Multimodal Text-to-Image Applications

Since Ollam does not support text-to-image functionality, we need to use other tools to implement local text-to-image functionality. Currently, voice-controlled text-to-image functionality is not supported, so the content of this article is the same as offline multimodal text-to-image applications.

---

1. Concept Introduction

1.1 What is Text-to-Image?

Text-to-image is an artificial intelligence technology that automatically generates corresponding images from **text descriptions**. Simply input a text (e.g., "A Shiba Inu wearing sunglasses is surfing on the beach"), and the AI model will generate an image that matches the description based on semantic understanding, without any drawing or design background.

Core Principles

1. **Language Understanding**: The AI first parses your text and identifies keywords (such as "Shiba Inu," "sunglasses," and "beach").
2. **Image Generation**: Based on training with massive amounts of image data, the AI converts text into visual elements and combines them into a reasonable image.
3. **Style Adaptation:** Allows specifying a style (realistic, cartoon, watercolor, etc.), and the AI will adjust the generated effect accordingly.

1.2 What is FastSDCPU?

FastSDCPU is an open-source project optimized for CPU devices, specifically designed for stable diffusion image generation. Through algorithmic and engineering optimizations, it enables the rapid generation of high-quality images on ordinary computers without GPUs, significantly lowering the hardware barrier for AI painting.

Core Features

1. **Pure CPU Execution:** Requires no dedicated graphics card and can run directly on laptops, desktops, or edge devices.
2. **High-Speed Inference:** Combined with technologies such as Latent Consistency Models (LCM), images can be generated in just 4–8 steps, significantly reducing waiting time.
3. **Lightweight Deployment:** Supports model quantization (such as INT8) and OpenVINO acceleration, reducing resource consumption and improving response efficiency.

Applicable Scenarios

- Localized AI Creation Tools
- Enterprise Intranet Image Generation Service
- Teaching Demonstration and Prototype Development
- Lightweight Deployment in Resource-Constrained Environments

2. Project Deployment

2.1 Deployment Environment

Open a terminal and execute the following code:

```
# If git is not installed on your motherboard, run the following first:
sudo apt update
sudo apt install git -y
sudo apt install python3.10-venv -y

# Add environment variables
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc

# Clone the project code
git clone https://github.com/rupeshs/fastsdcpu.git
cd fastsdcpu

# Create a virtual environment and install dependencies
python -m venv venv
source venv/bin/activate

# Install UV
curl -Ls https://astral.sh/uv/install.sh | sh (This step may fail if you are not
using a proxy in China. If not, you can skip this step and execute the next
command.)

#If the previous step of installing UV using curl failed, run these three
commands:
wget https://mirrors.huaweicloud.com/astral/uv/0.8.4/uv-aarch64-unknown-linux-gnu
-O ~/.local/bin/uv
chmod +x ~/.local/bin/uv
~/.local/bin/uv --version

chmod +x install.sh start-webui.sh
./install.sh --disable-gui
```

Installation successful. Press any key to exit:

```
 + tomesd==0.1.3
 + tomli==2.2.1
 + tomlkit==0.12.0
 - torch==2.7.1+cpu
 + torch==2.5.1
 + torchvision==0.20.1
 + tqdm==4.67.1
 + transformers==4.48.0
 + typer==0.16.0
 - typing-extensions==4.12.2
 + typing-extensions==4.8.0
 + tzdata==2025.2
 + urllib3==2.5.0
 + uvicorn==0.35.0
 + websockets==12.0
 + xxhash==3.5.0
 + yarl==1.20.1
 + zipp==3.23.0
FastSD CPU installation completed,press any key to continue...
```

2.2 Local Area Network Access

Before starting, you need to modify a file to support local area network access; otherwise, webUI can only be accessed locally:

```
vim ~/fastsdcpu/src/frontend/webui/ui.py
```

After opening the ui.py file, scroll to the last line and find the line `**webui.launch(share=share)**`. Modify it to `**webui.launch(server_name="0.0.0.0",share=share)**`

Then save.

Start:

```
./start-webui.sh
```

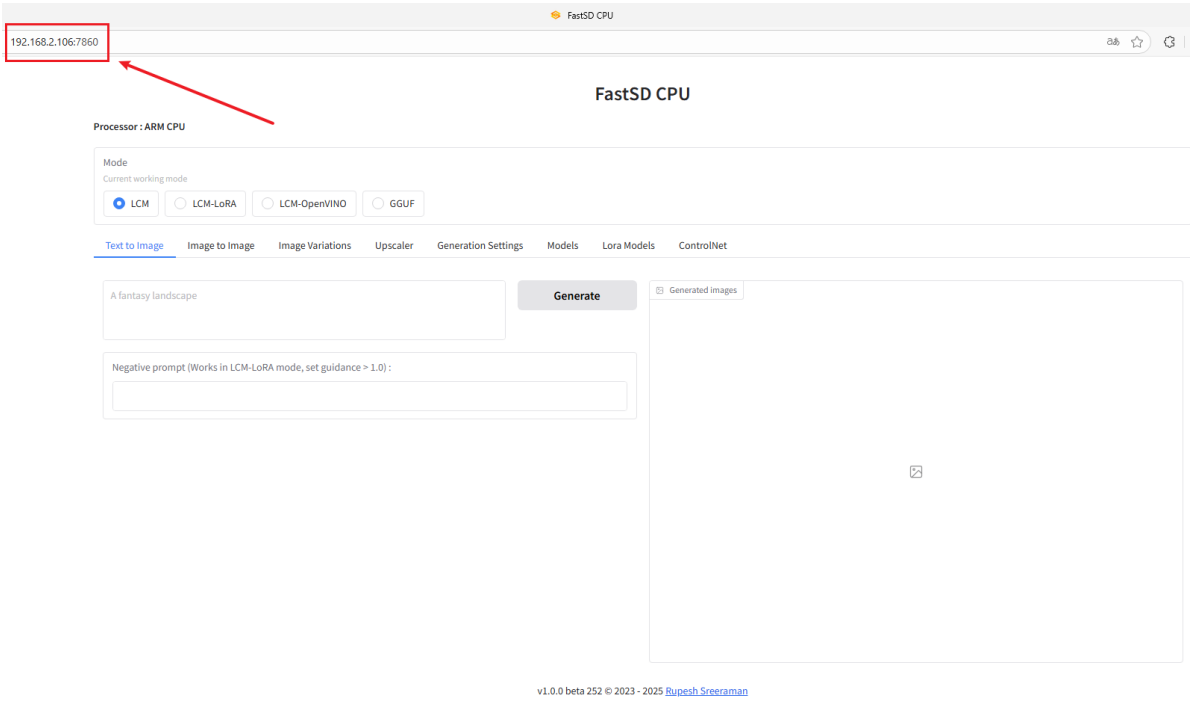
```
Found 7 stable diffusion models in config/stable-diffusion-models.txt
Found 4 LCM-LoRA models in config/lcm-lora-models.txt
Found 10 OpenVINO LCM models in config/openvino-lcm-models.txt
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/controlnet_aux/segment_anything/modeling/tiny_vit_sam.py:654:
 UserWarning: Overwriting tiny_vit_5m_224 in registry with controlnet_aux.segment_anything.modeling.tiny_vit_sam.tiny_
vit_5m_224. This is because the name being registered conflicts with an existing name. Please check if this is not exp
ected.
  return register_model(fn_wrapper)
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/controlnet_aux/segment_anything/modeling/tiny_vit_sam.py:654:
 UserWarning: Overwriting tiny_vit_11m_224 in registry with controlnet_aux.segment_anything.modeling.tiny_vit_sam.tiny
_vit_11m_224. This is because the name being registered conflicts with an existing name. Please check if this is not e
xpected.
  return register_model(fn_wrapper)
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/controlnet_aux/segment_anything/modeling/tiny_vit_sam.py:654:
 UserWarning: Overwriting tiny_vit_21m_224 in registry with controlnet_aux.segment_anything.modeling.tiny_vit_sam.tiny
_vit_21m_224. This is because the name being registered conflicts with an existing name. Please check if this is not e
xpected.
  return register_model(fn_wrapper)
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/controlnet_aux/segment_anything/modeling/tiny_vit_sam.py:654:
 UserWarning: Overwriting tiny_vit_21m_384 in registry with controlnet_aux.segment_anything.modeling.tiny_vit_sam.tiny
_vit_21m_384. This is because the name being registered conflicts with an existing name. Please check if this is not e
xpected.
  return register_model(fn_wrapper)
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/controlnet_aux/segment_anything/modeling/tiny_vit_sam.py:654:
 UserWarning: Overwriting tiny_vit_21m_512 in registry with controlnet_aux.segment_anything.modeling.tiny_vit_sam.tiny
_vit_21m_512. This is because the name being registered conflicts with an existing name. Please check if this is not e
xpected.
  return register_model(fn_wrapper)
Starting web UI mode
No lora models found, please add lora models to /home/sunrise/fastsdcpu/lora_models directory
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/gradio/components/dropdown.py:226: UserWarning: The value pas
sed into gr.Dropdown() is not in the list of choices. Please update the list of choices to include:  or set allow_cust
om_value=True.
  warnings.warn(
/home/sunrise/fastsdcpu/env/lib/python3.11/site-packages/gradio/components/base.py:194: UserWarning: show_label has no
 effect when container is False.
  warnings.warn("show_label has no effect when container is False.")
* Running on local URL:  http://0.0.0.0:7860

To create a public link, set `share=True` in `launch()`.
```

Then you can access webUI by entering your motherboard IP:7860 in your browser.

2.3 Using the Image Generator Function

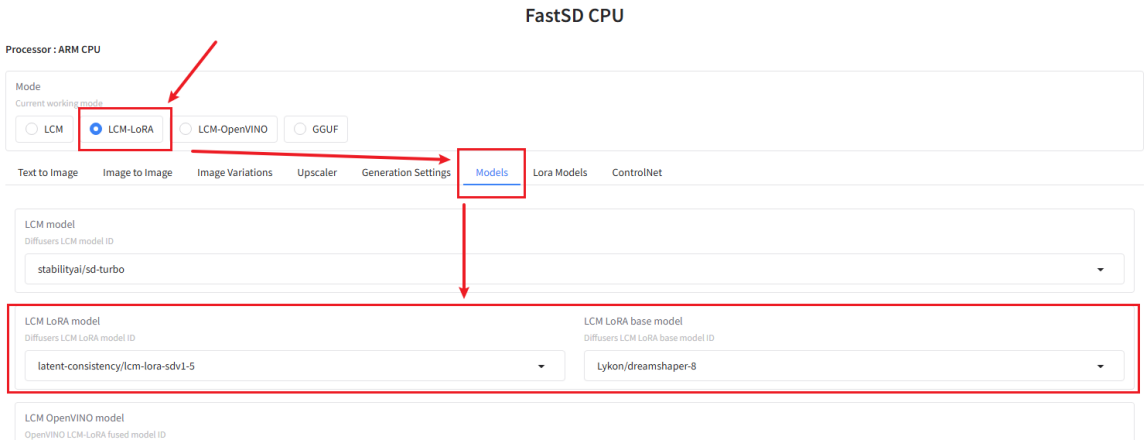
In the terminal, use `ifconfig` to find your motherboard's IP address. For example, mine is 192.168.2.106.

Then open your browser and enter **your motherboard IP:7860**. For example, I entered 192.168.2.106:7860, and then I was able to access the web UI.

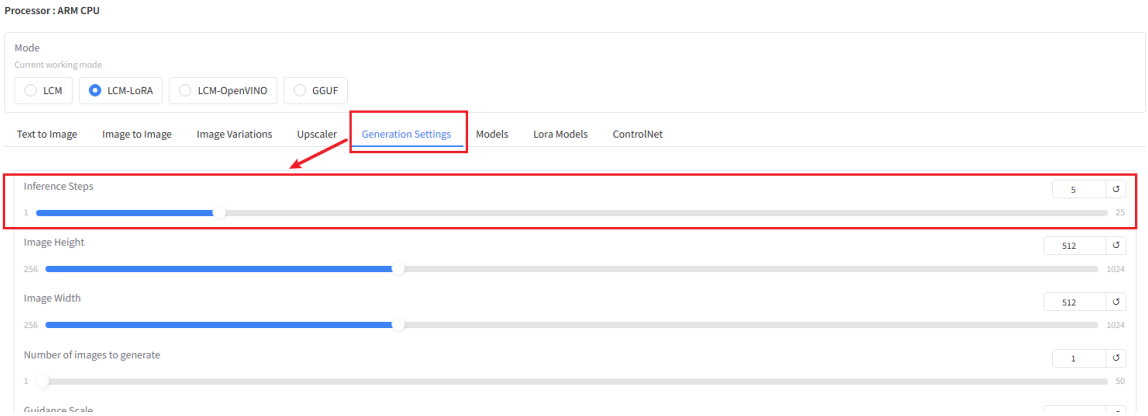


Next, click on LCM-LoRA. This model has a relatively small memory footprint. If you want to use other models, you can explore them on your own.

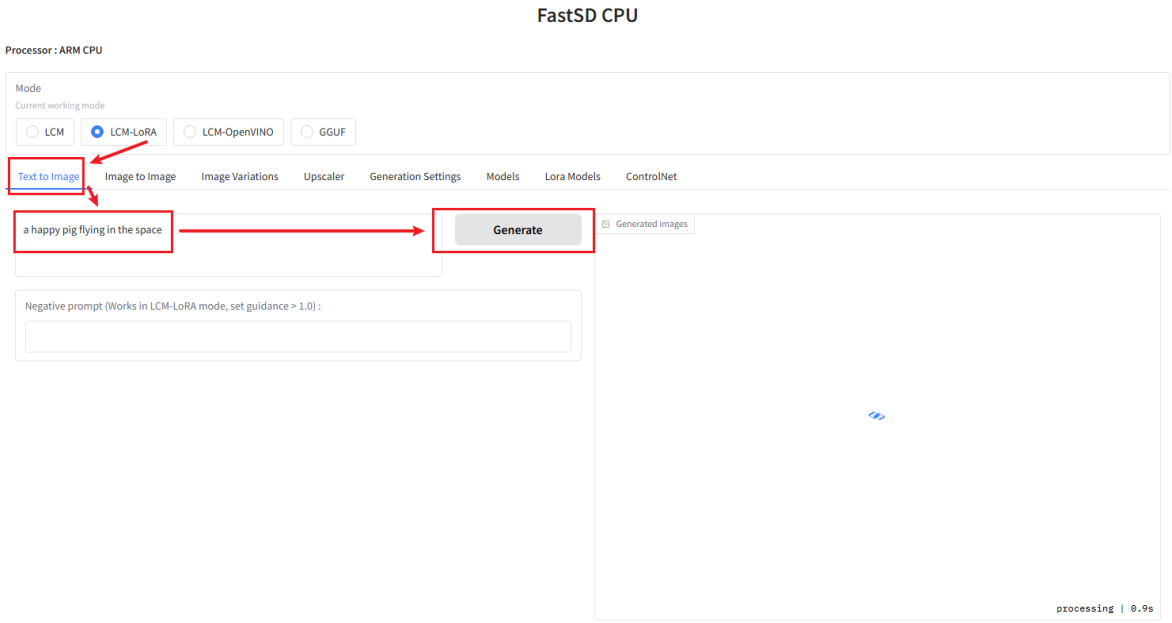
Then click on Models. You can see the LCM LoRA model settings. You can change the model you want, or you can choose the default settings like I did.



Next, click Generation Settings and increase the Inference Steps to improve the quality of the generated image. I've set it to 5 here.



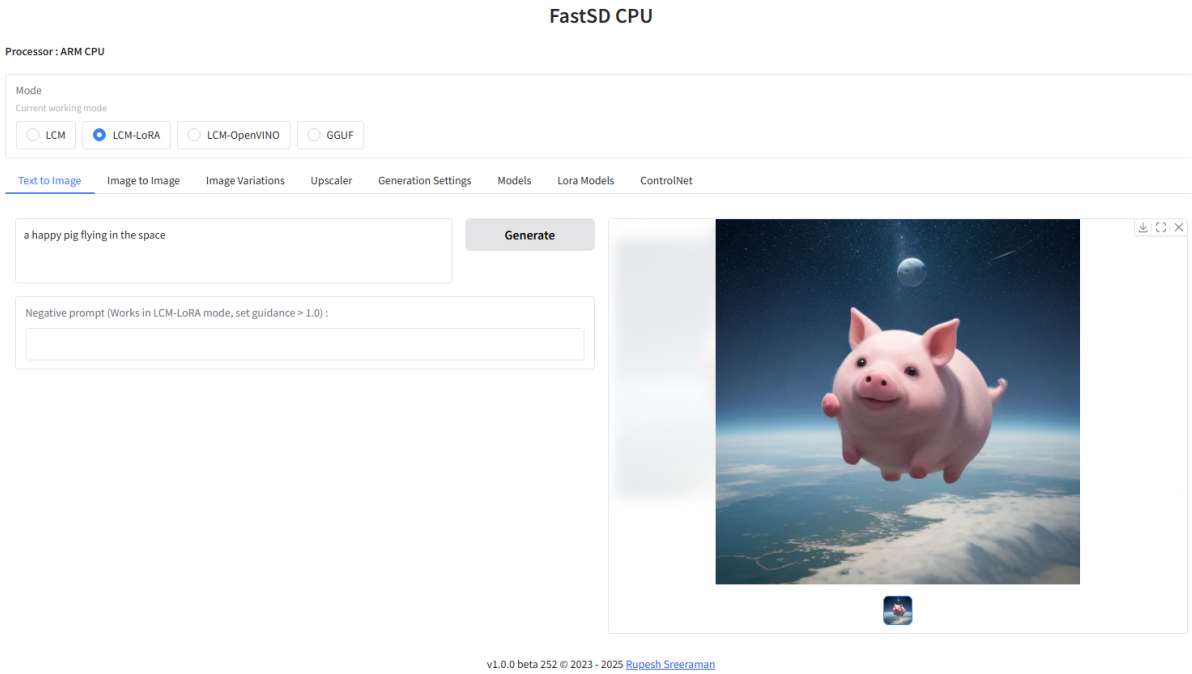
Then return to Text to Image. Enter the content you want to generate in the dialog box, and then click Generate to start generating the image.



For first-time use, you need to download the model. You can see the default model being downloaded in the terminal. Once the download is complete, the text-to-image generation function will begin.

```
'init_image': None,
'lcm_lora': {'base_model_id': 'Lykon/dreamshaper-8',
             'lcm_lora_id': 'latent-consistency/lcm-lora-sdv1-5'},
'lcm_model_id': 'stabilityai/sd-turbo',
'lora': {'enabled': False,
         'fuse': True,
         'models_dir': '/home/sunrise/fastsdcpu/lora_models',
         'path': '',
         'weight': 0.5},
'negative_prompt': '',
'number_of_images': 1,
'openvino_lcm_model_id': 'rupeshs/sd-turbo-openvino',
'prompt': 'a happy pig flying in the space',
'rebuild_pipeline': False,
'seed': 123123,
'strength': 0.6,
'token_merging': 0.0,
'use_gguf_model': False,
'use_lcm_lora': True,
'use_offline_model': False,
'use_openvino': False,
'use_safety_checker': False,
'use_seed': False,
'use_tiny_auto_encoder': False}
***** Init LCM-LoRA pipeline - Lykon/dreamshaper-8 *****
model_index.json: 100%|| 642/642 [00:00<00:00, 2.80MB/s]
config.json: 100%|| 796/796 [00:00<00:00, 3.81MB/s]
config.json: 100%|| 724/724 [00:00<00:00, 3.74MB/s]
special_tokens_map.json: 100%|| 472/472 [00:00<00:00, 2.27MB/s]
merges.txt: 525kB [00:00, 6.20MB/s]                 | 0.00/472 [00:00<?, ?B/s]
preprocessor_config.json: 100%|| 520/520 [00:00<00:00, 2.53MB/s]
tokenizer_config.json: 100%|| 737/737 [00:00<00:00, 3.54MB/s]
config.json: 1.87kB [00:00, 5.15MB/s]               | 0.00/1.22G [00:00<?, ?B/s]
vocab.json: 1.06MB [00:00, 9.26MB/s]                | 0.00/520 [00:00<?, ?B/s]
scheduler_config.json: 100%|| 614/614 [00:00<00:00, 2.99MB/s]
config.json: 100%|| 756/756 [00:00<00:00, 3.99MB/s]
model.safetensors:   2%|| 10.5M/492M [00:00<00:23, 20.3MB/s]
model.safetensors:  32%|| 157M/492M [00:36<01:24, 3.98MB/s]
model.safetensors:  12%|| 147M/1.22G [00:34<04:28, 3.98MB/s]
diffusion_pytorch_model.safetensors:  41%|| 136M/335M [00:34<00:50, 3.96MB/s]
diffusion_pytorch_model.safetensors:   4%|| 136M/3.44G [00:34<13:52, 3.97MB/s]
```

Generated Result:



In practice, this project supports English better, and the generated images are more closely aligned with the text. It is recommended to use English descriptions to generate images.

2.4 Changing the Model Download Location

The default download location for models is: /.cache/huggingface/hub

We can change the download path of our Huggingface models in the terminal before each model download:

```
export HF_HOME=/path/to/cache/directory # Replace "path to" with your desired path
```

This will change the download path of your Huggingface models. For example, if I want to place the model on an SSD and not use the motherboard's hard drive space, I can enter the following command:

```
export HF_HOME=/media/ssd/HF_Model
```

Here, /media/ssd is the path of my mounted SSD. You should modify it according to your own situation.
If you have already downloaded the model to the default path and want to change its location later, you can use the following command:

```
mv ~/.cache/huggingface /path/to/cache/directory # Replace "path to" with the path you want to move the model to
```

If you find it troublesome to use this command every time you download a model, you can directly add this export command to your .bahrc file.