1.05

1.5

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。

1.05

1.5

This is ABSTRACT.

You can write more than one paragraph here.

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

# 1 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。

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。

# 2 总体方案设计

## 2.1 GUI

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 ：

GUI模块table\_gui\_modules

| **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：

middleware 主要APItable\_gui\_modules

| **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-controlnet[[1]](#footnote-26)使用了ControlNet(Zhang, Rao, and Agrawala 2023)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-roop[[2]](#footnote-27)基于DeepFake(Van Huynh et al. 2021)， 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 以开源项目stable-diffusion-webui[[3]](#footnote-28)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

本项目使用了OpenAI[[4]](#footnote-30)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(Achiam et al. 2023) 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-Factory[[5]](#footnote-32)， 利用本项目提供的数据自动生成功能所生成的数据集，使用LoRa(Hu et al. 2021)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。

# 3 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。



Figure : 图像分割结果：原始图像，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）。



Figure : 关键词自动生成遮罩结果：原始图像，kewords=[’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）。



Figure : 基于已给出的点自动填充生成遮罩：标记后的图像，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。

1.3

原始遮罩，图像分割结果，配置文件 优化后的遮罩 获取遮罩与非遮罩的描边得到像素 从配置文件和图像分割结果获取 仅保留出现在中的 Apply MinFilter (in ) in [] 得到优化后的遮罩

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（,表示图片的长宽，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。



Figure : 遮罩优化结果：原始遮罩，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。

### 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，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。

[  
 {  
 "command" : "change",  
 "paras" : [ ["Background","Upper-clothes"] , "blue sky, white T-shirt"]  
 }  
 ]

### 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：

{  
 "face": {  
 "paras\_type": [  
 "<class 'str'>"  
 ],  
 "paras\_enum": null,  
 "paras\_min": null,  
 "paras\_max": null,  
 "combine": true,  
 "priority": 1  
 },  
 "change": {  
 "paras\_type": [  
 "<class 'list'>",  
 "<class 'str'>"  
 ],  
 "paras\_enum": null,  
 "paras\_min": null,  
 "paras\_max": null,  
 "combine": true,  
 "priority": 1  
 }  
}

指令校验的算法如 算法-2 所示。

1.3

待校验的指令、校验规则 校验后的指令 Add to

### 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：

1.3

待处理的指令、指令合并规则、指令优先性 处理后的指令 Combine to the same one in use 根据对进行排序

### 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。

1.3

指令、设置、默认参数、参数来源优先性 图像模型请求参数 Get from Choose the highest priority source use [] Set this parameter to []

## 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。

# 4 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

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。 和是通过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。这个API用于获取Stable Diffusion模型可用的LoRa模型列表。 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。 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，、和等API利用OpenAI的不同版本GPT模型来处理指令理解和生成任务，可调用微调后的ChatGLM2-6B模型来进行指令生成。

# 5 LLM的微调

大语言模型（LLM）的微调是一种调整预训练模型以更好地适应特定任务或应用场景的过程。 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， 通过针对特定任务的微调，可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-Factory[[6]](#footnote-55)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

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 Dataset[[7]](#footnote-56)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。

### 5.1.2 LLM指令生成任务性能评估方法

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(Hu et al. 2021)这篇论文提出了一种新颖且高效的微调方法，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。 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方法，通过开源项目LLaMA-Factory[[8]](#footnote-60)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。

## 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所示。

LLM 指令生成性能table\_llm\_scores

| **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% |

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紧准度指令生成任务。 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。

# 6 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-webui[[9]](#footnote-64)部署在揽睿星舟机器学习平台[[10]](#footnote-65)，通过API来调用Stable Diffusion模型， 主要使用的参数见表6-1。

主要使用的Stable Diffusion webui img2img API 参数sdapi\_params

| **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 |

## 6.2 ControlNet的使用及效果

sd-webui-controlnet[[11]](#footnote-67)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。



Figure : ControNet效果：原始图像，未使用ControlNet，使用ControlNet

## 6.3 roop的使用及效果

roop[[12]](#footnote-70)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。



Figure : roop效果：原始图像，目标图像，结果

# 7 系统实现效果与使用

## 7.1 系统实现效果

## 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

1. 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。

2. 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。

3. 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。

4. 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。

5. 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。

6. 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。

### 7.2.3 最佳实践

# 8 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。

GitHub[[13]](#footnote-80)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[[14]](#footnote-81)，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 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HuggingFace[[15]](#footnote-84)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。

# 9 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（middlreware）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。

# 10 基础模块示例

## 10.1 特殊文本类型

### 10.1.1 脚注

社交媒体是一种供用户创建在线社群来分享信息、观点、个人信息和其它内容（如视频）的电子化交流平台，社交网络服务（social network service, SNS）和微博客（microblogging）都属于社交媒体的范畴(“Social Media” 2018)，国外较为知名的有Facebook[[16]](#footnote-90)、Instagram[[17]](#footnote-91)、Twitter[[18]](#footnote-92)、LinkedIn[[19]](#footnote-93)等，国内较为知名的有新浪微博[[20]](#footnote-94)。

在社交媒体的强覆盖下，新闻信息的传播渠道也悄然了发生变化。(Vosoughi, Roy, and Aral 2018)

### 10.1.2 定义、定理与引理等

这是一条我也不知道在说什么的定义，反正我就是写在这里做个样子罢了，也没人会仔细读。(周兴 2017)

这是一条我也不知道在说什么的定理，反正我就是写在这里做个样子罢了，也没人会仔细读。

这是一条我也不知道在说什么的公理，反正我就是写在这里做个样子罢了，也没人会仔细读。

这是一条我也不知道在说什么的引理，反正我就是写在这里做个样子罢了，也没人会仔细读。

这是一条我也不知道在说什么的命题，反正我就是写在这里做个样子罢了，也没人会仔细读。

这是一条我也不知道在说什么的推论，反正我就是写在这里做个样子罢了，也没人会仔细读。

### 10.1.3 中英文文献、学位论文引用

根据美国皮尤研究中心的2017年9月发布的调查结果(“News Use Across Social Media Platforms 2017” 2017)，67%的美国民众会从社交媒体上获取新闻信息，其中高使用频率用户占20%。在国内，中国互联网信息中心《2016年中国互联网新闻市场研究报告》(“2016年中国互联网新闻市场研究报告” 2017)也显示，社交媒体已逐渐成为新闻获取、评论、转发、跳转的重要渠道，在2016年下半年，曾经通过社交媒体获取过新闻资讯的用户比例高达90.7%，在微信、微博等社交媒体参与新闻评论的比例分别为62.8%和50.2%。社交媒体正在成为网络上热门事件生成并发酵的源头，在形成传播影响力后带动传统媒体跟进报道，最终形成更大规模的舆论浪潮。(Yang et al. 2012)

在国内，新浪微博由于其发布方便、传播迅速、受众广泛且总量大的特点，成为了虚假信息传播的重灾区：《中国新媒体发展报告（2013）》(**唐绪军2013中国新媒体发展报告?**)显示，2012年的100件微博热点舆情案例中，有超过1/3出现谣言；《中国新媒体发展报告（2015）》(**唐绪军2015中国新媒体发展报告?**)对2014年传播较广、比较典型的92条假新闻进行了多维度分析，发现有59%的虚假新闻首发于新浪微博。

此等信息的传播严重损害了有关公众人物的名誉权，降低了社交媒体服务商的商业美誉度，扰乱了网络空间秩序，冲击着网民的认知，极易对民众造成误导，带来诸多麻烦和经济损失，甚至会导致社会秩序的混乱。针对社交媒体谣言采取行动成为了有关部门、服务提供商和广大民众的共同选择。(周兴 2017)

## 10.2 图表及其引用

此处引用了简单的表[[crowdwisdom\_TMP]](#crowdwisdom_TMP)。

请注意，LaTeX的图表排版规则决定了图表**不一定会乖乖呆在你插入的地方**，这是为了避免Word中由于图片尺寸不匹配在页面下部出现的的空白，所以请不要使用“下图”“下表”作为指向文字，应使用“图1-1所示”这样的表述。

基于浏览者行为的特征crowdwisdom\_TMP

| **特征** | **描述** | **形式与理论范围** |
| --- | --- | --- |
| 点赞量 | 微博的点赞数量 | 数值， |
| 评论量 | 微博的评论数量 | 数值， |
| 转发量 | 微博的转发数量 | 数值， |

此处引用了复杂的表[[complexcrowdwisdom\_TMP]](#complexcrowdwisdom_TMP)。

基于浏览者行为的复杂特征complexcrowdwisdom\_TMP

| **类别** | **特征** | **不知道叫什么的表头** | |
| --- | --- | --- | --- |
| 3-4 |  | **描述** | **形式与理论范围** |
| 正常互动 | 点赞量 | 微博的点赞数量 | 数值， |
| 评论量 | 微博的评论数量 | 数值， |
| 转发量 | 微博的转发数量 | 数值， |
| 非正常互动 | 羡慕量 | 微博的羡慕数量 | 数值， |

此处展示了更专业的表[[tab:abbr\_table]](#tab:abbr_table)，一个好的表格没有竖线。

红警2名词解释tab:abbr\_table

| **术语类别** | **缩略语** | **解释** |
| --- | --- | --- |
|  | 兵营 | 兵营（Barracks），《命令与征服 红色警戒2：尤里的复仇》游戏中的一种生产建筑，用以生产步兵单位 |
| 2-3 | 建造场 | 建造场（Construction Yard），《命令与征服 红色警戒2：尤里的复仇》游戏中的一种基础建筑，用以支持其他建筑的建造 |
| 2-3 | 矿厂 | 矿石精炼厂（Ore Refinery），《命令与征服 红色警戒2：尤里的复仇》游戏中的一种资源建筑，用以将矿车采集的矿石转化为游戏资金 |
| 2-3 游戏 | 空指 | 空指部（Airforce Command Headquarters），《命令与征服 红色警戒2：尤里的复仇》游戏中的一种资源建筑，用以提供雷达功能和T2科技及生产部分空军单位 |
| 2-3 | 相机 | 游戏术语，特指游戏内的观察区域和视角 |
| 2-3 | 重工 | 战车工厂（War Factory），《命令与征服 红色警戒2：尤里的复仇》游戏中的一种生产建筑，用以生产载具单位 |
| 2-3 | 战争迷雾 | 游戏术语，《命令与征服 红色警戒2：尤里的复仇》中指黑色的未探索区域 |

此处引用了一张图。图[[autoencoder\_TMP]](#autoencoder_TMP)表示的是一个由含有4个神经元的输入层、含有3个神经元的隐藏层和含有4个神经元的输出层组成的自编码器，代表偏置项。



Figure : 这是两个自编码器结构，我就是排一下子图的效果：左边的自编码器，右边的自编码器

## 10.3 公式与算法表示

### 10.3.1 例子：基于主成分分析

#### 10.3.1.1 主成分分析算法

下面对主成分分析进行介绍。

主成分分析是一种简单的机器学习算法，其功能可以从两方面解释：一方面可以认为它提供了一种压缩数据的方式，另一方面也可以认为它是一种学习数据表示的无监督学习算法。(**Goodfellow2016DeepLearning?**) 通过PCA，我们可以得到一个恰当的超平面及一个投影矩阵，通过投影矩阵，样本点将被投影在这一超平面上，且满足最大可分性（投影后样本点的方差最大化），直观上讲，也就是能尽可能分开。

对中心化后的样本点集（有），考虑将其最大可分地投影到新坐标系 ，其中是标准正交基向量，满足， （）。假设我们需要（）个主成分，那么样本点在低维坐标系中的投影是，其中，是在低维坐标系下第维的坐标。 对整个样本集，投影后样本点的方差是

|  |  |
| --- | --- |
|  |  |

由于我们知道新坐标系的列向量是标准正交基向量，且样本点集已经过中心化，则PCA的优化目标可以写为

|  |  |
| --- | --- |
|  |  |

由于是协方差矩阵，那么只需对它做特征值分解，即

|  |  |
| --- | --- |
| $$\label{PCA\_eigenvalue} \bm{X}^ \mathrm{ T }\bm{X} = \bm{W}\bm{\Lambda}\bm{W}^ \mathrm{ T } \\$$ |  |

其中，。

具体地，考虑到它是半正定矩阵的二次型，存在最大值，可对[[PCA\_goal\_TMP]](#PCA_goal_TMP)使用拉格朗日乘数法

|  |  |
| --- | --- |
| $$\bm{X}\bm{X}^ \mathrm{ T }\bm{w}\_i = \lambda\_i \bm{w}\_i \\$$ |  |

之后将求得的特征值降序排列，取前个特征值对应的特征向量组成所需的投影矩阵，即可得到PCA的解。PCA算法的描述如算法[[PCA\_algorithm]](#PCA_algorithm)所示。

1.3

样本集，低维空间维数 投影矩阵 对所有样本中心化 计算样本的协方差 对协方差矩阵做特征值分解 取最大的个特征值所对应的特征向量

#### 10.3.1.2 主成分分析可信度评估方法

记待判定微博的经典特征向量为，它的发布者在前发布的条微博为，这条微博对应的经典特征向量集为。令代表谣言，代表非谣言。算法的具体流程如算法[[PCA\_model]](#PCA_model)所示。

1.3

，，保留主成分数 标签 对所有特征向量应用PCA，保留前个主成分（） 计算中各向量的平均距离和标准差 计算阈值

## 10.4 代码表示

### 10.4.1 直接书写代码在.tex中

下面的代码[[plus]](#plus)是用Python编写的加法函数。

def plusFunc(a, b):  
 return a + b

### 10.4.2 引用代码文件

下面的代码[[recursion]](#recursion)是用Python文件中引入的倒序打印到的函数，请查看code文件夹。

def numbers(x):  
 if x > 0:  
 print(x)  
 numbers(x-1)

## 10.5 列表样式

### 10.5.1 使用圆点作为项目符号

* **第一章为基础模块示例**，是的，本章的名字就是基础模块示例，正如你看到这个样子。
* **第二章为不存在**，是的，其实它不存在。

### 10.5.2 使用数字作为项目符号

1. **第一章为基础模块示例**，是的，本章的名字就是基础模块示例，正如你看到这个样子。
2. **第二章为不存在**，是的，其实它不存在。

### 10.5.3 句中数字编号列表样式

**第一章为基础模块示例**，是的，本章的名字就是基础模块示例，正如你看到这个样子；

**第二章为不存在**，是的，其实它不存在。

# 11 为了目录撑到第二页

## 11.1 我不得不再添加一点内容

## 11.2 尽管这些章节一点正文都没有

## 11.3 是的

## 11.4 真的没有

## 11.5 我已经不知道说什么了

## 11.6 如果有，我们就祝愿一下学校教务处什么时候转变一下思维

## 11.7 把控制格式这种事情往前做

## 11.8 不要总是觉得折磨学生是合理的

## 11.9 你拿着教学管理岗位的工资

## 11.10 你需要折磨一下你自己才对

## 11.11 不要觉得我对别人要求太高，对自己太低

## 11.12 我对自己要求低的话也不至于想要修订这份模板

# 12 致谢

此处请写致谢的内容。

它可以有多段。

# 附录

## 附录1缩略语表

基于浏览者行为的特征crowdwisdom2

| **特征** | **描述** | **形式与理论范围** |
| --- | --- | --- |
| 点赞量 | 微博的点赞数量 | 数值， |
| 评论量 | 微博的评论数量 | 数值， |
| 转发量 | 微博的转发数量 | 数值， |

基于浏览者行为的复杂特征complexcrowdwisdom2

| **类别** | **特征** | **不知道叫什么的表头** | |
| --- | --- | --- | --- |
| 3-4 |  | **描述** | **形式与理论范围** |
| 正常互动 | 点赞量 | 微博的点赞数量 | 数值， |
| 评论量 | 微博的评论数量 | 数值， |
| 转发量 | 微博的转发数量 | 数值， |
| 非正常互动 | 羡慕量 | 微博的羡慕数量 | 数值， |

def minusFunc(a, b):  
 return a - b

|  |  |
| --- | --- |
|  |  |

## 附录2数学符号

|  |  |  |
| --- | --- | --- |
| **数和数组** | |  |
|  |  |  |
|  | 标量（整数或实数） |  |
|  | 向量 |  |
|  | 向量的维数 |  |
|  | 矩阵 |  |
|  | 矩阵的转置 |  |
|  | 单位矩阵（维度依据上下文而定） |  |
|  | 对角方阵，其中对角元素由向量确定 |  |

**真假新闻的在线传播**

# 第一章概述

## 1.1概述

决策、合作、通信和市场领域的基础理论全都将对真实或准确度的概念化作为几乎一切人类努力的核心。然而，不论是真实信息还是虚假信息都会于在线媒体上迅速传播。定义什么是真、什么是假成了一种常见的政治策略，而不是基于一些各方同意的事实的争论。我们的经济也难免遭受虚假信息传播的影响。虚假流言会影响股价和大规模投资的动向，例如，在一条声称巴拉克·奥巴马在爆炸中受伤的推文发布后，股市市值蒸发了1300亿美元。的确，从自然灾害到恐怖袭击，我们对一切事情的反应都受到了扰乱。

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

|  |  |
| --- | --- |
|  |  |

我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字s。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。我只是为了把第二章挤到下一页而凑的字。

# 第二章我也不知道是什么

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

|  |  |
| --- | --- |
|  |  |

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

新的社交网络技术在使信息的传播速度变快和规模变大的同时，也便利了不实信息（即不准确或有误导性的信息）的传播。然而，尽管我们对信息和新闻的获取越来越多地收到这些新技术的引导，但我们仍然对他们在虚假信息传播上的作用知之甚少。尽管媒体对假新闻传播的轶事分析给予了相当多的关注，但仍然几乎没有针对不实信息扩散或其发布源头的大规模实证调查。目前，虚假信息传播的研究仅仅局限于小的、局部的样本的分析上，而这些分析忽略了两个最重要的科学问题：真实信息和虚假信息的传播有什么不同？哪些人类判断中的因素可以解释这些不同？

# Reference

“2016年中国互联网新闻市场研究报告.” 2017. 中国互联网络信息中心; online. 2017. <http://www.cnnic.cn/hlwfzyj/hlwxzbg/mtbg/201701/P020170112309068736023.pdf>.

Achiam, Josh, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, et al. 2023. “Gpt-4 Technical Report.” *arXiv Preprint arXiv:2303.08774*.

Hu, Edward J, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. “Lora: Low-Rank Adaptation of Large Language Models.” *arXiv Preprint arXiv:2106.09685*.

“News Use Across Social Media Platforms 2017.” 2017. Pew Research Center; online. 2017. <http://www.journalism.org/2017/09/07/news-use-across-social-media-platforms-2017>.

“Social Media.” 2018. Merriam-Webster; online. 2018. <http://www.merriam-webster.com/dictionary/social media>.

Van Huynh, Nguyen, Dinh Thai Hoang, Diep N Nguyen, and Eryk Dutkiewicz. 2021. “DeepFake: Deep Dueling-Based Deception Strategy to Defeat Reactive Jammers.” *IEEE Transactions on Wireless Communications* 20 (10): 6898–6914.

Vosoughi, Soroush, Deb Roy, and Sinan Aral. 2018. “The Spread of True and False News Online.” *Science* 359 (6380): 1146–51.

Yang, Fan, Yang Liu, Xiaohui Yu, and Min Yang. 2012. “Automatic Detection of Rumor on Sina Weibo.” In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, 1–7. New York, NY, USA: ACM.

Zhang, Lvmin, Anyi Rao, and Maneesh Agrawala. 2023. “Adding Conditional Control to Text-to-Image Diffusion Models.” In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3836–47.

周兴. 2017. “基于深度学习的谣言检测及模式挖掘.” 中国科学院大学.

1. https://github.com/Mikubill/sd-webui-controlnet [↑](#footnote-ref-26)
2. https://github.com/s0md3v/sd-webui-roop [↑](#footnote-ref-27)
3. https://github.com/AUTOMATIC1111/stable-diffusion-webui [↑](#footnote-ref-28)
4. https://openai.com [↑](#footnote-ref-30)
5. https://github.com/hiyouga/LLaMA-Factory [↑](#footnote-ref-32)
6. https://github.com/hiyouga/LLaMA-Factory [↑](#footnote-ref-55)
7. https://github.com/lemondan/HumanParsing-Dataset [↑](#footnote-ref-56)
8. https://github.com/hiyouga/LLaMA-Factory [↑](#footnote-ref-60)
9. https://github.com/AUTOMATIC1111/stable-diffusion-webui [↑](#footnote-ref-64)
10. https://www.lanrui-ai.com [↑](#footnote-ref-65)
11. https://github.com/Mikubill/sd-webui-controlnet [↑](#footnote-ref-67)
12. https://github.com/s0md3v/sd-webui-roop [↑](#footnote-ref-70)
13. https://github.com [↑](#footnote-ref-80)
14. https://github.com/BinciLuo/multimodal-service [↑](#footnote-ref-81)
15. https://huggingface.co [↑](#footnote-ref-84)
16. http://www.facebook.com/ [↑](#footnote-ref-90)
17. https://www.instagram.com/ [↑](#footnote-ref-91)
18. http://www.twitter.com/ [↑](#footnote-ref-92)
19. http://www.linkedin.com/ [↑](#footnote-ref-93)
20. http://www.weibo.com/ [↑](#footnote-ref-94)