1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

%cd /content

!git clone -b totoro3 https://github.com/camenduru/ComfyUI /content/TotoroUI

%cd /content/TotoroUI

!pip install -q torchsde einops diffusers accelerate xformers==0.0.28.post2

!apt -y install -qq aria2

!aria2c --console-log-level=error -c -x 16 -s 16 -k 1M https://huggingface.co/camenduru/FLUX.1-dev/resolve/main/flux1-dev-fp8.safetensors -d /content/TotoroUI/models/unet -o flux1-dev-fp8.safetensors

!aria2c --console-log-level=error -c -x 16 -s 16 -k 1M https://huggingface.co/camenduru/FLUX.1-dev/resolve/main/ae.sft -d /content/TotoroUI/models/vae -o ae.sft

!aria2c --console-log-level=error -c -x 16 -s 16 -k 1M https://huggingface.co/camenduru/FLUX.1-dev/resolve/main/clip\_l.safetensors -d /content/TotoroUI/models/clip -o clip\_l.safetensors

!aria2c --console-log-level=error -c -x 16 -s 16 -k 1M https://huggingface.co/camenduru/FLUX.1-dev/resolve/main/t5xxl\_fp8\_e4m3fn.safetensors -d /content/TotoroUI/models/clip -o t5xxl\_fp8\_e4m3fn.safetensors

import random

import torch

import numpy as np

from PIL import Image

import nodes

from nodes import NODE\_CLASS\_MAPPINGS

from totoro\_extras import nodes\_custom\_sampler

from totoro import model\_management

DualCLIPLoader = NODE\_CLASS\_MAPPINGS["DualCLIPLoader"]()

UNETLoader = NODE\_CLASS\_MAPPINGS["UNETLoader"]()

RandomNoise = nodes\_custom\_sampler.NODE\_CLASS\_MAPPINGS["RandomNoise"]()

BasicGuider = nodes\_custom\_sampler.NODE\_CLASS\_MAPPINGS["BasicGuider"]()

KSamplerSelect = nodes\_custom\_sampler.NODE\_CLASS\_MAPPINGS["KSamplerSelect"]()

BasicScheduler = nodes\_custom\_sampler.NODE\_CLASS\_MAPPINGS["BasicScheduler"]()

SamplerCustomAdvanced = nodes\_custom\_sampler.NODE\_CLASS\_MAPPINGS["SamplerCustomAdvanced"]()

VAELoader = NODE\_CLASS\_MAPPINGS["VAELoader"]()

VAEDecode = NODE\_CLASS\_MAPPINGS["VAEDecode"]()

EmptyLatentImage = NODE\_CLASS\_MAPPINGS["EmptyLatentImage"]()

with torch.inference\_mode():

clip = DualCLIPLoader.load\_clip("t5xxl\_fp8\_e4m3fn.safetensors", "clip\_l.safetensors", "flux")[0]

unet = UNETLoader.load\_unet("flux1-dev-fp8.safetensors", "fp8\_e4m3fn")[0]

vae = VAELoader.load\_vae("ae.sft")[0]

def closestNumber(n, m):

q = int(n / m)

n1 = m \* q

if (n \* m) > 0:

n2 = m \* (q + 1)

else:

n2 = m \* (q - 1)

if abs(n - n1) < abs(n - n2):

return n1

return n2

2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

with torch.inference\_mode():

positive\_prompt = "black forest toast spelling out the words 'FLUX DEV', tasty, food photography, dynamic shot"

width = 1024

height = 1024

seed = 0

steps = 20

sampler\_name = "euler"

scheduler = "simple"

if seed == 0:

seed = random.randint(0, 18446744073709551615)

print(seed)

cond, pooled = clip.encode\_from\_tokens(clip.tokenize(positive\_prompt), return\_pooled=True)

cond = [[cond, {"pooled\_output": pooled}]]

noise = RandomNoise.get\_noise(seed)[0]

guider = BasicGuider.get\_guider(unet, cond)[0]

sampler = KSamplerSelect.get\_sampler(sampler\_name)[0]

sigmas = BasicScheduler.get\_sigmas(unet, scheduler, steps, 1.0)[0]

latent\_image = EmptyLatentImage.generate(closestNumber(width, 16), closestNumber(height, 16))[0]

sample, sample\_denoised = SamplerCustomAdvanced.sample(noise, guider, sampler, sigmas, latent\_image)

model\_management.soft\_empty\_cache()

decoded = VAEDecode.decode(vae, sample)[0].detach()

Image.fromarray(np.array(decoded\*255, dtype=np.uint8)[0]).save("/content/flux.png")

Image.fromarray(np.array(decoded\*255, dtype=np.uint8)[0])