北 京 邮 电 大 学

本科毕业设计（论文）中期进展情况检查表

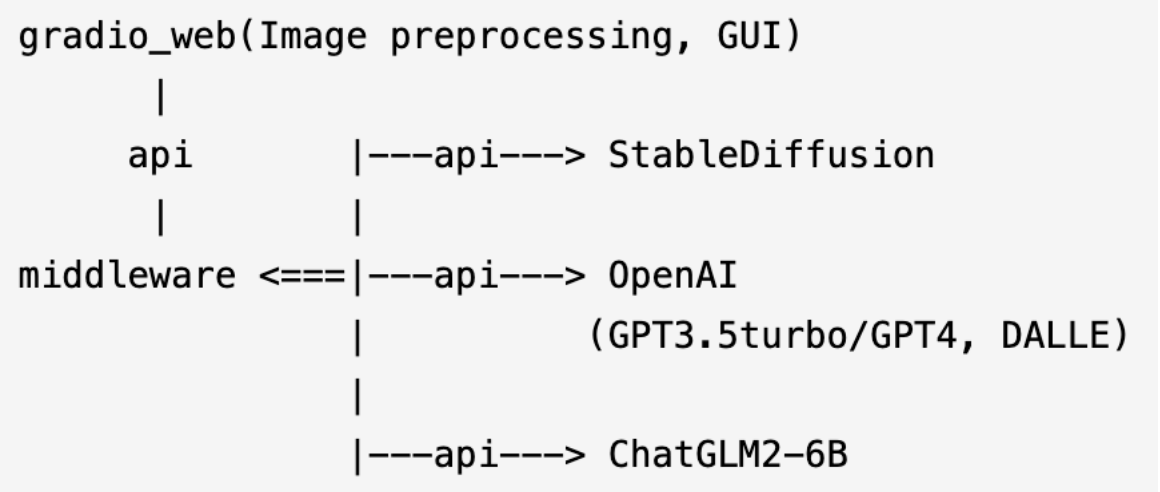
<table>
<tr><td>学院</td><td colspan="2">人工智能学院</td><td>专业</td><td colspan="2">智能科学与技术</td></tr>
<tr><td>学生姓名</td><td>罗彬慈</td><td>学号</td><td>2020212053</td><td>班级</td><td>2020219107</td></tr>
<tr><td>指导教师姓名</td><td>李佩佩</td><td>所在单位</td><td>人工智能学院</td><td>职称</td><td>副教授</td></tr>
<tr><td rowspan="2">设计（论文）题目</td><td colspan="5">（中文）基于 LLM 的交互式多模态图像编辑系统的设计与搭建</td></tr>
<tr><td colspan="5">（英文）Design and Construction of Interactive Multimodal Image Editing System Based on LLM</td></tr>
<tr><td>目前已完成任务</td><td colspan="5">
目前已完成的工作主要包括背景调研、系统设计、代码实现和系统部署三个方面：
一、背景调研
图像生成技术和 LLM（Large Language Models）都是深度学习领域的研究热点。Stable Diffusion 技术的推出和不断的迭代引发了在图像生成领域的浪潮；同时，像 ChatGPT 这样的大语言模型的发布和不断升级也引起了全球范围内的广泛关注。
图像编辑是图像生成技术中的关键技术之一，其旨在保留图像的主要信息的同时根据指令对特定内容进行修改。图像编辑技术在媒体娱乐、数字营销、智能医疗等领域具有广泛的应用前景，因此备受关注。然而，传统的图像编辑模型在交互性方面存在一定的局限性。本项目旨在利用大语言模型辅助进行交互式图像编辑，结合先进的图像生成模型以提升生成图像的质量，最终打造一个交互式图像编辑系统。
二、系统设计
1.系统框架
考虑到本项目需要整合多个模型且要满足用户使用的便利性，本项目使用 Golang 语言搭建了一个后端服务（middleware）对相同类型功能的请求进行整合并向 GUI 提供整合后的 API，因此本项目的 GUI 部分仅需安装一个 python 依赖库，降低了使用门槛。图像生成模型的请求通过 API 调用部署在云平台上的 Stable Diffusion 模型和由 OpenAI 提供的 DALL-E 模型；大语言模型的调用目前可选择通过 API 调用 OpenAI 的 GPT3.5turbo/GPT4 模型，后续准备对 ChatGLM 系列模型进行微调后部署在云平台上以通过 API 调用。这样的系统框架增加了用户使用的便捷性、系统迭代的稳定性，可在用户无感的条件下对系统进行升级。


图 2-1 系统框架
</td></tr>
</table>

2.大语言模型

大语言模型主要通过使用给定的 Prompt、图像分割的结果等将用户输入的自然语言转换为 json 格式的指令。系统会在大语言模型返回的文本中自动地抽取指令并结合配置文件提供的规则对指令的合规性进行校验。由于不同的语言大模型的表现并不一致，用户可以在配置文件中添加多个不同的 Prompt 模版以在各个模型上达到更好的效果。

3.图像生成模型

考虑到 Stable Diffusion 的功能更加丰富以及在开源社区的热度，图像生成主要采用 Stable Diffusion 模型，并在特定的任务中结合不同的模型与插件以满足任务的需求并提高结果的质量。当用户上传图片时，图像分割模型会对图像进行分割，并将分割的结果以对话的方式反馈给用户。在使用图像生成模型对图像进行修改时，首先会结合大语言模型生成的指令和分割的结果对不需要修改的部分进行遮罩，然后在生成的过程中通过 Control Net 对图像的基本框架进行固定以保证生成的质量。由于图像分割模型返回的分割结果不可避免的在一些细节上存在瑕疵（如标签 Hair 在部分区域存在将发丝周围的少量的像素一同分割），在生成遮罩时会对特定的标签进行不同程度的腐蚀--膨胀操作以提高遮罩的质量。同时，可以选择是否使用 ROOP 对图像中的人脸进行替换。当部署在云平台上的 Stable Diffusion 模型不可用时，会使用 DALL-E 模型完成相应功能。

三、代码实现

目前的总代码量为 3649 行，其中 GUI 代码量为 2323 行，middleware 代码量为 1120 行。

1.GUI

使用 Python3 实现了 GUI 的构建，主要使用了 gradio 库。在代码实现时将 GUI 分为 webui、controllers、modules、config、test 五个模块，分别实现 UI 布局（各个模块在 UI 界面的分布）、交互逻辑（不同可交互模块的处理逻辑）、请求处理（图像处理、request body 生成等）、系统配置（模版、路由、默认参数等）、代码测试（主要针对 modules 中的函数进行测试）。UI 界面主要由操作面板、图像预览、文本交互界面和图片编辑界面。

2.middleware

使用 Golang 语言配合 beego 框架实现，将大语言模型调用、图像生成模型调用等数十个 API 整合为 8 个 API 供 GUI 调用，同时在不需要标识请求用户的情况下支持多个 GUI 同时对其进行访问但互不影响（如对话的历史记录等）。

3.Stable Diffusion

基于开源项目 stable-diffusion-webui 进行修改，在其中添加了 ROOP、Control Net 等插件以满足项目的需求。

四、系统部署

1.项目运行

本项目提供了 Docker、Kubernetes、Terminal 等多种运行方式。

2.持续集成、持续部署

持续集成（Continuous Integration，CI）是一种软件开发实践，旨在通过频繁地将代码集成到共享存储库中，然后对代码进行自动化构建和测试，以确保代码变化不会破坏整个代码库的稳定性。CI 旨在尽早发现和解决集成问题，从而提高开发效率和软件质量。

持续部署（Continuous Deployment，CD）持续部署是自动将经过测试的代码部署到生产环境，没有人工干预。

本项目使用 GitHub Action 实现了自动化测试、镜像自动化构建与发布、自动部署至 Azure，提高了代码的可靠性并减少了非开发的工作量。

是否符合任务书要求进度	是 □	否 □

<table>
<tr><td rowspan="2">尚需完成的任务</td><td colspan="3">1. 实现建议功能，在对话时提供建议，并且在用户采纳建议后更新指令。由于传统的大语言模型只能接收文本输入，如果使用传统的大语言模型效果不理想，考虑使用 GPT4-V 模型来生成更好的建议。

2. 使用 LoRa 微调一个更加适合本任务的大语言模型，在与大语言模型交互时支持切换不同的大语言模型。</td></tr>
<tr><td colspan="3">是否可以按期完成设计（论文）　　是 □　　否 □</td></tr>
<tr><td rowspan="2">存在问题和解决办法</td><td>存在问题</td><td colspan="2">1. GUI 的交互性仍有待改善。

2. 在编辑图像时可选的参数较少。

3. 图像生生成质量不稳定。</td></tr>
<tr><td>拟采取的办法</td><td colspan="2">1. 通过对功能分区分类或简化提升 GUI 的交互性。

2. 实现更多的参数可自定义并增加一个设置面板，提高系图像生成的稳定性，将所有设置内容迁移到设置面板中并提供模版，避免在增加参数的过程中增加交互的复杂性。</td></tr>
<tr><td colspan="2">指导教师签字</td><td></td><td>日期</td><td>年　月　日</td></tr>
<tr><td>检查小组评分及意见</td><td colspan="3">评分：　　（总分：　　）

组长签字：　　年　月　日</td></tr>
</table>

注：此表仅供参考，各学院应围绕毕设目标达成度，结合人才培养目标、专业认证要求等进行个性化完善。