

基于 LLM 的交互式多模态图像编辑系统的设计 与搭建

摘要

这是中文摘要的部分。

它可以拥有多段。这是中文摘要的部分。

它可以拥有多段。

如果你写的太长，甚至可以到第二页。

关键词 北京邮电大学 本科生 毕业设计 模板 示例

Design and Construction of Interactive Multimodal Image Editing System Based on LLM

ABSTRACT

This is ABSTRACT.

You can write more than one paragraph here.

If your abstract is too long, it will take up more pages.

KEY WORDS BUPT undergraduate thesis template example

目 录

第一章 绪论	1
1.1 项目背景	1
1.2 项目意义	2
1.3 项目内容	2
第二章 总体方案设计	4
2.1 GUI	4
2.2 middleware	4
2.3 Stable Diffusion	5
2.4 OpenAI	6
2.5 ChatGLM2-6B	6
第三章 GUI 的构建	8
3.1 图像自动遮罩与优化	8
3.1.1 图像自动遮罩	8
3.1.2 图像遮罩性能优化	9
3.1.3 对自动生成的遮罩进行优化	10
3.2 多模态	11
3.2.1 JSON 指令生成	11
3.2.2 JSON 指令校验	12
3.2.3 多指令处理	13
3.2.4 图像模型请求参数生成	13
3.3 图像修改建议	13
第四章 middleware 的构建	15
4.1 对多个平台的 API 进行配置和整合	15
4.2 使用 beego 框架提供 API 服务	15
第五章 LLM 的微调	16
5.1 LLM 微调数据集生成与性能评估方法	16
5.1.1 微调数据集生成	16
5.1.2 LLM 指令生成任务性能评估方法	17
5.2 ChatGLM2-6B 针对指令生成任务的微调	17
5.3 各个 LLM 在本任务下的性能评估	18

第六章 Stable Diffusion 及扩展的使用	20
6.1 Stable Diffusion API 的使用	20
6.2 ControlNet 的使用及效果	21
6.3 roop 的使用及效果	21
第七章 系统实现效果与使用	22
7.1 系统实现效果	22
7.2 系统使用方法	24
7.2.1 系统部署	24
7.2.2 GUI 使用说明	24
第八章 项目管理与维护	26
8.1 代码管理	26
8.2 自动化测试	26
8.3 持续集成与持续部署	27
第九章 总结及未来展望	29
9.1 总结	29
9.2 未来展望	30
参考文献	
致 谢	

第一章 绪论

1.1 项目背景

随着技术的迅速发展，图像生成技术在多个行业中发挥着越来越重要的作用，尤其是在媒体娱乐、数字营销和智能医疗等领域。然而，尽管其应用广泛，传统的图像编辑技术仍面临许多挑战，尤其是在交互性和生成图像质量上的限制。

传统图像编辑工具往往依赖于专业的技术知识和复杂的操作界面，这对于普通用户来说是一个较大的门槛。用户需要花费大量时间学习如何使用这些工具，这限制了工具的普及性和易用性。这些工具的交互性通常较差，不能很好地根据用户的具体需求进行灵活调整和响应。

为了克服这些挑战，人们正在探索利用深度学习技术来改善图像编辑系统。深度学习，特别是扩散模型，已经在图像生成领域显示出了巨大的潜力。这些技术能够学习大量图像数据，自动提取复杂的特征，并生成高质量的图像。此外，结合最新的 GPT4Turbo 等性能优异的大语言模型，可以进一步提升系统的交互性，实现更加自然和直观的用户界面。

图像生成模型是深度学习领域中一个活跃的研究方向，主要致力于通过机器学习技术生成高质量的图像。这些模型在多种应用场景中均有广泛应用，包括艺术创作、游戏开发、电影制作等。目前，图像生成模型主要包括生成对抗网络 (GANs)、变分自编码器 (VAEs) 和扩散模型等。

扩散模型是近年来发展起来的一种新型生成模型，与传统的生成对抗网络 (GANs) 和变分自编码器 (VAEs) 相比，扩散模型在生成图像的质量和多样性方面展现出了卓越的性能。其基本原理是模拟从高质量数据分布到高熵噪声分布的逐步转变过程，然后再逆向这一过程以生成新的数据。扩散模型的工作流程可以分为两个阶段：前向扩散过程和反向生成过程。在前向扩散过程中，模型逐渐将数据中的信息转化为噪声，这一过程通常通过向数据中逐步加入高斯噪声来实现。在反向生成过程中，模型则需要学习如何从噪声状态恢复出原始数据的分布，这一过程通常通过训练一个参数化的神经网络来完成，网络的目标是最小化原始数据与生成数据之间的差异。扩散模型的关键优势在于其生成的图像具有较高的质量和自然性，这是因为模型在生成过程中能够更好地控制噪声的去除过程，并逐步精细化图像的细节。此外，扩散模型在训练过程中相对稳定，不易出现生成对抗网络中常见的模式崩溃问题。

大语言模型是人工智能领域中的一项核心技术，主要用于处理和理解自然语言。这些模型通过学习大量文本数据，能够生成文本、回答问题、翻译语言等。随着算力的提升和语料的增加，大语言模型已经取得了显著的进步，并在多个应用场景中展现出了巨大的潜力。近几年，随着深度学习技术的发展，尤其是 Transformers 架构的提出，大语言模型的性能得到了质的飞跃。Transformers 架构能够有效处理长距离依赖问题，并在许多自然语言处理任务中设定了新的性能标准，GPT (Generative Pre-trained Transformer)、

BERT (Bidirectional Encoder Representations from Transformers) 等主流模型，通过大规模的语料进行预训练，对语言的深层次结构和语义的理解能力有了显著的提升，在应用方面展现出广泛的适用性，包括但不限于文本生成、对话系统、语言翻译、内容摘要和情感分析等。

通过图像生成模型与大语言模型的结合，可以创建一个更加灵活且易用性强的图像编辑系统。这样的系统不仅能够提供更加直观的编辑界面，降低用户的操作难度，还能够根据用户的描述自动生成或修改图像，极大地提升生成图像的质量和编辑效率。例如，用户可以通过简单的语言指令，如“增加图片亮度”或“改变背景为海滩”，直接与编辑系统交互，系统能够理解这些指令并即时作出响应。

通过整合深度学习和语言模型技术，我们有望构建出一个全新的交互式图像编辑系统。这种系统不仅能够提供更高质量的图像编辑结果，更能够提供给用户更加自然的交互方式，为该领域的专业人士和普通用户都带来更多的可能性。

1.2 项目意义

通过融合先进的图像生成技术与大语言模型，本项目希望能搭建一个基于 LLM 的交互式图像编辑系统，提供更为易用、精准且高效的图像编辑工具，提升媒体娱乐、数字营销以及智能医疗等多个行业的图像处理能力。通过实现更加智能化和用户友好型的编辑系统，为广大用户带来前所未有的图像编辑体验，并为促进技术创新做出贡献。这样的研究与开发，为图像编辑技术的未来提供了一种可能性和一条新的探索和实践路径。

1.3 项目内容

该项目是一个多模态交互式图像编辑系统，它主要实现了 GUI、middleware、以及对 Stable Diffusion 和 ChatGLM2-6B 模型的修改与适配。在整体结构上，GUI、middleware、以及模型修改之间的相互关系和数据流向见图 1-1。

项目基于 Stable Diffusion 的开源项目 stable-diffusion-webui 进行了扩展，增强了其功能。系统可以调用不同的 Stable Diffusion 模型并结合多个扩展后的功能来对图像进行精细的修改和调整。这一功能的实现大大增强了系统对图像处理的灵活性和多样性。

项目还包括了对 ChatGLM2-6B 模型的微调。通过利用专门为本项目的需求生成的微调数据集进行微调，进一步提升了语言模型处理特定任务的能力和准确性，且能够利用 API 调用这些经过微调的 ChatGLM2-6B 模型。

通过调用 OpenAI 的 API，本项目实现了多项功能：利用 GPT4V 生成关于图像修改的建议，使用 GPT3.5Turbo 来辅助生成用于微调大语言模型的数据集以及图像修改指令，以及在 Stable Diffusion 模型不可用时，使用 DALL-E2 作为替代模型来进行图像修改。这些功能为基于 LLM 的交互式图像编辑系统提供了强大的工具。

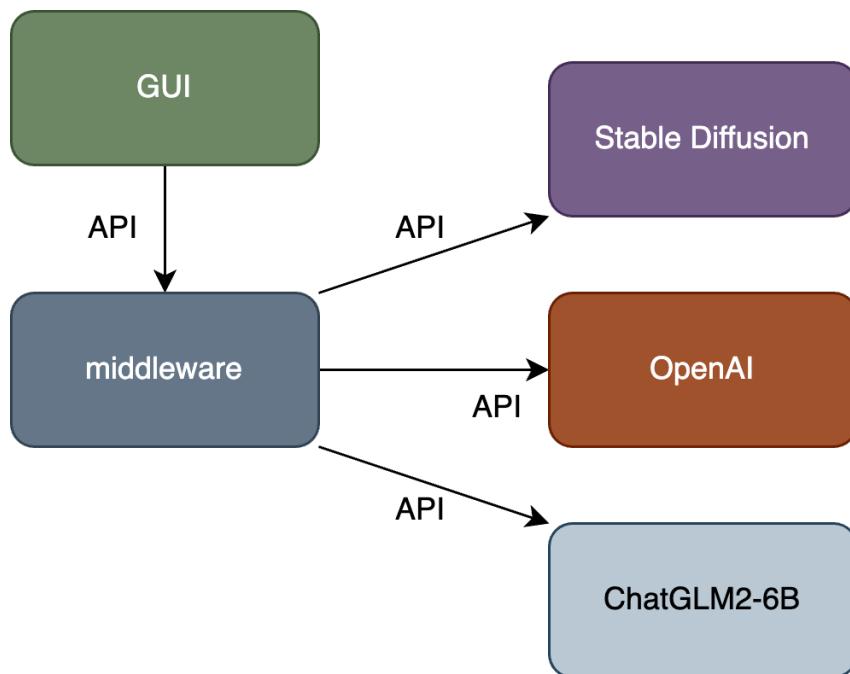


图 1-1

在 GUI 方面，本系统主要通过 Python 语言实现，构建了一个直观且用户友好的交互界面。该界面简洁易用，通过 API 调用向 middleware 发出请求，有效减轻了用户在执行计算密集型任务时对硬件资源的需求，从而降低了用户侧的使用门槛。

middleware 部分使用 Golang 语言构建了一个后端服务。这个服务不仅接入了 Stable Diffusion、ChatGLM2-6B、OpenAI 的 API，还将这些 API 进行了有效的整合，向 GUI 提供了一致风格的 API 接口。这种设计不仅提升了 GUI 调用多方 API 的便利性，也通过统一的配置和管理，极大地增强了系统的可维护性和稳定性。

第二章 总体方案设计

2.1 GUI

图形用户界面是现代软件项目中不可或缺的组成部分，它极大地提升了应用程序的可访问性和用户体验。GUI 通过可视化元素如按钮、图标和菜单等，允许用户以直观的方式与系统进行交互，简化了操作过程并降低了用户的使用门槛。本项目所构建的 GUI 为用户提供了一个交互方式简单且功能强大的图形用户界面，通过图形化的方式展示信息和选择，使得用户能够通过简单的点击或触摸来执行命令或更改设置。它减少了对复杂指令的依赖，从而使用户可以通过直观的方式与系统进行交互，对于非技术用户尤为重要。GUI 虽然承担计算任务较少，但却是承载本项目结构与逻辑的关键部分。通过使用符合规则的指令作为中枢，GUI 打通了大语言模型和图像生成模型之间的壁垒，使基于 LLM 的创新交互式图像编辑系统成为可能。GUI 的模块构成如表 2-1：

表 2-1 GUI 模块

模块	描述
BaseImage	接受上传的原始图片并预览
EditedImage	预览修改后的图片
Operation Board	执行指令
Settings	对系统进行设置
Chat	与大语言模型交互的聊天界面
Edit Image	对图像进行自定义遮罩和换脸等操作
Auto	执行自动化操作
Manual	系统使用说明

用户首先上传需要修改的图片，然后可在 Chat 模块中选择不同的大语言模型进行交互并得到相应的指令，最后在 Operation Board 模块中选择指令执行或一键全部执行。如果对自动生成的遮罩不满意，可在 Edit Image 中对遮罩进行修改。

在 Auto 模块中，用户可通过选择多张图片批量生成满足微调大语言模型微调所需的数据。其会循环地从给定的图片集中随机选择图片继续分割，将分割后的结果和特定的 prompt 通过 GPT3.5Turbo 生成对应的修改建议，再将分割的结果、生成的建议通过 GPT3.5Turbo 生成指令。

2.2 middleware

middleware 是项目的核心组件，通过整合多个平台的 API，为 GUI 提供统一的、简单易接入的 API 服务。其主要特点包括但不限于：1. API 整合：middleware 整合了多个

平台的 API，包括图像生成模型、语言模型等，使得 GUI 可以通过统一的接口调用不同功能模块；2. 统一风格：middleware 设计了统一的 API 风格和路由规范，使得 GUI 可以轻松使用 API 服务，提高开发效率；3. 简单易接入：middleware 提供了简单易用的 API 服务，GUI 无需关注具体实现细节，只需按照简单的请求规范调用 API 即可；4. 稳定性和可靠性：middleware 基于 Golang 语言实现，具有高效的并发处理能力和稳定的运行性能，保证了 API 服务的稳定性和可靠性；5. 易于维护：middleware 采用了 Beego 框架，具有清晰的代码结构和模块化设计，易于维护和扩展，保证了项目的长期可持续发展。其向 GUI 提供的主要 API 如表 2-2 所示：

表 2-2 middleware 主要 API

API	路由	描述
PostSDTxt2Img	/v1/pics/txt2img	通过 Stable Diffusion 模型生成图片
PostSDImg2Img	/v1/pics/img2img	通过 Stable Diffusion 模型修改图片
PostDALLE2Edit	/v1/pics/openai/img2img	通过 DALL-E2 模型修改图片
GetLoras	/v1/pics/loras	获取可用的 LoRa 模型列表
PostHuggingFaceImgSegment	/v1/pics/huggingface/segment	获取图像分割结果
PostGPT3Dot5Turbo	/v1/chat/gpt3dot5turbo	调用 GPT3.5Turbo
PostGPT4Turbo	/v1/chat/gpt4	调用 GPT4Turbo
PostChatGLM2_6B	/v1/chat/glm2_6b	调用 ChatGLM2-6B
PostGPT4V	/v1/chat/gpt4v	调用 GPT4V

2.3 Stable Diffusion

Stable Diffusion 是一种基于扩散模型的深度学习图像生成模型，它能够根据文本描述生成高质量的图像。这个模型采用了条件生成技术，允许用户通过文本指令来引导图像的生成，使其在艺术创作、媒体娱乐、广告和数字营销等多个领域具有广泛的应用。其核心在于从噪声逆向还原生成高质量的图片。

sd-webui-controlnet¹ 使用了 ControlNet^[1] 的原理，旨在增强原有 Stable Diffusion 模型的图像生成控制能力。通过集成一个额外的控制网络（ControlNet），允许用户精确指导图像的具体内容，显著提升了生成图像的细节质量和一致性。

sd-webui-roop² 基于 DeepFake^[2]，允许用户在图片中进行面部替换，简化了面部交换的过程，无需训练特定的模型，大大降低了使用复杂度。

由于本项目对于图像生成模型的要求较高且需求复杂，为了便于结合 Stable Diffusion 模型和其他前沿研究成果及开源社区项目，本项目在构建 Stable Diffusion 模块时以

¹<https://github.com/Mikubill/sd-webui-controlnet>

²<https://github.com/s0md3v/sd-webui-roop>

开源项目 stable-diffusion-webui¹为基础，结合 sd-webui-controlnet 和 sd-webui-roop 等扩展，通过 API 为 middleware 提供服务。

2.4 OpenAI

OpenAI²是一家成立于 2015 年的非盈利人工智能研究机构，由 Elon Musk、Sam Altman 等人联合创立，总部位于美国旧金山。该机构的主要目标是推进人工智能的安全发展，并确保 AI 技术的利益能够普及全人类，其在 AI 领域具有多项突破性成就，其中最著名的产品当属 GPT（Generative Pre-trained Transformer）系列，这是一系列的自然语言处理模型，能够生成连贯和相关性强的文本。GPT 模型已经更新多代，目前最新的为 GPT-4。此外，OpenAI 还推出了 DALL·E 系列，这是一种基于 GPT 的图像生成模型，能够根据文本描述生成图像。

本项目使用了 OpenAI 的 GPT3.5Turo、GPT4Turbo、GPT4V、DALL-E2 等模型，通过 API 调用 OpenAI 的模型。

GPT-3.5 Turbo 是 OpenAI 开发的一款先进的自然语言处理模型，属于 GPT-3 系列的增强版本。这个模型在处理大量文本和生成文本方面表现出色，适用于聊天机器人、内容创作、文本摘要等应用。GPT-3.5 Turbo 优化了处理速度和响应时间，提高了交互效率。

GPT-4^[3]是 GPT-3 的后续版本，代表了最新一代的语言预测和生成技术。它在模型结构和训练数据量上进行了大幅扩展，使其能够更准确地理解和生成复杂的文本。GPT-4 在理解上下文、维持一致性以及生成更自然的语言方面具有显著优势。

GPT-4V 是 GPT-4 的一个特殊版本，专门优化用于视觉任务，比如图像标注、视觉问答等。这个版本结合了文本和视觉处理能力，能够更好地理解和生成与图像相关的文本内容。

DALL-E 2 是一个先进的图像生成模型，专门设计用来创建新颖的图像和艺术作品。它可以根据用户提供的文本描述生成详细、高质量的图像。DALL-E 2 的核心优势在于其创造力和多样性，能够在遵循描述的同时，创造出独特和富有创意的视觉内容。

2.5 ChatGLM2-6B

ChatGLM2-6B 是由清华大学开发的第二代开源双语(中英)对话模型，基于 ChatGLM-6B 的成功经验并引入了多项新特性和性能提升。这款模型经过大规模预训练，实现了显著的性能提升，并在多个数据集上表现出色。ChatGLM2-6B 支持更长的对话上下文，并提高了推理速度和降低了显存占用，使得即使在资源有限的环境下也能有效运行。

本项目使用 ChatGLM2-6B 模型，使用开源项目 LLaMA-Factory³，利用本项目提供

¹<https://github.com/AUTOMATIC1111/stable-diffusion-webui>

²<https://openai.com>

³<https://github.com/hiyouga/LLaMA-Factory>

的数据自动生成功能所生成的数据集，使用 LoRa^[4] 方法对模型进行微调以在本项目所需的任务中获得更佳的表现。微调后的模型通过 fastapi 提供 API 服务。

第三章 GUI 的构建

在本项目中，GUI 作为关键组成部分，虽仅承担极少的计算任务，却在结构与逻辑上起着至关重要的作用。它通过使用标准化的指令连接大语言模型和图像生成模型，实现了基于 LLM 的创新交互式图像编辑系统。用户首先上传原始图片至 BaseImage 模块进行预览，之后可通过 Chat 模块与大语言模型进行交互，获取编辑指令。用户可以在 Operation Board 模块中选择单独或批量执行这些指令。若对自动生成的遮罩不满意，可在 Edit Image 模块中手动调整。此外，Auto 模块允许用户批量处理多张图片，自动生成数据以微调大语言模型。该过程包括图片的自动分割、利用 GPT3.5Turbo 生成编辑建议及相应的执行指令。这样的设计不仅提升了系统的效率，也优化了用户的交互体验。

3.1 图像自动遮罩与优化

由于本项目需要提供对图像进行部分修改的功能，所以需要在使用图像生成模型进行图像编辑时需要提供一个遮罩以明确需要修改的部分和不需要修改的部分。为了自动生成符合要求的遮罩，本项目借助图像分割和大语言模型的辅助，可通过两种方式生成自动遮罩：基于关键词对自动生成遮罩和基于已给出的点自动填充生成遮罩。两种方法都会首先使用图像分割模型对图像进行分割（如图 3-1），然后根据给出的要求对相应的部分进行遮罩生成原始的遮罩。受制于图像分割模型在边缘上的表现并不理想，需要对特定的分割部分进行处理以提高遮罩的质量，因此最后会通过本项目设计的优化算法生成最终的遮罩。



图 3-1 图像分割结果：(a)原始图像，(b)分割结果

3.1.1 图像自动遮罩

基于关键词自动生成遮罩的方法会根据关键词和图像分割结果生成自动原始的遮罩，该功能会遍历每个给出的关键词，若关键词与分割结果之一吻合，则会对相应的分割区域进行遮罩，生成原始的遮罩（如图 3-2）。

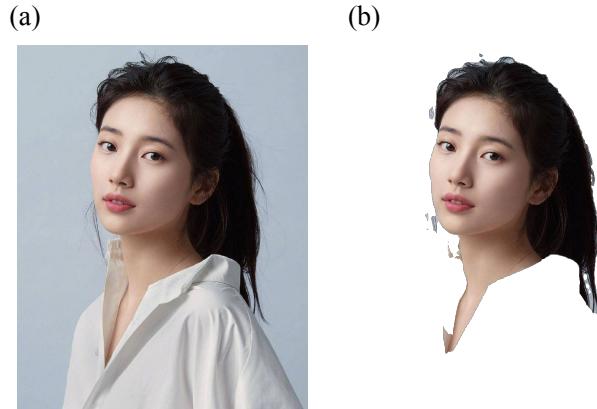


图 3-2 关键词自动生成遮罩结果: (a)原始图像, (b)keywords=['Background', 'Upper-clothes', 'Dress', 'Right-arm'] 得到的遮罩

基于已给出的点自动填充生成遮罩的方法会根据在图片中标记的点和图像分割结果生成自动原始的遮罩, 该功能会遍历每个给出的点, 将该点所在的部分全部进行遮罩, 最后生成原始的遮罩 (如图 3-3)。



图 3-3 基于已给出的点自动填充生成遮罩: (a)标记后的图像, (b)生成的遮罩

3.1.2 图像遮罩性能优化

LRU(Least Recently Used)缓存是一种常用的缓存淘汰算法, 用于在有限的缓存空间中管理数据。它的核心思想是优先淘汰最长时间未被使用的数据项。functools.lru_cache 是 python 标准库中 functools 模块提供的一个装饰器, 它实现了 LRU 缓存机制。该装饰器可以非常方便地被添加到任何想要进行缓存的参数可哈希的函数上, 自动地保存最近执行的函数调用结果并在后续相同的调用中直接返回缓存的结果, 避免重复计算的开销。

由于在本项目中图像遮罩存在一张图片多次调用的特点, 本项目使用了 LRU 缓存实现性能优化。由于 python 中 PIL.Image.Image 对象不可哈希, 缓存分割结果时将图像转为 base64 字符串进行映射。

3.1.3 对自动生成的遮罩进行优化

由于分割模型性能的限制，生成的原始遮罩可能在某些细节上表现不佳而影响图像编辑模型的结果，因此设计了一个算法对自动生成的遮罩进行优化。该算法可以根据配置文件的设置，对特定的未被遮罩的部分在遮罩的边缘进行收缩。

算法 1 遮罩优化算法

输入： 原始遮罩 $OriginMask$ ，图像分割结果 $SegmentResult$ ，配置文件 $Config$

输出： 优化后的遮罩 $OptimizedMask$

- 1: 获取遮罩与非遮罩的描边得到像素 $EdgePixels$
 - 2: 从配置文件和图像分割结果获取 $ConfigPixels$
 - 3: 仅保留出现在 $EdgePixels$ 中的 $ConfigPixels$
 - 4: **for** $pixcel$ in $ConfigPixels$ **do**
 - 5: Apply MinFilter Kernel(in $Config$) in $OriginMask[pixcel]$
 - 6: **end for**
 - 7: 得到优化后的遮罩 $OptimizedMask$
-

由于该算法仅会对遮罩边缘上的像素进行卷积且在设计时充分考虑到了内存中像素的存储顺序的原因，虽然需要复杂的处理过程，但经过多次的迭代后算法的时间复杂度降低至 $O(mnr)$ (m, n 表示图片的长宽， r 表示设置的优化强度)。算法实现的效果如图 3-4 所示，可见在发丝附近遮罩的质量得到了明显的改善。

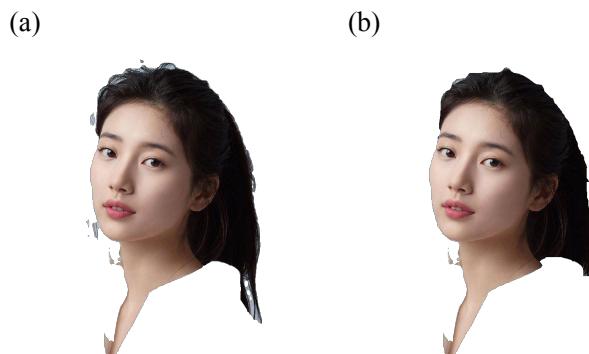


图 3-4 遮罩优化结果：(a)原始遮罩，(b)算法优化后的遮罩

3.2 多模态

如何打通大语言模型和图像生成模型是本项目的关键。本项目通过特定的 prompt 和图像分割结果，使用大语言模型生成 JSON 格式的指令并校验，并支持多轮对话。用户可有选择性地执行生成的指令或执行全部指令。系统首先会按照给定的规则对指令进行预处理和排序，然后通过指令生成请求参数来调用图像生成模型。多模态任务的实现方式如图 3-5。

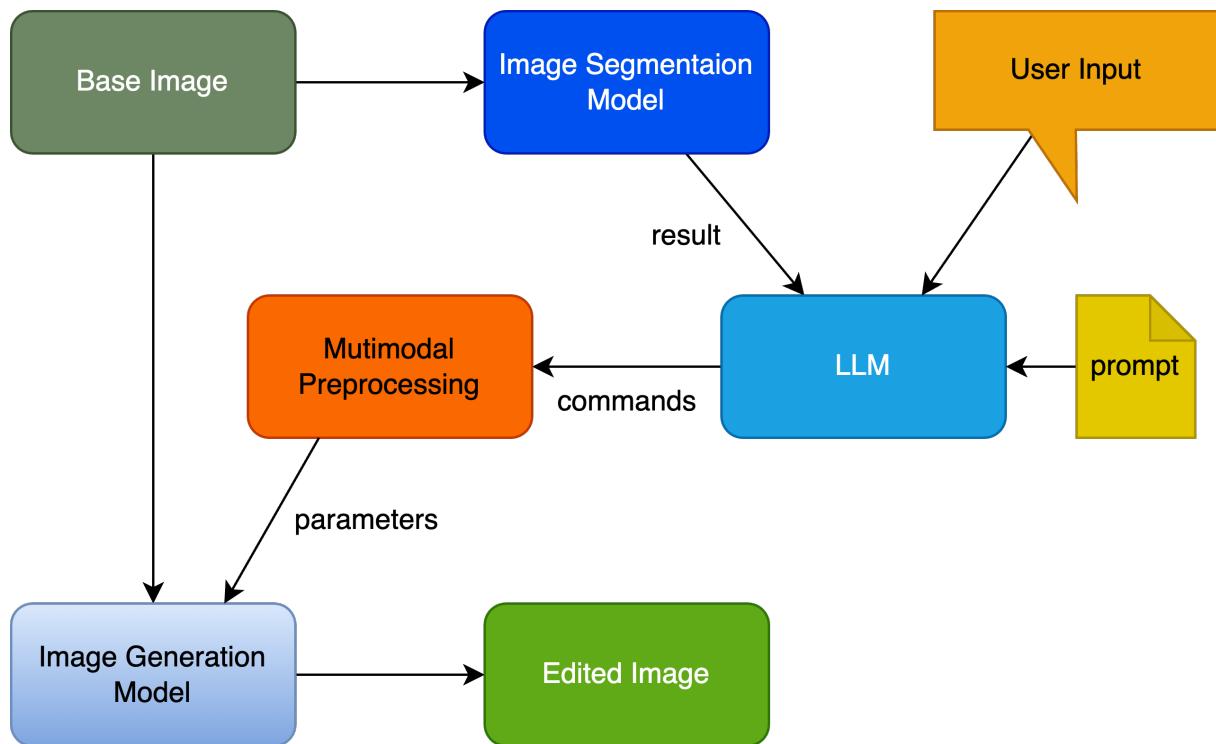


图 3-5

3.2.1 JSON 指令生成

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写，同时也易于机器解析和生成。它基于 JavaScript 的一个子集，但独立于语言，被广泛应用于许多编程语言中。JSON 主要用于网络应用间浏览器与服务器之间的数据传输。在 JSON 中，数据以键值对的形式存在，可以表示数组、布尔值、数字、对象或字符串。由于其简洁性和易于交互的特性，JSON 已成为 Web 应用中数据交换的主流技术。由于 JSON 应用范围广且大语言模型 JSON 处理能力较强，本项目采用此格式来承载大语言模型和图像生成模型的联系。

通过特定的 prompt 和图像分割结果以及用户输入的修改意图，本项目可使用 GPT3.5Turbo、GPT4Turbo、微调后的 ChatGLM2-6B 生成 JSON 指令。例：当图像分割结果为 *Background, Hair, Upper clothes, Dress, Face, Right-arm*，用户输入为“将背景更换为蓝天白云，将衣服更改为白色的 T-shirt”时，生成的 JSON 指令如代码 3-1 所示。

代码 3-1 生成的指令

```

1  [
2    {
3      "command" : "change",
4      "paras" : [ ["Background", "Upper-clothes"] , "blue\u2022sky,\u2022white\u2022T-shirt"]
5    }
6  ]

```

3.2.2 JSON 指令校验

由于大语言模型生成指令不稳定，需要对生成指令的合规性进行校验。校验规则存储为一个 JSON 文件，以修改非面部和面部的指令为例，校验规则如代码 3-2 所示：

代码 3-2 指令校验规则

```

1  {
2    "face": {
3      "paras_type": [
4        "<class\u2022'str'\u2022>"
5      ],
6      "paras_enum": null,
7      "paras_min": null,
8      "paras_max": null,
9      "combine": true,
10     "priority": 1
11   },
12   "change": {
13     "paras_type": [
14       "<class\u2022'list'\u2022>",
15       "<class\u2022'str'\u2022>"
16     ],
17     "paras_enum": null,
18     "paras_min": null,
19     "paras_max": null,
20     "combine": true,
21     "priority": 1
22   }
23 }

```

指令校验的算法如算法-2 所示。

算法 2 JSON 指令校验算法

输入：待校验的指令 *JsonCommands*、校验规则 *Rules*

输出：校验后的指令 *ValidJsonCommands*

```

1: for Command in JsonCommands do
2:   if Command satisfy Rules then
3:     Add Command to ValidJsonCommands
4:   end if
5: end for

```

3.2.3 多指令处理

由于一次执行可能会涉及到多个指令，会遇到指令重复、指令优先性等问题，所以会对需要执行的指令进行合并与排序。指令合并的算法如下：

算法 3 多指令处理算法

输入：待处理的指令 *OriginCommands*、指令合并规则 *Rules*、指令优先性 *Priority*

输出：处理后的指令 *ProcessedCommands*

```

1: for Command in OriginCommands do
2:   if Same Command type already in ProcessedCommands then
3:     Combine Command to the same one in ValidJsonCommands use Rules
4:   end if
5: end for
6: 根据 Priority 对 ValidJsonCommands 进行排序

```

3.2.4 图像模型请求参数生成

图像模型请求参数生成较为复杂，对于某个参数，其可能来源于 GUI 中可修改的设置，可能来源于指令，可能来源于模版，否则设置为默认参数。由于每个参数来说，其来源的优先性可能不一致，因此设计了算法-4 来生成图像模型请求参数。

3.3 图像修改建议

本项目提供了根据图像自动生成图像建议的功能。由于传统的大语言模型只能接受文本输入，因此本项目采用了 GPT4V 来自动生成图像修改建议。

GPT-4V 是由 OpenAI 开发的多模态大型语言模型，是 GPT 系列基础模型的第四代。该模型具有视觉能力，可以将图片作为输入，进行各种任务，例如描述图片中的幽默、总结截图文本、回答包含图表的考试题目等。

用户通过 GUI 界面的 Advise 按键，可以生成建议并将其转换为指令。

算法 4 图像模型请求参数生成算法

输入： 指令 *Command*、设置 *Settings*、默认参数 *Default*、参数来源优先性 *PriorityRules*

输出： 图像模型请求参数 *Parameters*

```
1: for ParaKey in Parameters do
2:   Get Template from Command
3:   if ParaKey found in Command or Settings or Template then
4:     Choose the highest priority source use PriorityRules[ParaKey]
5:   else
6:     Set this parameter to Default[ParaKey]
7:   end if
8: end for
```

第四章 middleware 的构建

在本项目中, middleware 作为核心组件, 通过整合来自不同平台的 API, 为 GUI 提供了统一且易于接入的 API 服务。其整合了多个图像生成和大语言语言模型, 通过统一的接口设计, 使得 GUI 能够方便地调用所需的功能。此外, middleware 采用了 Golang 语言和 Beego 框架, 不仅保证了 API 服务的高并发处理能力和稳定性, 还通过模块化的设计提高了系统的可维护性和可扩展性。主要 API 服务包括 Stable Diffusion 和 DALL-E2 模型、图像分割模型, 以及多种大语言模型。这样的架构设计不仅优化了开发效率, 也确保了系统的稳定运行和长期发展。

4.1 对多个平台的 API 进行配置和整合

middleware 通过整合不同平台如图像生成模型、语言模型等的 API, 使得 GUI 开发者可以通过一个统一的接口调用多种功能。这种整合不仅包括 API 的聚合, 还涉及统一 API 调用的风格和路由规范, 不仅保证了 API 服务的高稳定性和可靠性, 还便于日后的维护和扩展。例如, 无论是调用 ChatGLM2-6B 模型还是多种 GPT 模型, GUI 都能通过相同的结构化请求方式访问不同的服务。

4.2 使用 beego 框架提供 API 服务

beego 是一个使用 Go 语言开发的开源 Web 框架, 它支持快速开发各种应用程序, 包括 API、Web 应用和后端服务。该框架设计灵感来源于 Tornado、Sinatra 和 Flask, 但结合了 Go 的特定特性如接口 (interface) 和结构体嵌入 (struct embedding), 从而提供了高效的性能和简便的操作。

本项目使用 beego 框架构建了 middleware 组件提供了多个 API 服务, 以便 GUI 可以高效地使用各种模型和工具。*PostSDTxt2Img* 和 *PostSDImg2Img* 是通过 Stable Diffusion 模型来生成或修改图像的 API, 这使得用户能够通过简单的 API 调用, 进行复杂的图像生成和编辑操作。*GetLoras* 这个 API 用于获取 Stable Diffusion 模型可用的 LoRa 模型列表。*PostDALLE2Edit* 提供了利用 DALL-E2 模型修改图片的功能, 这进一步扩展了图像处理的能力, 在 Stable Diffusion 模型不可用时作为替代。*PostHuggingFaceImgSegment* API 通过可在部署在本地的分割模型不可用时通过调用 Hugging Face 上的图像分割模型 API 来实现图像分割。在文本处理方面, *PostGPT3Dot5Turbo*、*PostGPT4Turbo* 和 *PostGPT4V* 等 API 利用 OpenAI 的不同版本 GPT 模型来处理指令理解和生成任务, *PostChatGLM2_6B* 可调用微调后的 ChatGLM2-6B 模型来进行指令生成。

第五章 LLM 的微调

大语言模型（LLM）的微调是一种在预训练模型基础上通过特定数据集进一步训练的过程，用于优化模型在特定任务或场景中的表现。特别是对于参数量较小的 LLM，微调不仅可以提升其性能，还能增强其针对具体任务的适应性和泛化能力。微调的一个主要优势是性能提升，即使是较小的模型，通过针对性的微调，可以在特定任务上实现甚至超过大型模型未经微调时的性能，可见微调能够根据具体需求调整模型的行为，使其更加专注于特定的输出目标。微调技术提供了一种有效途径，通过少量的定制化数据提升模型的应用性能，特别是在参数量较小的模型中，这种优势尤为显著。

原始的 ChatGLM2-6B 模型已经通过大规模数据预训练，具备了强大的语言理解和生成能力，但受制于参数量较小，其在本项目需要高精准度的指令生成任务上表现不佳，因此本项目通过自动生成并进行筛选的高质量指令生成微调数据集对 ChatGLM2-6B 进行微调以进一步提升模型在指令生成任务的表现。

本项目利用开源项目 LLaMA-Factory¹和本项目中自动生成的指令生成任务数据集对 ChatGLM2-6B 模型进行微调。

5.1 LLM 微调数据集生成与性能评估方法

5.1.1 微调数据集生成

在大语言模型（LLM）的微调过程中，数据集的功能和作用至关重要。微调数据集不仅提供了模型训练所需的具体数据，还直接影响了模型微调后的性能和适应性。对于参数量较小的模型，高质量的微调数据集尤其重要，因为这些模型通常缺乏足够的参数量来从大规模数据中学习复杂的特征。通过高质量的微调数据集，参数量较小的大语言模型可以有效地提高这些模型的学习效率和最终性能。构造高质量的微调数据集涉及两个关键步骤：数据收集和数据预处理。数据收集需要确保获得足够多的、具有代表性的数据，这些数据能够覆盖模型在实际应用中可能遇到的各种情况。数据预处理是将收集到的原始数据转换成模型可以直接处理的格式，这可能包括数据清洗、特征提取、标签编码等。

由于没有适用于本任务的开源数据集，本项目尝试建立一个自动化工作流程，通过利用 ATR Dataset²调用多个模型和一定的校验规则来生成所需的数据并整合为一个数据集。数据集生成的流程如图 5-1。同时，在系统使用过程中产生的数据可设置是否保存，这些数据也会在运行数据集生成脚本添加到数据集中。

通过建立的自动化工作流程，本项目创建了一个包含 3000 个样本的数据集，数据集中的每一项数据都经过合法性校验以保证数据集的质量。数据集的格式满足 LLaMA-Factory 的要求并可使用此数据集对 ChatGLM2-6B 进行微调。

¹<https://github.com/hiouga/LLaMA-Factory>

²<https://github.com/lemondan/HumanParsing-Dataset>

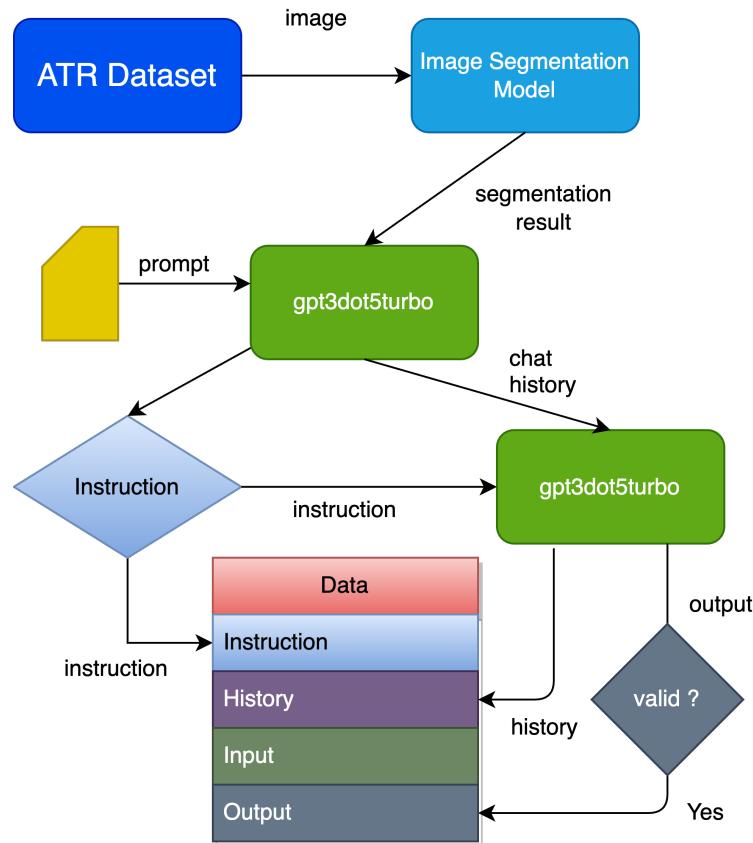


图 5-1

5.1.2 LLM 指令生成任务性能评估方法

性能评估是大语言模型（LLM）开发和应用的关键环节，尤其是在模型的实用性和可靠性方面。有效的评估方法可以帮助研究者和开发者了解模型在特定任务上的表现，从而进行进一步的优化和调整。

由于需要一种直观的性能评估方法来对不同的大语言模型在指令生成任务上的性能进行评估，本项目采用生成指令的合法率对不同的大语言模型在指令生成任务上的性能进行评估。

5.2 ChatGLM2-6B 针对指令生成任务的微调

LoRA: Low-Rank Adaptation of Large Language Models^[4] 这篇论文提出了一种新颖且高效的 LoRa 微调方法，用于微调大型预训练语言模型以适应特定任务。传统的微调方法往往需要重新训练模型的所有参数，而全参数训练的方法在模型参数规模庞大时需要巨大的量。LoRA 保持预训练模型权重不变，通过仅修改模型参数的一个小子集来进行微调，在模型的每层插入可训练的秩分解矩阵，只优化代表适应模型所需的最小可能变化的秩分解矩阵，大大减少了显存和计算开销。与全参数微调相比可训练参数的数量显著减少，并将显存需求减少了三倍。

本项目使用生成的指令生成任务数据集结合 LoRa 方法，通过开源项目 LLaMA-

Factory¹对 ChatGLM2-6B 模型进行微调。LoRA 微调在大语言模型上的训练损失随训练步数变化的情况如图 5-2 所示。从图中可以看出，最初损失值很高，但随着训练步数的增加，损失值迅速下降，特别是在前 50 步之内下降最为显著。在经过约 50 步之后，损失下降的速度开始放缓，但仍然持续下降，表明模型继续从训练数据中学习。在约 200 步之后，损失曲线趋于平缓，说明模型已经接近收敛，额外的训练步骤在减少损失方面的效果变得有限。图中还展示了一个平滑处理的损失曲线，更清晰地显示了训练过程的整体趋势，而不是每一步的波动。

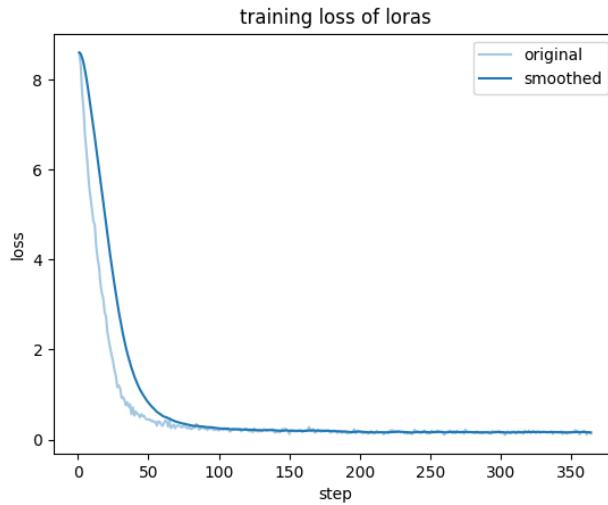


图 5-2 step-loss 图

5.3 各个 LLM 在本任务下的性能评估

结合本项目的 LLM 指令生成任务性能评估方法，本项目对不同的大语言模型进行了指令生成任务性能评估。各个模型的性能表现如表 5-1 和图 5-3 所示。

可观察到原始的 ChatGLM2-6B 模型在未经过微调时，在指令生成任务中的合格率仅为 6%。这一低下的性能表现暴露了 ChatGLM2-6B 模型在没有针对性训练的情况下难以完成需要高精准度指令生成任务。通过使用 LoRA 方法进行微调，模型性能随着微调步骤的增加显著提升。当微调步骤增至 80 步时，合格率提升至 77.8%；而在经过 200 步的微调后，合格率达到 95.8%，并在 300 步训练后稳定在 96% 左右。与 ChatGLM2-6B 相比，GPT3.5Turbo 和 GPT4Turbo 在无需额外训练的情况下即可达到分别为 97% 和 99% 的高合格率。

¹<https://github.com/hiyouga/LLaMA-Factory>

表 5-1 LLM 指令生成性能

模型	测试样本数	合格率
GPT3.5Turbo	100	97%
GPT4Turbo	100	99%
GhatGLM-6B(Origin)	100	6%
GhatGLM-6B(LoRa trained 50 steps)	1000	19.1%
GhatGLM-6B(LoRa trained 60 steps)	1000	43.7%
GhatGLM-6B(LoRa trained 80 steps)	1000	77.8%
GhatGLM-6B(LoRa trained 100 steps)	1000	88.1%
GhatGLM-6B(LoRa trained 150 steps)	1000	93.9%
GhatGLM-6B(LoRa trained 200 steps)	1000	95.8%
GhatGLM-6B(LoRa trained 250 steps)	1000	95.2%
GhatGLM-6B(LoRa trained 300 steps)	1000	96.7%

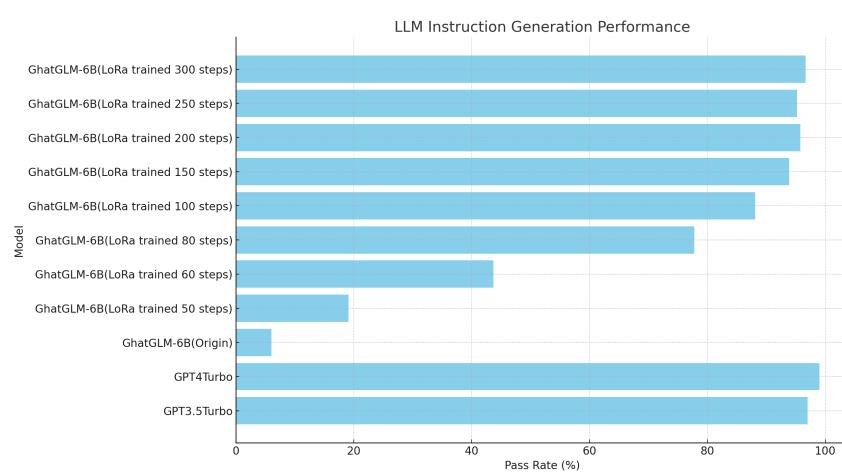


图 5-3 LLM 指令生成性能

第六章 Stable Diffusion 及扩展的使用

6.1 Stable Diffusion API 的使用

本项目通过将开源项目 stable-diffusion-webui¹部署在揽睿星舟机器学习平台²，通过 API 来调用 Stable Diffusion 模型，主要使用的参数见表 6-1。

表 6-1 主要使用的 Stable Diffusion webui img2img API 参数

参数	描述	形式
prompt	输出图像的期望修改或主题	str
negative_prompt	生成图像中应避免的内容	str(base64 Image)
mask	选择性编辑或生成的区域的图像遮罩	str
inpainting_fill	修补时的填充方法	int
inpaint_full_res	是否在图像的全分辨率下应用修补	bool
inpaint_full_res_padding	使用全分辨率时修补区域周围的填充	int
inpainting_mask_invert	是否反转修补遮罩	bool
mask_blur	遮罩边缘的模糊量	float
denoising_strength	去噪强度	float
sampler_index	采样算法	str
seed	初始化随机数生成器的值	int
steps	生成图像时的步骤数	int
width	输出图像的宽度	int
height	输出图像的高度	float
cfg_scale	输入提示的权重	float
restore_faces	是否修复生成图像中的面部	bool
alwayson_scripts	插件参数	dict

¹<https://github.com/AUTOMATIC1111/stable-diffusion-webui>

²<https://www.lanrui-ai.com>

6.2 ControlNet 的使用及效果

sd-webui-controlnet¹是一个用于 stable-diffusion-webui 的扩展，允许用户通过添加额外的条件来控制扩散模型的行为，从而增强生成图像的精确度和控制性。这一扩展可以实时添加到原始的 Stable Diffusion 模型中，不需要进行合并处理。本项目使用该插件以保持原始图像的主要轮廓和布局不受改变，其实现的效果如图 6-1。



图 6-1 ControlNet 效果：(a)原始图像，(b)未使用 ControlNet，(c)使用 ControlNet

6.3 roop 的使用及效果

roop²是一个用于 stable-diffusion-webui 的扩展，提供面部替换的功能。这个扩展需要一张原始图像和目标图像，并能将原始图像的面部替换到目标图像上。其实现的效果如图 6-2。



图 6-2 roop 效果：(a)原始图像，(b)目标图像，(c)结果

¹<https://github.com/Mikubill/sd-webui-controlnet>

²<https://github.com/s0md3v/sd-webui-roop>

第七章 系统实现效果与使用

7.1 系统实现效果

系统实现了一个简单易用的 GUI，可通过特定端口进行访问。GUI 的总体效果如图 7-1 所示。

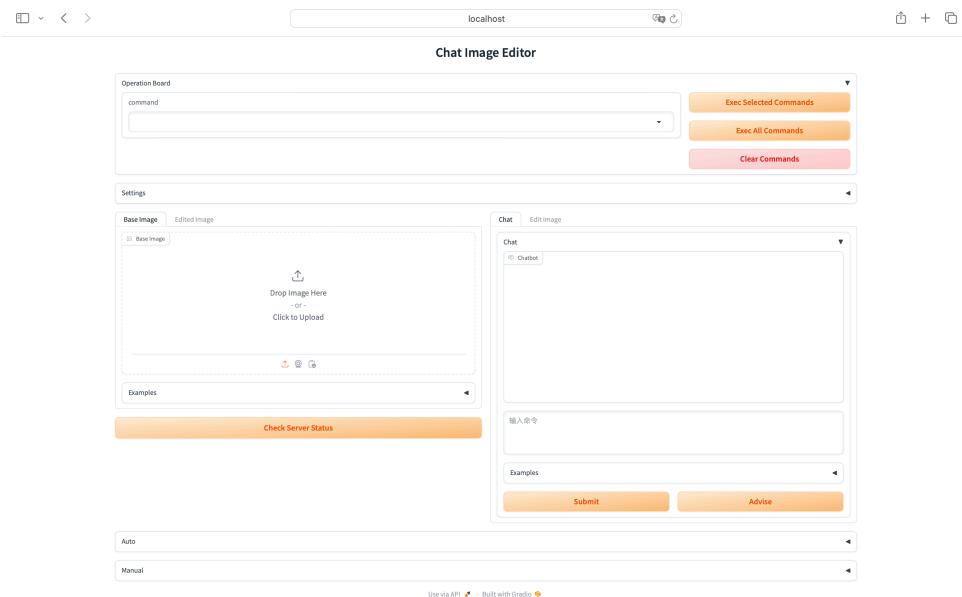


图 7-1 GUI 总体效果

可使用 GUI 的 Auto 模块进行 LLM 微调数据生成和 LLM 指令生成任务性能测试，其实现效果如图 7-2 所示。



图 7-2 Auto 模块

可使用 GUI 的 Manual 模块查看 GUI 使用方法，其实现效果如图 7-3 所示。

使用 GUI 进行基于 LLM 的交互式图像编辑时，系统主要模块效果如图 7-4 所示。

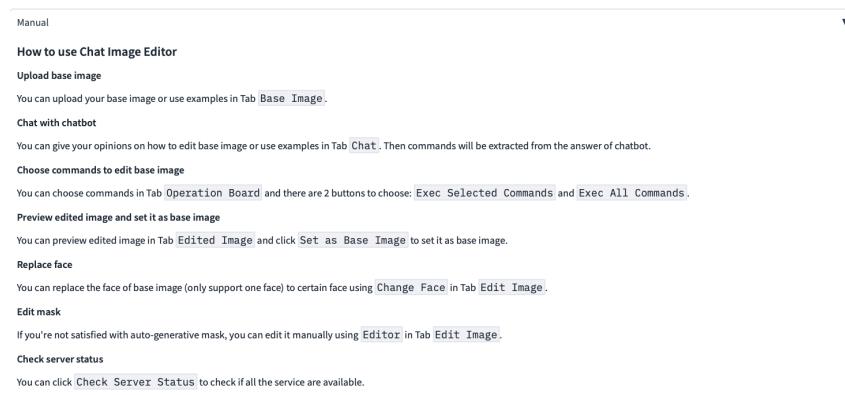


图 7-3 Manual 模块

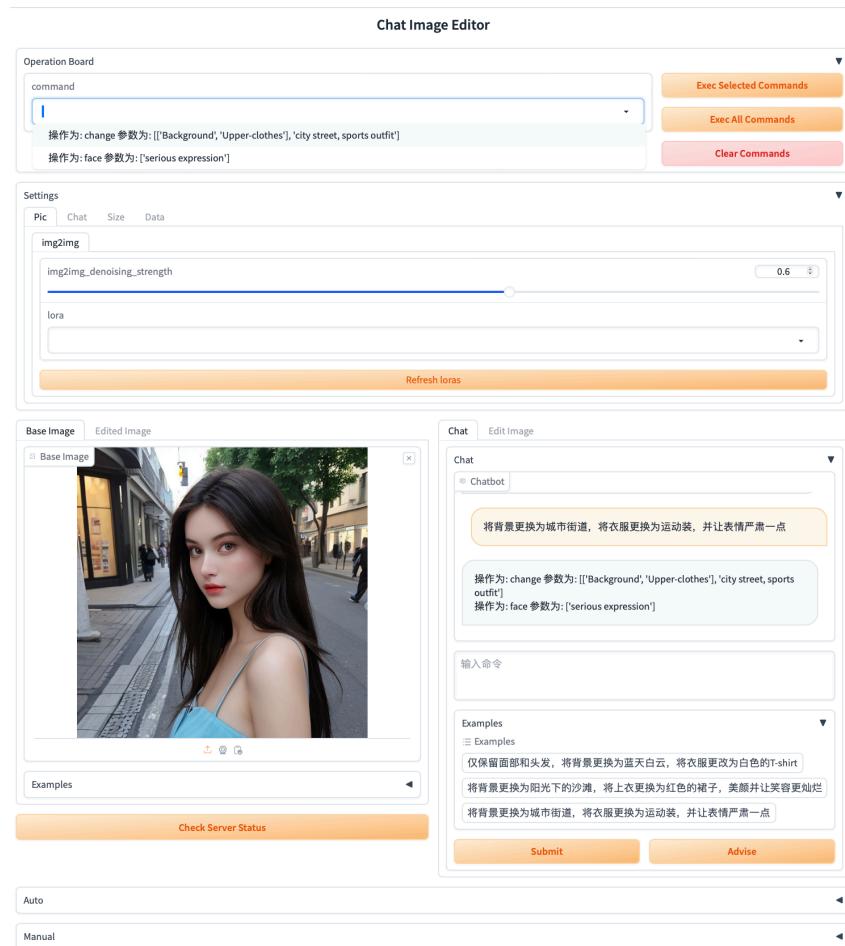


图 7-4 交互式图像编辑

7.2 系统使用方法

7.2.1 系统部署

本项目提供了多种常用的项目部署方式，包括使用 Docker、Kubernetes，以适应不同的用户需求和操作环境。如果选择通过 Docker 进行部署，用户可以根据需求选择不同的镜像。通过运行命令 `docker run --name multimodal -p 27777:80 binciluo/multimodal:latest` 可以在本地部署图像分割模型。如果希望通过 Hugging Face API 使用图像分割功能以在性能受限的设备上运行，则可以使用命令 `docker run --name multimodal -p 27777:80 binciluo/multimodal:mini_latest`。部署完成后，用户可通过访问本地地址 `127.0.0.1:27777` 来使用服务。对于 Kubernetes 部署，用户需要先切换到包含 Kubernetes 配置文件的目录，使用 `kubectl apply -f pod.yaml` 命令部署服务（对于使用 Arm 架构的用户，则需使用 `kubectl apply -f pod_arm.yaml`）。非 Linux 用户需运行 `kubectl port-forward mm-service-pod -n default 27777:27777` 以访问服务。服务可通过 `127.0.0.1:27777` 地址访问。

项目还支持本地直接运行。用户需要先安装 Golang 和 Beego 框架，然后安装 Python 所需的依赖包。首先安装 Golang，接着通过命令 `go install github.com/beego/bee/v2@latest` 安装 Beego，然后运行 `pip install -r gradio_web/requirements.txt` 安装 Python 依赖。最终，通过运行脚本 `runner.sh` 启动服务，或者设置环境变量 `SEG_MODEL_ENV='local'` 并运行 `runner_local.sh` 脚本以在本地使用图像分割模型。

7.2.2 GUI 使用说明

- 上传基础图像：您可以上传您的基础图像，或者使用在 Base Image 中的 Examples 提供的示例图像。图像上传后，系统会自动地进行图像分割任务并在 Chat 界面显示分割结果。

- 与 LLM 对话您可以就如何编辑基础图像提出您的看法，或者使用在 Chat 中的 Examples 提供的示例。LLM 的回复会被识别是否含有指令，若有指令存在则会对指令进行提取并自动添加至 Operation Board 中。

- 选择指令来修改基础图像您可以在 Operation Board 中选择单个或多个指令，并有两个按钮可供选择：Exec Selected Commands 和 Exec All Commands。Exec Selected Commands 会执行选中的指令，而 Exec All Commands 会执行所有指令。系统会对这些指令进行合并与排序等预处理，然后生成对应的参数向图像编辑模型发起请求。

- 预览编辑后的图像并设置为基础图像您可以在 Edited Image 标签中预览编辑后的图像，并点击 Set as Base Image 将其设置为新的基础图像。

- 替换面部您可以使用 Edit Image 标签中的 Change Face 功能，将基础图像的面部（仅支持单一面部）替换为特定面部。

- 编辑遮罩如果您对自动生成的遮罩不满意，可以在 Edit Image 标签的 Editor 中手动编辑，然后重新运行之前的指令。

7. 检查服务器状态您可以点击 Check Server Status 来确认所有服务是否可用。系统会在大约三秒后弹出提示框显示哪些服务目前不可用。
8. 自动生成 LLM 微调数据您可在 Auto 模块中的 Gen Data 选择图片，输入 OpenAI 的 API Key 并指定生成数量和线程数以生成 LLM 微调数据。
9. 对 LLM 指令生成合格率进行测试您可在 Auto 模块中的 Test LLM 选择测试集并指定测试样本数量和线程数以对 LLM 指令生成合格率进行测试。

第八章 项目管理与维护

8.1 代码管理

Git 是一个开源的分布式版本控制系统，它允许多个开发者在共同的代码基础上工作，同时能够追踪和记录所有文件的历史变更。其兼具高性能与灵活性，能处理从小到大的项目，让开发者能够在本地机器上工作，并保持代码的多个版本，以便在不同的分支上进行试验和开发新功能的同时不影响主代码库。

GitHub¹作为一个基于 Git 的代码托管和协作平台，为开发者提供了一个强大而便捷的环境来管理代码和协作。它不仅能够追踪和记录代码的变更历史，确保代码的完整性和回溯能力，还能通过分支管理支持多线程的工作流，允许多个开发者同时推进不同的功能。GitHub 的 pull 请求机制促进了团队成员之间的代码审查和讨论，这不仅提高了代码质量，也加强了团队协作。此外，GitHub 的集成系统支持持续集成和持续部署流程，与各种开发工具链的无缝连接，使得项目管理更加高效。通过开源项目的公开，GitHub 还为开发者提供了一个展示和交流的平台，促进了知识共享和技术交流。

为了便于进行代码管理和版本控制，本项目在 GitHub 上创建了一个仓库²，结合 GitHub 的其他功能，将其发展成了功能完整、文档详细的开源项目。

8.2 自动化测试

自动化测试是一种利用软件来控制执行测试的过程，它自动比较实际的运行结果与预期结果，以此来验证被测软件功能的一种测试方法。自动化测试的主要功能是提高测试的效率和覆盖率，它可以快速地执行大量的测试用例，并且可以反复运行这些测试，确保软件在新的开发迭代中未引入回归错误。此外，自动化测试可以在软件开发的早期发现缺陷，从而减少修复缺陷的成本。自动化测试还可以释放测试人员从繁琐的手动测试工作中解脱出来，使他们有更多时间专注于更复杂的测试任务和质量保障活动。在持续集成和持续部署（CI/CD）的实践中，自动化测试是不可或缺的一环，它提高了软件交付的速度和质量，是现代软件开发流程中的关键组成部分。

GitHub Actions 是 GitHub 提供的一个持续集成与持续部署（CI/CD）的平台，允许用户在代码仓库中直接自动化、自定义和执行软件开发工作流程。通过定义一系列的事件和操作，当指定事件发生时，如代码推送、合并请求或者发布时，GitHub Actions 会自动运行这些工作流程。这可以包括构建代码、运行测试、部署到生产环境等任务。GitHub Actions 的出现使得开发者无需离开 GitHub 环境就能自动化处理软件的构建、测试和部署过程，从而大幅提升开发的效率。它支持多种操作系统，提供了大量现成的 Actions 供用户使用，并且允许创建私有的、自定义的 Actions。作为 CI/CD 的解决方案，GitHub

¹<https://github.com>

²<https://github.com/BinciLuo/multimodal-service>

Actions 简化了开发流程，加快了从编写代码到部署产品的过程，同时还提高了软件的质量和交付的可靠性。

本项目使用 GitHub Actions 对代码中的部分模块进行自动化测试以保证代码的正确性和项目的稳定性。当有新的 pull 请求对 main 或 dev 分支进行更新时，自动化测试工作将在最新版本的 Ubuntu 运行环境上执行。其首先会使用 actions/checkout@v3 获取最新的仓库代码，利用 actions/setup-python@v3 来设置 Python 3.10 版本的 Python 环境。当设置好 Python 环境后，需要安装测试所需的依赖。在 gradio_web 目录下首先升级 pip，然后安装本项目自动化测试所需的代码检查和测试框架 flake8 和 pytest，如果存在 requirements.txt 文件，还会安装该文件中列出的依赖。最后，流程将继续在 gradio_web 目录下执行名为 test_utils.py 的测试脚本。

8.3 持续集成与持续部署

持续集成 (Continuous Integration, CI) 和持续部署 (Continuous Deployment, CD) 是现代软件开发中关键的实践，用于自动化软件开发和发布过程。CI 的核心是自动化地将代码变更频繁地合并到主分支，每次合并后自动运行构建和测试流程，这样可以迅速发现并解决集成错误，提高代码质量，缩短反馈周期。CD 扩展了 CI 的概念，不仅自动化测试，还包括自动化部署过程，确保经过测试的代码可以被自动且频繁地部署到生产环境中。这使得产品能够快速迭代，缩短从开发到产品投放市场的时间，同时减少了部署过程中的人为错误，提升了软件交付的速度和安全性。CI/CD 通过自动化的流程减少了手动工作，允许开发团队更加专注于功能开发和创新，而不是部署过程。

本项目使用 GitHub Actions 进行 CI 和 CD 流程。除了测试外，本项目还有几个关键的 CI/CD 流程，主要包括部署将项目部署到 Azure Web 应用和 Docker 镜像构建。

名为“Build and deploy container app to Azure Web App - gradio-app”的 Action 通过在代码推送到 main 分支或手动触发，自动化了在 Azure Web App 上的部署过程。其首先构建 gradio_web 的 Docker 镜像，并将其推送到 DockerHub，随后将镜像部署到 Azure 的生产环境，从而实现高效和一致的应用发布。名为“Build and deploy container app to Azure Web App - middleware-app”的 Action 以相同的方式实现了 middleware 在 Azure Web App 上的自动化部署。

名为“Docker Image CI”的 Action 主要用于构建和推送 Docker 镜像到 DockerHub。当代码被推送到 main 分支时，此工作流程触发并执行以下操作：使用最新的 Ubuntu 环境，首先通过 GitHub Secrets 进行 Docker 登录，然后分别从 docker/Dockerfile 和 docker/DockerfileMini 两个文件构建构建两个 Docker 镜像——标准镜像和更小的 Mini 镜像，其区别为标准镜像使用本地的模型进行图像分割而 Mini 模型使用 HuggingFace¹的 API 进行图像分割。这些镜像将在构建完成后被标记并推送到 DockerHub 上的账户下，确保最新的容器镜像版本可供部署和分发。此自动化流程加快了软件的交付速度，保证了镜

¹<https://huggingface.co>

像的最新状态和可用性。名为“Docker Image CI for ARM64”的 Action 通过相同的方式实现了适用于 arm64 架构的镜像的构建与发布。

第九章 总结及未来展望

9.1 总结

本系统的研发始于对当前图像编辑工具的局限性，这些工具往往需要用户具备专业知识和技能，且操作复杂，难以满足非专业用户的需求。为了解决这些问题，该项目通过打通最新的大语言模型和图像生成模型，开发了一个既强大又用户友好的图像编辑系统。系统的核心在于其能够通过简单的与大语言模型聊天的方式来自动化地生成图像编辑的指令从而调用图像生成模型来执行复杂的图像编辑任务。从系统架构层面，打通大语言模型和图像生成模型需要两个主要组件的协同工作：一个直观的图形用户界面（GUI）和一个功能强大的中间件（middleware）。

图形用户界面（GUI）的设计充分考虑了易用性和高效性，使得即使是没有图像编辑经验的用户也能够轻松地进行复杂的图像操作。通过各个简洁明了的模块，用户可以执行包括指令生成、图像自动遮罩、更换面部、参数设置等多种编辑任务。GUI 既为用户提供了足够简单易用的自动化操作，也能让用户对如遮罩生成等高精度要求的操作进行手动的调整。GUI 不仅提高了操作的直观性，还通过其与中间件（middleware）的高效交互，极大地降低了对用户设备性能的需求。

项目的中间件（middleware）部分是整个系统的枢纽。它整合了多个不同平台、不同请求格式、不同返回格式的 API，包括图像生成模型如 Stable Diffusion、DALL-E2 和大语言模型如 ChatGLM2-6B、GPT-3.5 Turbo 和 GPT-4 Turbo。中间件（middleware）的设计保证了这些模型的高效协同工作，支持了 GUI 所需的从文本交互到图像编辑的一系列高级功能。此外，中间件（middleware）还处理所有后端逻辑，包括图像数据的处理、一对多且互相隔离的服务、以及用户请求的响应，确保了系统的快速响应和高可靠性。

系统创新地引入了基于 LoRa 的大语言模型微调，通过训练专门的 LoRa 模型，原本在本任务下表现较差的 ChatGLM2-6B 模型能够理解复杂的用户输入并精准地将其转化为指令，然后自动对指令进行抽取并执行高质量的图像编辑任务。微调数据集的自动生成是其中的一个重要环节，系统采用了结合多个模型输出和验证规则的自动化工作流程。在已有的图像数据集下，利用现有的图像分割模型和文本生成模型，自动地生成图像编辑的要求、指令，并进行数据合格校验，生成了大量高质量的训练数据。系统设计了自动化的测试流程对各个大语言模型的指令生成能力进行评估，并对 ChatGLM2-6B 的多个微调模型和 GPT3.5 turbo、GPT4 Turbo 进行评估，以便对不同的模型的指令生成能力进行量化。从结果中可观察到 GPT3.5 turbo、GPT4 Turbo 在不进行微调的情况下就已经具有很强的能力，而参数量较小、初始能力极弱的 ChatGLM2-6B 模型在微调后在指令生成上也能达到接近 GPT3.5 turbo 的能力。

9.2 未来展望

最近几年，深度学习在许多领域都取得了显著进展，尤其是在图像和语音识别、自然语言处理和自动驾驶等技术中。在文本生成和图像生成任务上，深度学习技术已从理论探索逐步过渡到实际应用，随着 GPU 和 TPU 等专用硬件的发展，深度学习的训练和推理速度得到了极大的提升，使得复杂模型的实时应用成为可能。图像生成技术已经实现了从简单的图像生成到复杂的场景重构的跨越。在这一领域，生成对抗网络 (GANs) 和最新的扩散模型都已经能够生成高质量的图像内容。而大语言模型，如 OpenAI 的 GPT 系列和清华大学的 ChatGLM2-6B，已经能够理解并生成复杂的自然语言文本。在这些新技术的加持下，通过一定的方法对大语言模型和图像生成模型进行链接，从而使基于 LLM 的交互式图像编辑系统成为可能。本项目尝试了基于特定指令链接大语言模型和图像生成模型的方法，并搭建了基础框架且得到了优异的效果。

尽管大语言模型和图像生成技术的结合带来了许多机遇，但也存在不少技术和伦理方面的挑战。如何确保生成的内容的准确性和适当性，防止生成有偏见或不当信息的风险，是需要重点关注的问题。数据的隐私和安全问题也必须得到妥善处理，确保用户信息的保护同时不妨碍技术的有效应用。

随着深度学习技术的继续发展和优化，未来的交互式技术将更加智能和高效。大语言模型和图像生成技术的结合不仅能提升用户体验，还将开启全新的应用领域，为创新和改进现有服务提供强大动力。面对这些前景，行业和研究者需要共同努力，解决挑战，释放这些技术的全部潜力。

参考文献

- [1] Zhang Lvmin, Rao Anyi, Agrawala Maneesh. Adding conditional control to text-to-image diffusion models [C]. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023 : 3836–3847.
- [2] Van Huynh Nguyen, Hoang Dinh Thai, Nguyen Diep N et al. DeepFake: Deep dueling-based deception strategy to defeat reactive jammers [J]. IEEE Transactions on Wireless Communications. 20 (10). 2021: 6898–6914.
- [3] Achiam Josh, Adler Steven, Agarwal Sandhini et al. Gpt-4 technical report [J]. arXiv preprint arXiv:2303.08774. 2023.
- [4] Hu Edward J, Shen Yelong, Wallis Phillip et al. Lora: Low-rank adaptation of large language models [J]. arXiv preprint arXiv:2106.09685. 2021.

致 谢

此处请写致谢的内容。

它可以有多段。

附录