《General Image-to-Image Translation with One-Shot Image Guidance》阅读报告

班级：智能2204 姓名：朱思颖 学号：202208040402

**目录**

[一、论文信息 2](#_Toc13564)

[二、研究背景 2](#_Toc22576)

[三、VCT 的框架和核心设计 2](#_Toc21255)

[四、技术细节与原理拆解 3](#_Toc5978)

[1. Latent Diffusion 过程 3](#_Toc17737)

[2. 内容-概念反演（CCI） 4](#_Toc15985)

[（1）PTI（Pivotal Tuning Inversion） 4](#_Toc12044)

[（2）MCI（Multi-Concept Inversion） 5](#_Toc23092)

[3. 内容-概念融合（CCF） 5](#_Toc28227)

[（1） 构建双流扩散生成器（基于DDIM） 5](#_Toc3121)

[（2）软注入机制（注意力控制机制） 5](#_Toc5125)

[五、个人认为设计巧妙的地方 7](#_Toc18400)

[1、PTI 与 MCI 分离处理 7](#_Toc14782)

[2、 空间融合而非图像空间融合 7](#_Toc11649)

[3、 注意力控制策略 7](#_Toc24847)

[六、实验 7](#_Toc6420)

[七、总结 9](#_Toc15212)

[八、复现 9](#_Toc28128)

[1、搭建实验环境 10](#_Toc24575)

[2、调用VCT来做风格迁移任务 11](#_Toc11108)

[3、总结过程中遇到的问题和解决方法 12](#_Toc26971)

# 一、论文信息

|  |  |
| --- | --- |
| Authors | Bin Cheng, Zuhao Liu, Yunbo Peng, Yue Lin |
| Citations | NetEase Games AI Lab |
| Comments | accepted by ICCV 2023 |
| Subjects | **Computer Vision and Pattern Recognition (cs.CV)** |
| Cite as | [arXiv:2307.14352](https://arxiv.org/abs/2307.14352)**[cs.CV]** |

# 二、研究背景

图像到图像翻译（I2I）是一类广泛应用于人脸替换、艺术风格迁移、视觉重构的任务，其理想目标是在保留源图像结构信息的同时，实现对参考图像中某种“概念”（风格、对象属性、局部形状等）的迁移。

然而，已有方法各有其短：

**·基于GAN的方法**需要大量成对训练数据，难以泛化到新概念或新风格； -

·**基于文本引导的扩散模型**虽灵活，但难以精准表达图像中复杂而具象的视觉概念；

·**图像引导的方法**大多依赖多图指导或领域特定训练，难以在“one-shot”场景中应用。

于是作者产生一个自然的想法：**是否可以仅通过一张参考图像，就将其概念迁移到另一张源图像中，而不丢失后者的结构**，同时希望它具有通用性。 这就是论文所提出的VCT框架目标实现的事。

# 三、VCT 的框架和核心设计

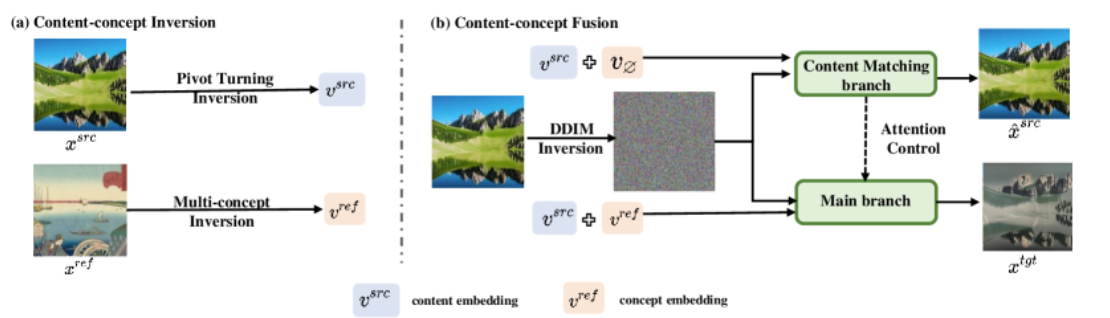
它的基本出发点是将“图像中的概念”看作一种视觉上的嵌入向量，并将其与结构嵌入分别处理后，再融合生成图像。

整个方法在扩散模型的潜空间中进行，分为两个阶段：

·内容-概念反演（**CCI**）：提取源图的“结构表示”，参考图的“概念表示”。

·内容-概念融合（**CCF**）：在扩散生成过程中将这两类表示融合，生成目标图像。

可以看出VCT设计的思路是**将结构与语义作为可组合模块**处理。



如图为它的工作流程：对于给定的一个源图像和一个参考图像 ，

·先进行CCI过程，应用 PTI处理 来获得源文本嵌入 。同时，应用MCI处理 来学习参考文本嵌入 。

·再进行CCF过程，采用DDIM反转架构进行图像翻译工作，包括主分支 B 和内容匹配分支 B\*，从使用 DDIM 反转的相同初始噪声开始。

·内容匹配分支重建源图像并提取注意力图用于注意力控制机制。

·最后，main 分支收集所有信息以获取目标图像 。

所以VCT的目标是：生成新图像——它迁移了的风格，保留了的结构和语义。

# 四、技术细节与原理拆解

## 1. Latent Diffusion 过程

潜在扩散模型LDM是**概率生成模型**，它是作为VCT的模型支柱，使用自动编码器在潜在空间中执行去噪操作。

对于输入图像x，在自动编码器中进行：

*（被编码到潜在空间，从潜在空间又解码到图像空间）*

· 编码过程：

·解码过程：

LDM 通过在潜空间 中逐步去噪来生成图像，其核心优化目标是：



· 是条件向量（文本嵌入），在论文中 和分别表示源图的结构嵌入和参考图的概念嵌入。

· 是原始图像的潜在表示。 · 是从正态分布 中采样的噪声。

·是从均匀分布 中采样的时间步。 · 表示欧几里得范数（L2 范数）的平方。

· 是神经网络，用于预测在给定潜在表示 、时间步 和文本嵌入 的情况下增加的噪声。

**那么LDM到底做了什么呢？**

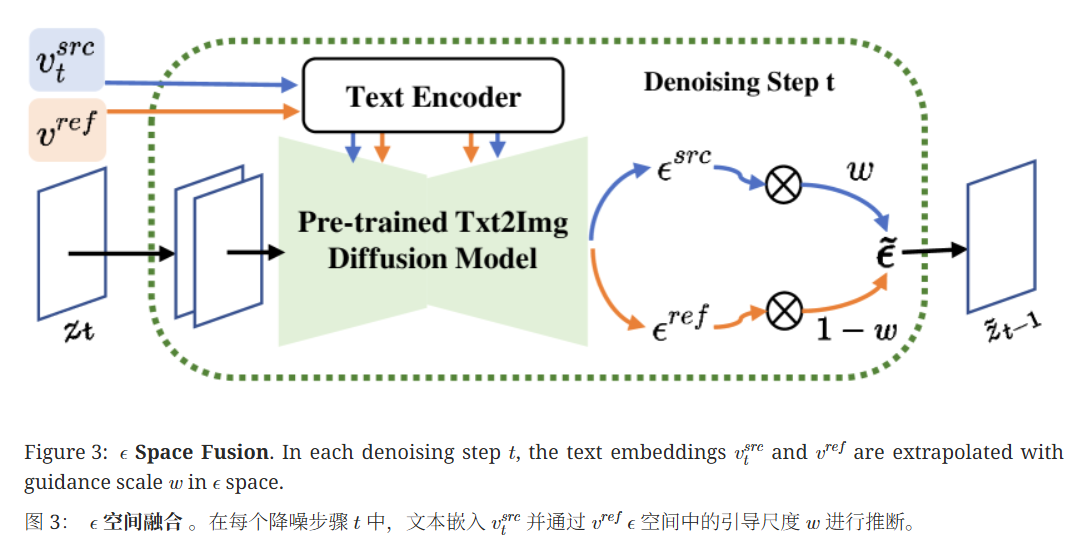
图像->潜在空间这一步通过编码来实现，是在降维编码成更小的潜在空间向量；在这个空间中扩散（前向加噪，反向去噪，形成扩散建模函数）；潜在空间->图像将生成好的结果解码回图像。

∴ 它把“在图像上学习如何生成”这件事，转移到一个名为潜在空间的“压缩空间”中去完成，从而在保持高质量的同时降低了计算量。

## 2. 内容-概念反演（CCI）

CCI 的目标是分别学习：

·一个能使源图在扩散模型中“重建出来”的结构表示；

· 一个能使参考图的“概念”在潜空间中得到充分捕捉的语义嵌入。 

输出：

Latents和注意力图

输入：

两张image

它的具体实现结合了两种方法：

### **（1）PTI（Pivotal Tuning Inversion）**

• 优化目标：

* 其中

也就是为源图 找到最优嵌入序列 来“重走”其扩散路径。

• 它的作用是将源图结构提取为嵌入向量，反向优化每一步的扩散过程，使模型在固定文本条件下能复原该图像。可以理解为让模型学会在自身语言中重新描述源图。

### **（2）MCI（Multi-Concept Inversion）**

• MCI的作用是提取参考图的风格、身份等，生成多个概念嵌入，，...

寻找的那个“最优”嵌入的特点是：不仅能指导模型生成视觉上接近参考图的图像，还能确保生成路径与参考图一致。

• 采用两种损失联合训练：

·标准扩散损失：

·潜空间重建损失：

∴ CCI中，PTI和MCI分别反演源图和引导图的语义信息，为 CCF 提供语义支撑。

## 3. 内容-概念融合（CCF）

CCF 我认为是 VCT 最具启发性的部分，它将CCI 提供的源图像的内容和参考图像的概念/风格在反向扩散过程中融合，并生成最终输出图像。

它设计了如下策略：

### 构建双流扩散生成器（基于DDIM）

“双流”一支以源图嵌入为主（结构对齐），一支以参考嵌入为主（引导概念）。

·主干支 B：根据 ，**生成目标图像** 。

·辅助分支 B\*：重建源图并生成 attention map，用于指导主干支注意力结构（也就是控制B分支）。

它们都采用DDIM反转机制，从相同的初始噪声开始反向扩散，但B 分支引入了参考图的概念注意力 ，B\* 分支只用了源图的 attention作为约束与参考。

### （2）软注入机制（注意力控制机制）

在B分支中，提出一种**“软注入机制”**，利用 attention map ，在扩散前期强制主干网络的注意结构对齐于源图，以保持图像轮廓。 具体为：

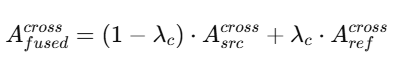
·在每一步降噪（扩散）中，使用 ε 线性融合策略：

·ε空间是两个文本嵌入分别引导产生两种不同的噪声预测而产生的空间

·ε融合发生在每步的残差预测上。

·它是**有选择地**将引导图像中的**注意力图**注入到正在进行的生成过程中（替换 + 混合的方式）。

***Cross-Attention 注入：***把“参考图的语义引导”部分地**替换并混入**源图注意力中，这样，最终生成图既遵循源图结构，又能接受参考图的“语义指令”。

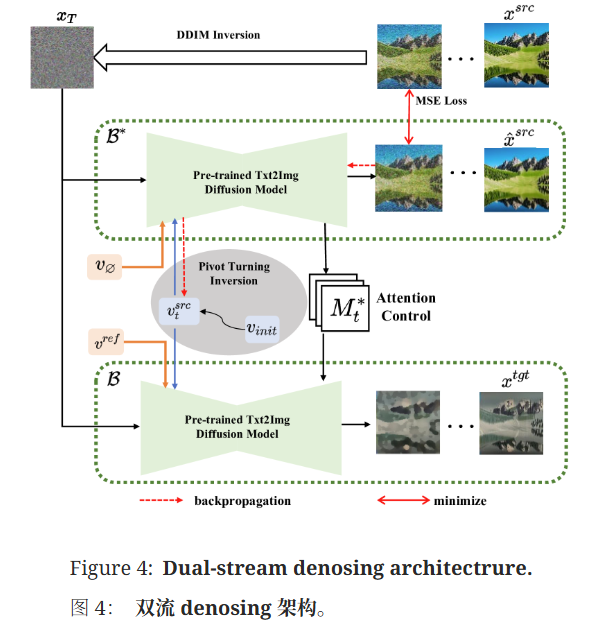


原本用于生成源图的 attention记作

***Self-Attention 注入：***替换局部风格特征的空间相关性。把 含有目标风格的空间分布特征注入源图生成过程中，实现风格迁移，使得生成图像在结构上保留源图，但在纹理、笔触、阴影等风格方面贴近参考图。



所谓“替换 + 混合”，就是指对扩散模型中 attention 模块的中间计算结果进行线性插值融合，让生成过程**部分采用参考图的注意力结果**，从而实现**风格迁移而非内容替换**。这种结构有效提升了结构保留能力。



Cross-attention injection：传递语义概念（某人）（某风格）。

Self-attention injection：负责捕捉图像中像素间的空间依赖，如纹理、边缘、布局等。

在时间步 t < 时，将B\* 的cross-attention 和self-attention map 注入主干支。

这一部分的设计是核心，作者借鉴扩散模型的部分正向过程，从源图像中提取内容特征，并从引导图像中获取注意力层的跨注意力和自注意力映射。

文中还有多个涉及注意力融合的公式，但本质上是通过对中间隐变量的处理，将“内容”和“概念”分别嵌入扩散过程的两个阶段中。

# 五、个人认为设计巧妙的地方

## 1、PTI 与 MCI 分离处理

对源图结构与参考概念的优化是分开进行的，训练过程很清晰，模块职责明确。而且它从单个图片来引入概念注入，提供图片中的视觉概念，而不仅仅依靠文本提示。这样图像中难以用文字准确描述的细节就可以被提取出来，进一步映射成嵌入文本来给到扩散模型，从而效果更好，也更通用。

## 空间融合而非图像空间融合

相比直接混合图像，融合去噪更自然。

## 注意力控制策略

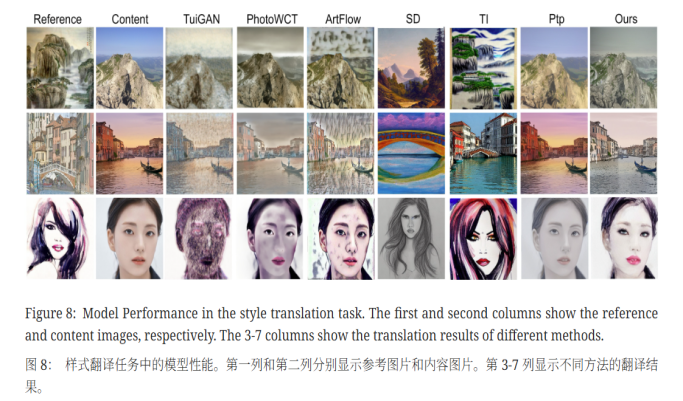
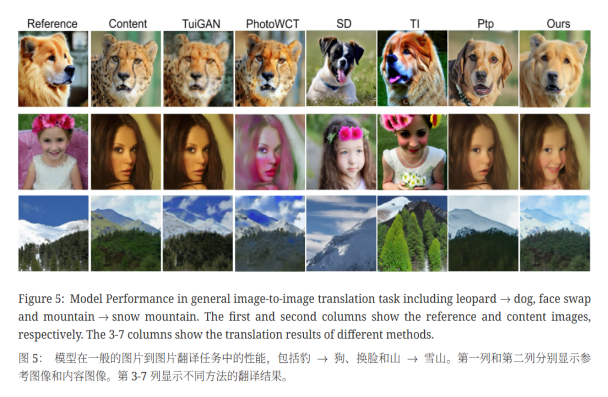
VCT是利用辅助支路B\*来生成 Attention Map，为主干提供结构约束的。这也是我认为最优雅的地方！

对于图像生成模型Stable Diffusion，它什么都没改——没有改网络、没有改损失函数，甚至没有引入任何控制器或外部模块。它只是选择在一个模型最关键但最容易被忽视的地方——注意力机制上动手，通过替换 attention map 实现对“生成重心”的转移。

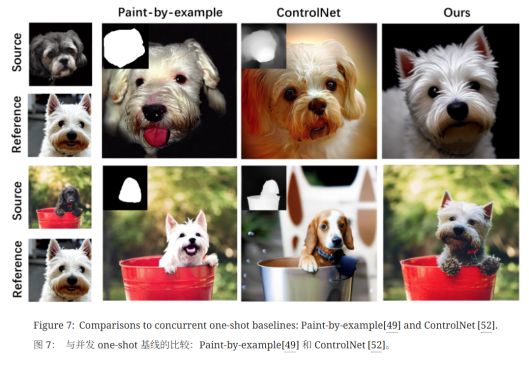
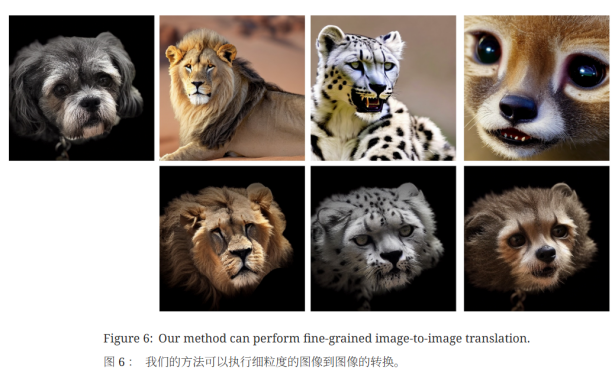
这也是予人启发的！在与大型预训练模型交互时，也许不一定要重新训练一个新模型，找到切入点、懂得如何与已有模型进行协同式对话或许会带来新的进展。

# 六、实验

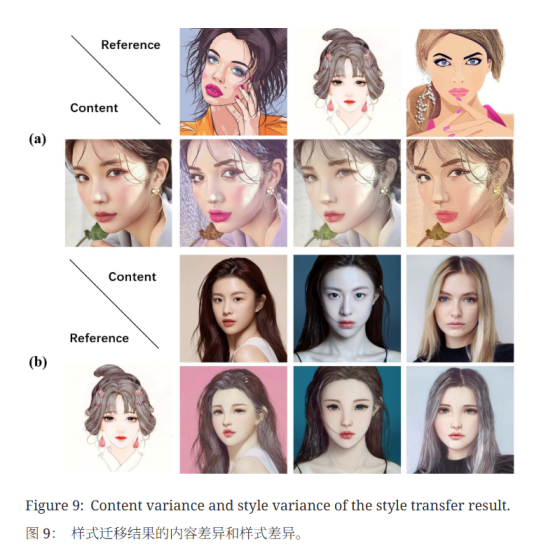
论文通过人脸转换、动物展示、艺术风格和材质数据集，高质量地进行了一带应用实验，可以看到效果非常好！既保留了内容，又进行了适度转换。



VCT 在结构还原与风格融合方面达到了很好的平衡效果。其他方法要么仅能迁移低层风格，要么缺乏局部对齐能力，而VCT则在双重目标下保持了稳定输出。

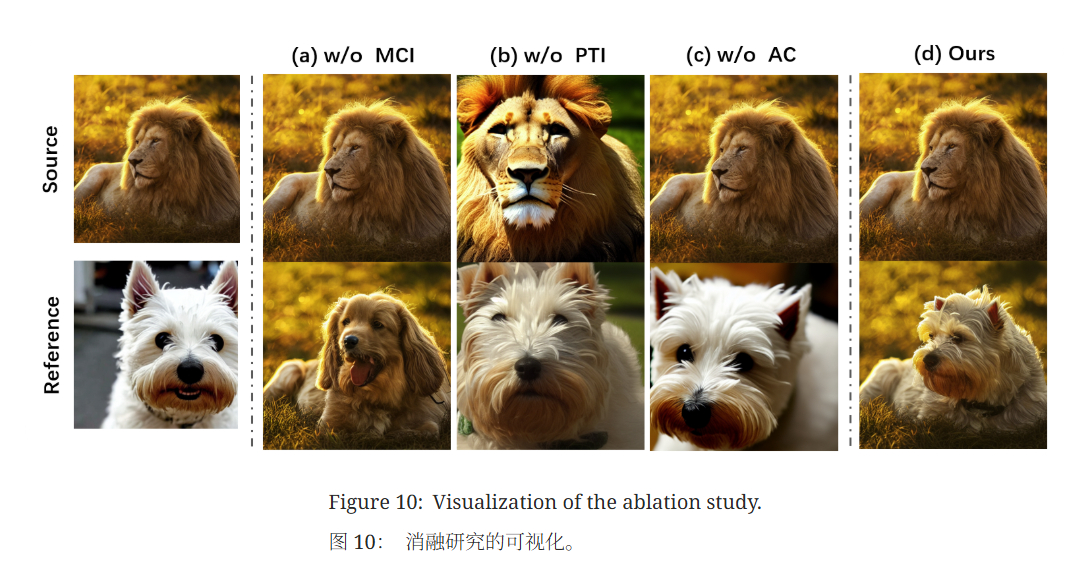


在细粒度的I2I任务上也表现优秀，而且只用一张参考图、不需要训练或使用描述性文本，就可以培育概念，完成基于图像的转换。



风格迁移做的也很好，转换出来的图像也很漂亮。

该团队还做了消融实验，展示删除 PTI、MCI、Attention Control（AC）后的效果对比。



·无 MCI —— 风格迁移失败，看起来只翻译出ref的身份信息“狗”，而没有更进一步的信息“品种”，说明参考嵌入对语义转换很重要；

·无 PTI —— 面部结构塌陷了，整体结构被破坏了，结构嵌入出现问题；

·无 AC —— 结构对齐丢失，图像生成变形，打破了生成流程中局部区域的一致性。

这也直接验证了各模块的重要性。

# 七、总结

VCT 通过实现灵活、高质量、一次性的图像到图像转换，无需重新训练或提示 ，弥合了 GAN 的表达能力和预训练扩散模型的能力之间的差距。

VCT的优势：

•无需模型训练，它是基于Stable Diffusion构建的，并与预训练的生成模型集成。

•在一次性设置中工作，它只需要一个引导图像来指定目标概念，不用数据集。

•传统翻译方法依赖成对数据、语义标签或者特定领域假设，VCT不需要此类监督。

VCT 的实验图也选得很好，隐含其机制的严密设计逻辑。从结构-语义分离，到控制注入的粒度与时机，每张图都在回答一个问题：为什么这个模块是必要的。

# 八、复现

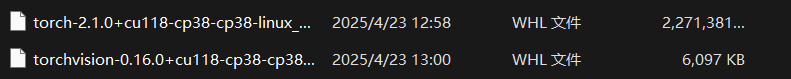
环境：主机windows+docker容器中ubuntu20.04

核心依赖：cuda11.8.0，python3.8，torch2.1.0，torchivision0.16.0。

## 1、搭建实验环境

（上传zip中包含我的Dockerfile，以及修改后的requirements.txt）

·首先准备好需要的轮子，解决镜像源没有对应轮子的问题。



·在项目目录下下载论文代码并解压。

·准备好Dockerfile，以及与自己的gpu、与所选torch等版本所适配的requirements.txt。

· 预先下载好预训练模型stable-diffusion-v1-5

<https://hf-mirror.com/stable-diffusion-v1-5/stable-diffusion-v1-5>

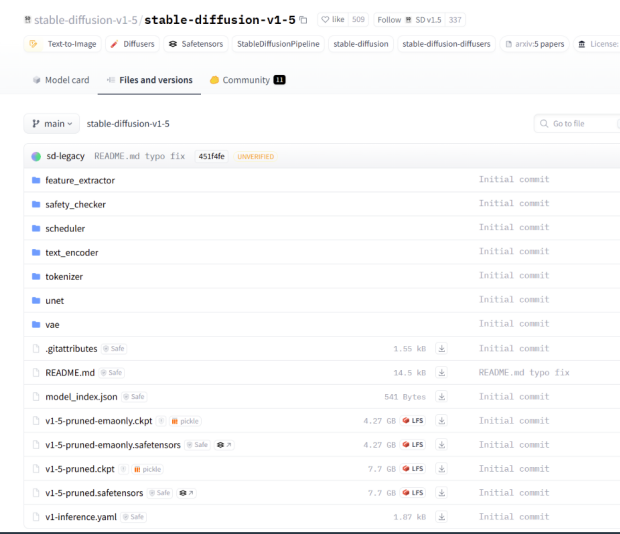
强烈推荐在HF Mirror用IDM来下载！

比写脚本用 snapshot\_download快很多，而且不会因为不知名原因断点。

缺点是偶尔需要你手动点一下，但是二十分钟基本就搞定了。不会像git有时莫名丢失内容，或者把发布的所有模型都下下来了。（之前下的时候也试过修改作者的代码，在用diffusion缓存模型的地方加个镜像路径，但是还是出现了timeout错误，所以最终还是选择下到本地先）

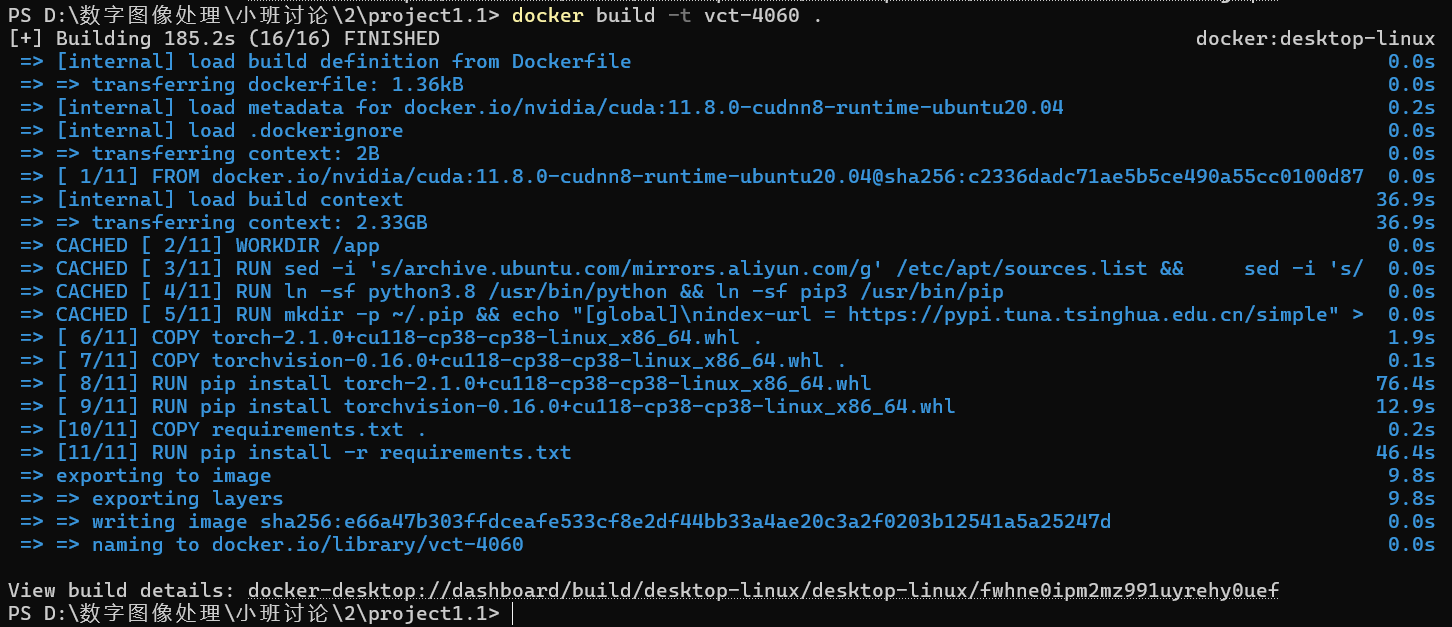
对于这篇论文的实现，因为作者的代码是基于diffussion的，所以不需要我们下载.ckpt或者.safetensors，但其他的文件夹格式、内容、以及配置文件要照例下载。

下好以后的目录结构

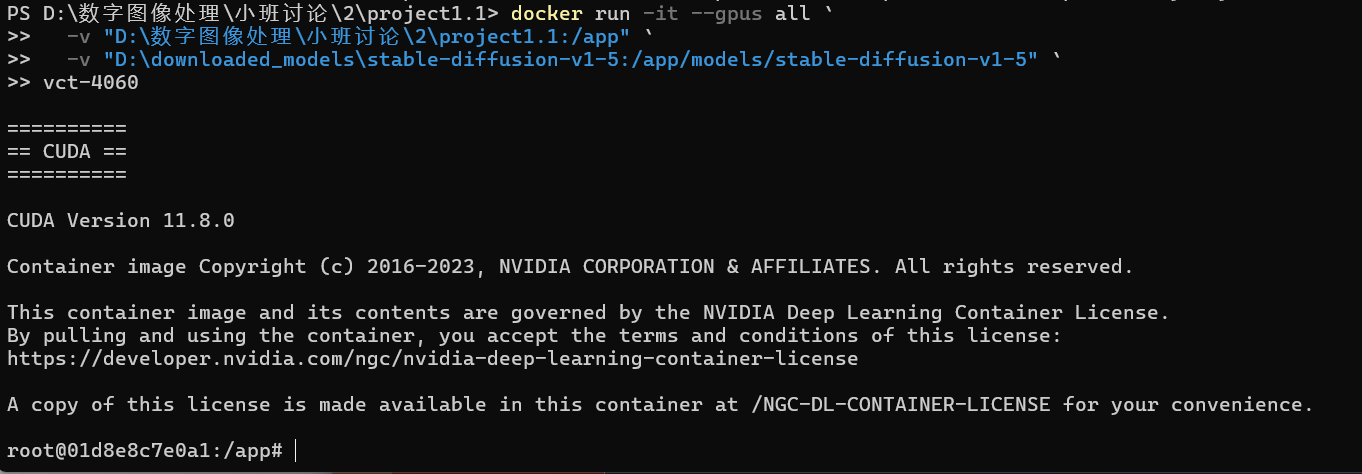
· 开始构建镜像

docker build -t visual-concept-translator .



·挂载项目文件和预训练模型，运行镜像

docker run -it --gpus all `  
 -v "D:\数字图像处理\小班讨论\2\project1.1:/app" `  
 -v "D:\downloaded\_models\stable-diffusion-v1-5:/app/models/stable-diffusion-v1-5" `  
vct-4060



## 2、调用VCT来做风格迁移任务

·进入容器后，参照作者给的调用VCT来做I2I任务的命令，写个配置+运行脚本：

#!/bin/bash  
# run\_vct\_lowmem.sh  
  
# 低显存设置：适配 RTX 4060 / 8GB 显存  
# - 启用 FP16 半精度  
# - 设置 PyTorch 显存碎片优化  
# - 减少 embedding 数量  
# - 保证只使用一张图测试  
  
export PYTORCH\_CUDA\_ALLOC\_CONF=max\_split\_size\_mb:64  
  
HF\_HUB\_OFFLINE=1 accelerate launch main.py \  
 --concept\_image\_dir="./examples/concept\_image" \  
 --content\_image\_dir="./examples/content\_image" \  
 --pretrained\_model\_name\_or\_path="/app/models/stable-diffusion-v1-5" \  
 --output\_image\_path="./outputs" \  
 --initializer\_token="girl" \  
 --max\_train\_steps=5 \  
 --concept\_embedding\_num=1 \  
 --cross\_attention\_injection\_ratio=0.2 \  
 --self\_attention\_injection\_ratio=0.9 \  
 --use\_l1 \  
 --mixed\_precision=fp16

·运行脚本

bash run\_vct\_lowmem.sh

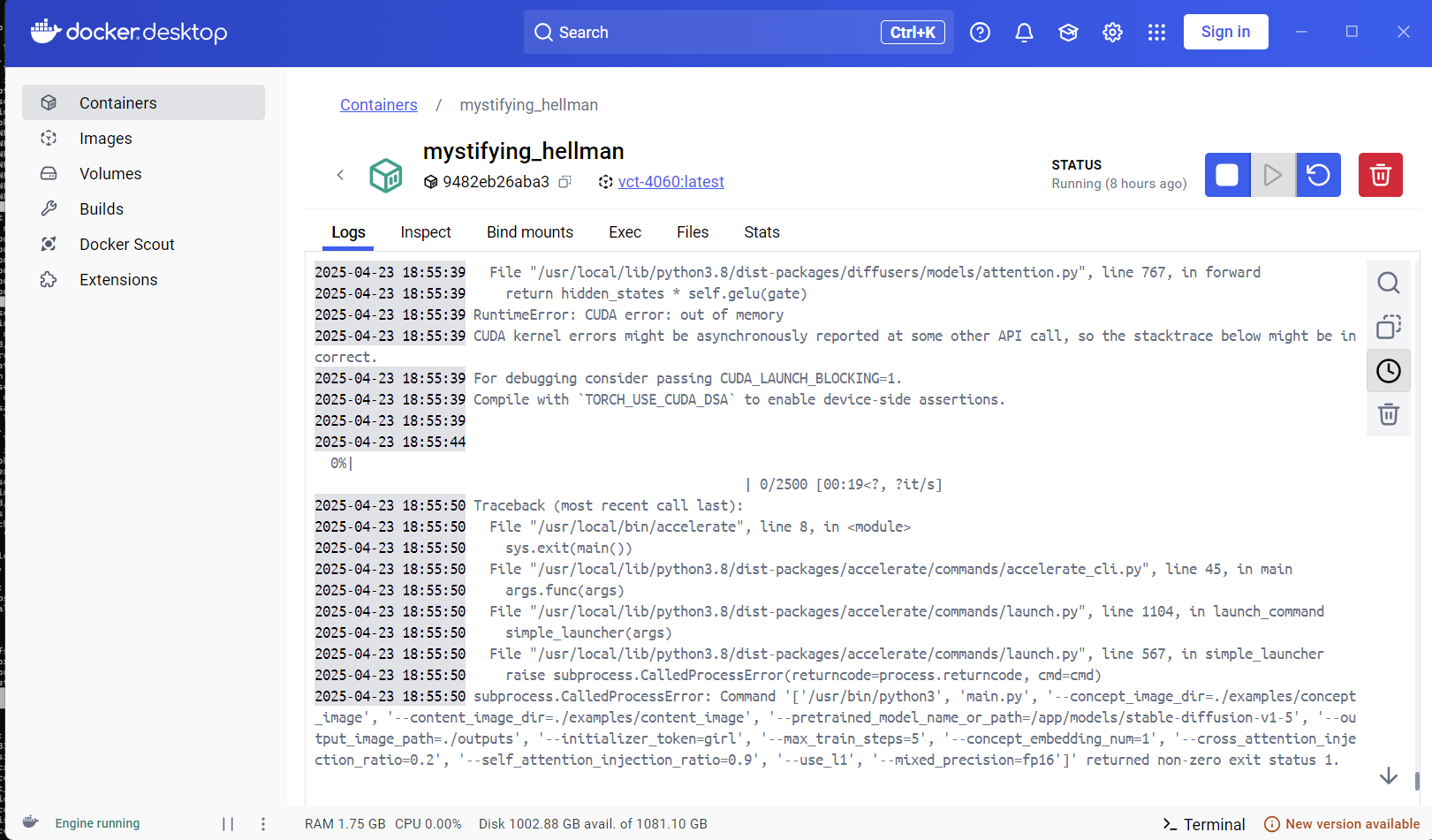
一切正常，除了**CUDA error: out of memory**

RuntimeError: CUDA error: out of memory

CUDA kernel errors might be asynchronously reported at some other API call, so the stacktrace below might be incorrect.

For debugging consider passing CUDA\_LAUNCH\_BLOCKING=1.

Compile with `TORCH\_USE\_CUDA\_DSA` to enable device-side assertions.

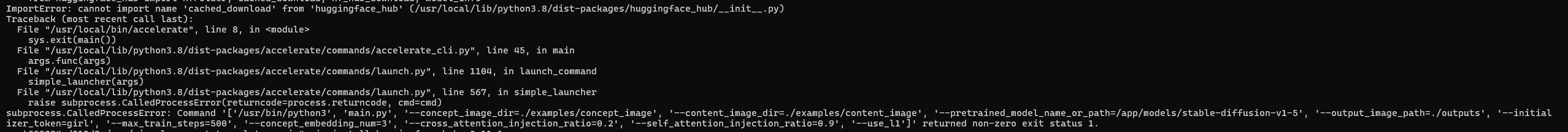




## 3、总结过程中遇到的问题和解决方法

（1）完全按照论文环境搭建时遇到的问题和解决方法

·hugging\_face版本问题

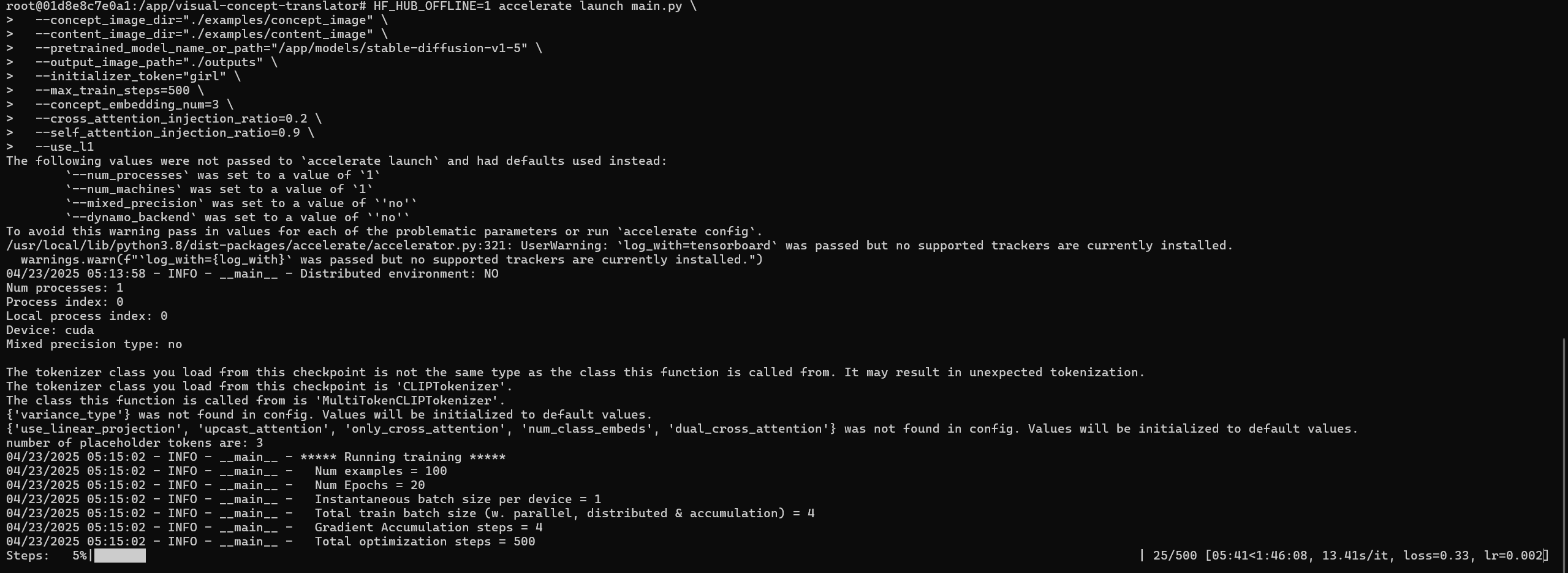


新版本里cached\_dowload函数似乎改名或者被移除了，所以需要更换到以前的版本，我们可以把这个也放到requirements.txt里。

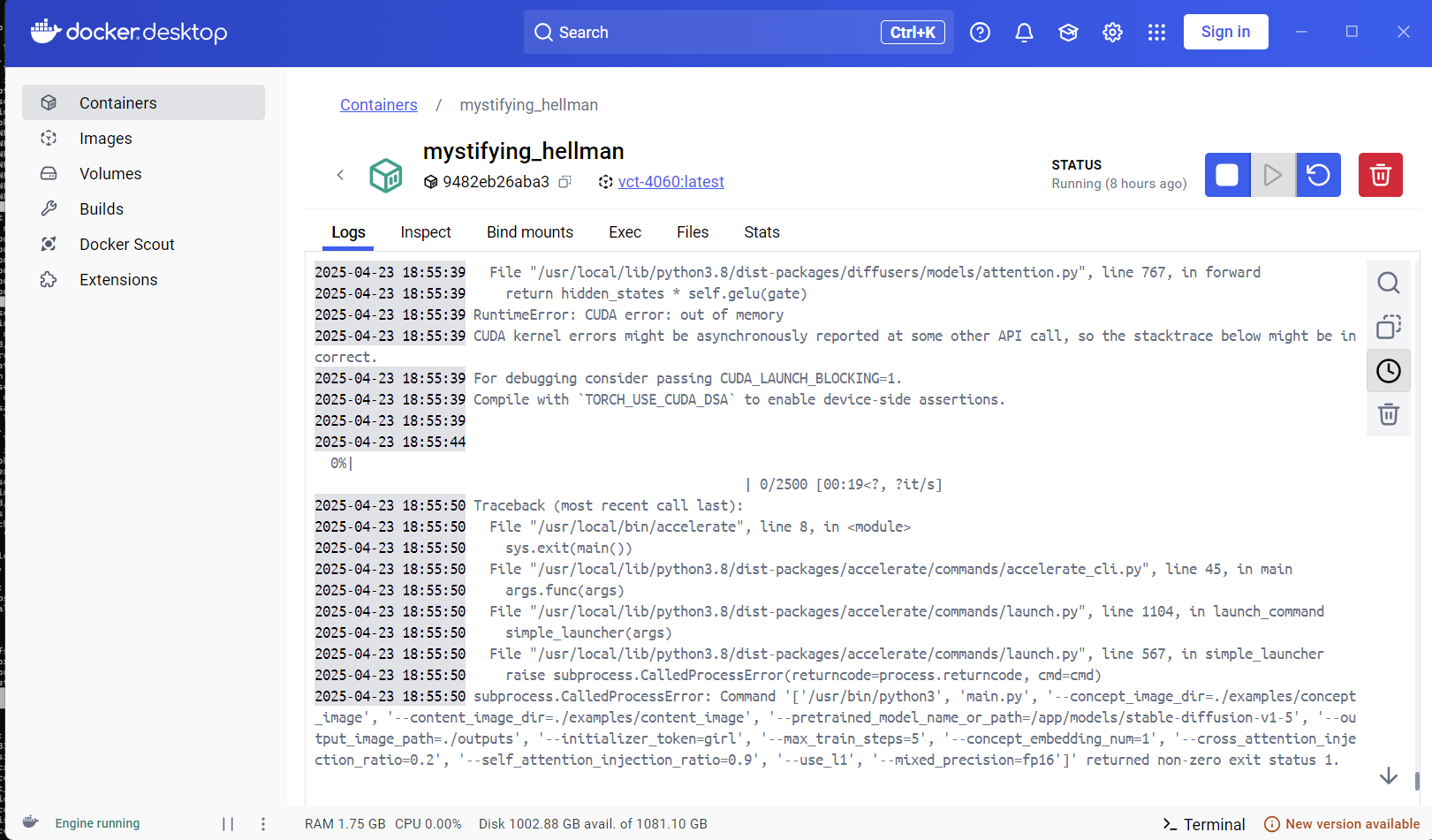
·出现cuda版本不兼容问题



虽然在dockerfile里指定了cuda:11.3.1-cudnn8-runtime-ubuntu20.04来匹配论文实验的环境，但我的GPURTX 4060（sm\_89 架构）是 PyTorch 1.10.1 无法支持的 ，所以只能改论文环境。

1. 为了适配我电脑的gpu，修改了torch、torchvision、cuda的版本，这之后再进行实验，镜像构建过程、使用论文代码仓的推荐指令来使用VCT的过程都很顺畅了！ （贴张过程图）它终于跑起来了 。 当 使用作者推荐的指令，迭代500次会很慢（看他后面显示预计要近两个小时才能跑完）为了快速验证，只处理一张源和一张概念，迭代次数为5。然后遇到了问题。

·GPU显卡爆了



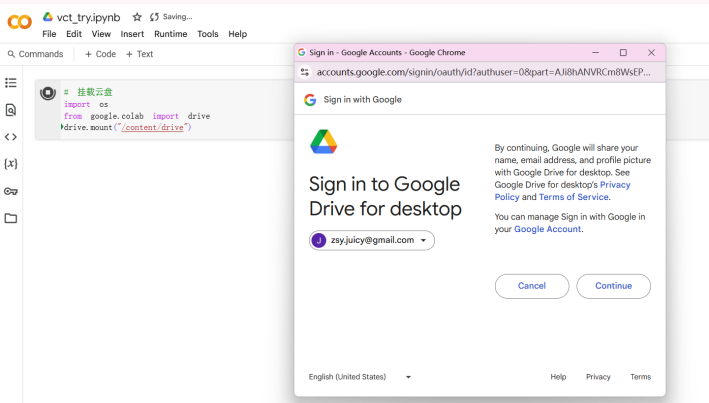
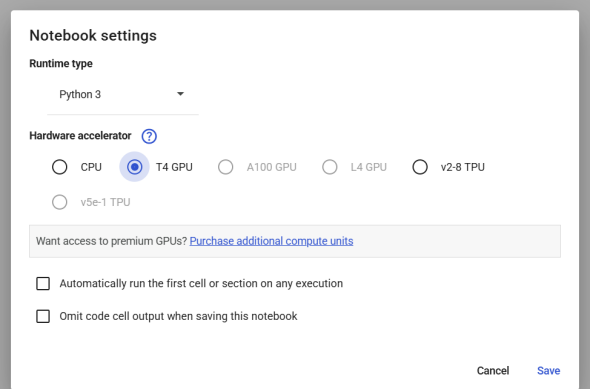
报“RuntimeError: CUDA error: out of memory”的错，查询发现，VCT 的反向编辑部分需要的计算资源很大，作者推荐的GPU为 A100 / A30（>= 24GB 显存），而我的笔电所带GPU为 RTX 4060 / 8GB 显存，所以就华丽丽地爆掉了。后边也尝试修改代码，让它默认能处理的最小分辨率512变成256，并且设置的参数也尽可能小，运行后还是会CUDA error: out of memory。

后边想起来谷歌有免费的GPU可以用，所以转战Google Colab了。

1. Google Colab的心酸之旅

·首先，想把预训练模型上传到GoogleDrive，但告诉失败，原因是网盘只有10+G额度。

·直接打开Colab，连接后修改notebook的Hardware accelerator为T4 GPU，获得了12h免费的16G显卡GPU！



·但部署环境时发现，由于所提供的cuda、python版本都很高，notebook里的相关依赖也很多，我们在尝试部署时，会发现各种版本不适配的报错层出不穷，而我们基于colab这个平台，很难把所有依赖都平衡好不报错的版本（而且它似乎不支持再创建虚拟环境），所以就失败了。