

# Gemma 3 多模态开发实践： 以RefSheet Chat为例



Xihan Li (李锡涵)  
University College London  
<https://snowkylin.github.io>

# Gemma 3 特点

## 多模态

(文字+图像输入, 出色的图片理解能力, 操作友好)

You



这幅图在说什么

Assistant `lmstudio-community/gemma-3-27b-it`

这张图片是一张幻灯片, 内容是关于Gemma 3的多模式开发实践, 以RefSheet Chat为例。

具体来说:

- **标题:** Gemma 3 多模式开发实践: 以RefSheet Chat为例 (Gemma 3 Multi-modal Development Practice: Using RefSheet Chat as an Example)
- **Logo:** Google Developer Experts 的logo
- **演讲者信息:**
  - Xihan Li (李锡涵)
  - University College London
  - 个人网站: <https://snowkylin.github.io>

总而言之, 这张幻灯片表明这是一场关于Gemma 3模型多模态开发实践的演讲或演示, 并且RefSheet Chat是一个具体的案例

## 多语言

(35+语言, 中文能力可圈可点, 很多小语种也不在话下, 出海友好)

You

请用中、日、韩、英、德、法、西、俄、阿拉伯语, 一句话介绍自己, 无需注音。

Assistant `lmstudio-community/gemma-3-27b-it`

好的, 以下是用九种语言自我介绍的一句话:

- **中文:** 我是一个大型语言模型, 致力于帮助人们解决问题。
- **日语:** 私は人々の問題を解決するのに役立つ大規模言語モデルです。(Watashi wa hitobito no mondai o kaiketsu suru node yakuataru daikibo gengo moderu desu.)
- **韩语:** 저는 사람들의 문제를 해결하는 데 도움을 주는 대규모 언어 모델입니다。(Jeoneun saramdeului munjeleul haegelhaneun de doumeul juneun daegimo eoneo modelimnida.)
- **英语:** I am a large language model dedicated to helping people solve problems.
- **德语:** Ich bin ein großes Sprachmodell, das sich der Lösung von Problemen für Menschen widmet.
- **法语:** Je suis un grand modèle linguistique dédié à aider les gens à résoudre des problèmes.
- **西班牙语:** Soy un modelo de lenguaje grande dedicado a ayudar a las personas a resolver problemas.
- **俄语:** Я большая языковая модель, предназначенная для помощи людям в решении проблем。(Ya bol'shaya yazykovaya model', prednaznachennaya dlya pomoshchi lyudyam v reshenii problem.)
- **阿拉伯语:** أنا نموذج لغوي كبير مخصص لمساعدة الأشخاص في حل مشكلاتهم。(Anā namūdhaj lughawī kabīr mukhassas li-muṣā'adat

## 可离线部署

(open model, 权重开放下载和微调, 隐私友好)

Hugging Face Search models, datasets, users...

google/gemma-3-27b-it like 979 Follow Google 10.5k

Image-Text-to-Text Transformers Safetensors gemma3 conversational text-generation-inference arxiv:28 papers License: gemma

Train Deploy Use this model

Model card Files Community 38

main gemma-3-27b-it

Go to file Ctrl+K 8 contributors History: 6 commits

+ Contribute

osanseviero patloeber Update README.md (#35) 005ad34 VERIFIED			
.gitattributes	2.45 kB	Super-squash branch...	14 days ago
README.md	25.1 kB	Update README.md (#...	4 days ago
added_tokens.json	35 Bytes	Super-squash branch...	14 days ago
chat_template.json	1.62 kB	Update chat_templat...	14 days ago
config.json	972 Bytes	Super-squash branch...	14 days ago
generation_config.json	215 Bytes	Adding top_k and to...	6 days ago
model-00001-of-0...	4.85 GB	Super-squash branch...	14 days ago
model-00002-of-0...	4.95 GB	Super-squash branch...	14 days ago
model-00003-of-0...	4.95 GB	Super-squash branch...	14 days ago
model-00004-of-0...	4.95 GB	Super-squash branch...	14 days ago
model-00005-of-0...	4.95 GB	Super-squash branch...	14 days ago

# 实践案例：RefSheet Chat

## 一个“character.ai升级版”

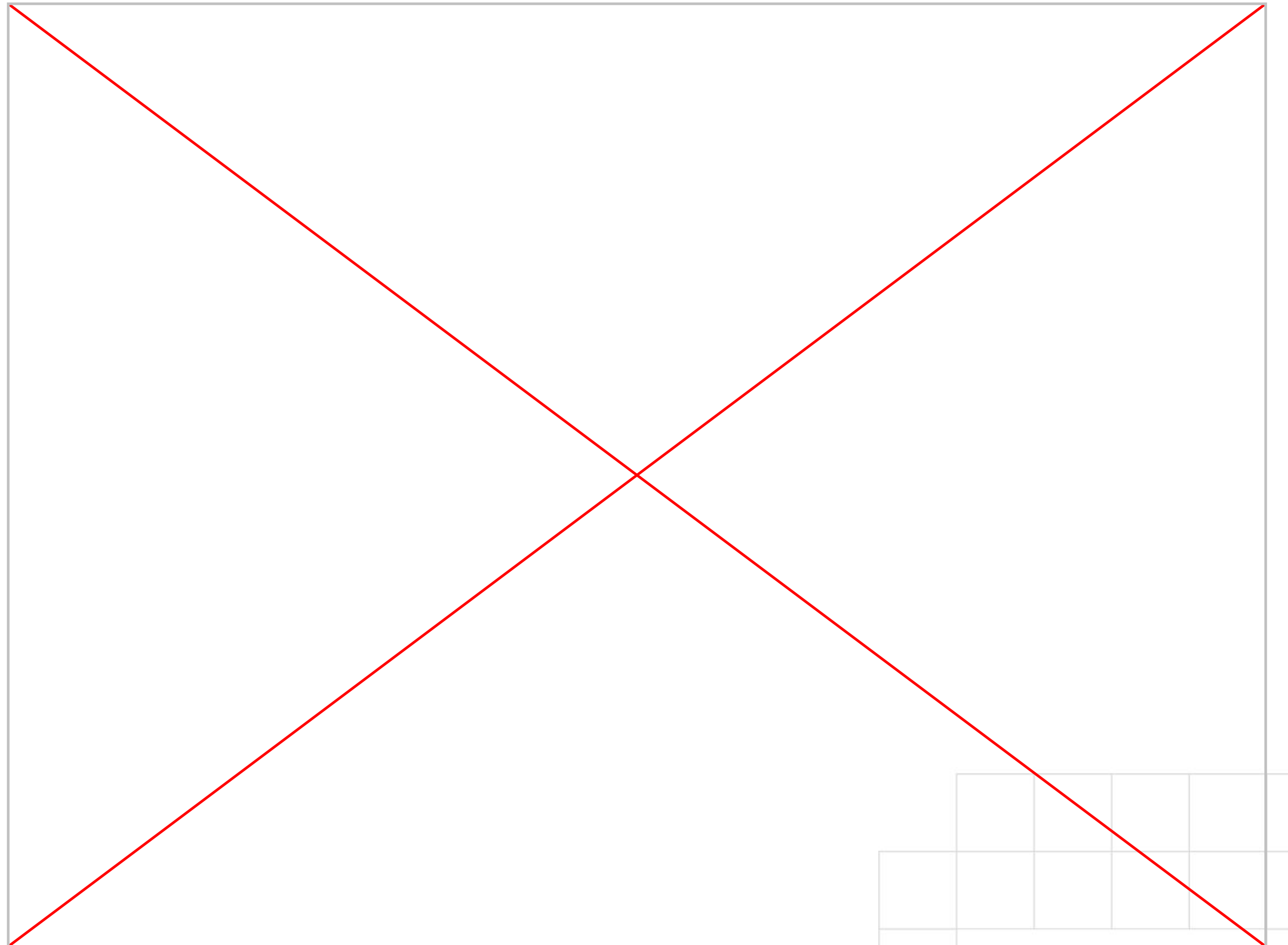
上传一张角色图片(设定图), 无需复杂的角色文字描述, RefSheet Chat将自动理解和“脑补”图中的信息, 并以这位角色的身份与用户对话。

支持离线运行, 角色图片完全在本地处理, 降低信息泄露风险。

GitHub: [https://github.com/snowkylin/refsheet\\_chat](https://github.com/snowkylin/refsheet_chat)

在线Demo: <https://refsheet.chat>

离线程序下载: <https://refsheet.chat/local>



# 本地部署 Gemma 3

## 基于HuggingFace的Transformers库

Gemma 3 有参数量 1B、4B、12B、27B四种版本可选  
1B版本是纯文本模型，从4B开始支持多模态(图片输入)  
本案例使用4B版本进行示范

<https://huggingface.co/google/gemma-3-4b-it>

模型大小约8G，内存16G即可运行，适合在一般开发环境进行部署调试

Model	Vision Encoder	Embedding Parameters	Non-embedding Parameters
<b>1B</b>	0	302M	698M
<b>4B</b>	417M	675M	3,209M
<b>12B</b>	417M	1,012M	10,759M
<b>27B</b>	417M	1,416M	25,600M

Table 1 | Parameter counts for the Gemma 3 models. Our vocabulary has 256k entries.

<https://arxiv.org/abs/2503.19786>

# 本地部署 Gemma 3

## 基于HuggingFace的Transformers库

### 1. 配置Python环境(安装Miniconda)

<https://www.anaconda.com/docs/getting-started/miniconda/install>

### 2. 创建conda环境“transformers\_gemma”并进入该环境

```
conda create -n transformers_gemma pip python=3.12
```

```
conda activate transformers_gemma
```

### 3. 安装支持Gemma 3的Transformers库

```
pip install git+https://github.com/huggingface/transformers@v4.49.0-Gemma-3
```

### 4. 安装PyTorch(参考 <https://pytorch.org/get-started/locally/> )

```
pip install torch torchvision torchaudio
```

# 本地部署 Gemma 3

## 基于HuggingFace的Transformers库

5. 注册一个HuggingFace账号，并访问

<https://huggingface.co/google/gemma-3-4b-it>，同意Google的使用协议，获得gemma 3的下载权限

6. 获取一个HuggingFace的Access Token(参考

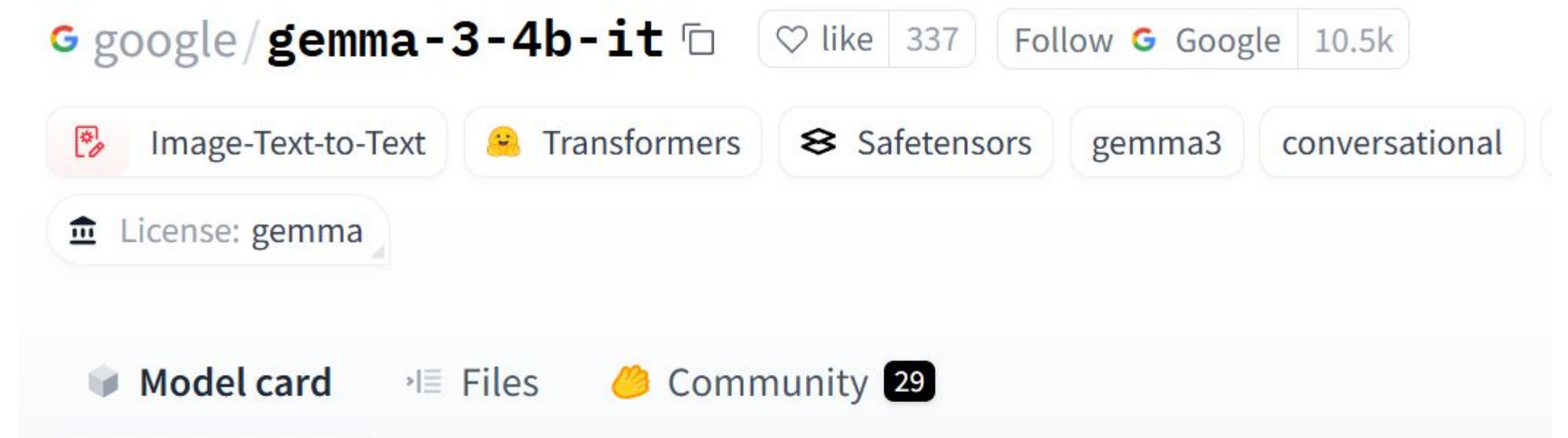
<https://huggingface.co/docs/hub/security-tokens>)，并在conda环境  
境内执行

```
huggingface-cli login
```

输入刚才获取的Token并回车，以在开发环境登录

HuggingFace账号

(也可以将环境变量HF\_TOKEN设置为获取的Token)



### Access Gemma on Hugging Face

This repository is publicly accessible, but you have to accept the conditions to access its files and content.

To access Gemma on Hugging Face, you're required to review and agree to Google's usage license. To do this, please ensure you're logged in to Hugging Face and click below.

Requests are processed immediately.

Log in

or

Sign Up

to review the conditions and access this model content.

# 本地部署 Gemma 3

## 基于HuggingFace的Transformers库

7. 建立 `main.py` 并输入以下代码

```
from transformers import pipeline
import torch
```

```
pipe = pipeline(
    "image-text-to-text",
    model="google/gemma-3-4b-it",
    torch_dtype=torch.bfloat16,
)
```

在conda环境内执行 `python main.py` 运行代码, 即会自动下载Gemma 3 4B模型。下载完毕后即可通过pipe实例, 在Python代码中进行调用。

# Gemma 3 本地图文推理

基本调用方法：

```
response = pipe(text=messages, generate_kwargs=generate_kwargs)
```

其中messages是对话记录，generate\_kwargs是推理参数(可选)

为了让Gemma 3进行基于图片的“角色扮演”，我们以一张角色图片“character.jpg”和一句指令“你的身份是图中的角色”作为输入。

```
messages = [  
  {  
    "role": "user",  
    "content": [  
      {"type": "image", "url": "character.jpg"},  
      {"type": "text", "text": "你的身份是图中的角色。"}  
    ]  
  }  
]
```



祈梦 Dream

外向活泼的蓝瞳绿色小龙，脸颊上有螺旋状花纹。擅长水彩和油画，希望用自己的画笔和创新风格，绘出心中的梦幻世界。

<https://www.wilddream.net/Art/index/introduction>  
已获授权使用



# Gemma 3 本地图文推理

推理参数设置:

```
generate_kwargs = {  
    'max_new_tokens': 1000,      # 最多生成的回复Token数量  
    'do_sample': True,          # 是否基于采样来生成（为False则直接选择概率最大的Token,  
                                # 每次生成的结果都一样）  
    'temperature': 1.0          # 温度越高，生成回复的随机性越强  
}
```

更多参数可参考：

[https://huggingface.co/docs/transformers/en/main\\_classes/text\\_generation#transformers.GenerationConfig](https://huggingface.co/docs/transformers/en/main_classes/text_generation#transformers.GenerationConfig)

# Gemma 3 本地图文推理

执行

```
response = pipe(text=messages, generate_kwargs=generate_kwargs)
```

返回

```
[{'input_text': [{'role': 'user',  
                  'content': [{'type': 'image', 'url': 'character.jpg'},  
                              {'type': 'text', 'text': '你的身份是图中的角色。'}]}]},  
 {'generated_text': [{'role': 'user',  
                     'content': [{'type': 'image', 'url': 'character.jpg'},  
                                 {'type': 'text', 'text': '你的身份是图中的角色。'}]}]},  
 {'role': 'assistant',  
  'content': '你好！我就是梦，一位喜欢绘画的绿龙！ \n\n我喜欢用水彩和油画来创作，  
并且很喜欢用自己的画作来启发和创造新的想法。很高兴认识你！ 😊 \n\n你今天过得怎么样？希望你也能拥有一个充满创造力的美好一天！ 🎨\n'}]]]
```

# Gemma 3 本地图文推理

多轮对话:在回复 generated\_text 的基础上再追加用户提问,并调用 pipe 生成回复

```
messages = response[0]['generated_text'] # 更新当前对话上下文
messages.append({"role": "user", "content": "你会画怎样的画呢"}) # 在上下文中追加用户提问
response = pipe(text=messages, generate_kwargs=generate_kwargs) # 再次调用pipe生成回复
```

Gemma 3 生成回复内容:

```
[{'input_text': [.....],
  'generated_text': [{'role': 'user',
                      'content': [{'type': 'image', 'url': 'character.jpg'},
                                   {'type': 'text', 'text': '你的身份是图中的角色。'}]}],
  {'role': 'assistant',
   'content': '你好!我就是梦, .....希望你也能拥有一个充满创造力的美好一天! 🎨\n'},
  {'role': 'user', 'content': '你会画怎样的画呢'},
  {'role': 'assistant',
   'content': '哈哈, 这个问题问得好! 我会画一些充满想象力、充满活力和梦幻色彩的画作。 \n\n* **奇幻生物: **
我会画各种各样的奇幻生物, 比如飞舞的精灵、藏在森林里的巨龙、或者在星空中遨游的星灵。 \n* **梦境场景: ** 我会把梦境中的景象变成画作,
比如漂浮在云端的城堡、在彩虹间跳跃的动物、 或者一个由糖果组成的城市。 \n* **人物与自然: ** 我喜欢画人物与自然和谐相处的画面, 比如坐在
湖边钓鱼的少年, 或者在森林里与动物玩耍的孩子。 \n* **抽象表达: ** 有时候, 我也会尝试用色彩和线条来表达自己的情感, 创造一些抽象的画
作。 \n\n我特别喜欢用鲜艳的颜色和流畅的线条来表现画面的生动和活力。 \n\n你觉得我应该画什么呢? 或者, 你有什么想让我画的吗?'}]}}
```

# Gemma 3 多语言对话生成

设定回复语言:直接在指令中说明即可

```
messages = [{ "role": "user",  
              "content": [ { "type": "image", "url": "character.jpg" },  
                           { "type": "text", "text": "你的身份是图中的角色, 使用英文。" } ] ] }
```

回复:

```
[ { 'input_text': [.....],  
  'generated_text': [ { 'role': 'user',  
                        'content': [ { 'type': 'image', 'url': 'character.jpg' },  
                                      { 'type': 'text', 'text': '你的身份是图中的角色。' } ] },  
                        { 'role': 'assistant',  
                          'content': 'Greetings! I\'m Dream, a little, moss-colored dragon with a fondness for  
water and paint. I love to use my own drawings as inspiration and create all sorts of wonderful things. It\'s a joy to  
splash around and bring a bit of color into the world! Don\'t be surprised if you see a bit of paint on me - it\'s all part of  
the creative process!" \n\n---\n\nHow would you like me to respond next? Do you want me to:\n\n* Describe a specific  
action I might be doing?\n* Answer a question about my character?' } ] ] }
```

# 交互界面与可执行文件打包

## 基于网页的交互界面：[Gradio](#)

```
import gradio as gr

def response(message, history)
    # message: 用户的提问 (字符串)
    # history: 之前的对话上下文 (列表)
    generated_text = ...
    return generated_text # 返回回复 (字符串)

demo = gr.ChatInterface(
    fn=response,
    type="messages"
)
demo.launch()
```

## 将网页转换成桌面端窗口：[PyWebview](#)

```
import webview
window = webview.create_window(
    "RefSheet Chat", demo.local_url)
webview.start()
```

## 可执行文件打包：[PyInstaller](#)

```
pyinstaller app.py
```

参考：<https://github.com/whitphx/gradio-pyinstaller-example>

# 案例资源访问

GitHub:

[https://github.com/snowkylin/refsheet\\_chat](https://github.com/snowkylin/refsheet_chat)

在线Demo:

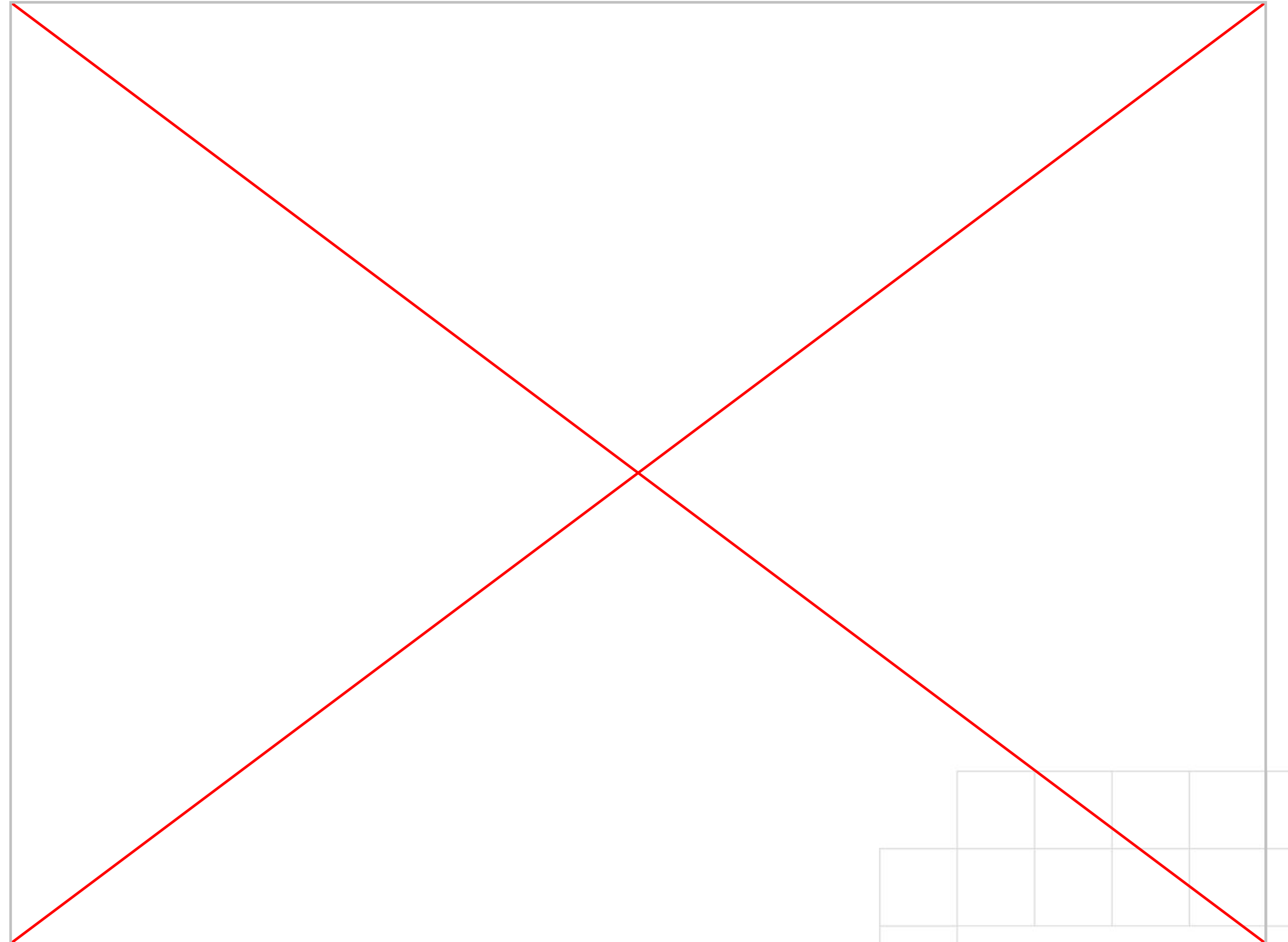
<https://refsheet.chat>

[https://huggingface.co/spaces/snowkylin/refsheet\\_chat](https://huggingface.co/spaces/snowkylin/refsheet_chat)

离线程序下载:

<https://refsheet.chat/local>

本幻灯片下载: <https://snowkylin.github.io/talks/>



# Thank You!



Xihan Li (李锡涵)  
University College London  
<https://snowkylin.github.io>

