

Multimodal Text-to-Image Applications

Multimodal Text-to-Image Applications

1. Concept Introduction
 - 1.1 What is Text-to-Image?
 - Core Principles
 - 1.2 What is FastSDCPU?
 - Core Features
 - Applicable Scenarios
2. Project Deployment
 - 2.1 Deployment Environment
 - 2.2 LAN Access
 - 2.3 Using the Vinyl Image Function
 - 2.4 How to start after successful deployment

Since Ollam doesn't support text-to-image functionality, we need to use other tools to implement local text-to-image functionality.

1. Concept Introduction

1.1 What is Text-to-Image?

Text-to-image is an artificial intelligence technology that automatically generates corresponding images based on **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, requiring no drawing or design skills.

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:** You can specify a style (realistic, cartoon, watercolor, etc.), and the AI will adjust the generated effect.

1.2 What is FastSDCPU?

FastSDCPU is an open-source project for stable diffusion image generation, specifically optimized for CPU devices. 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 Operation:** No dedicated graphics card required; can run directly on laptops, desktops, or edge devices.
2. **High-Speed Inference:** Combining 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 Demonstrations and Prototype Development
- Lightweight Deployment in Resource-Constrained Environments

2. Project Deployment

2.1 Deployment Environment

Note: If using our factory image, no deployment environment is required; skip the deployment steps. Refer directly to <2.4 Startup Method After Successful Deployment> at the bottom to start. Then return to <2.3 Using the Text-to-Image Function> to learn how to use it.

Open a terminal and execute the following code:

```
#If there is no git on the motherboard, run
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/rupehs/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 not be successful if
there is no proxy in China. If not, you can skip this step and execute the next
command)

#If the curl command failed to install uv in the previous step, run these three
instructions
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
+ tomkit==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 LAN Access

Before starting, you need to modify a file to support LAN access. Otherwise, the 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 **webui.launch(share=share)**. Change it to **webui.launch(server_name="0.0.0.0",share=share)**.

Save the file.

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 expected.
    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 expected.
    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 expected.
    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 expected.
    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 expected.
    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 passed into gr.Dropdown() is not in the list of choices. Please update the list of choices to include: or set allow_custom_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

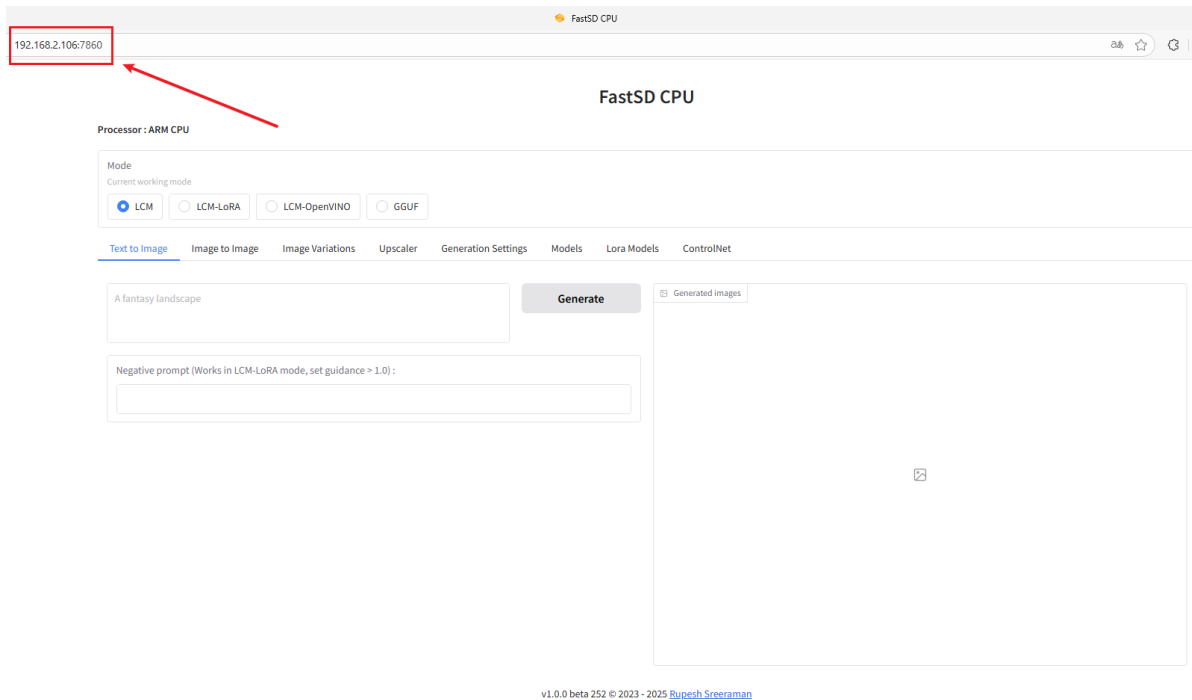
To create a public link, set `share=True` in `launch()`.
```

Then, enter your motherboard's IP address: 7860 in your browser to access the webUI.

2.3 Using the Vinyl Image Function

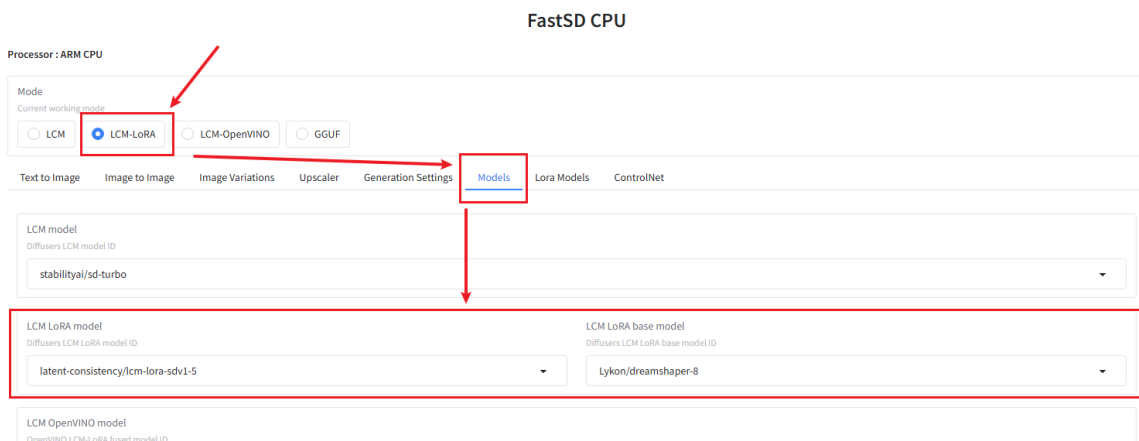
Use ifconfig in the terminal to query your motherboard's IP address. For example, mine is 192.168.2.106.

Then open your browser and enter **your motherboard's IP address: 7860** . For example, I entered 192.168.2.106:7860 . This will allow you to access the webUI.



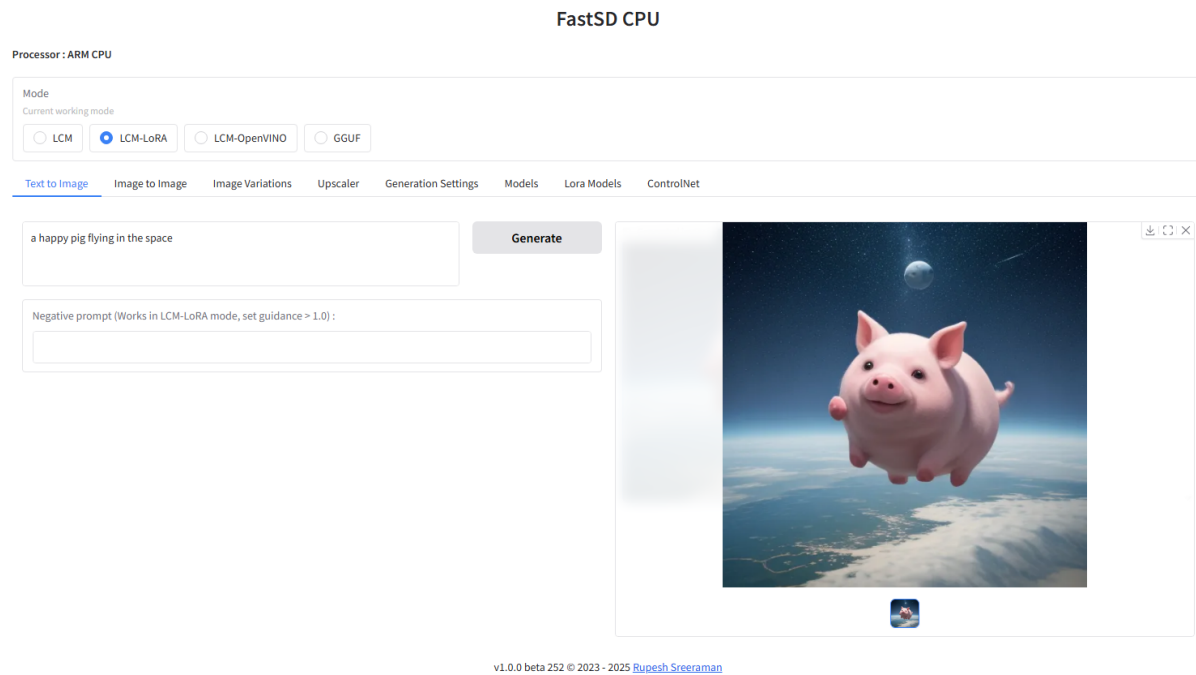
Next, click LCM-LoRA. This model uses less memory. If you prefer, feel free to explore other models.

Next, click Models to view the LCM LoRA model settings. You can change the model you want, or just stick with the default, as I did.



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

Generated result:



This project has better support for English, and the generated images are more consistent with the text. We recommend using English descriptions when generating images.

2.4 How to start after successful deployment

```
cd fastsdcpu #Enter the fastsdcpu directory
source venv/bin/activate #Enter the virtual environment
./start-webui.sh #Start the webui
```

After the webui is successfully started, enter your motherboard's IP address: 7860 in your browser to start the image generation function.