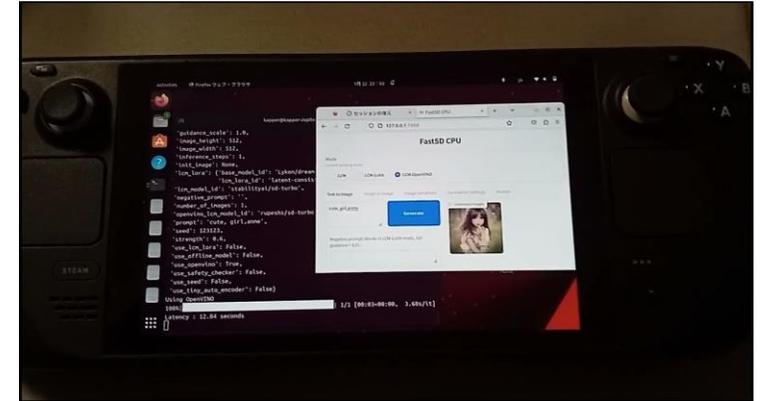


CPUでもOK,はじめてのローカル画像生成AI。Stable-Diffusion-ForgeとFastSD CPU,Draw Things他など。練習例題あり初心者向け講座。LCM-LoRa高速化,LLMと組み合わせ技などあり。NetBSDもOK。

1. 準備
2. ローカル画像生成AIとLLM
3. Stable-Diffusion演習
4. LCM-LoRa高速化他
5. その他



発表者 : Kapper

ガジェットハッキング  
ユーザーグループ

This Presentation:

Slideshare & PDF files  
publication of my HP

<http://kapper1224.sakura.ne.jp>

Gadget Hacking User Group

Speaker:Kapper

メインテーマ「ガジェット & 生成AI」

OSC京都2024 2024年7月27日

Place: 中小企業振興会館

# 自己紹介 Self Introduction

- My name: Kapper
- Twitter account : @kapper1224
- HP : <http://kapper1224.sakura.ne.jp>
- Slideshare: <http://www.slideshare.net/kapper1224>
- Mastodon: <https://pawoo.net/@kapper1224/>
- Facebook : <https://www.facebook.com/kapper1224/>
- My nobels : <https://ncode.syosetu.com/n7491fi/>
- My Together : <https://togetter.com/id/kapper1224>
- My Youtube : <http://kapper1224.sakura.ne.jp/Youtube.html>
- My Hobby : Linux、 \*BSD、 and Mobile Devices
- My favorite words : The records are the more important than the experiment.
- Test Model : Netwalker、 Nokia N900、 DynabookAZ、 RaspberryPi、 Nexus7、 Nexus5、 Chromebook、 GPD-WIN、 GPD-Pocket、 Macbook、 NANOTE、 Windows Tablet、 SailfishOS、 UBPorts、 postmarketOS、 NetBSD and The others...
- Recent my Activity :
  - Hacking Generative AI (Images and LLM) in a lot of devices.
  - Hacking Linux on Windows10 Tablet (Intel Atom) and Android Smartphone.
  - Hacking NetBSD and OpenBSD on UEFI and Windows Tablet.
  - I have over 200 Windows Tablet and 120 ARM Android, and test it now.
- 後、最近小説家になろうで異世界で製造業と産業革命の小説書いていますなう。

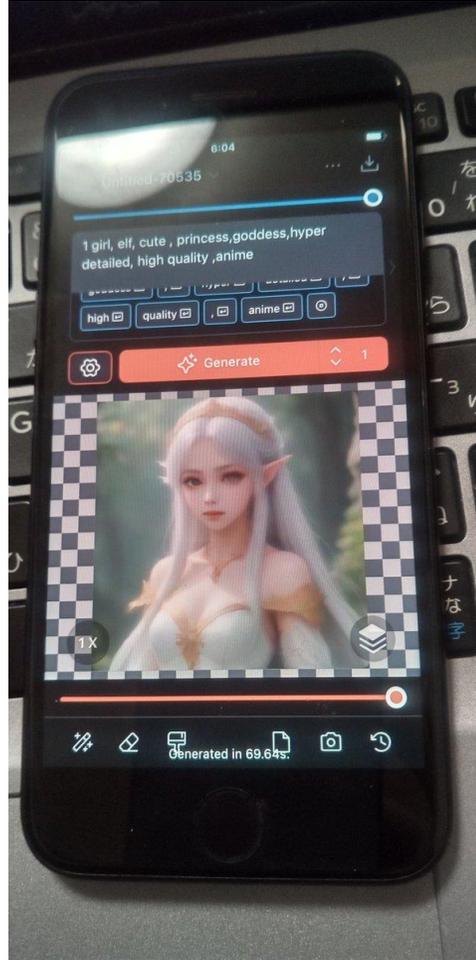


# Recently my activities

Ollama, llama.cpp



Stable-diffusion



Windows Tablet



UMPC



Book8088, Pocket386



OSC, KOF参加



# OSC名古屋2024で発表したローカル生成AIの資料を公開中。よろしくお願ひします。

- Slideshare、トクセル、Speakerdeck、PDFファイルなど  
<https://www.slideshare.net/slideshow/ai-llm-llama-cpp-llamafile/269337758>

## はじめての生成AIローカルLLM、llama.cppとllamafile。練習演習あり初心者向け講座

May 26, 2024 • 0 likes • 328 views

 Netwalker lab kapper

OSC名古屋2024の発表資料です。生成AIのローカルLLMを対象です。



2 of 47

 Download now

Download to read offline

Page 1



Page 2



Page 3



Page 4



Page 5



Page 6



## はじめての生成AIローカルLLM、llama.cppとllamafile。練習演習あり初心者向け講座

1. 準備
2. ローカルLLMと生成AI
3. llamafile演習
4. llama.cpp, Ollama他
5. その他



発表者: Kapper

ガジェットハッキング  
ユーザーグループ

This Presentation:  
Slideshare & PDF files  
publication of my HP  
<http://kapper1224.sakura.ne.jp>  
Gadget Hacking User Group  
Speaker: Kapper

メインテーマ「ガジェット&生成AI」

OSC名古屋2024 2024年5月25日  
Place: 中小企業振興会館 15:00~15:45

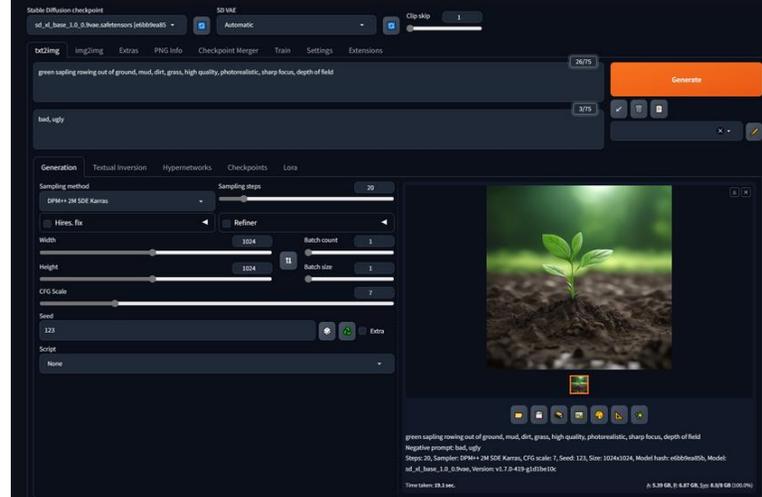
# 事前ご連絡

- 今回は画像生成AIの使い方練習講座です
- 演習にStable-Diffusion-Webui Forge, FastSD CPU, Stable-Diffusion.cppを使用します
- Githubからダウンロードお願いします  
Win, Linux, \*BSD, Mac, Android対応
- iOSの方はDraw Thingsをご使用ください
- GPUは必ずしも必須ではなく、CPUは遅いだけ  
AMD RadeonやIntel Arcでも動く

## Screenshots of Comparison

I tested with several devices, and this is a typical result from 8GB VRAM (3070ti laptop) with SDXL.

This is original WebUI:

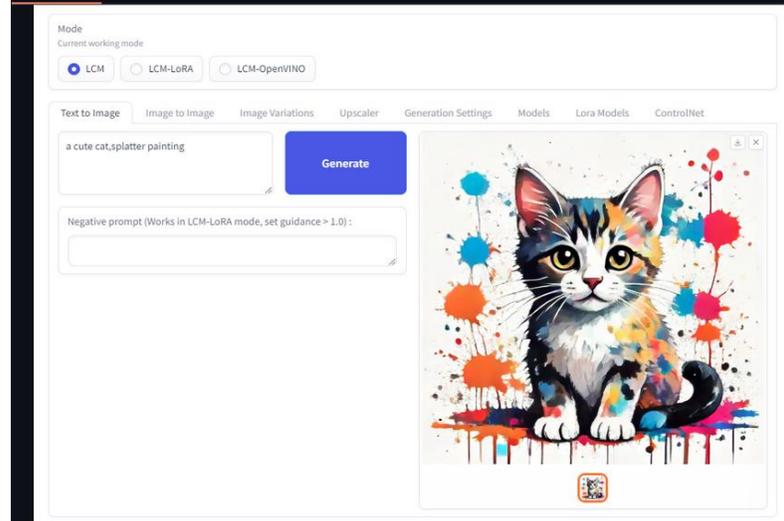


FastSD CPU  

 GITHUB TRENDING  
#6 Repository Of The Day

FastSD CPU is a faster version of Stable Diffusion on CPU. Based on [Latent Consistency Models](#) and [Adversarial Diffusion Distillation](#).

[README](#)  MIT license



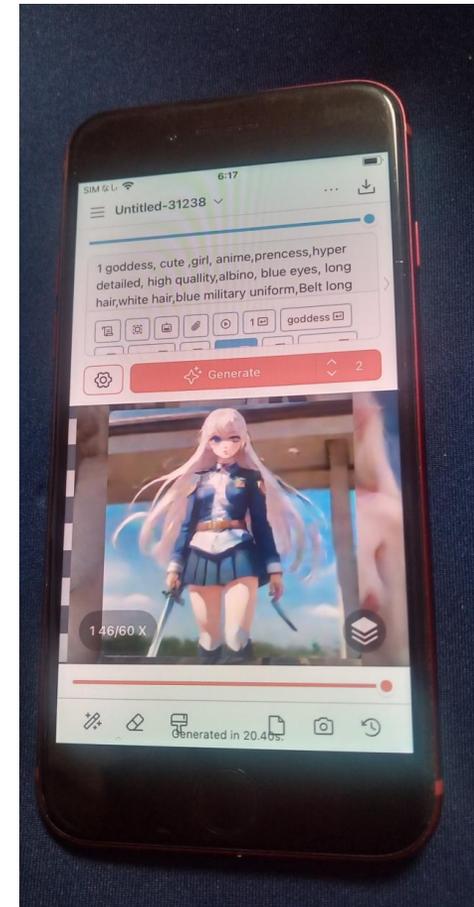
# 事前アンケートお願い (挙手にてお願いいたします)

- ①ChatGPTやStable-Diffusionなど生成AIを時々使っている人？
- ②仕事もしくは商売で生成AIを使いたい人？
- ③ローカル画像生成AIを使った事がある人？
- ④やっぱりLLMより画像生成AIの方が興味がある人？

今日のセミナーのペース時間配分と今後の参考とさせて下さい

# 生成AIとは?

- テキストなど 小さい情報から画像、動画、文章などを生成
- 短時間で複雑なデータを生成可能で生産性が良い
- 複雑な計算が必要なので高性能なCPU、GPUが必要
- 大規模言語モデル (Large Language models) が最近有名
- 最近ではAndroidやiPhoneでもOK
- 他にも画像生成AIなど  
定番モデル pony diffusion、Animage XL他  
参考としあき wiki [https://wikiwiki.jp/sd\\_toshiaki/SDXLモデル](https://wikiwiki.jp/sd_toshiaki/SDXLモデル)  
もちろん blog <https://chimolog.co/bto-gpu-stable-diffusion-specs/>



# 最近の画像生成AIとは？

- 画像生成AIとはStable-Diffusion類似品の総称でText to Image
- 業務用は主に広告業界。5秒の動画を繋いで短いCMも低コストで作成
- 様々なオープンソースモデルが作られ色々な種類の画像生成。動画も。
- LCM-LoRAにより**少ないステップ数で短時間画像生成**
- 出力される情報を他のAIと組み合わせ活用範囲が拡大
- 計算するGPUコストかかるが、**CPUやNPU**でも計算出来る様になった
- 英語圏中心でキーワードは英語。
- 本質的に何でも出来るAIの必要は無い。高コスト
- 自由に改変出来て、自由に使えるライセンスのモデルに注目
- CPUで動かすならWebサーバ、ブラウザも重し。xさえ不要

# プロプラChatGPT VS ローカル生成AI

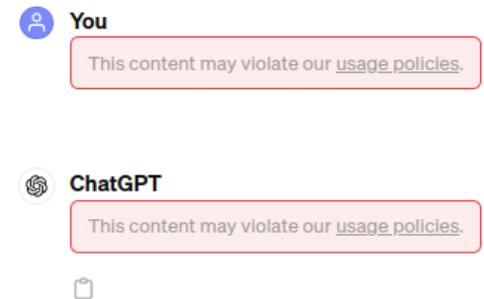
- ここが凄いここが厳しいChatGPT
  - とにかく賢い。Web検索可能。画像や動画なども？
  - 一部無料。多分赤字経営。MicroSoftが助けてくれる？
  - Wikipediaや論文など校正済み文を利用しているため、マニアックな知識に弱い
  - どんどん規制が厳しくなる。将来どうなることか？
- ここが凄いここが厳しいローカル生成AI
  - 安価。オープンソース。API無料**
  - 独自辞書、独自知識を学習可能
  - 規制は自由。付けるも無くすも学習済モデル次第。
  - サーバの経済的理由よりそのうち全てのパソコン、スマホに入るだろう
  - 実はCPU、GPUベンチマークの代替え手段として重宝しているw

# しかし何故今更ローカル生成AI？

- 手元の機種で学習し自由に動かせるという**ロマン**
- ライセンスを侵害しない範囲で**自由に学習・改造**出来る
- 今後 厳しくなるであろう規制が無い。**API無料**。有料化対策。
- UNIXなどの**マニアックな情報を沢山学習**させたい
- オリジナルアプリへライセンスフリーで組み込みたい**パクり**たい
- ~~エロい~~の**を作りたい人向け**
- ~~アンダーグラウンド~~で**危険なLLMを作りたい人向け**
- ~~キャラクター~~を**模倣させたい人向け**
- ~~ギャンブル~~関係の**LLMを作りたい人向け**
- ~~ベンチマーク~~の**代わりw**

ChatGPT 3.5 ▾

しかしChatGPTは  
規制用語が入ると...



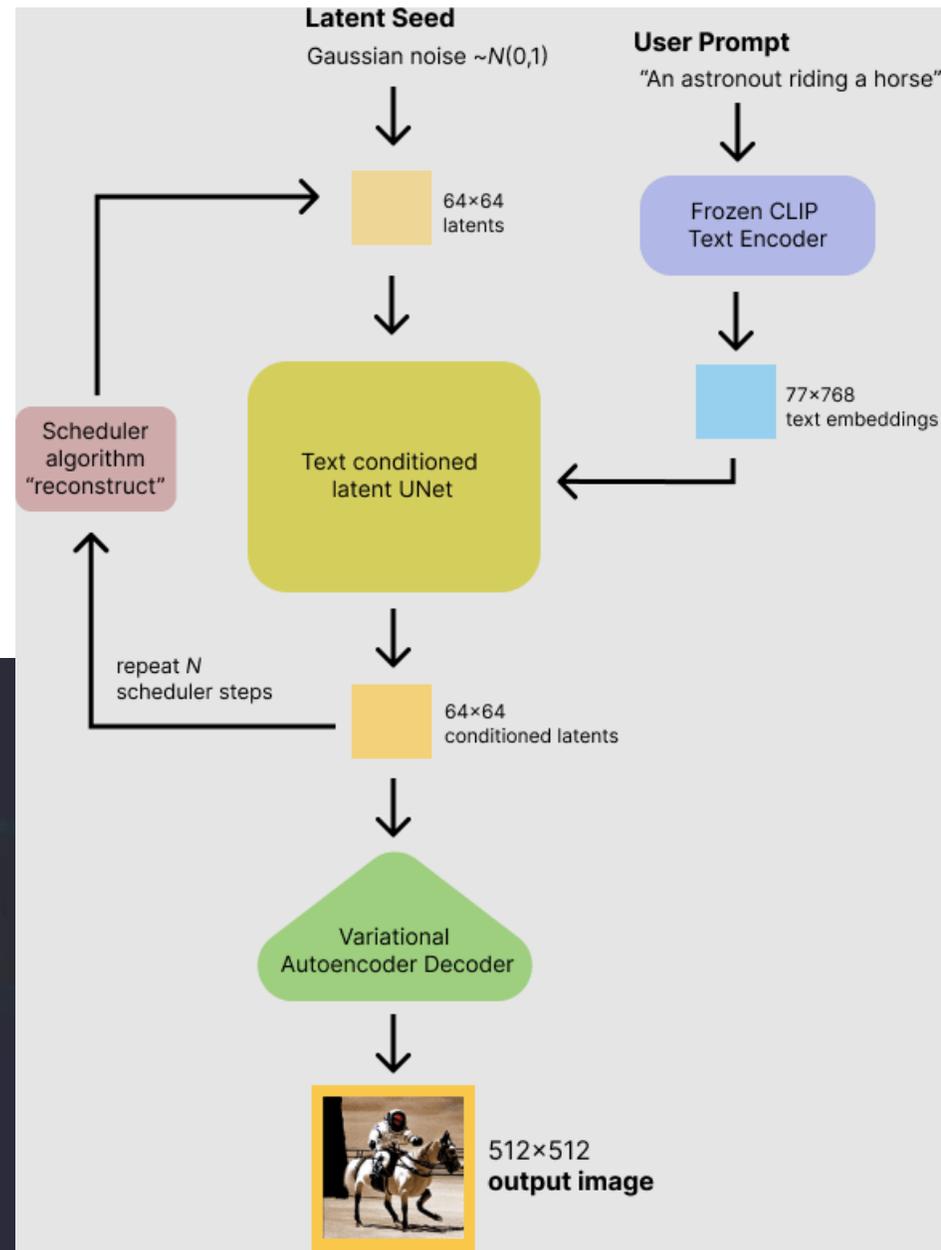
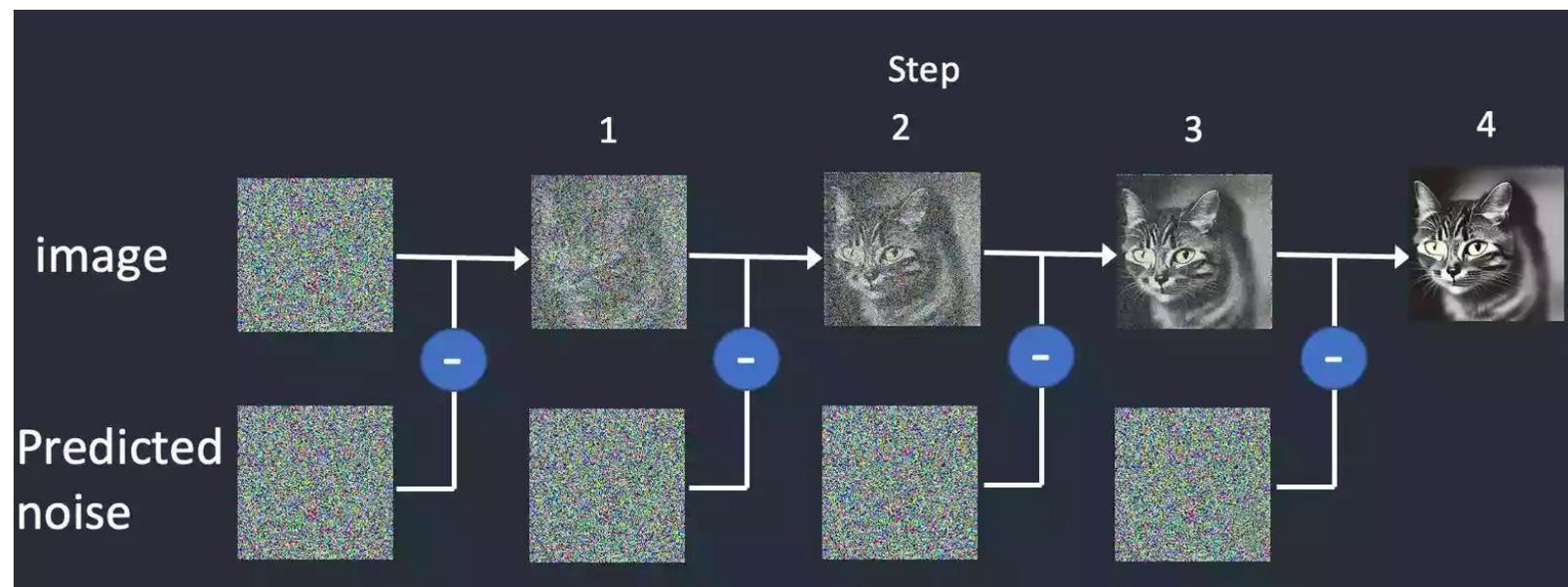
# 画像生成AIの動作原理

- 英語で登録したキーワードを元に画像生成

Stable-Diffusionはノイズ除去・生成する過程でキーワードで合成

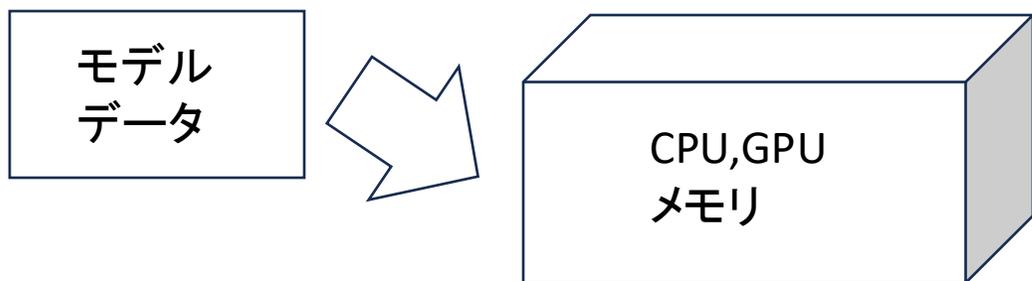
→学習データに大きく依存する

標準Step数は20だが、1~8Stepも可能

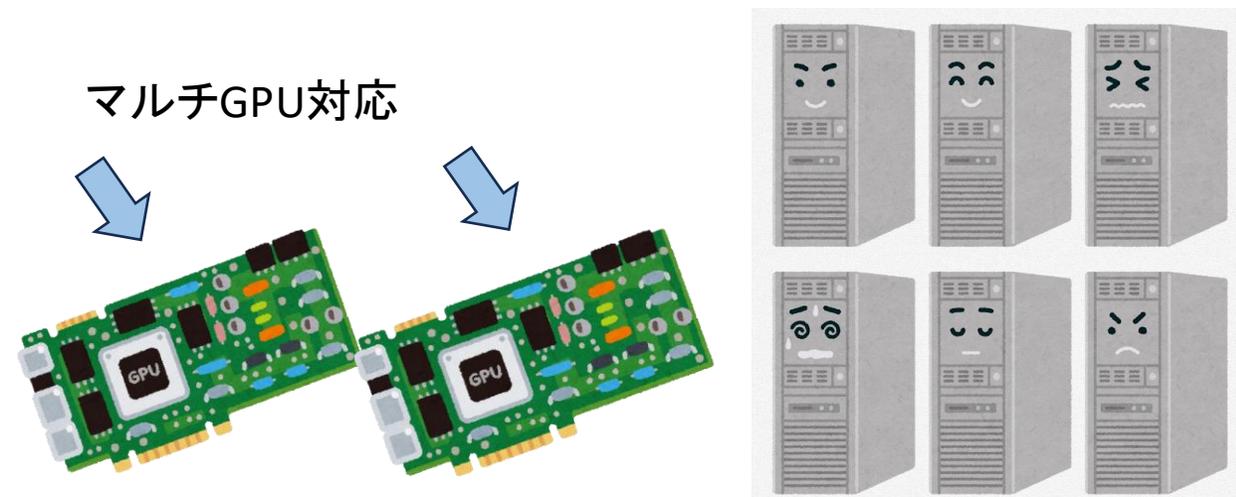


# 動作イメージ

- 一旦モデルをメモリの中に入れる  
(大量のメモリが必要)



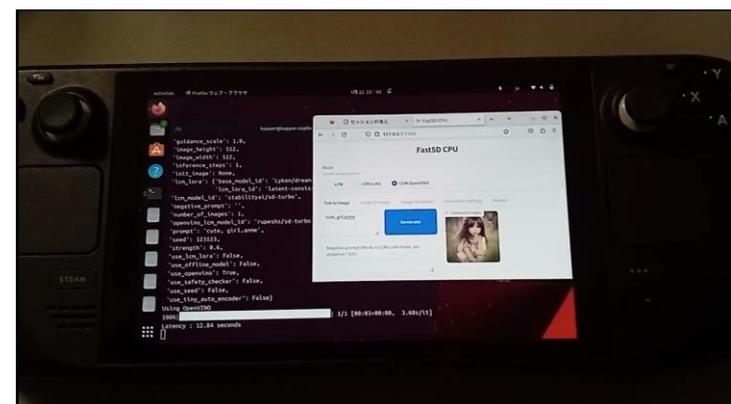
- CPU・GPUで計算



- LoRaがあれば追加



- 結果出力

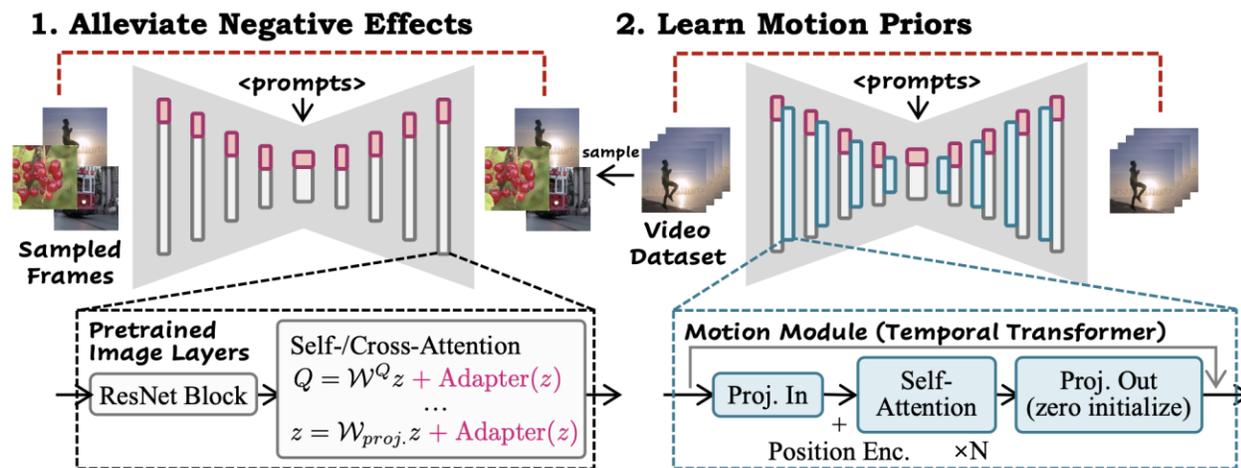


# 画像生成AIの有名な技術

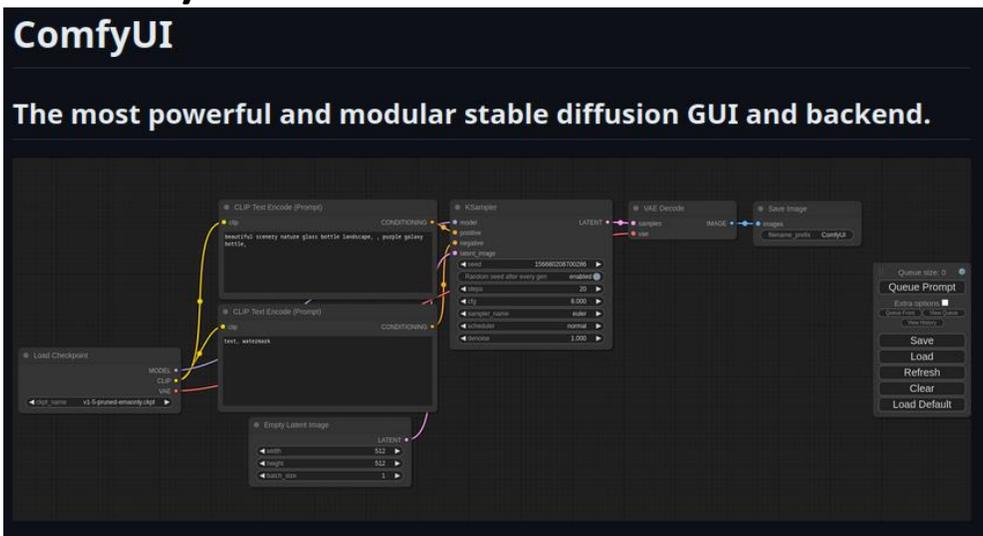
## ControlNet



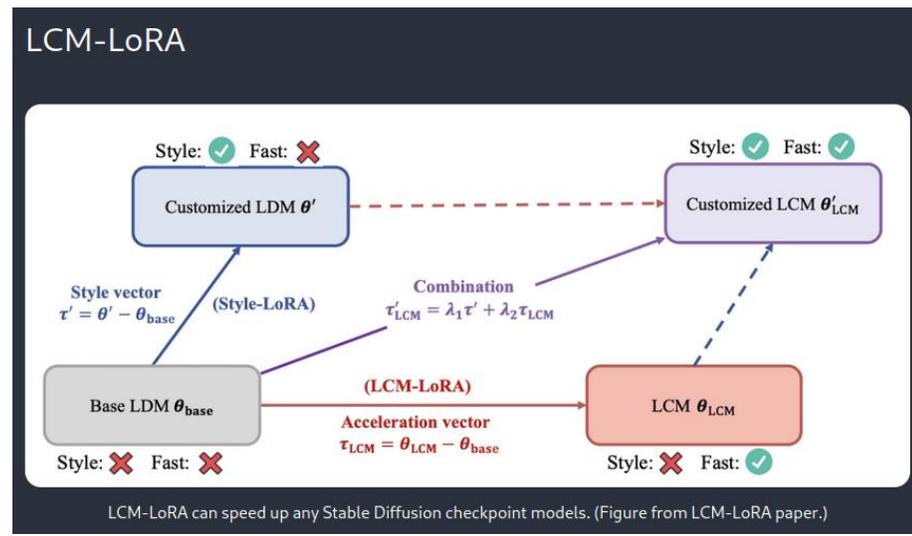
## Animate Diff



## ComfyUI



## LCM-LoRA

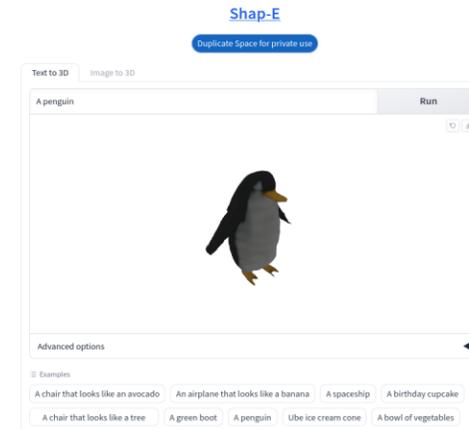


# 最近のプロプラエタリ画像生成AI関係

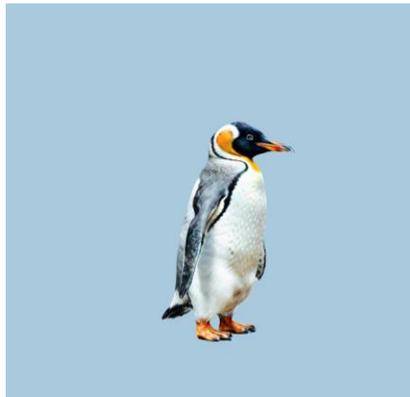
- LumaAI DreamMachine, Kling\_AI  
: 5秒～までの動画作成



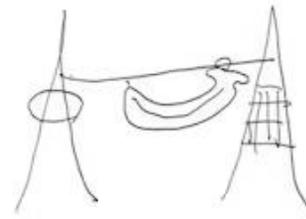
- Shape-E、TripoSR他  
: 3D ポリゴンモデル作成、出力



- TripoSR他 : 3Dプリンター出力  
GLB,FBX,OBJ,USD,STLサポート



- Adobe Firefly、Stability AI他  
: 手書きを画像認識



# OS別Stable-Diffusion この手の亜流は山ほどあるので一部

	Linux	Windows	Mac	iOS	Android (Termux)	Free BSD	Open BSD	Net BSD	コメント
Stable-Diffusion-WebUI A1111	◎	◎	◎	—	◎	△	—	—	
Stabel-Diffusion-WebUI-reForge	◎	◎	◎	—	◎	—	—	—	省メモリGPU
Stable-Diffusion-WebUI-amdgpu	◎	◎	○	—	—	—	—	—	DirectML,ROCm
SD.Next	◎	◎	◎	—	—	—	—	—	IntelGPU他
FastSD CPU	◎	◎	◎	—	◎	—	—	—	OpenVINO
Stable-Diffusion.cpp	◎	◎	◎	—	◎	○	△	△	GCC,cmake
sd.cpp-webUI	◎	◎	◎	—	?	—	—	—	Python
OnnxStream	◎	◎	◎	—	△	?	?	?	512MB RAM,GCC
Koboldcpp	◎	◎	◎	—	◎	—	—	—	簡単
Stablity Matrix	◎	◎	◎	—	—	—	—	—	簡単
Draw Things	—	—	◎	◎	—	—	—	—	A12,M1以降

UNIX系が使いにくいのはPip,CondaとCUDAとPytorchの依存性の関係  
◎: 公式サポート, ○: 公式非サポート, △: 動作報告あり

# ハードウェア目安

- 最低 (動くだけ。速度は一切考慮しない。QemuなどPCエミュレーション環境も想定)  
x86: **SSE3搭載CPU**、メモリ+Swap: 3.7GB以上 (2.7GB+OS:モデル量子化の例外あり)  
Android メモリ+Swap 3.7GB以上 (2.7GB+OS:モデル量子化の例外あり)  
**iPhone SE2**以降 (2020年度～ A12, M1以降)  
RaspberryPi4,5 メモリ+Swap 3.7GB以上
- 必要  
x86: 第8世代Intel、AMD Ryzen以降、**メモリ+Swap 13GB以上 (WebUIを使う場合必須)**  
Android メモリ+Swap 13GB以上  
GPU: **メモリ4GB以上**
- 推奨 (GPU前提)  
x86: 第8世代Intel、AMD Ryzen以降、メモリ16GB以上 (GPUを使う場合、CPUは重要ではない)  
GPU: NVIDIA RTX3050、1660Ti以降 **メモリ6GB以上 (12GB以上が望ましい)**

詳しくは、「もちろぐ」さんのブログを参照

# 今ならこの程度の型落ちでも・・・幸せに？

状態：液晶ドット抜け/スポット抜け/ムラ、バームレストスレキズ、天板キズスレ、底面打痕/凹み/キズ、キーテカリ（軽度）/キーボード隙間汚れあり  
※メモリ16GB→24GBに増設済み  
付属品：電源ケーブル、ACケーブル付属

【CPU】 Core i7 9750H  
【CPUクロック】 2.6GHz  
【搭載OS】 Windows 10 Home 64bit  
【標準メモリ容量】 24GB  
【内蔵ストレージ容量】 512GB(SSD)  
【ドライブ】 -  
【グラフィックコントローラ名称】 nVIDIA GeForce GTX 1660 Ti  
【TVチューナ】 -  
【有線LAN】 ○  
【無線LAN/その他】 ○  
【ディスプレイ（画面）サイズ】 15.5インチ  
【付属ディスプレイ仕様】 1920x1080  
【発売時期】 2019/10

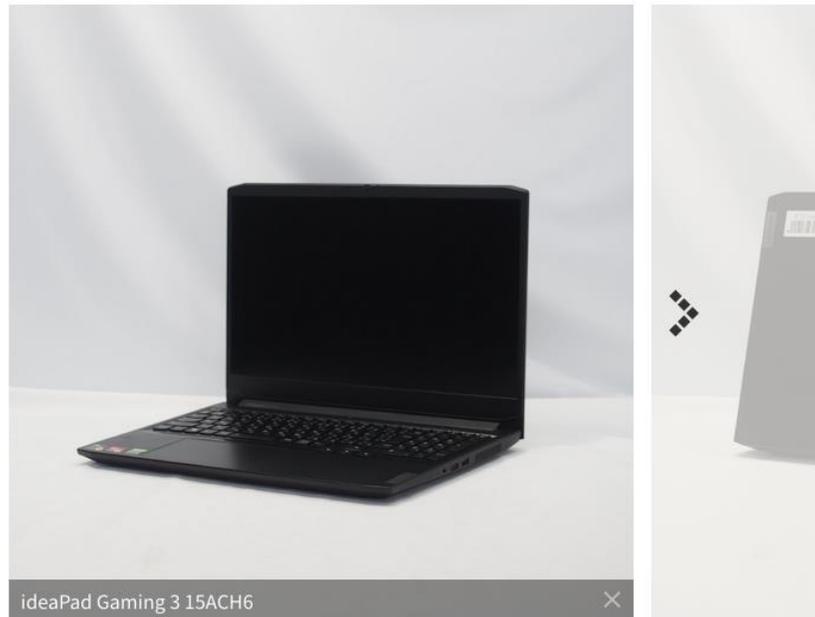
ディスプレイ（画面）サイズ	15.5インチ
付属ディスプレイ仕様	1920x1080
搭載OS	Windows 10 Home 64bit
CPU	Core i7 9750H
CPUクロック	2.6GHz
標準メモリ容量	24GB
内蔵ストレージ容量	512GB(SSD)
ドライブ	-
グラフィックコントローラ名称	nVIDIA GeForce GTX 1660 Ti

商品番号：83269566

**¥59,980**

カートに入れる

ただし来年になるとNPUパソコン登場で大幅値下げ？？？



ideaPad Gaming 3 15ACH6

画像クリックで拡大できます



Lenovo ideaPad Gaming 3 15ACH6 AMD Ryzen 5 5600H  
3.3GHz/16GB/SSD512GB/15インチ/OS無/動作未確認【栃木出荷】 [ストア](#)

現在 **59,000円** (税込 64,900円)

送料 岐阜県は**1,100円** (税込) (離島を除く)

配送方法：佐川急便（一部地域は日本郵便）[配送方法一覧](#)  
※ 条件により送料が異なる場合があります

👁️ 22件 🕒 終了 [詳細](#) 7/15 (月) 21:58 終了予定

🌟 やや傷や汚れあり [質問する](#)

出品者 [ストア](#)



ショップ・インバース [+フォロー](#)

★★★★★ 99.7% 総合評価：87,156

発送元の地域：栃木県 那須烏山市

注目の商品

本日終了



Panasonic デジタル4Kビデオカメラ HC...

**49,500円**



Lenovo Think Pad X13 Core i5-10210U 1...

**16,500円**



Lenovo Think Pad X13 Core i5-10210U 1...

**16,500円**



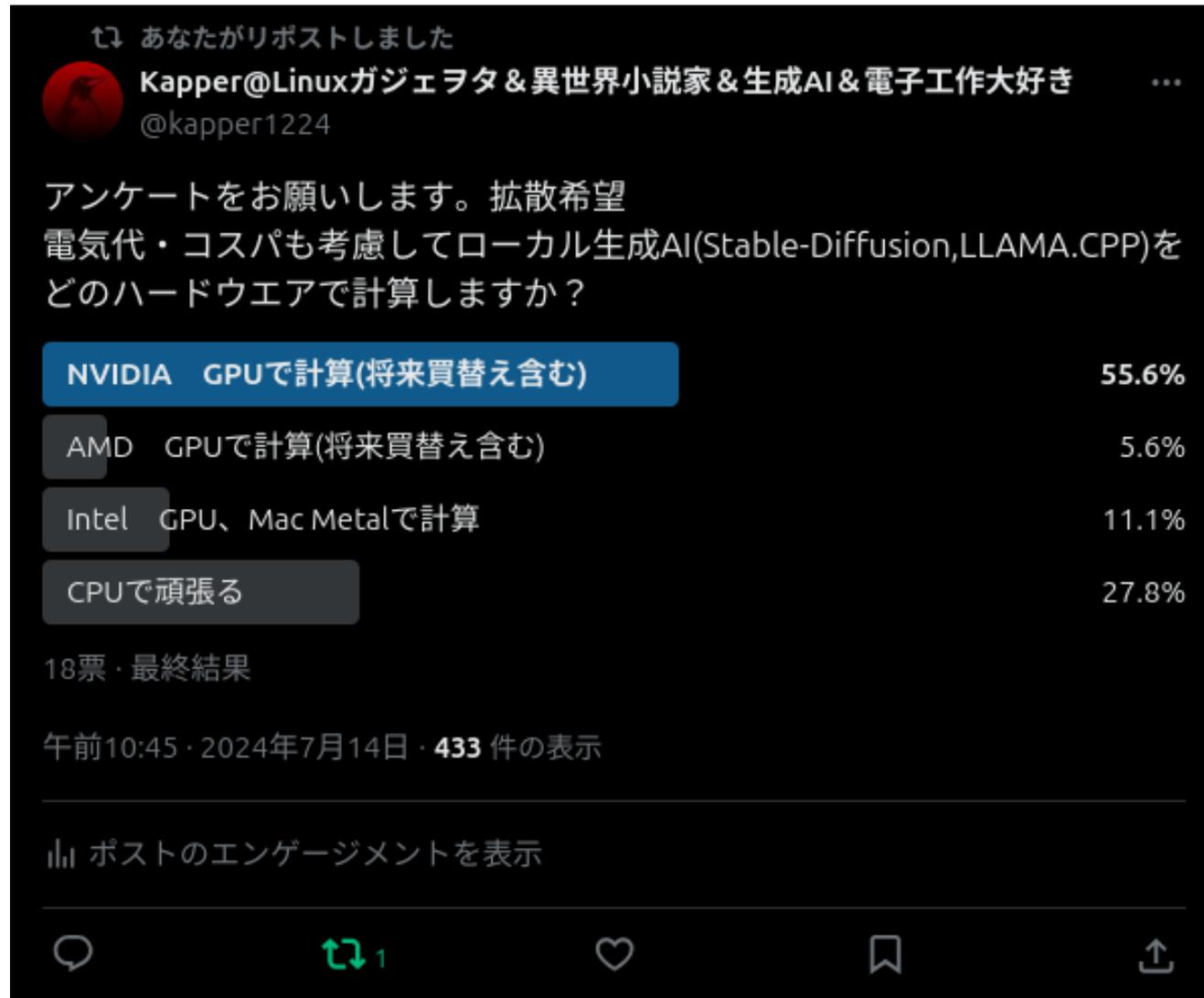
Apple Mac mini 2018 MRTR 2J/A Core i3...

**26,000円**

Apple Mac mini 2018 MRTR 7th Gen i-Fi

**22,000円**

# アンケート結果ご協力ありがとうございました



# 人気あるStable Diffusionモデルデータ

- 主にckpt,safetensorsファイルをCivitaiやHuggingfaceなどからダウンロードしてmodels/Stable-diffusion/のフォルダに保存して使用

## 高画質系

- ベース基本 :SD1.5系、SD XL系
- 風景 :A Regular Deep Background、architecture\_Urban\_SDlife\_Chiasedamme他
- アニメ系最強クラス:Anything-XL、AniagineXL、Anime Pastel Dream
- 実写+アニメ:Pony Diffusion
- 実写日本人:fuduki\_mix、ChilloutMix

## 高速化系

- 実写幻想風高速LCM 4~8 step : LCM\_Dreamshaper
- アニメ系高速化LCM 4~8 step : Anything LCM
- 高速1~8step : SDXL-Lightning、
- 高速1~8step:Hyper-SD、SDXL-Turbo

モデルは数え切れないほどありますので、おすすめサイトをググって色々と試してみてください

# LLMモデルと混迷するオレオレライセンス

- LLMモデルはオープンソース化したMetaのLLaMaを中心。オレオレライセンス注意
- Huggingfaceダウンロード方法：一例 wgetは使えないので注意。間に - を入れる事

curl -LOC - <https://huggingface.co/runwayml/stable-diffusion-v1-5/resolve/main/v1-5-pruned-emaonly.safetensors>

## • Civitaiダウンロード方法：

```
wget "https://civitai.com/api/download/models/#####?type=Model&format=SafeTensor&token=****"
```

##### はモデル番号、 \*\*\*\* は api key

### Curl

```
curl -Lo add-detail-xl.safetensors https://civitai.com/api/download/models/135867?token=\$CIVIT\_TOKEN
```

### Wget

```
wget -O add-detail-xl.safetensors https://civitai.com/api/download/models/135867?t
```

The screenshot shows the Hugging Face website interface. At the top, there is a search bar with the text "Search models, datasets, users...". Below the search bar, there are navigation tabs for "Models", "Datasets", "Spaces", and "Posts". On the left side, there are tabs for "Tasks", "Libraries", "Datasets", "Languages", and "Licenses". The "Tasks" tab is selected, and there is a search bar for "Filter Tasks by name". Below this, there are several task categories: "Multimodal" (Image-Text-to-Text, Visual Question Answering, Document Question Answering) and "Computer Vision" (Depth Estimation, Image Classification, Object Detection, Image Segmentation, Text-to-Image, Image-to-Text). On the right side, there is a list of models. The first model is "mradermacher/Mirai-Nova-Mistral-LocalAI-7B-v0.1-i1-GGUF" with "Updated 1 minute ago". The second model is "thesven/microsoft\_WizardLM-2-7B-GGUF" with "Updated 4 minutes ago". The third model is "acsankar/InsuranceGPT-4bit\_ver0.2-GGUF" with "Updated 4 minutes ago". The fourth model is "RichardErkhov/nonetrix\_-\_pippafeet-11B-0.1-gguf" with "Updated 4 minutes ago". Two blue arrows point to the search bar and the "GGUF" filter.

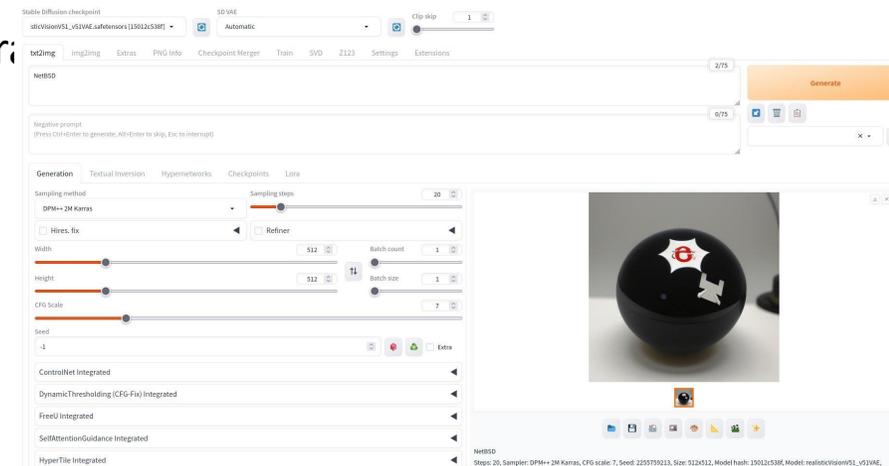
# Stable-Diffusionの使い方

- Windowsの場合。webui-user.batを実行
- Unix系の場合。chmod +x ファイル名で実行形式にして  
chmod +x webui.sh
- ./webui.sh で実行。ブラウザで操作
- モデルを追加する時はmodels/Stable-diffusion/に保存して更新
- LoRaを追加する時はmodels/lora/に保存して更新
- CPUで動かす人はコマンドラインならメモリ節約、高速化

# アプリ紹介

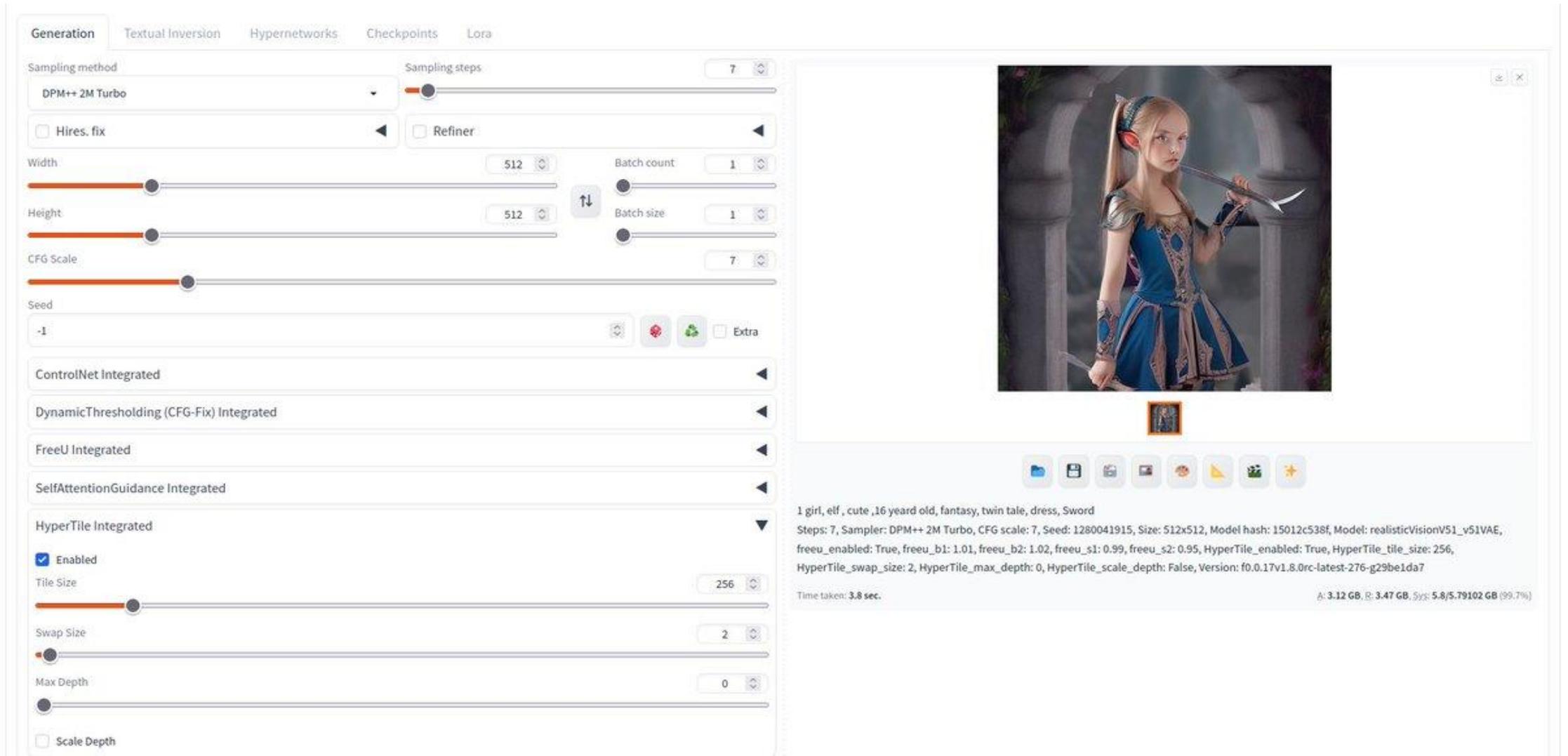
## Stable-Diffusion-webui-forge/reForge

- 現在人気の高いStable-Diffusion改良版
- 少ないGPUのVRAMで生成出来る様にチューニング、高速化
- VRAM 4GB GTX1650~  
VRAM 6GB GTX1660Ti,Super~  
こんな古い世代でも512x512なら十分使える
- インストール方法  
git clone <https://github.com/lllyasviel/stable-diffusion-webui-forge.git>
- cd stable-diffusion-webui-forge
- python -m venv venv
- source venv/bin/activate
- pip3 install --upgrade --force-reinstall torch torchvision torchaudio --extra-index-url <https://download.pytorch.org/whl/cu125>
- deactivate
- ./webui.sh  
注) CUDAを使わない場合は ./webui.sh --skip-torch-cuda-test



# Stable-Diffusion-webui-forge モバイル1660Ti 6GB

512x512 で 3.8 sec /枚。この古い世代のGPUのレベルでも割と速い。



The screenshot displays the 'Generation' tab of the Stable-Diffusion-webui-forge interface. The settings are as follows:

- Sampling method: DPM++ 2M Turbo
- Sampling steps: 7
- Hires. fix:
- Refiner:
- Width: 512
- Height: 512
- Batch count: 1
- Batch size: 1
- CFG Scale: 7
- Seed: -1
- ControlNet Integrated:
- DynamicThresholding (CFG-Fix) Integrated:
- FreeU Integrated:
- SelfAttentionGuidance Integrated:
- HyperTile Integrated:  Enabled
  - Tile Size: 256
  - Swap Size: 2
  - Max Depth: 0
  - Scale Depth:

The generated image shows a girl elf in a blue dress holding a sword. Below the image are icons for file operations and a metadata section:

1 girl, elf, cute, 16 year old, fantasy, twin tale, dress, Sword  
Steps: 7, Sampler: DPM++ 2M Turbo, CFG scale: 7, Seed: 1280041915, Size: 512x512, Model hash: 15012c538f, Model: realisticVisionV51\_v51VAE, freeu\_enabled: True, freeu\_b1: 1.01, freeu\_b2: 1.02, freeu\_s1: 0.99, freeu\_s2: 0.95, HyperTile\_enabled: True, HyperTile\_tile\_size: 256, HyperTile\_swap\_size: 2, HyperTile\_max\_depth: 0, HyperTile\_scale\_depth: False, Version: f0.0.17v1.8.0rc-latest-276-g29be1da7  
Time taken: 3.8 sec. A: 3.12 GB, R: 3.47 GB, S: 5.8/5.79102 GB (99.7%)

# Stable-Diffusion-Webui-Forgeの画面

モデル選択(手動でダウンロードしてmodels/Stable-diffusion/フォルダに保存しておく)

The screenshot shows the Stable-Diffusion-Webui-Forge interface with several key elements highlighted by red arrows and text:

- Model Selection:** A red arrow points to the "Stable Diffusion checkpoint" dropdown menu, which currently shows "isticVisionV51\_v51VAE.safetensors".
- Prompt Input:** A red arrow points to the "txt2img" tab and the "NetBSD" text input field.
- Step Count Selection:** A red arrow points to the "Sampling steps" slider, which is set to 20.
- Size Selection:** A red arrow points to the "Width" and "Height" sliders, both set to 512.
- Generate Button:** A red arrow points to the orange "Generate" button on the right side of the interface.

Additional interface details include:

- Navigation tabs: txt2img, img2img, Extras, PNG Info, Checkpoint Merger, Train, SVD, Z123, Settings, Extensions.
- Generation settings: Sampling method (DPM++ 2M Karras), Hires. fix, Refiner, Width (512), Height (512), Batch count (1), Batch size (1), CFG Scale (7), Seed (-1).
- Integrated features: ControlNet, DynamicThresholding (CFG-Fix), FreeU, SelfAttentionGuidance, HyperTile.
- Output window: Displays a generated image of a black sphere with a white logo and a white arrow pointing to it.
- Footer: "NetBSD Steps: 20, Sampler: DPM++ 2M Karras, CFG scale: 7, Seed: 2255759213, Size: 512x512, Model hash: 15012c538f, Model: realisticVisionV51\_v51VAE,"

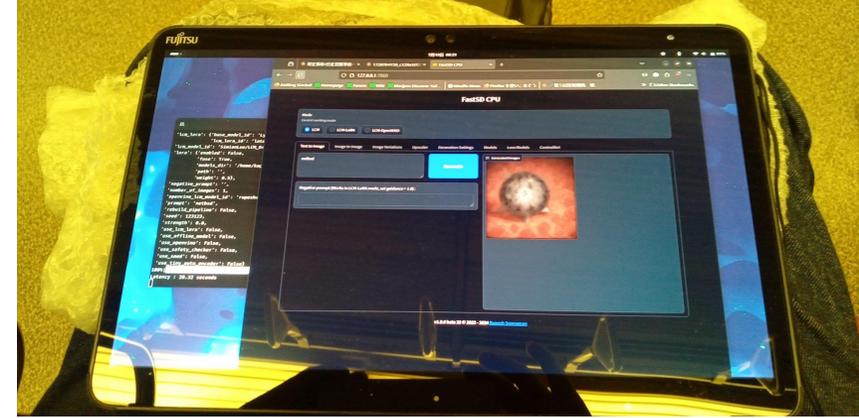
# インストール時のトラブル

- PythonとPipのPathが通っていない(Windows)・・・設定必要
- Pipのプロキシが通っていない(Windows他)・・・必要なら設定必要
- Python3-venv仮想環境が入っていない・・・venvを入れる
- TorchのCUDAエラー・・・CUDAを使わない場合は、./webui.sh --skip-torch-cuda-testオプション
- Torchのバージョンエラー・・・A1111はPytorch3.12をまだサポートしていない。  
Python3.10を使う必要がある。  
sudo add-apt-repository ppa:deadsnakes/ppa
- sudo apt update  
sudo apt install python3.10 python3.10-venv  
python3.10 -m venv "venv"
- webui.sh に python\_cmd="python3.10" を修正

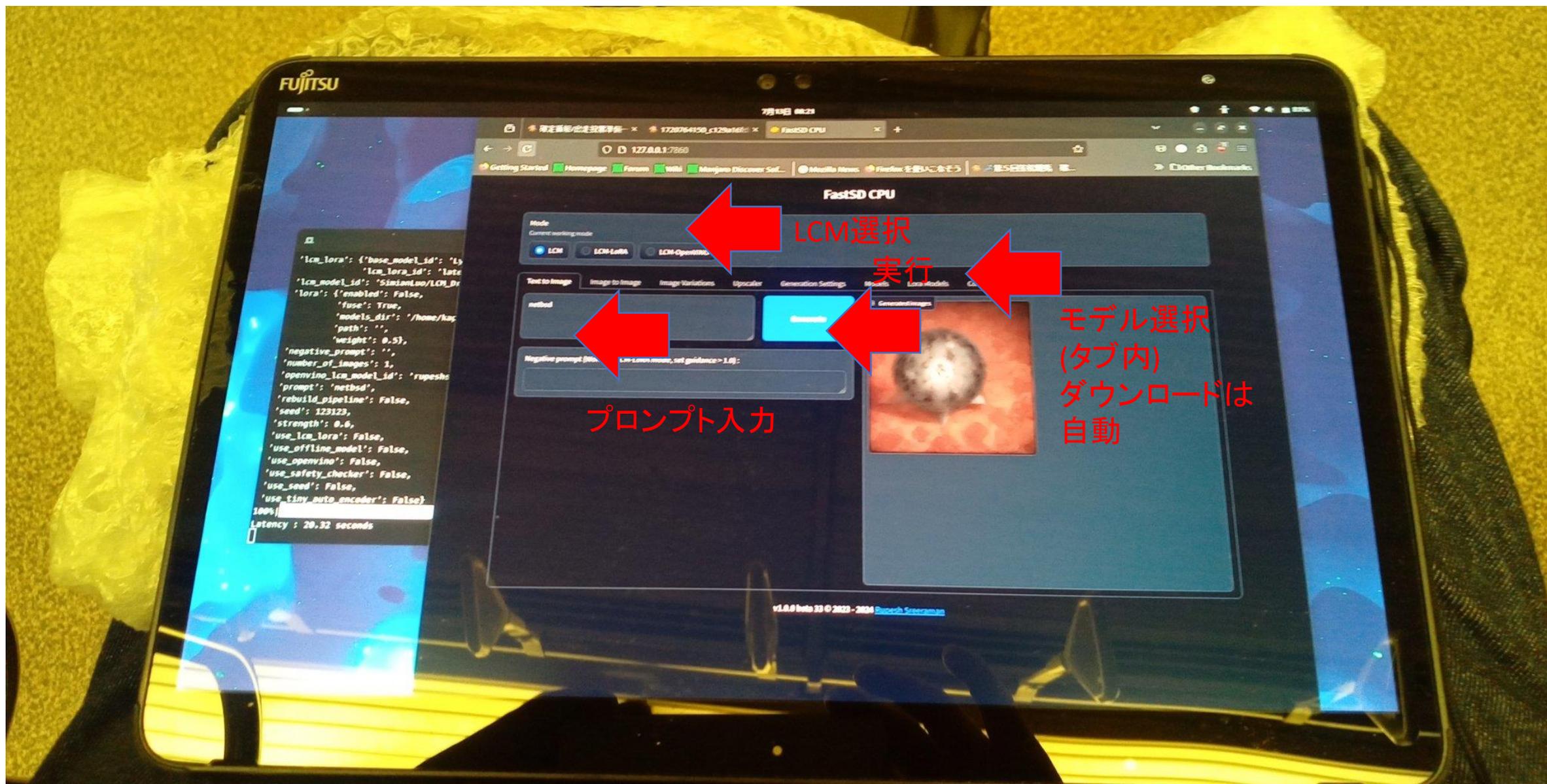
# アプリ紹介

## FastSD CPU

- CPU向けに最適化されたStable-Diffusionモデル自動ダウンロード、初心者向け
- SDXL Lightningを使えば1~4stepで生成可能で4.8秒/枚程度まで高速化可能  
ARMでも40秒/枚前後とかなり高速
- Termuxを使えばAndroidでも動くはず
- インストール  
git clone <https://github.com/ruphes/fastsdcpu.git>  
cd fastsdcpu  
chmod +x install.sh  
./install.sh  
chmod +x start-webui.sh  
./start-webui.sh
- インストールエラーが出たらrequirement.txtのバージョン指定削除  
(作者はWindowsのため環境が古い)



# FastSD CPUの画面



# FastSD CPUはCPUでも結構高速

Comet-Lake Core-i5 10310Uでも16.8 sec/枚

FastSD CPU - Falkon

1月20日 21:32

https://twitter.co... (Windowsノート)... 深層学習サーバに... ページの読み込み... ページの読み込み... 【Stable Diffusio... ページの読み込み... ページの読み込み... Install and Run o... GitHub - rupeshs... FastSD CPU

http://127.0.0.1:7860/ DuckDuckGo

ブックマーク

検索...

ブックマークツールバー  
ブックマークメニュー  
未整理のブックマーク  
じゃんばら【スマホ・P...  
ホーム/X

FastSD CPU

```
kapper@ubuntu: ~/fastsdcpu
```

```
'guidance_scale': 1.0,  
'image_height': 512,  
'image_width': 512,  
'inference_steps': 1,  
'init_image': <PIL.Image.Image image mode=RGB size=600x467 at 0x7F2A2532B700>,  
'lcm_lora': {'base_model_id': 'Lykon/dreamshaper-8',  
            'lcm_lora_id': 'latent-consistency/lcm-lora-sdv1-5'},  
'lcm_model_id': 'stabilityai/sd-turbo',  
'negative_prompt': '',  
'number_of_images': 1,  
'openvino_lcm_model_id': 'rupeshs/sd-turbo-openvino',  
'prompt': 'Miku, Blue Hair',  
'seed': 123123,  
'strength': 0.6,  
'use_lcm_lora': False,  
'use_offline_model': False,  
'use_openvino': False,  
'use_safety_checker': False,  
'use_seed': False,  
'use_tiny_auto_encoder': False}  
Strength: 0.6,2  
100% [Progress bar] | 1/1 [00:04<00:00, 4.62s/it]  
Latency : 19.36 seconds
```

Generated images

v1.0.0 beta 23 © 2023 Rupesh.Sreeraman

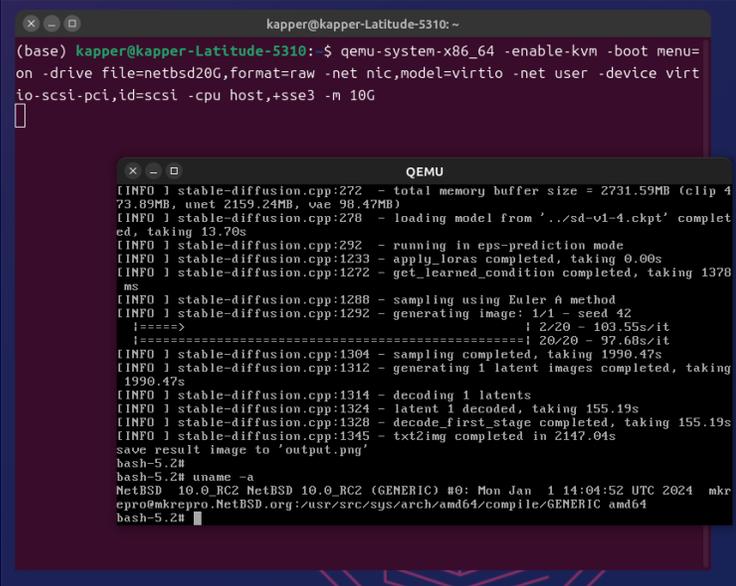
# アプリ紹介

## Stable-Diffusion.cpp

- 生成AIのLlama.cppを参考にC,C++で作られたStable-Diffusion。
- Python,CUDA,Pytorchに依存しないので、色々なOSで動かせる  
\*BSDで動かすなら最適。消費メモリ小さい
- 使用できるモデルは登録されていないと使えない。限定あり
- llama.cpp同様にAMD GPUやIntel GPU、vulkan、CUDA対応
- WebUIはないのでコマンドライン使用。中級者向け
- 初期設定ではSeedは42固定。-s -1などマイナスオプションでランダム動作
- -b オプションでbatch count回数指定するか、zshのRepeat機能で放置プレイ推奨。
- インストール方法  
git clone --recursive https://github.com/leejet/stable-diffusion.cpp
- cd stable-diffusion.cpp  
mkdir build
- cd build
- cmake ..
- cmake --build . --config Release

CPUの場合はFlash Attention オプションを使用して更に省メモリ化

```
cmake .. -DSD_FLASH_ATTN=ON  
cmake --build . --config Release
```



```
kapper@kapper-Latitude-5310: ~  
(base) kapper@kapper-Latitude-5310: ~$ qemu-system-x86_64 -enable-kvm -boot menu=on -drive file=netbsd20G,format=raw -net nic,model=virtio -net user -device virtio-scsi-pci,id=scsi -cpu host,+sse3 -m 10G  
[INFO ] stable-diffusion.cpp:272 - total memory buffer size = 2731.59MB (clip 473.89MB, unet 2159.24MB, vae 98.47MB)  
[INFO ] stable-diffusion.cpp:278 - loading model from './sd-v1-4.ckpt' completed, taking 13.70s  
[INFO ] stable-diffusion.cpp:292 - running in eps-prediction mode  
[INFO ] stable-diffusion.cpp:1233 - apply_loras completed, taking 0.00s  
[INFO ] stable-diffusion.cpp:1272 - get_learned_condition completed, taking 1378ms  
[INFO ] stable-diffusion.cpp:1288 - sampling using Euler A method  
[INFO ] stable-diffusion.cpp:1292 - generating image: 1/1 - seed 42  
|=====| 2/20 - 103.55s/it  
[INFO ] stable-diffusion.cpp:1304 - sampling completed, taking 1990.47s  
[INFO ] stable-diffusion.cpp:1312 - generating 1 latent images completed, taking 1990.47s  
[INFO ] stable-diffusion.cpp:1314 - decoding 1 latents  
[INFO ] stable-diffusion.cpp:1324 - latent 1 decoded, taking 155.19s  
[INFO ] stable-diffusion.cpp:1328 - decode_first_stage completed, taking 155.19s  
[INFO ] stable-diffusion.cpp:1345 - txt2img completed in 2147.04s  
save result image to 'output.png'  
bash-5.2# uname -a  
NetBSD 10.0_RC2 NetBSD 10.0_RC2 (GENERIC) #0: Mon Jan 1 14:04:52 UTC 2024 mkr  
epro@mkrepro.NetBSD.org:/usr/src/sys/arch/amd64/compile/GENERIC amd64  
bash-5.2#
```

# アプリ紹介

## OnnxStream

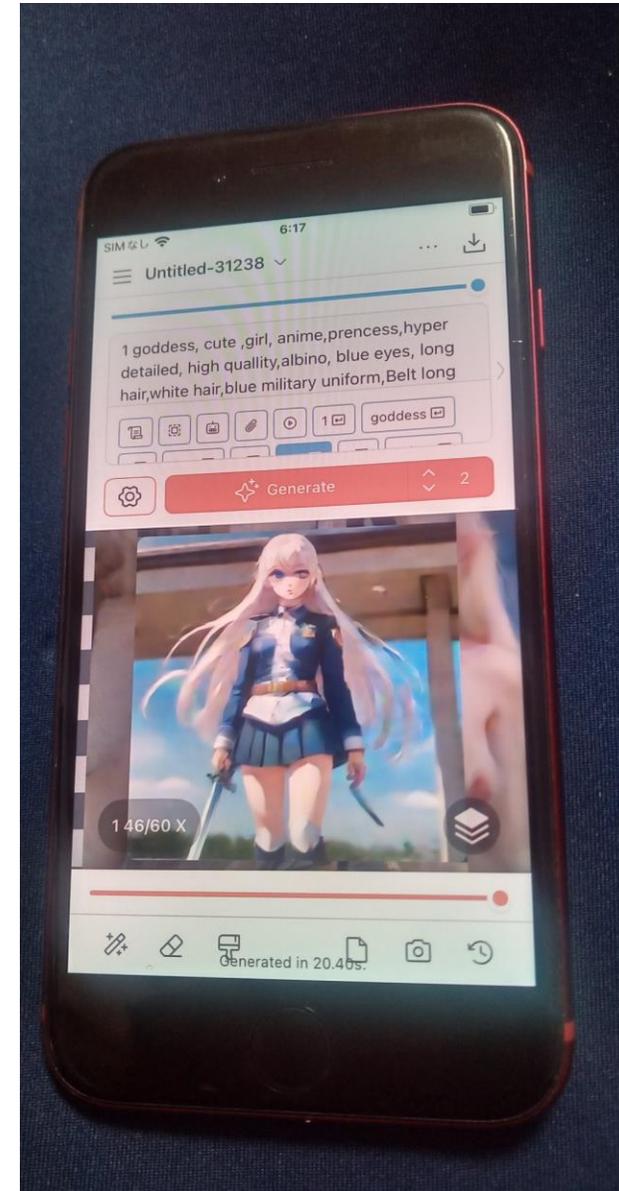
- RaspberryPiで動かすために少ないメモリとC,C++で作られたStable-Diffusion,Local-LLM
- ~512MBのRAMで動かせる様に分割して計算。その代わりに遅い
- WebUIはないのでコマンドライン使用。上級者向け
- インストール方法

```
git clone https://github.com/vitoplantamura/OnnxStream.git
cd OnnxStream
cd src
mkdir build
cd build
cmake -DMAX_SPEED=ON -DOS_LLM=OFF -DOS_CUDA=OFF -
DXNNPACK_DIR=<DIRECTORY_WHERE_XNNPACK_WAS_CLONED> ..
cmake --build . --config Release
```

# アプリ紹介

## Draw Things

- iOS用Stable-Diffusion互換
- IOS 15.4かつA12 bionic以降
- iPhone SE2,3推奨。高速。
- Apple storeよりインストール



# アプリ紹介

## Stability Matrix

- 簡単インストールStable-Diffusionパッケージ
- オールインワンタイプ。容量注意
- 初心者におすすめ。インストールにFuse必要
- インストール方法  
公式サイトからApplImageファイルダウンロードして展開  
sudo apt install fuse  
chmod +x StabilityMatrix.ApplImage  
./StabilityMatrix.ApplImage
- Linuxの場合、フォントが表示されない現象あり注意

# NVIDIA CUDAインストール

- Ubuntu22.04の場合  
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86\_64/cuda-ubuntu2204.pin
- sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin-600
- wget https://developer.download.nvidia.com/compute/cuda/12.0.0/local\_installers/cuda-repo-ubuntu2204-12-0-local\_12.0.0-525.60.13-1\_amd64.deb
- sudo dpkg -i cuda-repo-ubuntu2204-12-0-local\_12.0.0-525.60.13-1\_amd64.deb
- sudo cp /var/cuda-repo-ubuntu2204-12-0-local/cuda-\*-keyring.gpg /usr/share/keyrings/
- sudo apt-get update
- sudo apt-get -y install cuda

```
sudo nano .bashrc
```

テキストエディタで下記を保存

```
export PATH="/usr/local/cuda/bin:$PATH"
```

- export LD\_LIBRARY\_PATH="/usr/local/cuda/lib64:\$LD\_LIBRARY\_PATH"
- <https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html>

# その他生成AI

- 最近生成AI増えたね
- ガジェットハッキングLLM作りたいまニアックなファインチューニングエモいキャラクター生成してw
- ローカルコミュニティ用LLM必要
- 東海道らぐLLM作りませんか？他所に真似できない凄いコアなのを

## The Generative AI Application Landscape v2

A work in progress

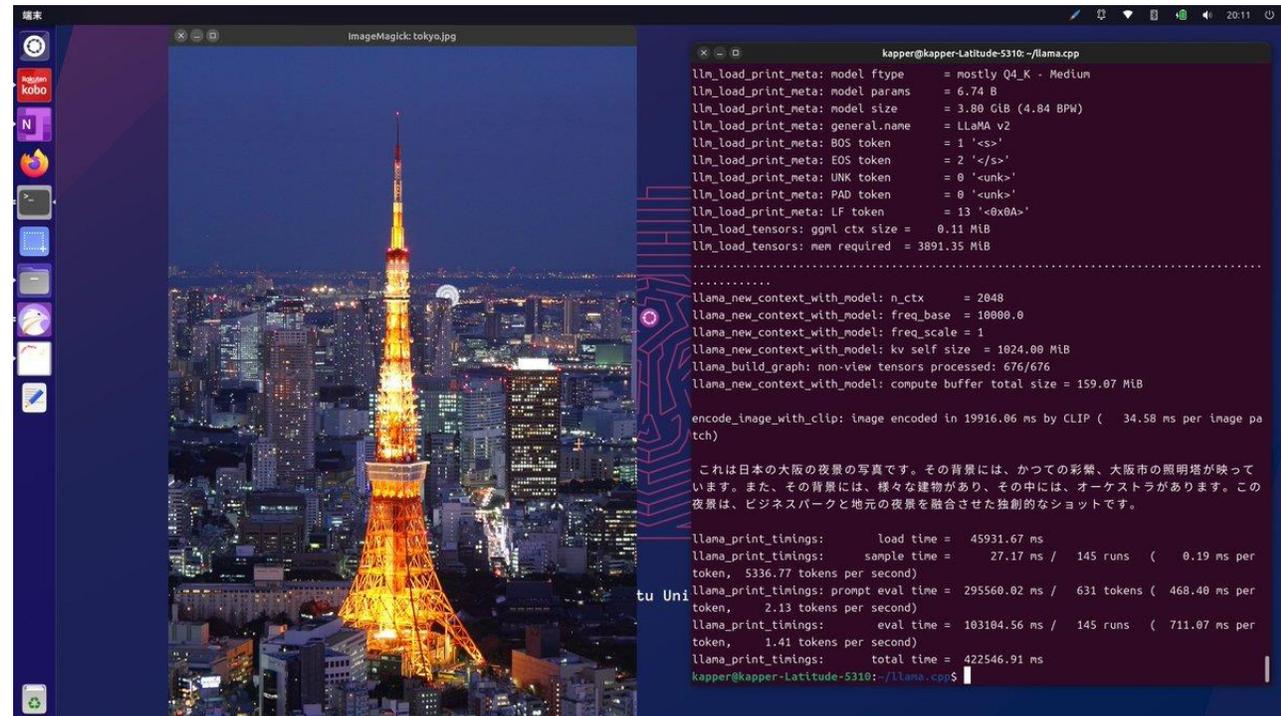


# お仕事で生成AI

- プログラムレスで色々
- 翻訳、メール作成
- 対話型業務をAIチャットで
- 画像、資料を作成
- いつのまにかRPAはどこにいった？  
今後のビジネスの中心
- 賢さ ≠ 語彙力、情報量

# 他にもマルチモーダル化

- 画像を読み取ったり、出力したり
- オープンソースではLLAVAモデルが有名
- 複数のAIを組み合わせてさらに色々な事をさせる事ができる



# 有名キャラクターの模倣(LLMの場合)

- LLMなら色々なモノマネをしてくれます
- ChatGPTくらい賢ければ色々な事が出来ます
- さらに音声合成で出力させるとヲタクの皆様御用達w

ウマ娘のライスシャワーの口調で返事して下さい



「えっと、LinuxをUSBメモリにインストールする方法だね？まず、好きなLinuxのISOイメージをダウンロードしてね。それから、USBメモリをフォーマットして、RufusとかEtcherを使ってブータブルUSBを作成するの。次に、BIOSの設定を変更してUSBからブートできるようにしてから、インストールを進めてね。大丈夫、ライスも応援してるから、がんばってね！」

エヴァンゲリオンの惣流・アスカ・ラングレーの口調で返事して下さい



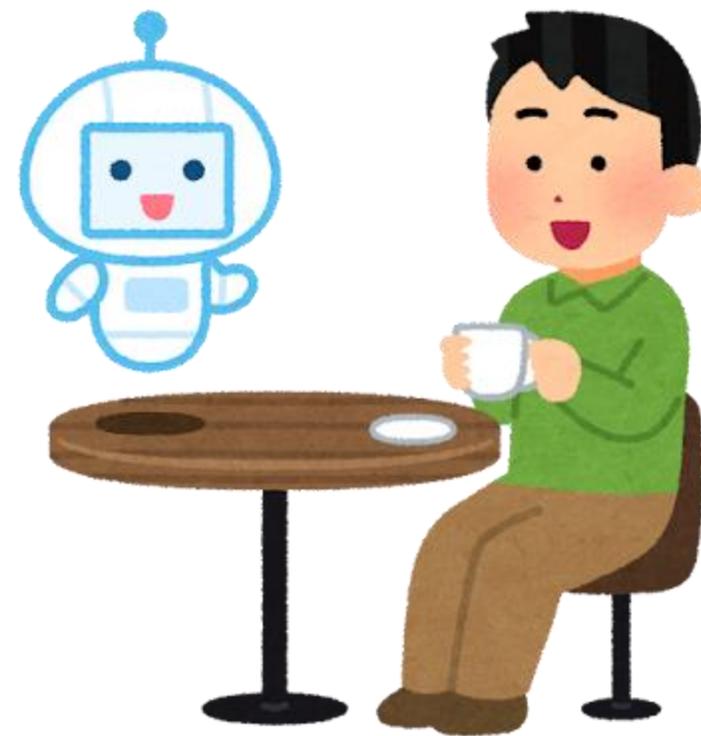
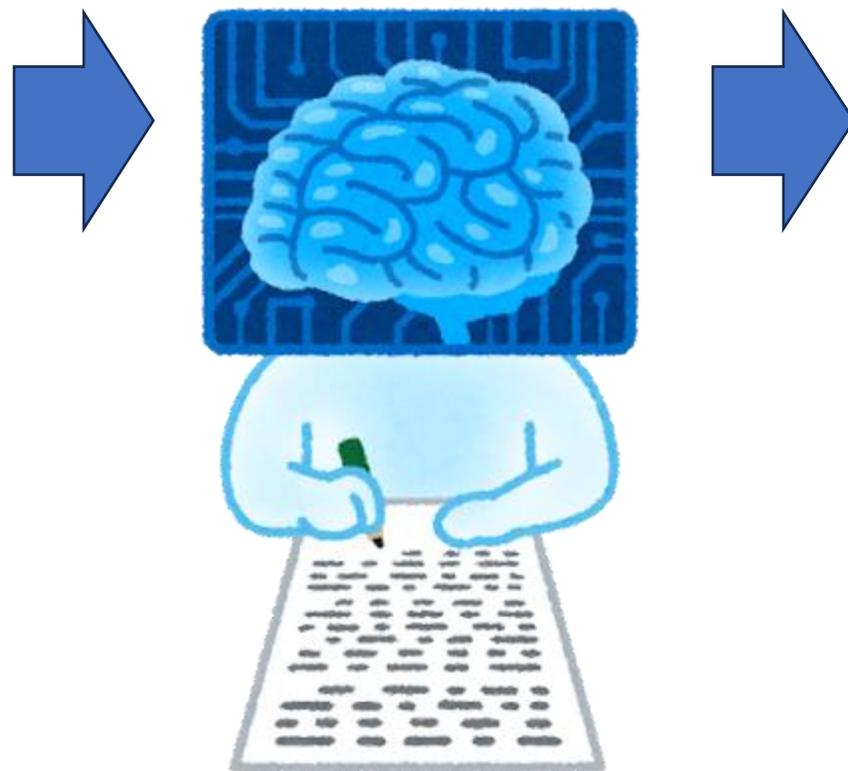
「はいはい、LinuxをUSBメモリにインストールしたいのね？まずは、好きなLinuxのISOイメージをダウンロードしなさいよ。それから、USBメモリをフォーマットして、RufusとかEtcherってツールを使ってブータブルUSBを作成するのよ。次に、BIOSの設定を変更してUSBからブートするようにして、インストールを進めればいいわ。これでうまくいくはずよ、ちゃんとやってみなさい！」

# 生成AIでこんな使い方も



①音声入力  
Whisper.cpp  
Windowsキー+H  
など

②生成AI文章作成  
ChtGPT  
llama.cpp  
など

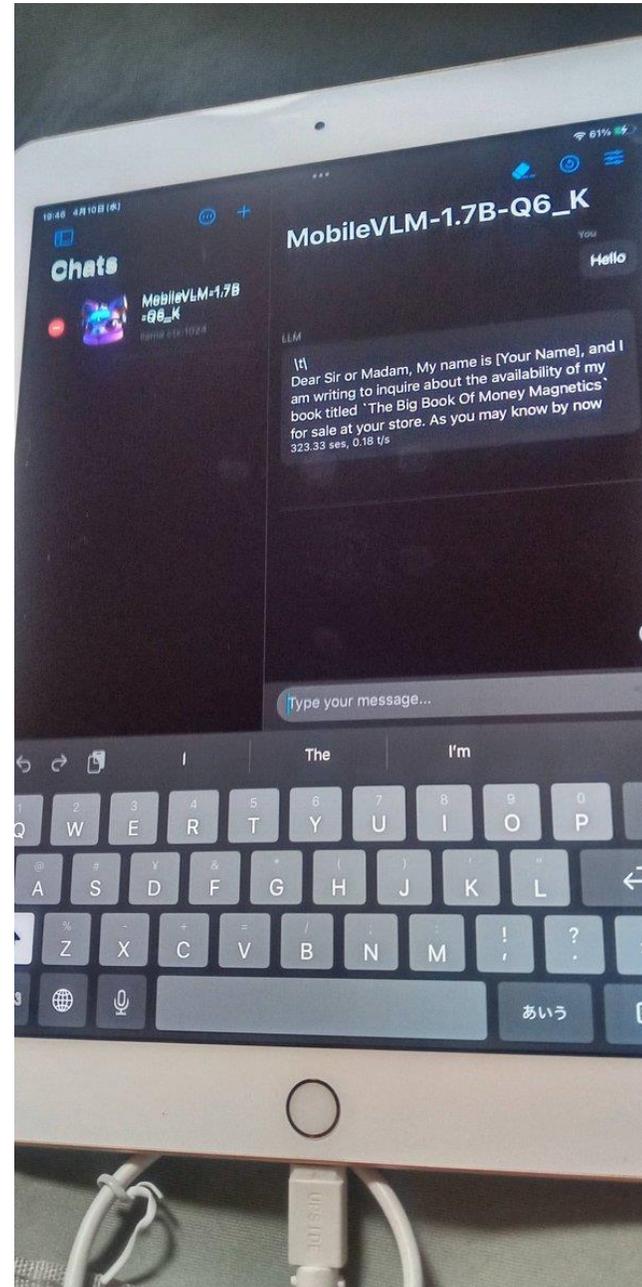


③音声合成  
Voicevoxなど

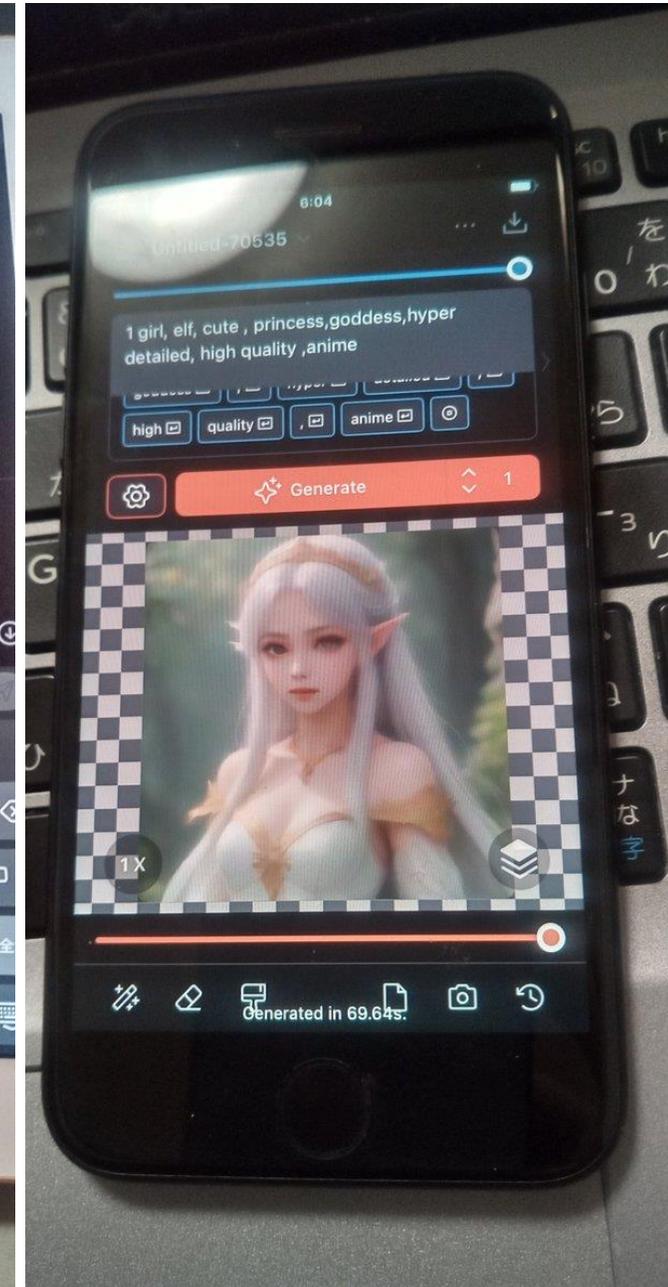
# iPhone,iPadの場合

- 画像生成AIはDraw Things
- LLMはLLMFarmがあります
- 2020年世代以降推奨
- iPhoneSE第2世代が安い

LLMFarm



Draw Things

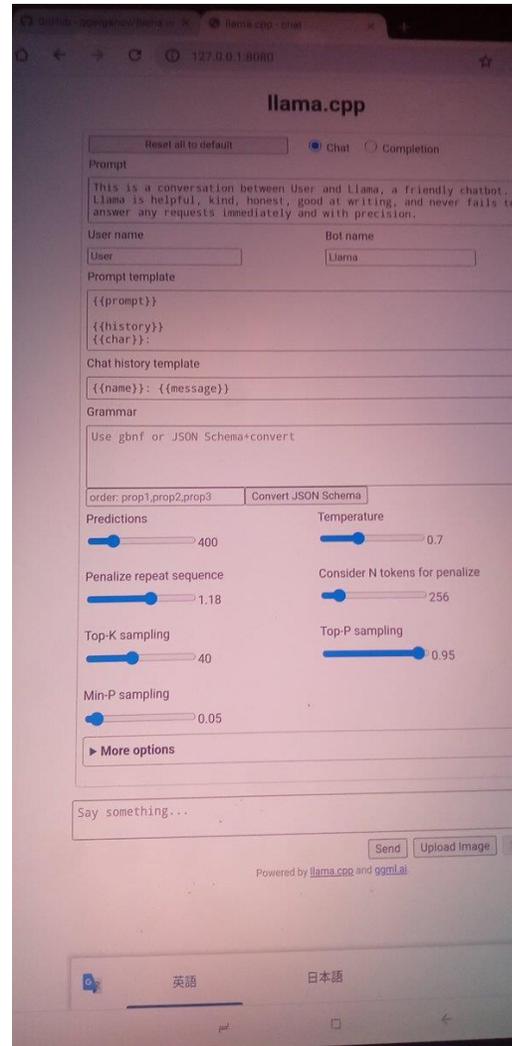


# Termux + Android など

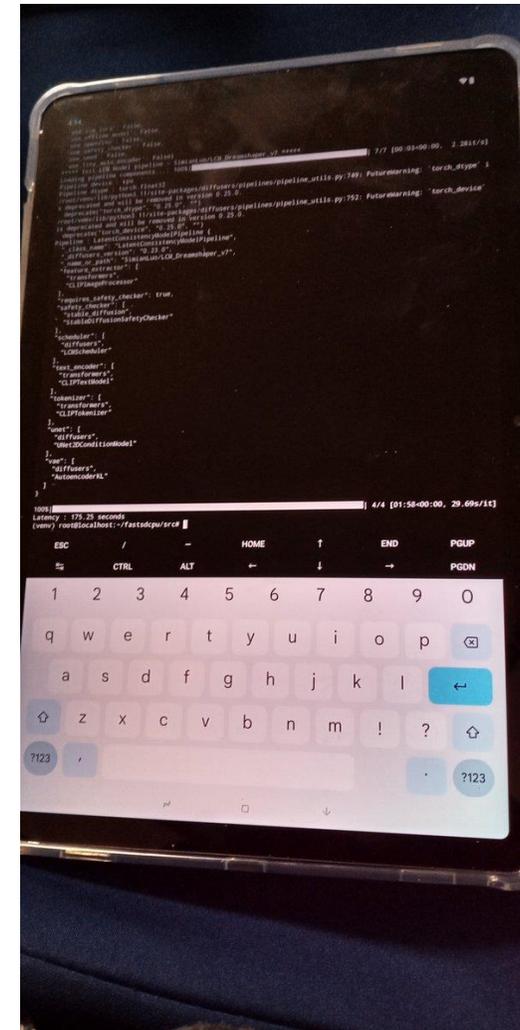
Ollama



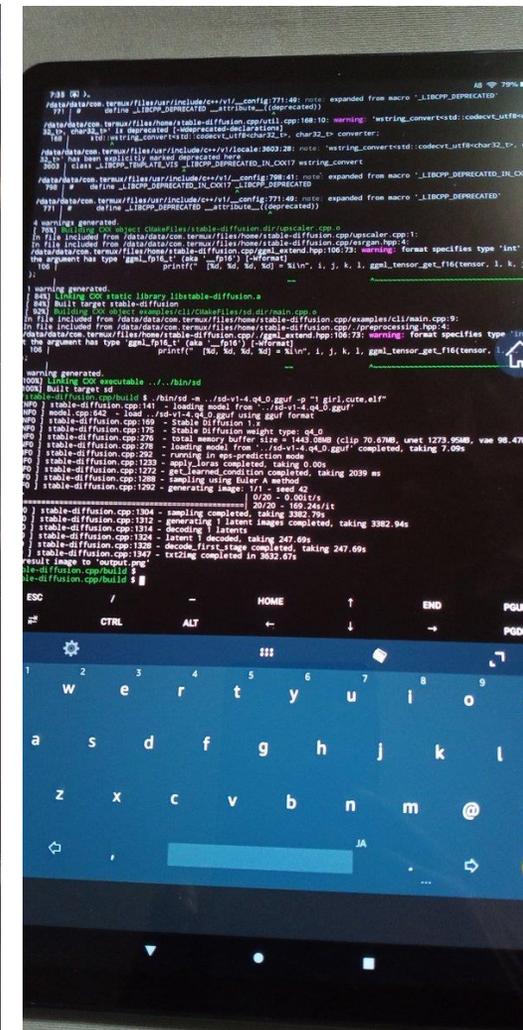
llama.cpp



FastSD cpu



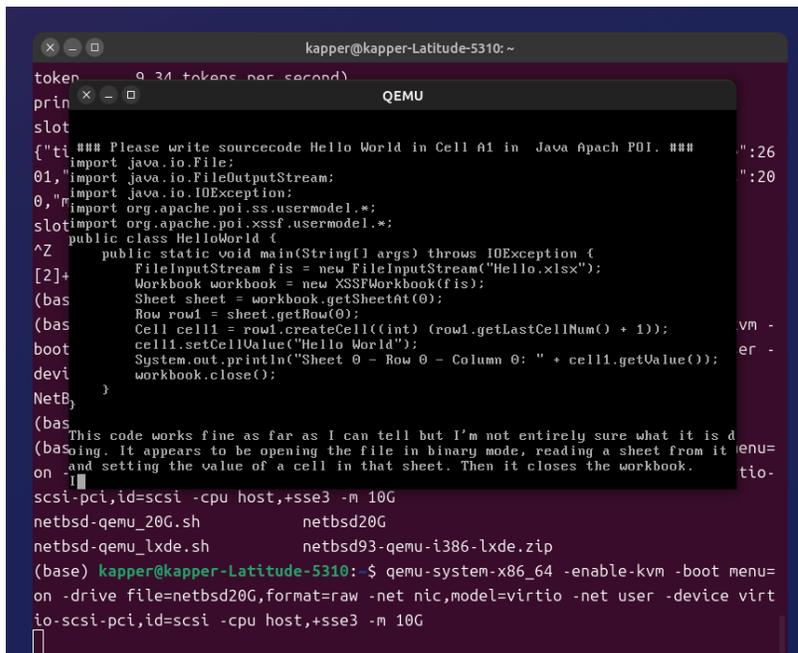
Stable-Diffusion.cpp



# NetBSDの場合(\*BSD含む)

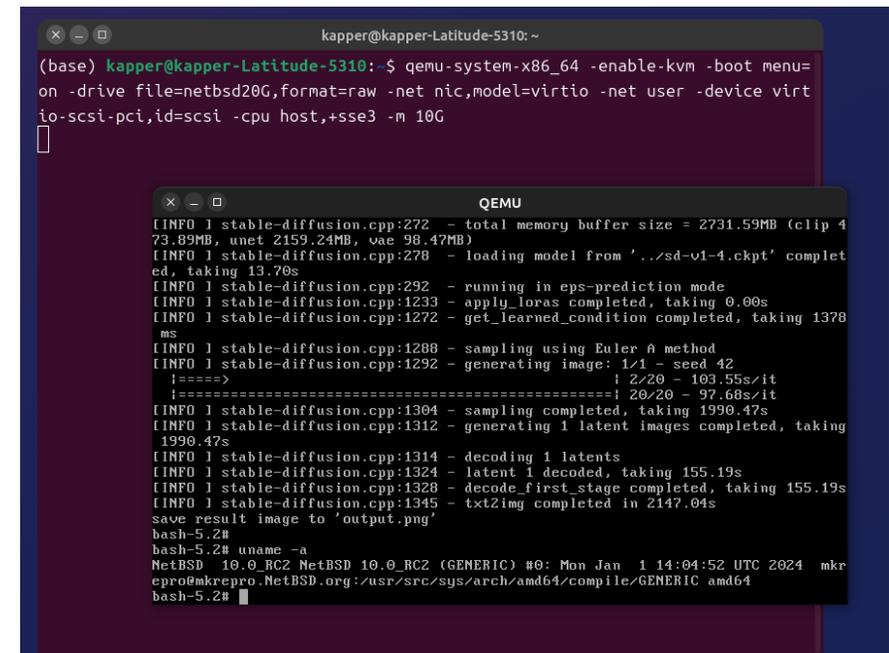
- llamafileがNetBSD + 9.2(amd64)に正式対応(要SSE3:エミュレータ注意)
- llama.cppはcmakeでNetBSD環境でもビルド可能
- Stable-diffusion.cppもcmakeでビルド可能。画像生成AIも完璧。GUIもXもCPU負荷とメモリ負荷になるからスティックに不要。

llama.cpp, llamafile



```
kapper@kapper-Latitude-5310: ~  
token (9.34 tokens per second)  
print  
slot  
{"### Please write sourcecode Hello World in Cell A1 in Java Apache POI. ###":26  
"import java.io.File;":20  
"import java.io.FileOutputStream;":20  
"import java.io.IOException;":20  
"import org.apache.poi.ss.usermodel.*;":20  
"import org.apache.poi.xssf.usermodel.*;":20  
slot  
public class HelloWorld {  
    public static void main(String[] args) throws IOException {  
        FileInputStream fis = new FileInputStream("Hello.xlsx");  
        Workbook workbook = new XSSFWorkbook(fis);  
        Sheet sheet = workbook.getSheetAt(0);  
        Row row1 = sheet.getRow(0);  
        Cell cell1 = row1.createCell((int) (row1.getLastCellNum() + 1));  
        cell1.setCellValue("Hello World");  
        System.out.println("Sheet 0 - Row 0 - Column 0: " + cell1.getValue());  
        workbook.close();  
    }  
}  
NetBSD  
(base) kapper@kapper-Latitude-5310: ~  
This code works fine as far as I can tell but I'm not entirely sure what it is doing. It appears to be opening the file in binary mode, reading a sheet from it and setting the value of a cell in that sheet. Then it closes the workbook.  
io-  
scsi-pci,id=scsi -cpu host,+sse3 -m 10G  
netbsd-qemu_20G.sh netbsd20G  
netbsd-qemu_lxde.sh netbsd93-qemu-i386-lxde.zip  
(base) kapper@kapper-Latitude-5310: ~  
$ qemu-system-x86_64 -enable-kvm -boot menu=  
on -drive file=netbsd20G,format=raw -net nic,model=virtio -net user -device virt  
io-scsi-pci,id=scsi -cpu host,+sse3 -m 10G
```

Stable-diffusion.cpp



```
kapper@kapper-Latitude-5310: ~  
(base) kapper@kapper-Latitude-5310: ~  
$ qemu-system-x86_64 -enable-kvm -boot menu=  
on -drive file=netbsd20G,format=raw -net nic,model=virtio -net user -device virt  
io-scsi-pci,id=scsi -cpu host,+sse3 -m 10G  
QEMU  
[INFO] stable-diffusion.cpp:272 - total memory buffer size = 2731.59MB (clip 4  
73.89MB, unet 2159.24MB, vae 98.47MB)  
[INFO] stable-diffusion.cpp:278 - loading model from './sd-v1-4.ckpt' complet  
ed, taking 13.70s  
[INFO] stable-diffusion.cpp:292 - running in eps-prediction mode  
[INFO] stable-diffusion.cpp:1233 - apply_loras completed, taking 0.00s  
[INFO] stable-diffusion.cpp:1272 - get_learned_condition completed, taking 1378  
ms  
[INFO] stable-diffusion.cpp:1288 - sampling using Euler Å method  
[INFO] stable-diffusion.cpp:1292 - generating image: 1/1 - seed 42  
=====| 2/20 - 103.55s/it  
=====| 20/20 - 97.68s/it  
[INFO] stable-diffusion.cpp:1304 - sampling completed, taking 1990.47s  
[INFO] stable-diffusion.cpp:1312 - generating 1 latent images completed, taking  
1990.47s  
[INFO] stable-diffusion.cpp:1314 - decoding 1 latents  
[INFO] stable-diffusion.cpp:1324 - latent 1 decoded, taking 155.19s  
[INFO] stable-diffusion.cpp:1328 - decode_first_stage completed, taking 155.19s  
[INFO] stable-diffusion.cpp:1345 - txt2img completed in 2147.04s  
save result image to 'output.png'  
bash-5.2#  
bash-5.2# uname -a  
NetBSD 10.0_RC2 NetBSD 10.0_RC2 (GENERIC) #0: Mon Jan 1 14:04:52 UTC 2024 mkr  
epro@mkrepro.NetBSD.org: /usr/src/sys/arch/amd64/compile/GENERIC amd64  
bash-5.2#
```

# NetBSD, 低メモリPCでStable-Diffusion.cppを使う

- 本資料P7で書いた様にgitからビルドしたらモデルデータをどこかにダウンロード  
cd stable-diffusion.cpp/  
curl -LOC - <https://huggingface.co/runwayml/stable-diffusion-v1-5/resolve/main/v1-5-pruned-emaonly.safetensors>
- Stable-diffusion.CPPをコマンドラインで実行  
cd build/  
./bin/sd -m ../v1-5-pruned-emaonly.safetensors --cfg-scale 1 --steps 20 -p "a lovely cat" -o output\_test1.png -v
- 出力されたoutput\_test1.pngを移動して保存。画像ビューアで見る。
- Batch Count指定には -b 10 などオプション追加



# NetBSD, 低メモリPCでStable-Diffusion.cppを使う

- せっかくなのでダウンロードしたモデルを量子化して小さく  
./bin/sd -M **convert** -m ../v1-5-pruned-emaonly.safetensors -o ../v1-5-pruned-emaonly.q8\_0.gguf -v --type q8\_0
- 量子化したGGUFモデルをStable-diffusion.CPPをコマンドラインで実行  
./bin/sd -m ../v1-5-pruned-emaonly.q8\_0.gguf --cfg-scale 1 --steps 20 -p "a lovely cat" -o output\_test2.png -v
- 出力されたoutput\_test1.pngを移動して保存。画像ビューアで見る。

## Memory Requirements

precision	f32	f16	q8_0	q5_0	q5_1	q4_0	q4_1
<b>Memory</b> (txt2img - 512 x 512)	~2.8G	~2.3G	~2.1G	~2.0G	~2.0G	~2.0G	~2.0G
<b>Memory</b> (txt2img - 512 x 512) <i>with Flash Attention</i>	~2.4G	~1.9G	~1.6G	~1.5G	~1.5G	~1.5G	~1.5G



# NetBSD, 低メモリPCでStable-Diffusion.cppを使う

- 高速化としてLCM-LoRaを使う。LCMモデルをlcmフォルダに保存。  
mkdir ../lcm  
curl -LOC - [https://huggingface.co/latent-consistency/lcm-lora-sdv1-5/resolve/main/pytorch\\_lora\\_weights.safetensors](https://huggingface.co/latent-consistency/lcm-lora-sdv1-5/resolve/main/pytorch_lora_weights.safetensors)  
mv pytorch\_lora\_weights.safetensors ../lcm
- 保存したLCMモデルを指定して再びStable-diffusion.cppを実行  
./bin/sd -m ../../v1-5-pruned-emaonly.safetensors --lora-model-dir ../lcm/ --sampling-method lcm --cfg-scale 1 --steps 4 -p "a lovely cat<lora:pytorch\_lora\_weights:1>" -o output\_lcm\_text1.png -v
- 出力されたoutput\_clm\_test1.pngを移動して保存。画像ビューアで見ると。  
一例) 20 step(421 sec/枚) → 4step(91 sec/枚)へ高速化しました。
- LCMを使うと、出力される画像が変わりますw
- Seedをランダムは -s -1 オプション追加

FP32



Q4\_1



# NetBSD,低メモリPCでStable-Diffusion.cppを使う

メモリ使用量を比較。Gnome起動時のメモリを1.4GBから逆算。非常に省メモリ

- ・FP32 :2.7GB使用
- ・Q4\_1:2.0GB使用

Manjaro FP32 モデル

```
top - 01:15:00 up 2 min, 2 users, load average: 3.45, 1.41, 0.54
Tasks: 283 total, 6 running, 277 sleeping, 0 stopped, 0 zombie
%Cpu(s): 64.8 us, 2.6 sy, 0.2 ni, 32.3 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7752.5 total, 430.1 free, 4088.6 used, 3717.1 buff/cache
MiB Swap: 4096.0 total, 4095.2 free, 0.8 used. 3663.9 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2942	kapper	20	0	2905208	2.7g	5760	R	395.7	35.5	4:15.90	sd
1643	kapper	20	0	4968792	282356	128108	R	31.5	3.6	0:10.85	gnome-s+
3042	kapper	20	0	574972	48504	31864	S	21.9	0.6	0:00.66	gnome-c+
3038	kapper	20	0	1094936	50996	39608	S	19.2	0.6	0:00.58	nautilus
3045	kapper	20	0	447280	31520	23552	R	18.5	0.4	0:00.56	seahorse
3044	kapper	20	0	2717568	60452	45360	R	16.6	0.8	0:00.50	gnome-c+
3036	kapper	20	0	509656	30188	23276	S	8.6	0.4	0:00.26	gnome-c+
1476	kapper	20	0	8316	5876	4096	S	3.3	0.1	0:00.64	dbus-da+
3039	kapper	20	0	463080	16512	14848	S	1.7	0.2	0:00.05	gnome-c+
230	root	-51	0	0	0	0	S	1.0	0.0	0:00.63	irq/162+
1922	kapper	20	0	398820	11712	6656	S	1.0	0.1	0:01.00	ibus-da+
2415	kapper	39	19	744568	26280	17852	S	1.0	0.3	0:00.29	tracker+
78	root	-51	0	0	0	0	S	0.7	0.0	0:02.26	irq/9-a+
2489	kapper	20	0	3159340	66692	51188	S	0.7	0.8	0:00.90	gjs
1	root	20	0	23500	14456	9848	S	0.3	0.2	0:02.23	systemd
36	root	20	0	0	0	0	S	0.3	0.0	0:00.12	ksoftir+
99	root	20	0	0	0	0	I	0.3	0.0	0:00.13	kworker+

Manjaro Q4\_1 量子化モデル

```
top - 01:17:52 up 5 min, 2 users, load average: 3.87, 2.28, 1.02
Tasks: 282 total, 2 running, 280 sleeping, 0 stopped, 0 zombie
%Cpu(s): 65.5 us, 3.8 sy, 0.1 ni, 30.3 id, 0.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7752.5 total, 179.2 free, 3352.7 used, 4765.9 buff/cache
MiB Swap: 4096.0 total, 4094.2 free, 1.8 used. 4399.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3243	kapper	20	0	2023836	1.9g	5760	R	397.7	24.4	4:13.66	sd
3315	kapper	20	0	4710796	145400	97012	S	25.6	1.8	0:00.77	gnome-c+
1643	kapper	20	0	5007260	308972	128252	S	22.6	3.9	0:21.20	gnome-s+
3319	kapper	20	0	574960	48800	32448	S	18.9	0.6	0:00.57	gnome-c+
3311	kapper	20	0	874312	51628	40340	S	17.3	0.7	0:00.52	nautilus
3309	kapper	20	0	509524	30440	23528	S	7.6	0.4	0:00.23	gnome-c+
3307	root	20	0	16220	6912	6144	S	4.0	0.1	0:00.12	systemd+
1476	kapper	20	0	8708	6240	4096	S	3.3	0.1	0:00.76	dbus-da+
90	root	20	0	0	0	0	S	1.0	0.0	0:00.37	kswapd0
2415	kapper	39	19	744568	29580	18108	S	1.0	0.4	0:00.40	tracker+
3314	kapper	20	0	536820	17024	15104	S	1.0	0.2	0:00.03	gnome-c+
1	root	20	0	23500	14584	9848	S	0.7	0.2	0:02.32	systemd
78	root	-51	0	0	0	0	S	0.7	0.0	0:03.43	irq/9-a+
230	root	-51	0	0	0	0	S	0.7	0.0	0:01.59	irq/162+
339	root	19	-1	58268	25216	24192	S	0.7	0.3	0:00.46	systemd+
1097	root	20	0	2286760	32016	20224	S	0.7	0.4	0:00.76	snappd
1922	kapper	20	0	398820	11712	6656	S	0.7	0.1	0:02.76	ibus-da+

# FreeBSDの場合

PytorchとCondaの壁をなんとかする方法が書かれています

☐ README

This tutorial was written on FreeBSD 13.1.

## Table of Contents

- [Verify and Setup CUDA](#)
- [Initial setup](#)
- [Conda](#)
- [PyTorch](#)
- [Stable Diffusion WebUI](#)
- [Submitting requests or changes](#)

## Notes

For now this only covers those with an [NVIDIA](#) card with CUDA support.

You must have the [Linuxulator](#) installed for any of this to work. It has only been tested with the default CentOS from ports.

This does not use the official [Stable Diffusion](#) repository but the WebUI written by and located at [AUTOMATIC1111](#).

If you wish to install the official repository none of the contents in this documentation should change other than checking out the official repository.

## Verify and setup CUDA

This involves two steps the first is to install `nv-sglrun` in order to check for CUDA support which only works for FreeBSD binaries. The second step is to build `uvm_ioctl_override.c` to have the same work for Linux binaries.

# Stable-Diffusionを練習してみよう

- ①モデルを選択
- ②text-imageに適切なプロンプト(キーワード)「NetBSD」を入力してGenerateボタンを押して実行
- ③しばし待つと画像が生成される
- ④プロンプト(キーワード)を変更して色々と試してみる

# Stable-Diffusionのプロンプト(キーワード)

- プロンプトは非常に多い。専用サイトを見ると良い  
高画質化のプロンプトはほぼ必須  
一人ポーズ : 1 girl, 1 man  
髪型 : long hair, short hair, gold hair, silver hair  
年齢 : 18 years old, など  
背景 : clear sky, midnight, nature, river, cave, inside, outside, bed room  
指 : beautiful fingers, beautiful hands  
高画質 : masterpiece, hyper detailed, high quality
- ネガティブプロンプト  
指 : extra hands, extra fingers, bad hands, bad fingers, extra arms,

# Stable-diffusionの高画質化

- Step数を増やしてみる
- cfg-scaleを調整7～20へ
- VAEを入れ替えてみる
- モデルを変更
- Samplerを変更してみる
- 高画質プロンプトを入れる masterpiece, hyper detailed, high quality

# Stable-Diffusionの高速化

- GPUを使う
- LCM-LoRaモデルなどでstep数を減らす
- Tiny Encoderを使う
- サンプラーをLCMなどする
- CPU計算ならOpenVINOを使う
- モバイルPCなら電源コンセント繋いでターボブーストON推奨

# Stable Diffusionで練習

LCM-LoRA例

お題:「猫」を描いてみる

プロンプト "a lovely cat"



# Stable Diffusionで練習

お題: 宇宙で馬に乗る人

プロンプト "a photograph of an astronaut riding a horse"



# Stable Diffusionで練習

お題: スターウォーズ 複数のプロンプト

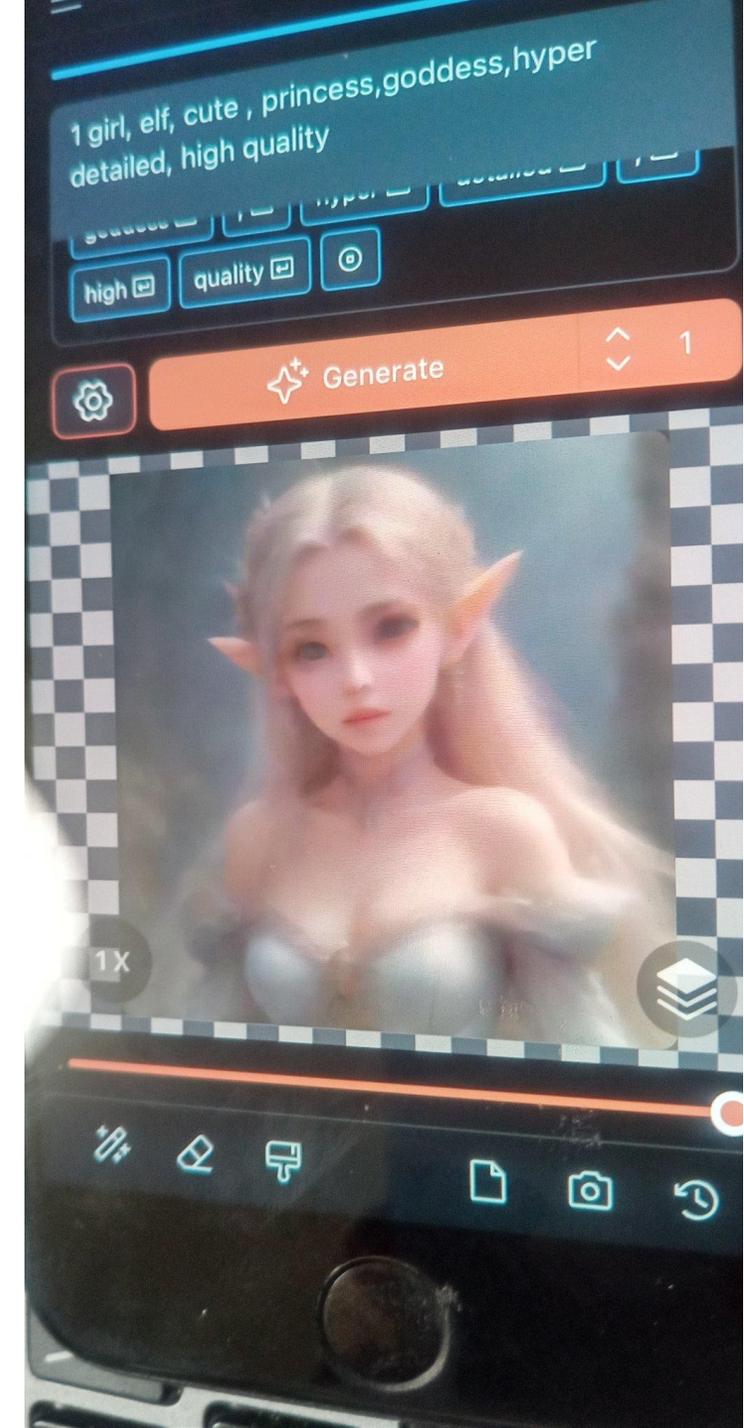
プロンプト: starwars, spacecraft



# Stable Diffusionで練習

お題: エルフの女性、高画質化  
プロンプト

"1 girl, elf, cute, princess, **goddess, hyper detailed, high quality**"



# Stable Diffusionで練習

お題: ChatGPTにプロンプトを作成させる  
4コマ漫画風を作成  
CharGPT任せで色々

①



②

②



③



④



起: ドラえもんがゴルゴ31に町中でばったり出会う。 承: お互いに戦闘モードになり構える。 転: のびたが現れてゴルゴ31が驚く。 結: のびたがゴルゴ31を銃撃して撃退する



## ストーリー

起: ドラえもんが町中でゴルゴ31にばったり出会う。

承: 二人はお互いに戦闘モードになり構える。

転: のび太が現れてゴルゴ31が驚く。

結: のび太がゴルゴ31を銃撃して撃退する。

## プロンプト

### 1. 起 - ドラえもん和ゴルゴ31の出会い

- プロンプト: "Doraemon is strolling through a bustling town when he suddenly bumps into Golgo 31. Both look at each other with surprise. [image of Doraemon and Golgo 31 standing face to face, with a crowded town street in the background]"

### 2. 承 - 戦闘モード

- プロンプト: "Doraemon and Golgo 31 quickly shift into combat mode, ready to face off. Both adopt defensive stances, eyes locked in a tense stare. [image of Doraemon with a serious expression, gadgets ready, and Golgo 31 aiming his weapon, both ready to fight]"

### 3. 転 - のび太の登場

- プロンプト: "Just as the tension peaks, Nobita suddenly appears on the scene, catching Golgo 31 off guard. Golgo 31 looks shocked and distracted by Nobita's unexpected arrival. [image of Nobita running towards Doraemon and Golgo 31, Golgo 31 with a surprised expression]"

### 4. 結 - のび太の撃退

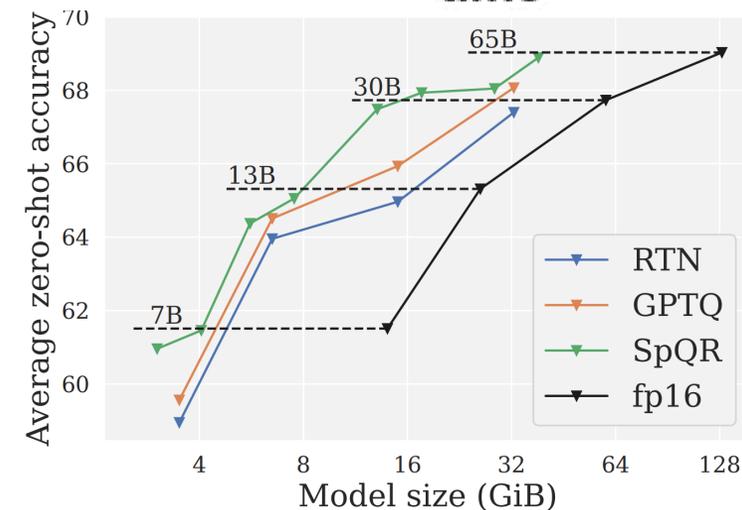
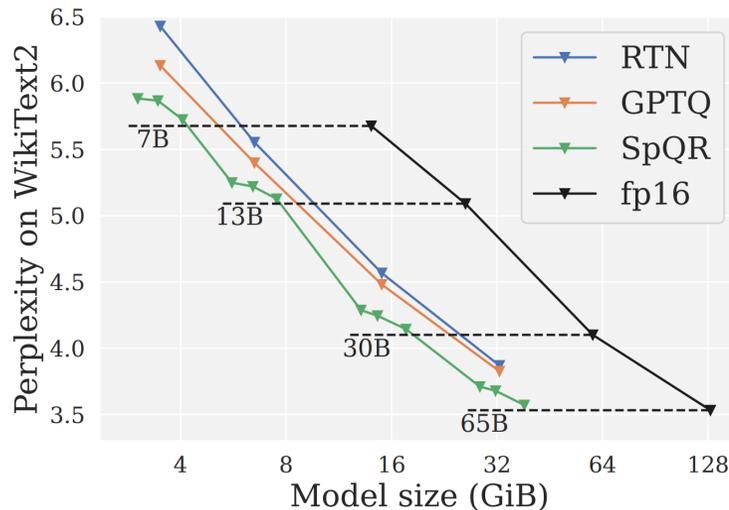
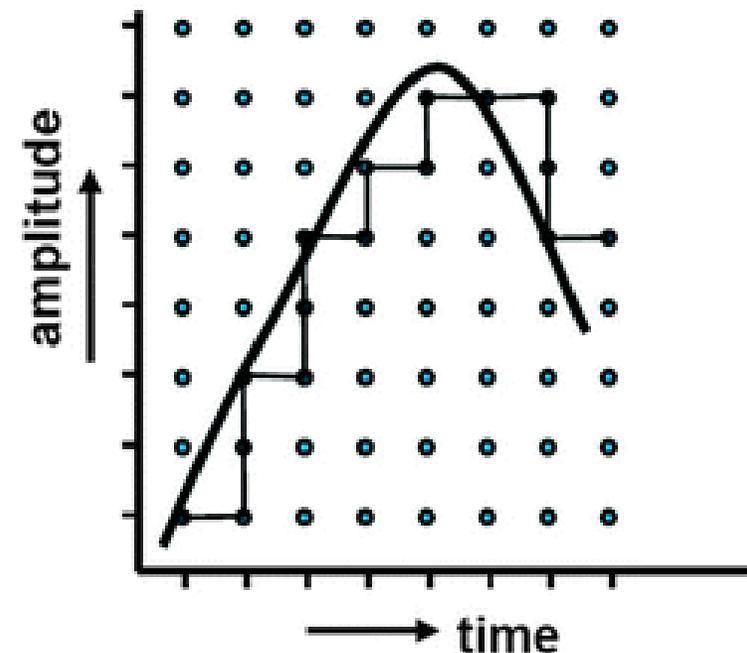
- プロンプト: "Seizing the opportunity, Nobita pulls out a toy gun and shoots at Golgo 31, managing to fend him off. Doraemon and Nobita celebrate their victory. [image of Nobita holding a toy gun, Golgo 31 retreating, and Doraemon giving Nobita a high-five]"



このプロンプトを使って、4コマ漫画を生成してみてください。楽しいストーリーが出来上がるこ

# 量子化

- 確率分布の小数点桁数を削って計算
- メモリと負荷低減。数字程精度が落ちない
- 画像生成AIは16bit、LLMは3~16bit
- モデルサイズ1GBを切ると回答が難しくなる
- BitNet 1.58bitが出たがまだ普及していない
- 画像生成AIはStable-Diffusion.cppのみ対応



# ファインチューニング・LoRa

- モデルに追加学習。入れたい不足する画像情報を追加
- メモリ不足でまだ試していないけどツールも使える
- 要求スペックが高いためGPUメモリを十分に確保かGoogle Colabを借りる

## TrainTrain

- This is an extension for [AUTOMATIC1111's stable-diffusion-webui](#).
- You can create LoRA, iLECO, and differential LoRA.

lang English 言語 日本語 Support

## Overview

This is a tool for training LoRA for Stable Diffusion. It operates as an extension of the Stable Diffusion Web-UI and does not require setting up a training environment. It accelerates the training of regular LoRA, iLECO (instant-LECO), which speeds up the learning of LECO (removing or emphasizing a model's concept), and differential learning that creates slider LoRA from two differential images.

## Table of Contents

- [Requirements](#)
- [Installation](#)
- [Usage](#)
  - [LoRA](#)
  - [iLECO](#)
  - [Difference](#)
- [Settings](#)
  - [Mandatory Parameters](#)
  - [Optional Parameters](#)
- [Queue \(Reserving Training\)](#)
- [Plot](#)
- [Acknowledgments & References](#)

README GPL-3.0 license

## LoRA\_Easy\_Training\_Scripts

A set of training scripts written in python for use in Kohya's [SD-Scripts](#). It has a UI written in pyside6 to help streamline the process of training models.

Old scripts can be found [here](#)

## Table of contents

- [Installation](#)
  - [Windows](#)
  - [Linux](#)
- [Usage](#)
- [Configuration](#)
- [Changelog](#)

## Installation

**You Must have python 3.10 installed, and it Must be the running version of python**

### Windows

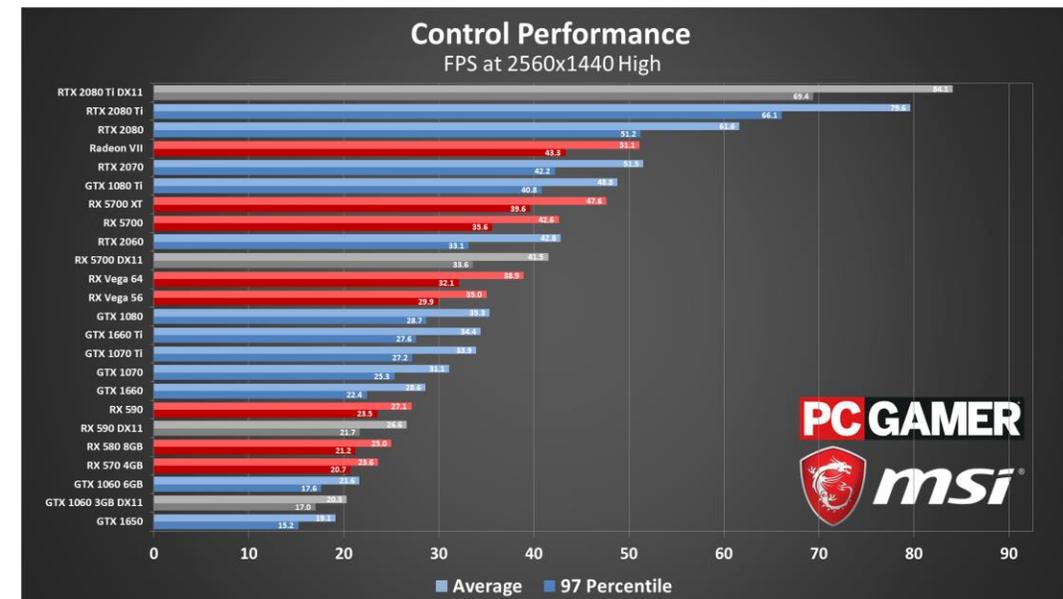
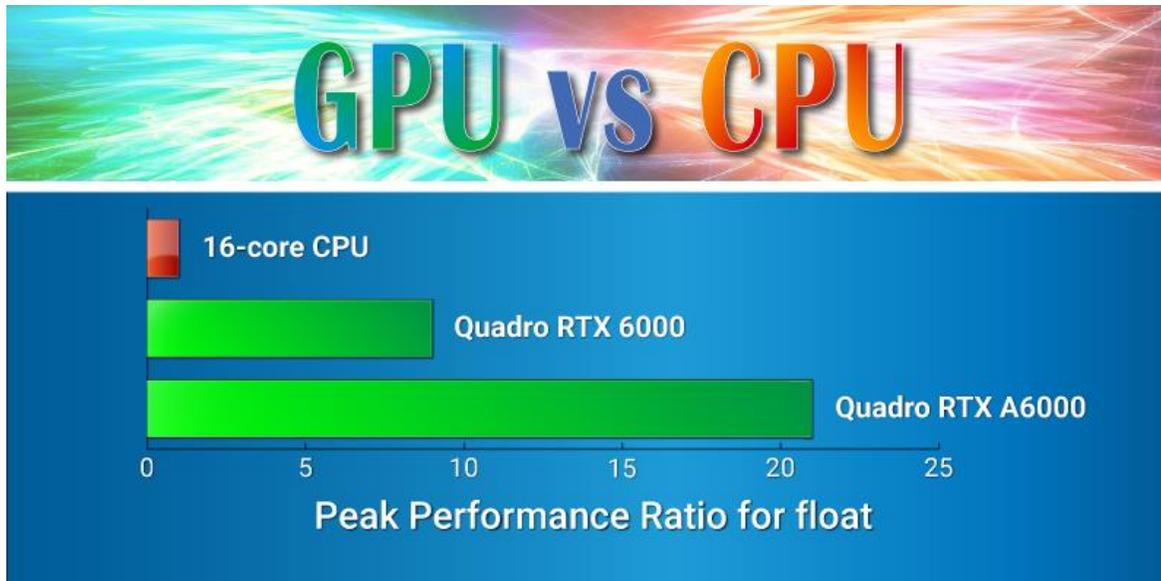
If you are on windows all you need to do to install the scripts is follow these commands. Open up a command line within the folder that you want to install to then type these one line at a time

```
git clone https://github.com/derrian-distro/LoRA_Easy_Training_Scripts
cd LoRA_Easy_Training_Scripts
install.bat
```

after that, it will begin installing, asking a few questions along the way. Just make sure to answer them.

# CPU VS GPU

- 現在高速化としてGPUを使うのが主流
- CPU VS GPUの速度は1:20~100くらいの速度差
- WebサーバとブラウザはCPU、計算はGPUで分担している
- CPUで計算するとWebサーバとブラウザの分だけ負担が増える  
OllamaなどCUIで使うのも良い(xも不要)
- AI学習は完全にGPU優位。GPUはメモリ確保が大変高価



# GPUクラウドを格安でレンタルしたい

- 実験やAIモデル学習時にはできればGPUサーバをレンタルしたい
- Vast.AI: 個人サーバから間借りレンタル。格安
- NovelAI: 数少ない限定的月額式。小説は無料、画像は有料
- Google Colab
- Runpod: サーバレスモード
- Replicate: モデルが設置済
- Hostkey: 色々なGPU

#GPUs:	ANY	0X	1X	2X	4X	8X	8X+	On-Demand	Any GPU	Planet Earth	Price (inc.)
m:19354	host:52271	Taiwan, TW	X10DRG	↑241 Mbps ↓247 Mbps	10 ports	verified Max Duration 7 mon.	\$0.098/hr				
	1x RTX 3070	19.8 TFLOPS Max CUDA: 12.2	8 GB 377.3 GB/s	Xeon® E5-2686 v4 9.0/72 cpu 32/258 GB	INTEL 742 MB/s 287.1 GB	16.2 DLPerf 164.5 DLP/\$/hr	Reliability 96.6%	RENT			
m:12468	host:80527	United Kingdom, GB	X470 GAMING P...	↑94 Mbps ↓681 Mbps	50 ports	verified Max Duration 1 mon, 9d	\$0.103/hr				
	1x RTX 2080 Ti	14.6 TFLOPS Max CUDA: 12.2	11 GB 508.8 GB/s	AMD Ryzen 9 39... 12.0/24 cpu 32/64 GB	WD Green SN350... 1288 MB/s 338.0 GB	8.7 DLPerf 83.9 DLP/\$/hr	Reliability 99.49%	RENT			
m:11258	host:53115	British Columbia, CA	X99	↑749 Mbps ↓729 Mbps	8 ports	verified Max Duration 9 days	\$0.105/hr				
	1x RTX A2000	11.2 TFLOPS Max CUDA: 12.2	12 GB 243.0 GB/s	Xeon® E5-2680 v4 3.5/28 cpu 8/64 GB	SPCC Solid State 862 MB/s 471.3 GB	10.9 DLPerf 104.0 DLP/\$/hr	Reliability 98.8%	RENT			
m:7155	host:46971	Indiana, US	X99	↑570 Mbps ↓617 Mbps	50 ports	verified Max Duration 17 days	\$0.108/hr				
	1x GTX 1660 S	3.6 TFLOPS Max CUDA: 12.2	6 GB 272.6 GB/s	Xeon® E5-2698 v3 16.0/32 cpu 64/129 GB	TEAM T253512GB 350 MB/s 315.5 GB	1.7 DLPerf 15.9 DLP/\$/hr	Reliability 99.90%	RENT			
m:18119	host:78246	Poland, PL	PRIME B365	↑577 Mbps ↓433 Mbps	10 ports	verified Max Duration 13 days	\$0.122/hr				
	1x RTX A4000	20.6 TFLOPS Max CUDA: 12.2	16 GB 364.4 GB/s	Pentium® Gold G... 2.0/4 cpu 16/32 GB	nvme 1652 MB/s 357.0 GB	20.4 DLPerf 166.7 DLP/\$/hr	Reliability 99.47%	RENT			

# よくある質問・疑問点

- CPUではだめなんですか？ --- 遅いだけです。その分CPUメモリは安い。
- 自動で放置プレイしたい --- BatchCountを指定。zshのRepeatコマンドでも
- AndroidスマホのGPUは動きますか？ --- GPUメモリが小さくてほぼ使われていない
- NVIDIA以外のGPUは動くのですか？ --- ROCm, DirectML, Metal, OpenVINOでも動く
- なぜIntel Arcは生成AIであまり使われないの？ --- 動作問題なし。モデルをOpenVINOに変換
- Intel NPUでも動きますか？ --- OpenVINOを使えば動くはずだが動作実例なし
- ノートパソコンの内蔵GPUは使えますか？ --- メモリ500MB設定なのできつい
- GPUは借りた方が安いのですか？ --- 学習させるなら時間に応じて
- 指や体が奇形になります --- ネガティブプロンプトを工夫。サイズを縦長へ調整。  
手を隠すとかもあり。
- そもそもローカルで生成AIを使う意味は？ --- ロマン、商売用、独自絵
- モバイル中に計算が遅いんですが --- 節電設定。電源繋いでターボブーストON推奨

# まとめ

- 簡単な生成AIの使い方と説明
- ローカル画像生成AIを実例に各OSで動作
- Windows, Linux, MacOS, \*BSDで生成AI動作確認
- ファインチューニング、LoRaを説明
- 各種Stable Diffusionのインストール、ビルド方法を説明