden dimension of 768. For the semantic decoder, we use a causal attention mask to facilitate next token prediction when integrating it with an LLM. As the image tokens are generated autoregressively, the semantic decoder must learn to process it in a causal way. The semantic decoder learns to convert discrete tokens after quantization into semantic representations through knowledge distillation from continuous visual encoder, such as SigLIP.

A.2. Training

Training Procedure. The semantic tokenizer is trained from scratch in a two-stage manner. In the first stage, we train the model on the ImageNet-1k dataset for 40 epochs. In the second stage, we fine-tune the model for 1 epoch on 50 million images. These images come from the visual generation data used during the Janus pretraining process. We use a constant learning rate of $1e-4$ and a batch size of 128.

Training Loss. The training loss of the semantic tokenizer consists of two parts. On one hand, we use the loss for RGB reconstruction as described in LlamaGen, including MSE loss, perceptual loss, as well as GAN loss. On the other hand, we use SigLIP-Large-Patch16-384 as the teacher to supervise the semantic feature reconstruction results by the semantic decoder. We adopt the loss in BEiT-v2. Specifically, we maximize the cosine similarity between the semantic feature predicted by the semantic decoder and the SigLIP output. The weight for the semantic reconstruction loss is set to 0.25.

A.3. Integrating with LLM

Take the process of image generation as an example, we present the integration of the semantic tokenizer and the LLM in Figure 1 (b). The image is first transformed into continuous features through the CNN encoder, vector quantization, and the semantic decoder. Then, the LLM processes these features and generates predictions for the image IDs. Finally, the pixel decoder converts these discrete IDs into generated images.

B. Additional Qualitative Results

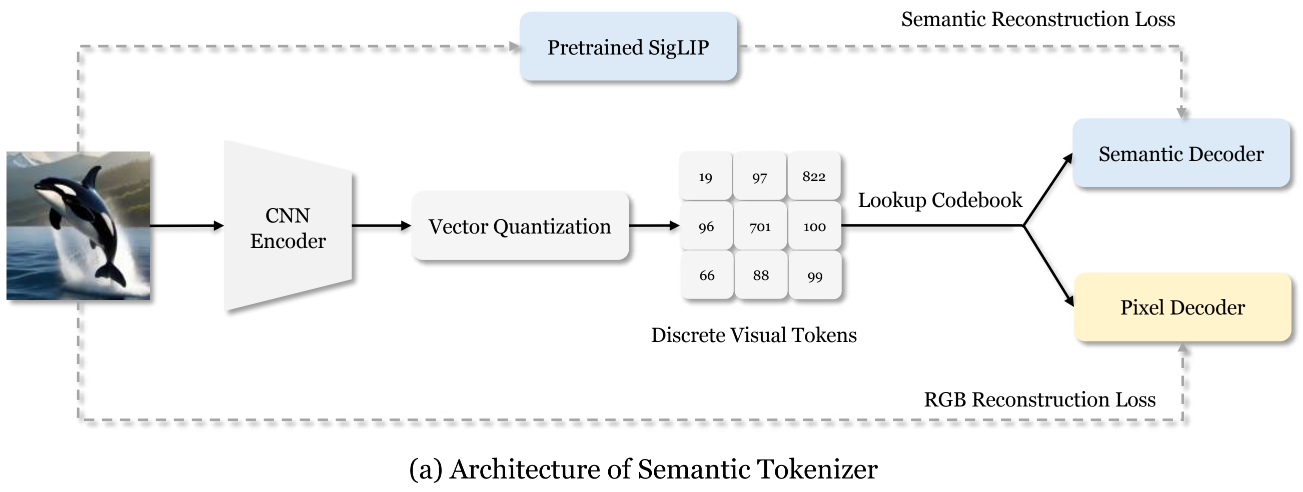
More Visualizations of Text-to-Image Generation. We present more text-to-image generation results in Figure 2. It is evident that Janus is capable of producing high-quality images that adhere closely to the given prompts. We further explore the multilingual text-to-image capabilities of our model, as shown in Figure 3. We are pleasantly surprised to find that, despite our training data consisting solely of English text-to-image data, Janus can still process text-to-image tasks in other languages. We attribute this multilingual ability to the original large language model's inherent traits. The LLM initially translates various languages into a unified semantic space, allowing Janus to perform text-to-image tasks naturally without additional training.

More Multimodal Understanding Results. Additional results on multimodal understanding are presented in Figure 4. Janus demonstrates remarkable comprehension abilities when processing inputs from diverse contexts such as scientific charts, artwork images, and LaTeX formula images, among others. These examples illustrate Janus's powerful high-level capabilities in seamlessly integrating and understanding information from multiple modalities, reflecting its advanced cognitive processing and contextual adaptability.

C. Additional Quantitative Results

Evaluation of inference speed, real-time performance, and resource consumption. We evaluated the throughput and latency for both Janus and LlamaGen on A100, with and without vLLM optimization. From Tab. 1, we observe Janus achieves ∼3x higher throughput and lower latency than LlamaGen in the standard setting, while using slightly less GPU memory (3648.43 MB vs. 3797.83 MB). With vLLM, throughput of Janus increases to 253.53 tokens/s and latency drops to 2.82 sec.

Further evaluation of generation results. We evaluate Janus on DPG-Bench, where it achieves an impressive score of 79.68, outperforming other models, including SDXL (74.65), PixArt-α (71.11), and SDv1.5 (63.18), as shown in Table 2. Notably, Janus surpasses SDXL, which has 2.6B parameters, despite having only 1.3B parameters, highlighting its efficiency and strong generative capabilities. These results demonstrate Janus's ability to generate



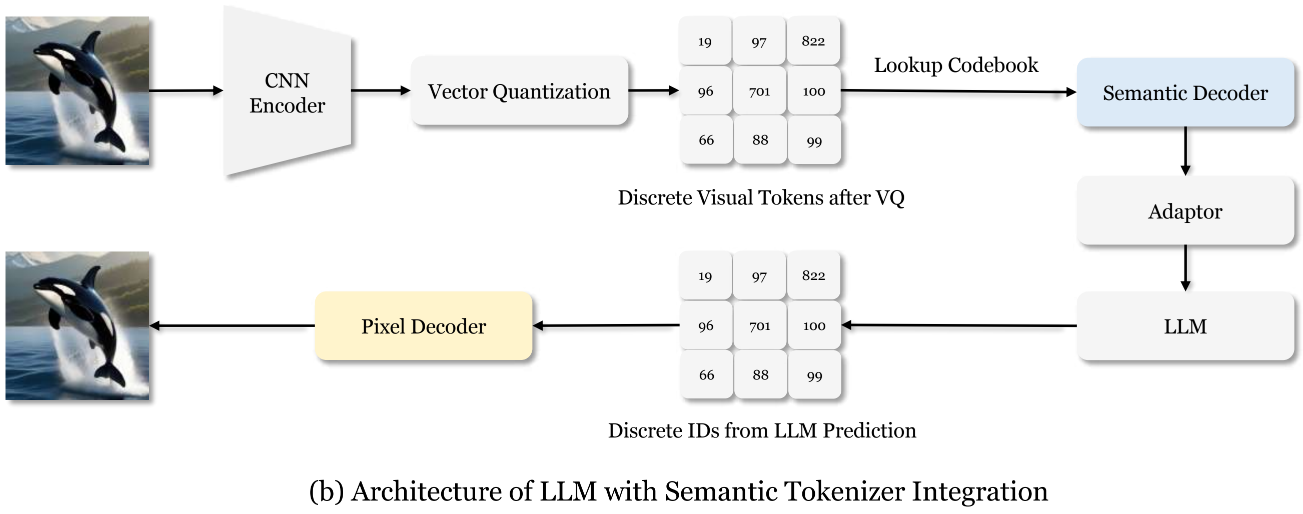

CNN Encoder
Vector Quantization
19 97 822
96 701 100
66 88 99
Lookup Codebook
Semantic Decoder
Discrete Visual Tokens after VQ
Adaptor
LLM
Pixel Decoder
Discrete IDs from LLM Prediction
(b) Architecture of LLM with Semantic Tokenizer Integration

Figure 1. **Architecture and usage of the semantic tokenizer.** (a) Architecture used during training of the semantic tokenizer. We use pre-trained SigLIP to supervise the reconstruction of semantic information, while using raw image to supervise the reconstruction of RGB values. (b) Integrating LLM with the semantic decoder. The semantic decoder outputs continuous features with high-level semantics, which are passed through an adaptor and then used as input for the LLM. **Please note that the semantic tokenizer is only used in the ablation study, not in the main experiment.**

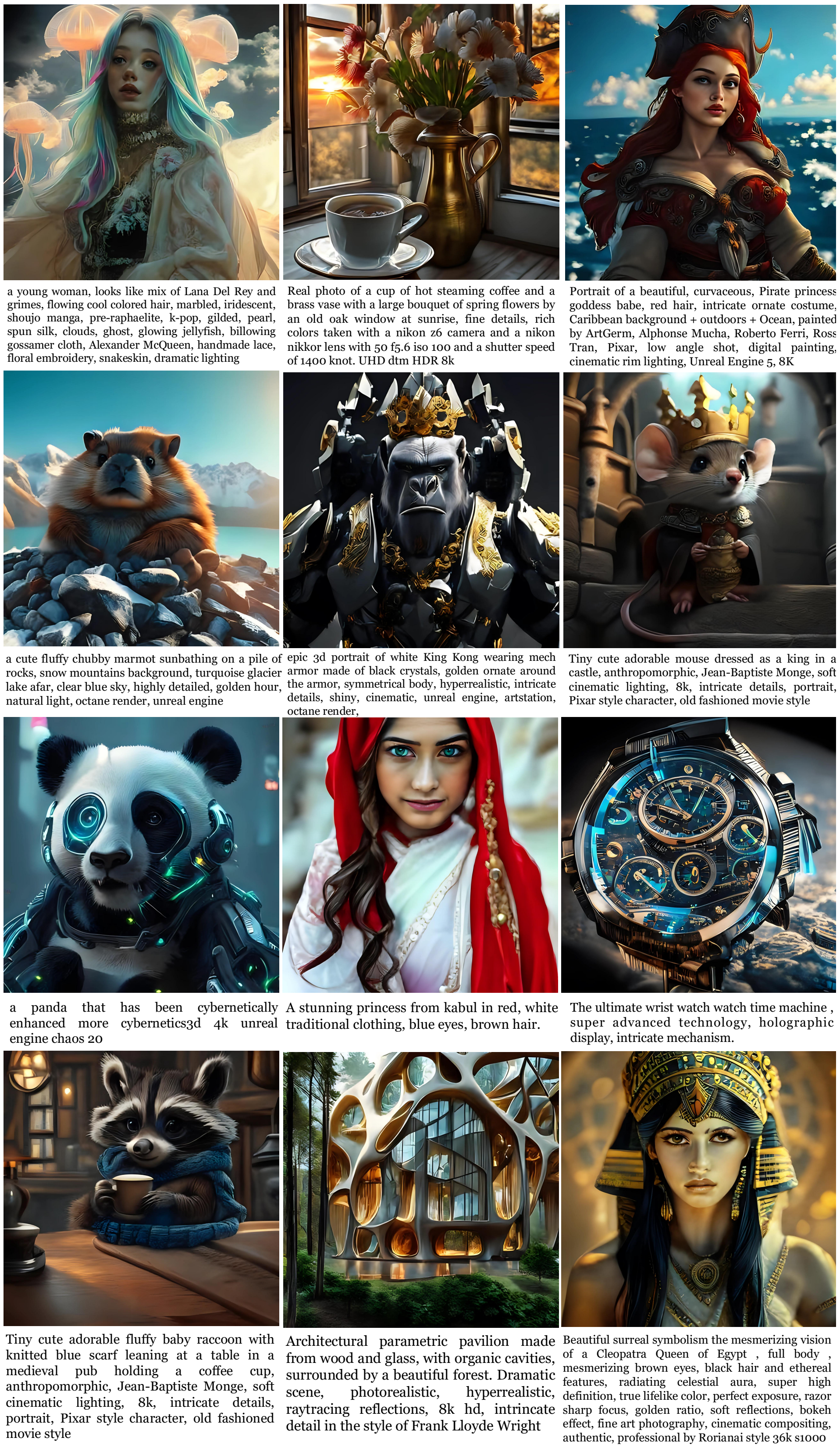
high-quality outputs while maintaining a relatively compact model size, making it a competitive choice for generative tasks. The performance gap suggests that Janus benefits from improved architecture or training techniques, enabling it to outperform even larger counterparts.

Model	Token Throughput (tokens/s) ↑	Latency (sec) ↓
LlamaGen-1.4B	22.26	26.38
Janus-1.3B	**63.72**	**9.54**
LlamaGen-1.4B w. vLLM	98.29	6.36
Janus-1.3B w. vLLM	**253.53**	**2.82**

Table 1. Comparison of Janus and LlamaGen on A100 GPU in terms of throughput (tokens/s) and latency (sec).

Method	# Params	DPG-Bench ↑
SDv1.5	0.9B	63.18
PixArt-α	0.6B	71.11
SDXL	2.6B	74.65
Janus	1.3B	79.68

Table 2. Comparison of generation results on DPG-Bench.



a young woman, looks like mix of Lana Del Rey and grimes, flowing cool colored hair, marbled, iridescent, shoujo manga, pre-raphaelite, k-pop, gilded, pearl, spun silk, clouds, ghost, glowing jellyfish, billowing gossamer cloth, Alexander McQueen, handmade lace, floral embroidery, snakeskin, dramatic lighting

Real photo of a cup of hot steaming coffee and a brass vase with a large bouquet of spring flowers by an old oak window at sunrise, fine details, rich colors taken with a nikon z6 camera and a nikon nikkor lens with 50 f5.6 iso 100 and a shutter speed of 1400 knot. UHD dtm HDR 8k

Portrait of a beautiful, curvaceous, Pirate princess goddess babe, red hair, intricate ornate costume, Caribbean background + outdoors + Ocean, painted by ArtGerm, Alphonse Mucha, Roberto Ferri, Ross Tran, Pixar, low angle shot, digital painting, cinematic rim lighting, Unreal Engine 5, 8K

a cute fluffy chubby marmot sunbathing on a pile of rocks, snow mountains background, turquoise glacier lake afar, clear blue sky, highly detailed, golden hour, natural light, octane render, unreal engine

epic 3d portrait of white King Kong wearing mech armor made of black crystals, golden ornate around the armor, symmetrical body, hyperrealistic, intricate details, shiny, cinematic, unreal engine, artstation, octane render,

Tiny cute adorable mouse dressed as a king in a castle, anthropomorphic, Jean-Baptiste Monge, soft cinematic lighting, 8k, intricate details, portrait, Pixar style character, old fashioned movie style

a panda that has been cybernetically enhanced more cybernetics3d 4k unreal engine chaos 20

A stunning princess from kabul in red, white traditional clothing, blue eyes, brown hair.

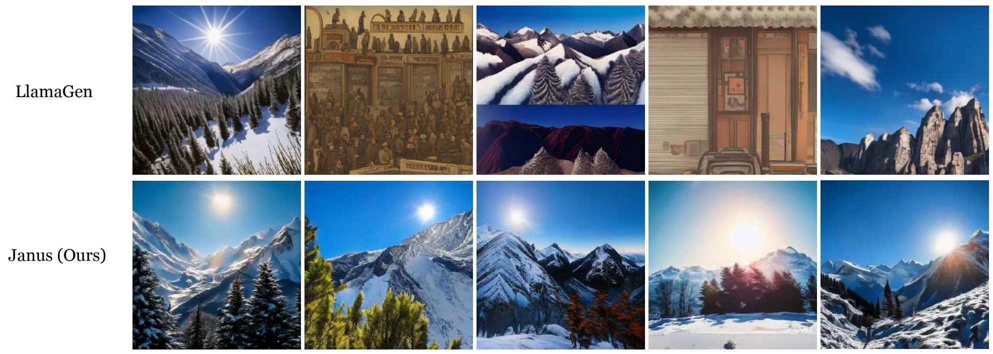
The ultimate wrist watch watch time machine , super advanced technology, holographic display, intricate mechanism.

Tiny cute adorable fluffy baby raccoon with knitted blue scarf leaning at a table in a medieval pub holding a coffee cup, anthropomorphic, Jean-Baptiste Monge, soft cinematic lighting, 8k, intricate details, portrait, Pixar style character, old fashioned movie style

Architectural parametric pavilion made from wood and glass, with organic cavities, surrounded by a beautiful forest. Dramatic scene, photorealistic, hyperrealistic, raytracing reflections, 8k hd, intricate detail in the style of Frank Lloyde Wright

Beautiful surreal symbolism the mesmerizing vision of a Cleopatra Queen of Egypt , full body , mesmerizing brown eyes, black hair and ethereal features, radiating celestial aura, super high definition, true lifelike color, perfect exposure, razor sharp focus, golden ratio, soft reflections, bokeh effect, fine art photography, cinematic compositing, authentic, professional by Rorianai style 36k s1000

Figure 2. **More text-to-image generation results**. We upsample the images to 1024×1024 for better visualization.



A realistic photo of snowy mountains with sunlit peaks glistening under a clear blue sky	这是一张雪山写实照片，阳光照耀下的山峰在晴朗的蓝天下熠熠生辉。前景中高大的常青树被白雪覆盖，为这令人惊叹的高山景色增添了深度。	Une photo réaliste de montagnes enneigées dont les sommets éclairés par le soleil scintillent sous un ciel bleu limpide. Au premier plan, de grands arbres à feuilles persistantes saupoudrés de neige ajoutent de la profondeur à cette scène alpine à couper le souffle.	澄み切った青空の下、太陽に照らされた山頂が輝く雪山のリアルな写真。前景の雪に覆われた背の高い常緑樹が、息をのむようなアルプスの風景に深みを与えている。	A realistic photo of ❄️ ⛰️ with ☀️ under a clear blue sky. Tall, snow-dusted 🌲 in the foreground add depth to the breathtaking alpine scene.

Figure 3. **Multilingual text-to-image generation samples compared to LlamaGen**. Note that we only use English text-to-image data in training, and this is an emergent capability of our model. The languages used in the prompt, from left to right, are: English, Chinese, French, Japanese, and English with emoji.

Prompt

Generate the latex code of this formula.

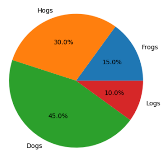
$$A_n = a_0\left[1 + \frac{3}{4}\sum_{k=1}^{n}\left(\frac{4}{9}\right)^k\right]$$

Response

A _ { n } = a_ 0 \left[1 + \frac{3}{4} \sum_{ k=1}^{n} \left(\frac{4}{9} \right)^k \right]

Prompt

Generate the matplotlib.pyplot code for this plot.



Response

The image you've provided is a pie chart that represents the distribution of different types of animals based on their frequency. The chart is divided into four segments, each representing a different type of animal and its corresponding percentage.
Here's the code to generate the pie chart:

```python
import matplotlib.pyplot as plt
# Data
animals = [
  {"name": "Hogs", "frequency": 30.0},
  {"name": "Frog", "frequency": 15.0},
  {"name": "Logs", "frequency": 10.0},
  {"name": "Dogs", "frequency": 45.0}
]
# Create the pie chart
plt.pie(animals, labels=animals, autopct='%1.1f%%',
startangle=90)
# Set the title and labels
plt.title('Distribution of Animals')
plt.title.set_text('Frequency of Animals')
# Show the plot
plt.show()
```

Prompt

What algorithm does the image depict?

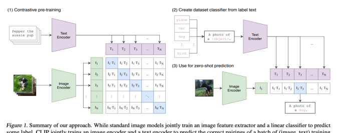

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T₁ T₂ T₃ ... T_N
I₁ I₂ I₃ ... I_N
I₁·T₁ I₁·T₂ I₁·T₃ ... I₁·T_N
I₂·T₁ I₂·T₂ I₂·T₃ ... I₂·T_N
I₃·T₁ I₃·T₂ I₃·T₃ ... I₃·T_N
I_N·T₁ I_N·T₂ I_N·T₃ ... I_N·T_N
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I₁
I₁·T₁ I₁·T₂ I₁·T₃ ... I₁·T_N
A photo of a dog.

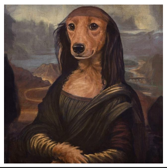
Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Response

The image depicts a CLIP (Contrastive Language-Image Pre-training) model. The CLIP model is a type of pre-training algorithm that combines the power of language and vision to learn a unified representation of the world. The image encoder processes the input image, and the text encoder processes the text associated with the image. The output of the text encoder is then used to predict the correct pair of images, which is a key component of the CLIP model.

Prompt

Why is this painting interesting?



Response

This image is an interesting piece of art because it combines a well-known painting, the "Mona Lisa," with a dog. The dog is depicted in the same pose as the Mona Lisa, but with a different expression and a different background. The painting is a unique blend of art styles and subjects, making it stand out.

Figure 4. **More multimodal understanding results**. Janus has a strong multimodal understanding capability and can handle inputs from various contexts, such as scientific charts, artwork images, LaTeX formula images, and more.