南京大学

# 研 究 生 毕 业 论 文

# （申请硕士学位）

论 文 题 目　基于对抗攻击换脸的人脸保护系统设计与实现

作 者 姓 名　迟 宇 翔

专 业 名 称　计 算 机 科 学 与 技 术

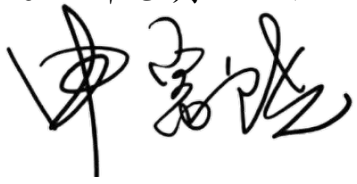
研 究 方 向　计 算 机 视 觉

指 导 教 师　申 富 饶　教 授

2022 年 5 月 25 日

学　　号：**MG1937004**

论文答辩日期：**2022** 年 **5** 月 **17** 日

指 导 教 师：  　（签字）

# Face-Swapping Adversarial Attack Based Face Protection System Design and Implementation

by

**Yuxiang Chi**

Supervised by

Professor Furao Shen

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
MASTER
in
Computer Science and Technology


School of Artificial Intelligence
Nanjing University

# 南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于对抗攻击换脸的人脸保护系统设计与实现

计算机科学与技术 专业 2019 级硕士生姓名：迟宇翔

指导教师（姓名、职称）：申富饶 教授

## 摘　要

近年来，随着深度学习与计算机视觉技术的不断深化与发展，在互联网中出现了一种以 Deepfake 为代表的人脸交换技术，这种技术可以逼真地交换两幅图像或两段视频中的人脸，使得一幅图中人脸相貌及代表人物身份的特征出现在另一幅图中。然而由于其带来诸如肖像权、隐私权侵犯、交换政治人物、敏感图像等问题，对社会产生不良影响，众多换脸应用被下架。因此，若要发挥该项技术的用途，必须解决其隐私问题与安全问题。尽管目前存在一些对伪造人脸的检测手段，但这些检测方法可以被绕过。通过在人脸交换系统中加入敏感图像检测技术，来限制部分人物图片的交换也是一种手段，但对于未识别的人脸则无法避免。针对上述问题，本文设计并实现了基于对抗攻击换脸模型保护人脸的系统，通过破坏换脸结果以起到防止换脸的作用。本文主要研究工作如下：

1. 本文提出了从特征层面对换脸模型进行攻击的人脸保护方法，基于对抗攻击方法，在给定的人脸图像中添加微小扰动，使其换脸后的图片出现变化，起到保护人脸的作用。该方法在传统的攻击方案基础上，通过攻击换脸模型中的特征提取部分，以使得模型提取的特征发生变化，使其使用错误的特征进行换脸，可以很好地掩盖换脸图片中被换脸者的真实身份信息，导致换脸无效。同时，由于在特征提取层面特征就已经发生变化，因此无论后续采用何种模型，换脸都将无效，这使得攻击方法具有更强的鲁棒性和稳定性，能在一定程度上解决换脸技术存在的隐私风险。

2. 本文提出了从换脸结果层面进行对抗攻击的人脸保护方法，结合 PGD 攻击与 StarGAN 判别器对换脸模型进行攻击，通过破坏换脸结果以使得换脸失败。与特征层面攻击的区别在于，基于结果的攻击以换脸结果的破坏为目的，对完整的模型进行攻击，且通过设计不同的攻击目标，能产生不同的改变换脸

图像人物认知的破坏形式，其中包括真实性降低、特定位置出现黑块、语义特征发生改变等效果，从而实现多种类、稳定、可靠的人脸保护。

3. 本文基于上述人脸保护方法设计并实现了攻击换脸的人脸保护系统，其目的是提供有效的可控攻击服务，对换脸算法与攻击算法进行实际应用、测试与验证。系统的优势在于能对外提供算法服务，实现网页版的换脸与攻击，并能在服务器中更为快捷地计算。系统基于 B/S 架构，使用轻量级的 Flask 作为后端框架，使得开发过程便捷；使用 NodeJS 与 React 作为前端框架，能高效完成页面设计，运行时具有优良的性能；使用 NginX 实现反向代理，能提供高效的服务；基于双 token 验证机制实现了身份验证模块；基于多进程框架与任务队列实现了单服务器多任务并发调度方案，实现多任务并发运行。经测试，系统在正常并发环境下能稳定运行，具有良好的可扩展性。

**关键词：** 人脸交换技术；人脸交换攻击；投影梯度下降；生成对抗网络

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Face-Swapping Adversarial Attack Based Face Protection System Design and Implementation

SPECIALIZATION: Computer Science and Technology

POSTGRADUATE: Yuxiang Chi

MENTOR: Professor Furao Shen

## Abstract

In recent years, with the emergence and development of deep learning and computer vision technology, a new technology called Deepfake has appeared on the Internet, especially face swapping, which swaps faces in two images or videos while keeping the whole image consistent. However, due to the improper use of exchanging political figures or sensitive images, such behavior leads to violations of portrait rights, causing personal privacy issues and negative impact on society. Eventually, many face-swapping applications have been taken down for various reasons. In order to maintain the value of this technology, its privacy issues must be addressed. Although there are already some detection methods available that can detect fake images, these methods can be easily bypassed. It is also worth mentioning that by adding sensitive image and celebrity detection to the system, thus preventing those images from being swapped can reduce infringements, but it is powerless against unrecognized faces. In response to the above problems, this paper designs and implements a face privacy protection system based on attacking face-swapping model. The system provides methods that can corrupt the face-swapping result, which helps preventing face-swapping from happening. The main research work of this paper is as follows:

1. This paper proposes a face protection method based on adversarial attack from feature level. Based on the PGD attack method, a small disturbance is added to a given face image, so that the image after face-swapping would change, thus provides protection for the original image. In addition to the adversarial attack scheme, this method attacks the feature extraction part of the face-swapping model, so that the feature extracted by the model changes. Since the wrong features are used in the face-swapping

process, the real identity information of the person in the final image is well covered, which makes the face-swapping process invalid. Meanwhile, since the features have already changed at the feature extraction level, no matter what model is used subsequently, face-swapping will be invalid, which makes the attack method more robust and stable, and can solve the privacy issue of face-swapping technology to some extent.

2. This paper proposes a face protection method based on adversarial attack from result level. Based on the PGD attack method, by attacking the face-swapping model, the result image can be corrupted, which protects the original image from being swapped. The difference from the feature-level method is that the result-based method aims at corrupting the result image. By attacking the complete model, and by designing different target functions, the corruption on the result image can become a specific form, including reduction of authenticity, the appearance of black blocks in specific positions, and the change of semantic features, etc., thus achieves a variety of stable and reliable face protection results.

3. Based on the above-mentioned face protection method, this paper designs and implements a face protection system for attacking face-swapping. The system is based on B/S architecture, uses Flask framework for the back-end, NodeJS and React framework for the front-end, uses NginX for reverse proxy and cross-domain problems, and implements authentication module based on the double token authentication method, based on the multi-process framework and task queue to implement a single-server multi-task concurrent scheduling scheme to achieve multi-task concurrent operation. After testing, the system can run stably in a normal concurrent environment and has good scalability.

**keywords:** Face Swapping Technology; Face Swapping Attack; Project Gradient Descent; Generative Adversarial Network

# 目　　次

# 插图清单

# 附表清单

# 第一章　绪论

## 1.1　研究背景

人脸深度伪造技术是使用机器学习算法对图像或视频中的人脸进行编辑、替换、合成等操作以生成伪造人脸的技术，这种技术早在二十年前就已出现，Christoph Bregler 等人在 1997 年的一个项目中根据语音对视频中的人脸嘴型进行编辑[1]，文中使用 HMM 检测语音中的三音素，并标记视频中的面部特征点，根据人声发音改写视频中的嘴型。近年来，随着深度学习与计算机视觉领域的不断深化发展，产生了一系列深度伪造算法，例如 Face2Face 可以实时将一个视频片段中的人物模仿另一个人的面部表情[2]。一些算法能将新的表情或新的姿势应用与原图像人物上，还可以修改图像中人脸的面部特征，例如性别、年龄、发色等。这些深度伪造算法不仅可以编辑人脸，还能使生成的人脸难辨真假。

除人脸编辑之外，人脸交换也是人脸深度伪造技术之一，这项技术可以交换两张图像或视频中的人脸，使被交换的人脸融合到另一张图像中，图像整体具有高度一致性。换脸技术可应用于电影制作、人像匿名化等领域，在电影制作中被用于替换演员人脸，不过目前仍存在很多技术问题，且换脸成本的高昂使其离达到电影标准还有一定距离。人像匿名化是一种基于换脸进行隐私保护的方法，通过将原图人脸替换为一匿名人脸，以起到隐藏原图人物身份的作用。

图像篡改的方法大体可分为三类：身份篡改，表情篡改和面部属性修改。其中身份篡改将图像中的人脸替换为另一个人脸，例如以 Face Swap 为代表的一系列人脸交换应用[3]，以及 Huang 等人的工作中将图像中的人脸替换成完全虚构的人脸[4]，该方法实现了人像匿名化的功能，因此也可用于人像的隐私保护。表情篡改是将图像中的人脸表情修改为指定的表情，Thies 等人提出的 Face2Face 可将一人的表情迁移到另一幅图像的人脸上[2]。面部属性修改可对人脸部分的年龄、性别、发色等属性进行修改，例如 Choi 等人提出的 StarGAN[5] 是一种单领域到多领域的图像转换模型，可基于单模型生成器将图像从原领域

转换至多个目标领域的分布上。

近年来互联网中出现了众多含有人脸交换功能的手机应用。Face Swap 于 2014 年在 Android 上推出[3]，让人们可以轻松地将手机拍摄照片中的人脸进行交换。Face Swap Live 在图像换脸基础上进行了创新[6]，使其可以实时在视频中交换人脸。2016 年，Snapchat 推出了换脸滤镜功能[7]，可在照相过程中给人脸添加包括换脸功能在内的特殊效果，显著推动了这一趋势。然而，随着新型换脸 App 的不断推出，这项技术也引发了大量争议，其主要原因在于换脸技术侵犯了被换脸人的肖像权，对知名人物会出现丑化、侮辱、诽谤该人物形象的问题，并造成不良社会影响。一些提供换脸服务的平台还会在用户不知情的情况下使用用户图像为其他人进行换脸，也涉及到侵犯肖像权的问题。为此，以 ZAO 为代表的众多 App 陆续下架。

随着人脸深伪技术日益成熟，市面上具有人脸编辑功能的应用也不断涌现。为了减少上述问题的发生，部分研究者转而对深伪模型的攻击进行研究，通过使用对抗攻击技术，基于已知深伪模型进行白盒攻击，通过在图像上添加扰动的方式破坏深伪模型所生成的图像。Ruiz 等人于 2020 年提出了一种针对图像翻译网络与人脸编辑系统的攻击方法[8]，这种方法能够破坏深度伪造模型所生成的输出图像，并能成功泛化到不同类别的可转移对抗性攻击，使得攻击者无需知道条件类别即可实现攻击，同时也是第一个提出对鲁棒性图像转换系统进行对抗性攻击的工作。

目前，市面上存在一系列成熟的人脸交换工具与系统，但尚未发现对图像进行保护的工具，而实现一种攻击换脸模型的系统，通过保护图像以起到对换脸的防范作用，能很好解决诸多换脸技术带来的侵权现象及不良社会影响。本文基于此对目前公开、换脸效果最好，且具有较好鲁棒性与泛化能力的 FaceShifter 换脸模型，提出了基于特征层面和基于结果层面攻击的方法，从多角度对换脸模型进行攻击，并将攻击方案进行应用，设计并实现了人脸保护系统。系统实现了对给定图片进行隐私保护，通过在图片中加入微量扰动以破坏换脸结果，且可以通过设定扰动上界控制扰动的大小，其研究及开发的意义在于既能起到保护特定人物隐私权或肖像权的作用，又能减少不良换脸行为的发生，使得换脸技术在不带来不良社会影响的情况下得以服务于大众。

## 1.2 国内外研究现状

### 1.2.1 人脸交换研究现状

人脸交换的主要难点在于如何准确地提取图像中人物的身份，并将其与另一幅图的人物重新组合。早期基于替换的工作只是将人脸部分的像素进行替换，这类方法对人物姿势和视角的变化很敏感。通过 3D 重建人脸模型可以解决人物姿势或透视差异的问题，但其准确性和鲁棒性均不理想。近年来，基于 GAN 的方法取得了较好的效果，但一些算法合成结果的逼真性和高质量性仍得不到保证。

从功能上分，目前市面上流行的包含人脸交换功能的系统，例如于 2019 年推出的 FaceApp[9]，为用户提供以美颜为主的各类特效滤镜功能，例如年轻化、衰老化、去皱纹、性别交换、添加胡须等，以及与其他人进行面部交换的功能。也有部分研究设计匿名化人像功能的系统，例如 Mahajan 等人提出的 SwapItUp 系统[10]，该系统通过将图中人脸替换为另一匿名人脸的方式，隐藏图像中的人物身份。

目前实现人脸交换有以下几类主流方法，有使用特征点对齐法将人像部分剪切并拼接的方法，例如 SwapItUp 系统[10]，该系统首先检测第一幅图的面部特征点，将第二幅图的脸部进行变换以对齐特征点，并对图像色彩平衡进行调整，以使得图像的整体观感更自然，最后使用一个掩膜将第二幅图像的面部关键区域嵌入第一幅图像中。与其他方法相比，这种方法得到的图像能保证其高质量性。还有使用 3D 建模法对人物形状与位姿进行估计，并在三维空间中重构人脸，然后进行换脸操作，例如 Blanz 等人提出的系统可以在包含巨大光照差异以及视角差异的场景下实现较高质量的人脸交换[11]，该系统首先估计图像中人物的三维形状、姿势、光照等相关场景参数，再手动标记面部约 7 个特征点，并标记目标发际线后实现人脸的交换。该系统可用于图像处理、虚拟发型试穿和人脸识别。近年来也有基于对抗生成网络的相关工作，Zeno 等人提出一种基于对抗生成网络的能适应人物不同姿态下的人脸增强框架[12]，该框架允许使用任意姿态下的原图像，并使用另一张人脸图像控制源图像的姿态，且生成的图像中含有原始图像的人物身份。Zhao 等人提出了一种双 Agent 的 GAN 模型[13]，其中双 Agent 是指在 GAN 中使用两个判别器，使用未标记的真实数据和感知身份信息判别合成人脸图像的真实性，而生成器则需同时对抗识别真

实性的判别器和身份判别器。Karras 等人采用渐进增长的方式训练 GAN[14]，通过不断提高图像分辨率的方式训练，在生成较小的图像时由于所包含的类别信息和特征较少，因此虽然生成的图像更稳定，但仍存在一些问题，例如所生成的图像对语义敏感，以及未能理解图像中所存在的约束等问题。Karras 等人提出的基于风格进行编辑的 GAN[15]，在生成器中通过计算隐空间的风格来控制图像的高阶属性，从而可以通过对风格参数进行特定比例的修改来控制图像合成效果，同时使用混合正则化的方式使用两组隐变量参数按比例生成图像，使得图像同时应用两组风格参数，从而防止网络假设相邻风格是相关的。Pumarola 等人建立了一个与人脸动作编码系统具有相似表达水平的人脸生成模型[16]，并在不使用任何面部标记的情况下能够生成解剖学意义上连贯的人脸表情图像。

此外，对人脸交换性能与质量方面也有相关研究。Korshunova 等人提出了一种快速人脸交换算法[17]，该算法基于风格迁移方法，其输入图像经过人脸变换、卷积神经网络、分层遮罩、再次变换等操作后得到输出图像，其中神经网络基于内容损失、风格损失和光照损失函数训练，能够在没有用户输入的情况下实时进行面部交换，并产生高度逼真的结果。在人脸深度伪造方面常见的两个任务，即人脸交换与表情重现任务通常是分开处理，而 Peng 等人则利用了两种任务的潜在相似性，提出了一种同时具有高保真面部交换和表情重现功能的框架[18]，通过分解三维位姿、形状及表情因素，将其重组以用于不同的任务，这些任务被输入到一个常见的图像转换模型中，以直接生成最终的合成图像。

### 1.2.2 针对深度伪造的防御研究现状

由于人脸深度伪造可能带来的问题，目前已有很多工作均从检测角度对深度伪造图像进行判别。根据伪造类型的不同，检测内容大体可分为对身份交换的检测、面部属性修改的检测、面部表情修改的检测这，检测方法包含基于传感器硬件的检测、提取篡改特征的检测、基于图像缺陷的检测、基于生物特征的检测等方法。

Bhattacharjee 等人使用基于硬件的方法实现面部呈现攻击检测 (PAD)[19]，分别使用深度成像摄像机与热成像摄像机，能有效检测出使用面具等方法进行的面部伪造。Khodabakhsh 等人基于最近的深度学习方法和传统的基于纹理描述符的方法，对不同假人脸检测方法的普遍性进行了全面的探索性研究，并提出了有效评估基于纹理和基于深度学习方法的泛化性方案[20]。Zhou 等人提出

了一种双流网络检测篡改的人脸的方法[21]，其中第一个流通过训练 GoogLeNet 对图像是否经过篡改进行分类，第二个流提取隐写分析特征并基于三元组损失进行训练，利用图像局部噪声和相机特征对人脸图像是是否经过篡改进行分类，最后融合两个流的结果得到最终识别结果。Rössler 等人构建了一个包含未篡改和经过两种类型篡改的图像的数据集[22]，提供了不同质量标准下图像取证任务分类与分割的基线标准。Guo 等人提出了一种自适应篡改痕迹提取网络(AMTEN)[23]，该模型使用 CNN 自动学习图像篡改痕迹的高阶特征，其中模型第一层用于生成图像预测像素值，并用预测像素值与真实像素值之差作为篡改痕迹，另外设计了一种基于 CNN 的层次特征提取 (HDE) 模块，用于进一步学习篡改痕迹的高阶特征。与其相似的工作还有 Rich 等人提出的一种构建数字图像噪声分量的模型[24]，其中考虑了像素之间许多不同性质的关系，该过程首先将噪声分量的丰富模型组装为许多不同子模型的联合，首先基于不同的滤波器计算图像的噪声残差，这些噪声残差相邻样本的联合分布构成了不同子模型，这种方法计算复杂度低，能有效处理高维特征空间和大型训练集。Li 等人对深度神经网络生成的图像与真实图像的色彩分量进行分析，发现合成图像在色度分量在残差域中比真实图像的区分度更高，并据此提出了一种用于区分合成图像与真实图像的特征集[25]，在该研究中认为基于 GAN 生成的图像基于 RGB 色度空间，而在其他色度空间中与真实图像相比具有更大的统计差异，因此基于卡方距离对色彩分量差异进行建模。Agarwal 等人提出了一种深度伪造视频的检测方法[26]，该方法利用面部表情与人物动作的相关性，通过追踪视频中人物的面部和头部运动，提取特定动作单元的强度，并使用单分类 SVM 将人物与其他人物及深度伪造人物进行区分。

一些研究工作则针对生物特征的不一致性进行检测，例如 Matern 等人通过检测特定人脸部位的缺陷来检测深度伪造[27]，例如提取眼睛、牙齿等部位的反射缺陷、细节缺陷和颜色差异等特征，以及面部边缘、鼻尖等部位的特征，实现深度伪造图像与真实图像的分类。Li 等人根据视频中人的眨眼频率判别视频真伪[28]，使用带有长期记忆机制的 LRCN 模型并以眼部区域的图像序列作为输入，进行眨眼状态的预测并计算眨眼频率，根据深度伪造的视频眨眼频率低的特点进行分类。Ciftci 等人使用面部生物信号检测伪造视频[29]，该研究工作对真伪视频中人物不同面部区域分别提取生物特征信号，通过对生物信号进行时空域分析，计算经过变换后的生物信号的相关性以在特征集中捕捉信号特征，并训练 SVM 与 CNN 对视频进行分类，该方法在不同伪造方法、视频质量、分

辨率等场景下均取得了较高的准确率。Wu 等人通过分析视频色彩变化进行检测[30]，该方法对视频空间频域分解后按帧进行时域滤波，并根据放大后的信号进行检测。

也有部分工作从防止伪造的角度出发攻击深度伪造模型，这类方法通过给原图添加肉眼不可分辨的扰动破坏模型所生成的伪造图。Ruiz 等人于 2020 年提出了一种针对基于图像翻译的人脸深度伪造进行攻击的方法[8]，这种方法不对图像是否进过篡改进行检测，而是破坏人脸深度伪造生成图片的结果，在研究中使用了 FGSM、I-FGSM 和 PGD 传统对抗攻击方法进行训练，所使用的深度伪造模型为将人脸属性进行修改的图像翻译系统，并成功对基于 GAN 的抗攻击模型进行了攻击。同时，在灰盒场景下使用模糊图像的方法也取得了一定的效果。Yeh 等人使用 PGD 方法针对图像翻译网络进行攻击[31]，其中包括一种使得图像翻译模型输出被破坏的扭曲攻击，以及一种身份无效化攻击。Segalis 等人提出了振荡对抗生成网络 (OGAN) 攻击[32]，这是一种抗训练的新型攻击，通过在迭代过程中交替优化，同时训练生成器和换脸模型来解决优化问题。该方法构建一个双级优化问题，在其中训练了一个生成器和一个人脸交换模型，将每个输入图像与目标失真成对地输入到生成对抗性图像的生成器中。当对其应用人脸交换自动编码器时，该图像将表现出失真。实验表明它们可以在不同的目标模型和目标人脸之间转移，包括对抗性攻击未经训练的人脸。Willetts 等人针对 VAE 通过保持重建性能的方式进行防御[33]，并定量分析了这种防御方法对 VAE 任务影响的鲁棒性。

### 1.2.3 现有问题与不足

目前市面上广泛存在的换脸工具及系统会造成肖像权及隐私权侵犯，带来不良社会影响，虽然部分系统包含敏感人物检测，能阻止部分图像的编辑，但仍然存在绕过该机制的方式，且检测机制不足以涵盖所有需要保护图像权利的人物形象。部分合成的图像能被现有一些伪造图像检测技术检出，但这些检测系统仍能通过对抗攻击手段来绕过，例如 Li 等人与 2021 年提出的一种通用的对抗伪造检测方法[34]，因此从检测角度出发不足以对抗这种行为，而基于图像添加扰动的方式对换脸模型进行攻击，从而对图像进行保护是较为有效的手段。目前存在的各种人脸深度伪造系统，包括人脸交换、人脸编辑等人脸深度伪造功能，也有一些将人脸交换应用于人脸隐私保护的系统，但尚未出现对图片进行保护以防止人脸交换的功能。此类功能不仅可以大幅改善人脸交换所带

来的问题，使得人脸交换能正常用于网络环境，还同时起到了个人、名人明星等肖像权与隐私权保护的作用。

在攻击换脸的方面，现有工作例如 Yeh 等人基于 PGD 攻击方法对 Cycle-GAN 图像风格转换网络进行攻击[31]，这种方法攻击能使得图像中出现随机且不规则的斑块，虽然能达成主动保护图像的目的，但攻击后所合成的图像出现的破坏痕迹具有随机性，难以控制合成图像被破坏的形式，且有部分图片在攻击后所生成的图像仍然保留原图人物身份特征，无法起到隐私保护的作用。另外攻击形式较为单一，其攻击目标仅使得图像被破坏，虽然有控制攻击图像出现破坏的形式，但效果并不理想。

在系统方面，目前只存在提供换脸服务的应用与工具，尚未出现对换脸模型进行攻击，可提供人脸图像保护服务的系统。本文所设计系统可提供多种图像攻击、换脸的服务，用户可选择不同的攻击效果与参数，直接在浏览器中操作即可调用服务 API 执行多种保护操作。

## 1.3 本文工作

人脸交换技术具有很大的应用价值，但由于一些不良换脸现象造成的社会影响导致该技术无法被充分应用。本文基于这一存在的问题设计并实现了对抗攻击换脸的人脸保护系统，其核心是防止指定图片被换脸，通过对抗攻击换脸模型，在人脸图片中添加具有攻击性质的扰动，能使得换脸后的图像实质产生变化，使换脸结果无效，从而起到保护人脸图像不被用于换脸的作用。根据攻击形式的不同，本文分别提出了基于特征层面的攻击和基于换脸结果的攻击，前者不对换脸结果进行破坏，而是针对特征提取过程进行攻击，通过破坏所提取的特征来使得后续过程无效；后者针对换脸结果进行攻击，通过改变所设定的攻击目标来实现对换脸图片不同形式的破坏，从而起到保护人脸的作用，有效解决上述问题。本文研究内容主要包含两部分，一是从基于特征和基于结果两个角度出发设计了多种针对换脸模型的对抗攻击方案，能对换脸后图像造成不同形式的破坏，起到保护图片中人物隐私的作用；二是从需求分析、系统设计、系统实现及系统测试四个环节完整实现了基于换脸攻击的人脸保护系统，该系统具有完整的功能，能对外提供可靠的换脸与攻击服务，对攻击方案进行充分应用，具体如下：

本文首先提出了对换脸模型基于特征层面的攻击，该方法不直接攻击整个

模型，而是对模型中的特征提取部分进行攻击，以使得模型的对人物的特征提取失败。当提取了错误的特征后，后续采用任意结构的模型进行换脸，所得的换脸图像都将呈现出另一个人物特征的效果，虽然没有破坏换脸结果，但使得原图像人物的特征得到了保护。通过攻击换脸模型中的身份编码器 arcface，基于最大化身份差异进行攻击，能使得换脸图像身份发生改变，使换脸结果无效化。

其次，本文提出了基于结果层面的攻击方法，该方法将以破坏换脸结果为目标，直接使得换脸失败。与传统攻击方法不同的是，传统方法以攻击前后生成图像的差异或其他指标作为攻击目标，而该方法以生成图像在语义层面或其他层面产生特定破坏形式为目标，以实现稳定、特定形式的攻击。例如，传统方法能使得被破坏的图片中出现随机性的斑块，但这些斑块的大小、质量均无法控制。通过在模型后串联一算法，由算法提取所需攻击的特征，并对整体进行攻击，可以实现对换脸破坏结果的控制。根据攻击目标的不同，本文设计了三种不同的攻击方案，其中包括：以最小化真实度为目标进行攻击，能使得攻击后换脸图像的真实性降低；在换脸模型后通过一个构造的 mask 计算损失，攻击能使得换脸图像特定位置出现黑色区域；在换脸模型后接一语义判别器，用于识别图像人物面部特征，并基于该特征与目标特征的均方差损失进行攻击，能使换脸图片发生语义层面上的变化，且可以人为指定所改变的人脸特征，例如可以指定人脸性别、年龄、发色等属性发生变化。

由于目前只存在一些基于人脸交换的隐私保护系统，但并未出现保护人脸以防止人脸交换的系统，本文基于该换脸攻击方案设计并实现了包含完整攻击功能的人脸保护系统，能够提供有效的可控攻击服务并解决换脸带来的肖像权等问题，并对攻击技术进行实际应用、测试与验证。系统总体实现方案采用前后端架构，实现网页版的攻击功能，便于使用；系统后端基于轻量级的 Flask 框架，使得开发过程便捷；前端基于 NodeJS 与 React 框架实现，能高效完成页面设计，运行时具有优良的性能；使用 NginX 实现反向代理与负载均衡，能提供高效的服务，并解决浏览器跨域无法访问的问题；使用 MongoDB 作为数据库，作为一种非关系型数据库，拥有灵活的查询和索引方式。后端分别对外开放 gRPC 服务与 RESTFul API，并实现与数据库的交互。前端基于 react-router-dom 实现页内路由机制，实现单次页面请求获取多个页面并进行多页面导航的功能，优化网络开销。

在系统实现方面，系统由身份验证模块、人脸交换模块、攻击模块以及任

务管理模块组成。在身份验证模块中，基于双 token 验证机制实现了对用户的身份验证机制；人脸交换模块在现有 FaceShifter 人脸交换算法基础上对图像进行预处理与后处理的改进，以形成更完整的换脸实现；系统任务管理模块基于多进程框架与任务队列实现了单服务器多任务并发调度机制，实现多任务并发运行。系统的实现过程围绕需求分析、系统设计、系统实现以系统测试四方面进行，涵盖了完整的软件开发测试过程，系统功能健全，具有良好的扩展性与内聚性。

## 1.4 章节安排

全文章节组成如下：

第一章 绪论。针对当前人脸深度伪造及人脸交换技术发展现状进行阐述与总结，介绍了国内外研究现状及现有研究的不足，并对本文工作进行概括。

第二章 相关工作。对所使用的换脸与攻击算法技术及相关系统开发工具进行罗列与介绍，并说明这些技术的使用意义、在该系统中所体现的价值。

第三章 需求分析。对人脸交换任务的市场需求、需要解决的问题等进行详尽的分析，准确定义系统整体及各模块所需功能、性能以及可靠性。

第四章 系统设计。在系统总体设计中根据系统需求提出了基于特征的攻击和基于结果的攻击，并设计攻击方案及具有良好可扩展性、低耦合性、安全性的系统架构。在前后端设计及各模块设计中，分别对系统各组成模块作进一步规划与设计。

第五章 模块实现。系统各功能模块的算法及工程实现，系统层面包括身份验证模块、攻击模块、任务管理模块实现，其中算法又分为换脸算法与四种不同的攻击方法实现。

第六章 系统测试。定义系统质量标准并对系统进行模块测试、集成测试、性能测试等，并展示系统运行效果。

第七章 结语。对本文工作进行总结，分析系统及算法所存在的不足，并对未来研究提出展望。

# 第二章　相关工作

## 2.1　换脸相关技术

### 2.1.1　生成对抗网络 (GAN)

生成对抗网络是由 Ian Goodfellow 等人于 2014 提出的一种无监督机器学习框架[35]，它由一个生成器 $G$ 判别器 $D$ 构成，使用一个随机向量 $\boldsymbol{z}$ 作为生成器 $G$ 的输入，由 $G$ 生成候选样本，再输入判别器 $D$ 并由判别器根据随机混合的生成样本与样本空间 $p_{\text{data}}$ 中的真实样本 $\boldsymbol{x}$ 进行分类，判别样本的真伪。其中判别器的训练目标是尽可能区分样本为真实样本还是由生成器生成的假样本，而生成器的训练目标则是使其生成的样本能混淆判别器，使其错误分类为真实样本。给定一数据集用作判别器的初始训练数据，使用训练样本训练判别模型直到它达到可接受的准确度，然后生成器以生成能让判别器误判的样本为目标进行训练，该过程交替进行。生成器的输入一般是从预定义的潜在空间中的随机采样，由生成器合成的样本由判别器进行评估。两个网络使用相互独立的反向传播，以使得生成器能生成更好的样本，而判别器能更好地区分。在训练过程中，训练 $D$ 以最大化正确区分样本标签的概率，并最小化 $\log(1 - D(G(\boldsymbol{z})))$ 的值，其优化目标如下：

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(x)] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \qquad (2\text{-}1)$$

许多效果较好的人脸交换模型基于 GAN 实现，由自动编码-解码器作为生成网络，生成换脸样本，并由判别器对所生成的样本真伪进行区分。CycleGAN 是 GAN 的一种改进，它是一种图像翻译模型，包含两个生成器 $F, G$ 以及两个判别器 $D_X, D_Y$，两个生成器可以实现图像在两个图像域 $p_{\text{data}}(x)$ 和 $p_{\text{data}}(y)$ 之间的双向转换。CycleGAN 的损失包含两部分，其中一部分是生成对抗损失，另一部分是新引入的一种循环损失。CycleGAN 的整体损失函数

如下：

$$\mathcal{L}(G,F,D_X,D_Y)=\mathcal{L}_{\text{GAN}}(G,D_Y,X,Y)+\mathcal{L}_{\text{GAN}}(F,D_X,Y,X)+\lambda\mathcal{L}_{\text{cyc}}(G,F) \quad (2\text{-}2)$$

其中循环损失 $\mathcal{L}_{\text{cyc}}$ 通过计算图像经过第一个生成器转换到另一个图像域后经过另一个生成器转换回原图像域后，要使得转换前后的图像差异最小化，称为循环损失：

$$\mathcal{L}_{\text{cyc}}(G,F)=\mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x))-x\|_1]+\mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y))-y\|_1] \quad (2\text{-}3)$$

对抗生成损失表示为：

$$\mathcal{L}_{\text{GAN}}(G,D_Y,X,Y)=\mathbb{E}_{y\sim p_{\text{data}}(y)}[\log D_Y(y)]+\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1-D_Y(G(x))] \quad (2\text{-}4)$$

将图像翻译模型在人脸的两种不同属性域进行训练，例如年轻和衰老，即可得到人脸年龄的互相转换。但需要注意的一点是，这种模型只能用于一对一的转换，即一次训练得到的模型只能用于人脸某两个属性的转换，如果要实现一对多的转换则需要训练多个模型分别对应不同的领域。StarGAN 则解决了上述问题，可以实现一对多的转换，其判别器可用于人脸的多个属性特征进行提取。StyleGAN 是另一种常见的模型，使用了一种新的生成器架构，可以在生成图像中无需监督和随机变化的情况下学习高级属性。

### 2.1.2 FaceShifter 换脸模型

FaceShifter 换脸模型是由 Li 等人提出的一种具有良好泛化能力的换脸模型[36]，该模型由 AEI 网络和 HEAR 网络组成，其中 AEI 网络分为三部分：身份编码器，多级属性编码器，AAD 生成器。使用两张人脸图像作为模型的输入：源图像 $X_s$ 与目标图像 $X_t$，模型的生成结果为换脸后的图像 $Y_{s,t}$，该图像同时包含源图像的人脸身份信息 $z_{id}(X_s)$，以及目标图像的面部属性信息 $z_{att}(X_t)$，即将源图像的人脸身份置换到目标图像人脸中，以实现人脸交换的效果。身份编码器以源图像 $X_t$ 作为输入，提取出源图像的隐空间特征向量 $z_{id}$ 作为输出，表示源图像人物的身份，在本文中使用 arcface 模型实现。多级属性编码器以目标图像 $X_s$ 作为输入，通过多层的编码器提取目标图像的多个属性向量 $z_{att}^i$，每一层都输出一个属性向量，描述目标图像的属性，包括姿态、轮廓、发色、肤色、光照等。AAD 生成器由多层的 AAD 跳层连接块组成，通过逐层增加分辨

率的形式，多次去正规化、多级自适应，每一层以前一层的低分辨率图像作为输入，输出更高分辨率的图像。AAD 生成器的输入包括前两部分所提取出的源图像 $X_s$ 的身份特征以及目标图像 $X_t$ 的多级属性特征，其训练目标是使得生成图像的这两类特征与输入特征尽可能接近。其整体的损失函数表示如下：

$$\mathcal{L}_{\text{AEI-Net}} = \mathcal{L}_{adv} + \lambda_{att}\mathcal{L}_{att} + \lambda_{id}\mathcal{L}_{id} + \lambda_{rec}\mathcal{L}_{rec} \tag{2-5}$$

其中 $\mathcal{L}_{adv}$ 为对抗损失。身份损失 $\mathcal{L}_{id}$、属性损失 $\mathcal{L}_{att}$、重构损失 $\mathcal{L}_{rec}$ 的计算如下：

$$\mathcal{L}_{id} = 1 - \cos\left(z_{id}\left(Y_{s,t}\right), z_{id}\left(X_s\right)\right) \tag{2-6}$$

$$\mathcal{L}_{att} = \frac{1}{2}\sum_{k=1}^{n}\left\|z_{att}^{k}\left(Y_{s,t}\right) - z_{att}^{k}\left(X_t\right)\right\|_2^2 \tag{2-7}$$

$$\mathcal{L}_{rec} = \begin{cases} \frac{1}{2}\left\|Y_{s,t} - X_t\right\|_2^2 & \text{if } X_t = X_s \\ 0 & \text{otherwise} \end{cases} \tag{2-8}$$

## 2.2 攻击换脸相关技术

### 2.2.1 投影梯度下降 (PGD) 攻击

神经网络在很多任务上都表现出卓越的性能，但当数据分布发生略微变化时则并不具有很强的鲁棒性。Goodfellow 等人于 2014 年提出了一种基于梯度的快速符号梯度法 (FGSM) 攻击神经网络模型[37]，通过反向传播算法为输入图片计算一个扰动向量，使模型将输入图像进行错误的分类。其后陆续有研究者提出了不同的改进方案，包括 Kurakin 等人提出的基本迭代法 (BIM)[38]，Dong 等人提出的使用动量法对 FGSM 的改进[39] 等。

投影梯度下降 (PGD) 法最早由 Bubeck 等人提出[40]，被应用于受限优化问题，后来由 Madry 等人将 PGD 方法应用于攻击深度学习模型[41]。PGD 攻击是一种简单有效的白盒攻击方法，被认为是实现对抗鲁棒性最有效的一阶攻击方法之一，即在对抗攻击过程中只利用相对输出结果的损失函数的梯度，只包含模型局部的一阶信息，同时试图找到在特定输入上最大化模型损失的扰动，并保持扰动的大小小于指定阈值。PGD 是快速符号梯度法 (FGSM)[37] 或基本迭代法 (BIM)[38] 的一种改进，FGSM 只沿梯度方向走一步来生成扰动，虽然速度上比 PGD 更快，但只在线性模型上表现好，在复杂网络下的对抗鲁棒性不高。相

比之下，PGD 则在复杂非线性模型上有更好的表现，这是由于复杂模型局部变化较大，就有可能导致 FGSM 的单步攻击法失效，因此 PGD 方法将 FGSM 中的单步改为多步进行。PGD 算法从样本点附近的一个随机扰动点出发，通过多次迭代的方式，每次迭代沿梯度下降一步，并将扰动大小进行约束，直到算法收敛。其计算公式如下：

$$\begin{aligned} L_{\text{adv}}\left(x_t^*\right) &= \text{Loss}\left(C\left(x_t^*\right),\ \text{label}_{\text{target}}\right) \\ x_0^* &= x + \eta, \\ x_{t+1}^* &= \prod_{x+S}\left(x_t^* + \alpha \cdot \text{sign}\left(\nabla_x L_{\text{adv}}\left(x_t^*\right)\right)\right) \end{aligned} \tag{2-9}$$

其中 $C$ 为待攻击模型。也可使用 PGD 进行基于目标的白盒攻击，例如对于含标签的分类模型，PDG 的下降方向是使得输入图像的输出标签 j 向目标标签方向变化，即其损失是关于图像输出标签与目标标签距离的函数。

### 2.2.2 StarGAN

StarGAN 由 choi 等人于 2018 年提出[5]，是一种对原有的单领域到单领域图像转换算法的改进，原有的许多 GAN 模型无法使用单个网络实现单领域图像转换至多个领域的任务，需训练多个模型分别对应不同领域才能实现，而 StarGAN 模型可以只使用单个生成器完成单领域到多领域的映射。

StarGAN 模型基于传统的生成对抗框架，由生成器 $G$ 与判别器 $D$ 构成，其生成器的训练借鉴了 CycleGAN 的思想。在训练中首先将原始图片和目标领域标签输入生成器 $G$，训练使生成器学习将原始图片转换至目标领域，同时训练使得判别器 $D$ 认为该生成图像是真实的，然后将目标领域下的生成图片和原始图片领域标签一同输入生成器 $G$，将目标领域下的图片再次转换回原始领域，使得重构后的图片与原始图片差异最小化。判别器 $D$ 以生成图像或真实图像作为输入，输出包括图像的真实度 $D_{src}(x)$，以及图像的领域标签 $D_{cls}(x)$。其中，用于优化生成器 $G$ 的损失为：

$$L_G = L_{adv} + \lambda_{cls} L_{cls}^f + \lambda_{rec} L_{rec}, \tag{2-10}$$

其中 $L_{adv}$ 为对抗损失，用于度量所生成图片被判别器正确区分的难易程度：

$$L_{adv} = \mathbb{E}_x\left[\log D_{src}(x)\right] + \mathbb{E}_{x,c}\left[\log\left(1 - D_{src}(G(x,c))\right)\right] \tag{2-11}$$

$\lambda_{cls}$ 为分类损失系数，$L_{cls}^{f}$ 为生成图像的分类损失，用于计算生成图像分类与目标领域标签 $c$ 之间的差异：

$$L_{cls}^{f} = \mathbb{E}_{x,c}\left[-\log D_{cls}\left(c|G(x,c)\right)\right], \tag{2-12}$$

$\lambda_{rec}$ 为重构损失系数，$L_{rec}$ 为重构损失，用于计算源图像 $x$ 经生成器 $G$ 转换至目标领域 $c$ 后再转回原始领域 $c'$ 所得图片 $G(G(x,c))$ 与原始图像的差异：

$$L_{rec} = \mathbb{E}_{x,c,c'}\left[\|x - G(G(x,c),c')\|_1\right]. \tag{2-13}$$

用于优化判别器 $D$ 的损失为：

$$L_D = -L_{adv} + \lambda_{cls}L_{cls}^{r}, \tag{2-14}$$

其中 $L_{cls}^{r}$ 为真实图像的分类损失：

$$L_{cls}^{r} = \mathbb{E}_{x,c'}\left[-\log D_{cls}\left(c'|x\right)\right]. \tag{2-15}$$

StarGAN 生成器 $D$ 的网络结构是在 CycleGAN 基础上的改进，由 2 个降采样卷积层、6 个残差块、2 个转置卷积层组成。StarGAN 在本文工作中的优势在于其一对多领域的图像转换与领域识别，使用其判别器可以同时提取人脸图像所包含的多个语义特征，并能根据标签损失计算梯度对前置模型进行攻击。StarGAN 的生成器可用于实现图像领域间的转换，与 PGD 攻击方法结合可改变图像所包含的语义特征。

## 2.3 系统相关技术

### 2.3.1 Flask 框架

Flask 是一个使用 Python 编写的轻量级 Web 框架，由于其核心简单、开发便捷、可扩展性强等特点，且基于 python 编写，更易于调用深度学习框架执行系统所需各算法，因此适合作为本文系统的后端框架。Flask 与其他大型框架如同样基于 python 的 Django 相比，Flask 不需要特定的工具或库，不对数据库、模板引擎等进行限制，使得开发者可以更灵活地编写应用程序与扩展，并自由选择配置。Flask 本身不包含数据库抽象层、表单验证工具或其他通用功能组

件，在使用时可根据需要自行添加第三方组件或扩展，例如用户帐户管理和身份验证、静态代码检查、表单处理、会话管理、缓存支持等。Flask 主要包含 5 个依赖项，其中 Werkzeug 提供路由、Web 服务器网关接口等功能；Jinja 提供用于渲染页面的模板语言，能在沙盒环境执行不受信任的模板，包含模板集成功能，具有模板即时编译、异常调试便捷、语法可扩展性强等特点，同时在 HTML 模板中编写的代码会编译为 python 代码；MarkupSafe 会在渲染模板时将不受信任的文本生成转义字符以避免注入式攻击，使得运行安全；ItsDangerous 进行数据签名以保护会话 Cookie；Click 提供命令行集成功能[42]。

虽然 Flask 内置了 web 服务器，但它需要放在真正的 web 服务器之后，并通过 WSGI 协议与 Flask 通信，不适合用于生产环境，因此通常可以选用 gunicorn 作为 Flask 的扩展。gunicorn 同样也是基于 WSGI 的 HTTP 服务器，它使用 pre-fork 模式，可在多个线程上运行多个服务器。对于关系型数据库扩展可使用 SQLAlchemy，并使用相应数据库驱动，例如 psycopg2 用于 PostgreSQL，MySQLdb 用于 MySQL 等，此外还可添加 Alembic 扩展作为轻量级数据库迁移管理工具。对于非关系型的数据库可使用 PyMongo、dynamo 等扩展，分别作为 MongoDB 以及 DynamoDB 等非关系型数据库的接口。此外，Flask 还可搭配 celery 与 Redis 创建分布式异步任务队列，使用 Flask-WTF 用于表单验证，使用 Flask-limiter 限制 Web 请求速率等。

与 Django 相比，Flask 更适合于快速构建中小型项目，具有易于配置、部署灵活等特点，而 Django 则更适用于大型项目，易于管理及维护，但其学习曲线上升更为缓慢。因此在不需要使用多数 Django 的功能时，选择 Flask 框架更合适。使用 Python 和 Flask 的优点在于它在 Web 开发中可进行快速原型设计和动态语义，同时也可以将新模块添加到 Flask 中以扩展其核心功能[43]。

### 2.3.2 双 token 验证机制

用户在经过身份验证后使用系统时涉及到会话保持与限制资源访问权限的问题。常见的会话保持解决方法包括 session、token 与 cookie。session 机制在服务端维护用户登录状态，其缺点是消耗服务器资源，在扩展服务器数量时需要将 session 状态同步到多台服务器中，增大服务器开销。token 机制是 session 的改进，在用户登录时生成一个 token 并发送给用户，当用户访问资源时将 token 发送给服务器验证其权限。其缺点是使用明文 token 易受攻击，并且当 token 有效期结束后用户需要重新登录，无法延续会话长度。因此在双 token

验证机制中采用访问 token 和刷新 token 两个 token 进行，其中访问 token 用于一般的身份验证，并存放于浏览器内存中，供随时使用；刷新 token 用于在访问 token 有效期结束后发送给后端重新获取一组新的 token，刷新 token 存放于 HTTP-Only 的 cookie 中，这种 cookie 的特点是无法被 Javascript 读取，因此是安全的。刷新 token 同样具有有效期，但其有效时间相对更长。

### 2.3.3 Nginx

Nginx 是一个提供 Web 服务、反向代理、缓存、负载均衡等功能的开源 Web 服务器[44]，最初是一个专为实现高性能和稳定性而设计的服务器。当服务端存在多个节点时使用 Nginx 作为反向代理节点可解决用户跨节点请求资源时的跨域问题，当服务端需要对同一功能部署多个服务器节点时使用 NginX 可起到负载均衡作用，并且可减少对外开放的端口数与 IP 数量。Nginx 可处理静态文件、索引文件以及自动索引，并可对打开的文件描述符进行缓存，通过缓存加速反向代理。NginX 还能实现负载均衡功能和容错机制，加速支持远程 FastCGI 服务器的缓存等功能。Nginx 提供多种过滤模式，包括流式压缩、分块响应、合并请求、图像大小调整等过滤器，用于对经过的数据进行处理以优化效率。[45]。

当下互联网中的应用服务需要同一系统的多个副本同时运行以实现高并发及高可用机制，大量应用的水平扩展需求使得高性能负载均衡成为必不可少的组成部分。因此 Nginx 提供了包括 HTTP、TCP、UDP 负载均衡等多项技术，其中包含多种负载均衡算法，并能监控上游服务器的连接状态或对客户端的响应，从而检测到上游服务器的异常，以适应现代网络架构的需要[46]。许多网络架构通常采用无状态应用的形式，其状态被储存在共享内存或数据库中，但也有部分交互式应用由于数据量过大或其他原因需要将用户的会话状态储存在应用服务器本地，并需要保证用户的一系列请求都发送至同一台服务器，同时服务器需要等到会话终止才释放，这在 Nginx 或 Nginx Plus 中都得到很好地解决。

Nginx 的流量控制功能能调节流量的速率、连接数、带宽限制等，并实现 A/B 测试功能。Nginx 中包含的缓存加速机制将请求取得的响应进行缓存，以便在下一次请求时利用，其中提供了设置缓存键、缓存绕行、客户端缓存、大文件切分等功能。Nginx 可对客户端请求进行验证，其中包含基础 HTTP 验证、子请求验证、token 有效性验证等功能。Nginx 中的安全控制功能包括基于

IP 地址的通行权限控制、跨域资源共享、上游流量加密、重定向 HTTPS 请求及对特定位置进行保护等[45]。

### 2.3.4 React 框架

React 是一个由 Facebook 开发的开源、高效、灵活、基于组件的 JavaScript 库，用于构建网页前端用户界面，负责移动和网络应用的视图层[47]。它包含 Javascript 扩展语法 (JSX)，组件可重用性、虚拟文档对象模型等特性，允许用户构建和管理有状态的封装组件，将组件进行组合以制作复杂的界面。React 有快速的学习曲线，具有代码可重用性高、测试方便等特点。与其他多向数据流的框架不同，React 框架将所有数据对象绑定为自顶向下关系，子组件的变化不直接影响父组件的变化，因此为单向数据流结构，数据在不同层与应用状态之间进行单向的流动，事件更新则沿相反方向移动。

在 React 中，每个组件分别维护属于自己的状态，并在其状态更新时高效率地对其下游组件进行更新，同时刷新 UI 的显示。React 使用虚拟 DOM 解决低效率刷新问题，在该机制中对每一个 DOM 对象都会有一个相对应的虚拟 DOM 对象，所有虚拟 DOM 对象在内存中以一个树型数据结构储存，每个虚拟 DOM 对象是树上的一个节点。对每个对象使用观察者模式监听状态变化，每当有修改或更新时就会更新 DOM 树，并与原先的树计算差异，实现分批发送形式的更新，而非针对组件状态的单个更改发送更新，每批次更新会对真实 DOM 对象进行修改，并更新浏览器的显示。在这种机制下，编码时每作一次改动 React 仅更新显示实际更改的组件，使得 React 具有很高的刷新效率，[48]。

React 支持添加 React Router，该路由库能快速向应用中添加视图和数据流，实现页内路由机制，保持页面与 URL 的同步。使用 React Router 可以使得包含多个 URL 的 UI 组件解析 URL 时变得智能化，以简化代码逻辑。

### 2.3.5 MongoDB

MongoDB 是一种非常流行的面向文档的非关系型数据库。它广泛用于与 AWS、Azure 和许多其他数据源协作进行应用程序开发和运行，与关系型数据库相比它具有更良好的可扩展性，允许存储和查询大量数据，并提供了二级索引、区间查询、排序、分析框架、地理空间索引等功能。在 MongoDB 中通过

适当的索引和处理功能可以更好地执行查询，使用临时查询可进行实时分析和优化数据处理，通过强大的复制功能可以提高数据可用性和灵活性，数据分片机制允许拆分大数据块以实现分布式和更快的查询执行过程。传统的关系型数据库在处理有限数量数据，且需要进复杂的查询时有良好的性能，而对于海量数据的持久性储存则不如非关系型数据库。非关系型数据库不使用关系数据库管理系统，数据不像关系型数据库一样存储在表中，有固定的模式，而是使用丰富的数据模型，同时使用标识键，可以根据分配的键找到数据[49]。

MongoDB 的大部分数据存储在内存中，在查询时将直接从内存中读取数据，减少磁盘读写的时间消耗，与一般关系型数据库相比速度上有显著提升。MongoDB 的环境配置十分简便，比关系型数据库要更快速搭建与运行，并且具有动态的示意图架构，支持非结构性数据和存储。在处理大量数据集时 MongoDB 采取分片机制，将数据集分为多部分，并分布式存储到多台服务器中，每个分片环境中的操作由一个轻量级进程 mongos 负责，该进程会根据分片键直接查询到正确的分片，这使得数据库拥有良好的水平扩展性以及更好地完成负载均衡。除此之外 MongoDB 对即席查询进行了优化，同时支持字段查询、区间查询、正则表达式查询等，使得用户有更灵活的选择。

除此之外，MongoDB 还自带大规模负载均衡功能，开发人员无需自行添加额外的负载均衡服务器。MongoGB 支持数据的并发读写，使用并发控制和读写锁来处理同一数据的多个并发读写请求，从而确保数据的一致性。MongoDB 通过主从复制机制，将数据同步存放在多个服务器中，实现读写分离以及数据的高可用性、安全性，并允许在出现故障时恢复数据。数据的恢复通过一个主节点与多个副本节点服务器来实现，用户对数据的交互均由主节点完成。同时，MongoDB 还实现了副本集机制，当主节点失效后副本节点接管主节点成为新的主节点，从而进一步提升数据库的可用性。

# 第三章　需求分析

随着计算机视觉与深度学习等技术的快速发展，人脸编辑、人脸交换技术逐渐出现在大众视野中，也拥有了一些市场。然而换脸技术的滥用会造成名人、敏感图像的篡改及传播，造成对个人隐私权、名人肖像权的侵犯，对明星、政治人物等产生不良社会影响，也使得基于人脸识别的身份验证系统的安全性受到挑战，因此设计一个安全、具有隐私保护功能的换脸保护系统，保证网络数字信息的规范化十分有必要。现有的一些技术手段，如检测伪造人脸技术，能轻易被对抗攻击方法绕过；在换脸系统中植入敏感图片检测机制，只能涵盖检测系统所用于识别的人物，并不能检测大量其他未经识别的人物。因此一种较为有效的手段是对原始图像进行保护，破坏换脸效果从而在根源上阻止换脸的发生。

本文所设计的系统需要实现两方面的需求：从保护图像角度出发的算法设计需求，以及系统设计层面出发的系统需求。本章首先探讨该系统的可行性，然后分别从功能性需求与非功能性需求两方面对实现系统所需要满足的条件进行分析与归纳。

## 3.1　可行性分析

本文所实现人脸保护系统应具备攻击换脸模型并破坏换脸效果的能力，在算法上前人已取得一些研究成果[31]，实现了破坏合成人脸的效果，破坏效果显著，并且所使用的基于 FGSM、PGD 等的白盒攻击方法能应用于其他模型，具有很好的通用性，此外在开发本系统之前已对包括对图像语义的攻击在内的攻击方式进行了实验验证，证明其有效性，因此系统算法功能可达到预期效果。

在经济可行性方面，本文所实现的人脸保护系统以 Web 应用的形式实现，仅需要部署在云端服务器中即可对外提供算法服务，并基于 github 进行版本控制，能很方便地部署；系统采用模块化设计，其高内聚性和低耦合性使得后期对系统进行维护也不需要作过多改动，因此系统的运营成本与维护成本都会很低。系统的数据吞吐中占比较大的部分只有图片的上传和下载，不包含视频等

大型流式数据的传输，因此并不要求较高的网络带宽。

在技术可行性方面，系统使用 Flask 作为后端框架，并基于 Python 语言开发，前端基于 NodeJS 和 React 框架，主要使用 Javascript，数据库使用 MongoDB，并使用 NginX 作为反向代理服务器，所有技术均广泛使用且形成成熟的技术生态，前后端技术框架均包含完整的文档、官方或第三方开发的模块或插件，这使得开发过程中不会因遇到关键的技术瓶颈或技术难点而导致开发成本急剧增加。前后端所使用的 NodeJS 与 Flask 技术框架均包含热更新机制，更新代码后无需重启后台即可将新代码应用于系统，可大幅缩减开发成本。

综上分析，该系统具有良好的算法功能可行性、经济可行性、技术可行性。下面进一步分析系统的功能性需求与非功能性需求。

## 3.2 功能性需求分析

随着市面上换脸应用的流行，滥用换脸技术的现象也不断出现，为了在使用换脸技术的同时也能避免换脸造成肖像权与隐私权的侵犯，需设计一种具有隐私保护功能的系统。从整体上，本文所设计的系统需要实现的主要功能有以下几个方面：

- 实现人脸保护功能，防止图像被换脸。人脸保护功能是系统的核心功能，系统需要实现完整的人脸保护机制以从根本上防止换脸带来的不良影响，通过使用一种对换脸模型进行攻击，在原图片基础上添加扰动的方法，使其经过换脸所产生的图像发生损坏。
- 实现换脸功能，换脸功能不仅作为攻击算法的依赖项，还能为用户验证攻击效果提供服务。
- 实现用户身份验证和用户访问控制，包含用户验证功能的系统在整体上更安全，使得个人信息非公开化，对避免个人隐私的泄露有重要作用。
- 实现异步任务机制，用户所提交的请求以任务形式暂存在任务队列中，在后台进行计算，并以用户查询的形式获得计算结果。由于攻击算法耗时长短不一，无序地提交任务会使得系统不稳定，因此需要以任务队列的形式暂存所有待处理的任务，使系统资源利用更高效。

基于上述主要功能需求，系统采用前后端的形式，用户在浏览器中登录系统，上传图片并提交任务，在后端将创建任务并进行计算，在计算完毕后用户提出查询请求，获得计算结果。

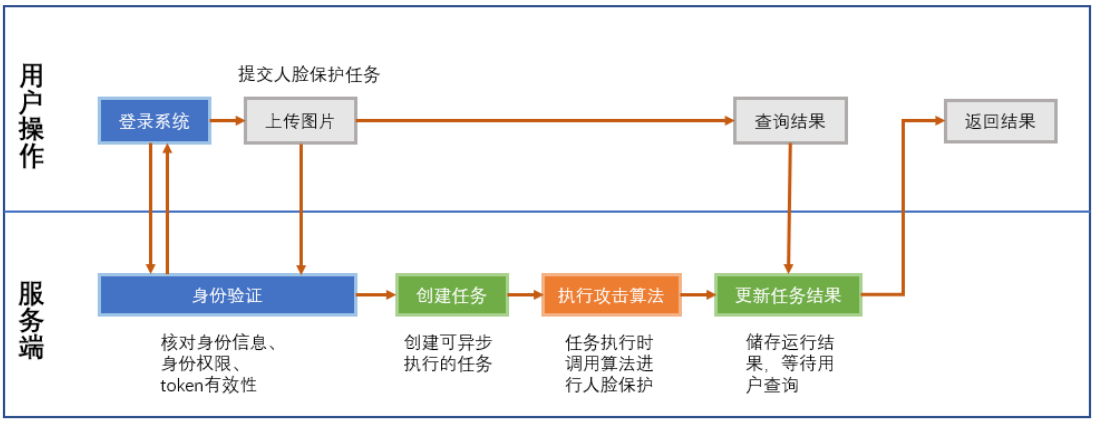



图 3-1: 人脸保护过程图

图3-1展示了用户使用系统进行人脸保护所需包含的过程，其中分为用户操作与服务端处理两部分，首先需要用户登录系统，登录时与后端进行身份验证，服务端在检查用户信息合法性后，授权用户访问系统与算法功能。用户在系统攻击页面上传需要保护的图片并选择攻击参数，向后端提交任务。所有用户对后端 API 的调用均需经过后端验证，后端在检查 token 有效性以及用户身份拥有执行权限时放行，并将算法请求交至任务管理模块处理。任务管理模块分配资源并创建相应的算法任务，分配进程异步执行任务。创建后的任务每次状态的更新都将相关任务信息同步至数据库，用户可随时查询任务状态，包括正在运行、运行成功、运行失败等状态。当任务执行完毕后将结果返回给任务管理模块，将人脸保护后的图片储存在服务端的同时更新任务信息，并以图片链接的形式记录任务结果。当用户查询任务时可获得任务执行结果所给链接，通过该链接即可访问算法执行结果图。

下面进一步对系统算法方面的功能与其他辅助性系统功能作详细的需求分析。

### 3.2.1 算法功能需求

在系统算法功能方面，为了加强对人脸数据隐私性的保护，系统中需要实现对人脸图像的保护功能，即使得指定图像经过换脸后图像出现明显破坏痕迹。系统最核心的部分是对换脸模型的攻击，首先需要完成换脸系统对人脸图片的保护，使其不受模型换脸的影响，从而在一定程度上限制换脸行为。人脸保护功能通过对用户图像添加肉眼不可见的噪声，使得该图像经过同类人脸交换模型后得到的生成图像质量受到破坏，表现出明显差异，例如使得生成图像

特定区域出现黑块，性别、年龄等属性发生明显变化等，其中发生变化的属性可人为指定，例如指定性别翻转、发色变化，或指定水印图像、标记区域等，并且这种标记是由换脸模型自动生成出的缺陷，并非人为添加，因此其效果取决于换脸模型的优劣，但其缺陷必须使得人眼可辨认，最终需要使得换脸后的人脸受到破坏，例如身份无法辨认，或包含明显篡改痕迹，以起到保护被换脸人隐私的作用。同时保护算法还需具有一定的泛化性能，即对同类型但不同模型的换脸算法都能表现出较好的效果。人脸图像保护功能要求提供一种迅速、短期出结果但不要求结果高质量的方法，以及多种需经过一段时间训练模型，但能得到更丰富结果、更具实用意义的方法，从而满足用户多层次的需求。

为进一步提升人脸保护的效果，系统应包含一种可控制人脸保护效果的算法，用户可指定经保护的图片再次经过换脸模型所得换脸图片所发生的语义变化形式，其目的是提升用户交互性与自定义程度，提升用户体验。实现用户数据的安全形式储存，系统实现用户所创建任务的合理调度及用户对任务进度、历史记录的查询。将人脸交换服务、人脸保护服务等算法功能以多种接口形式对外提供，使得前端有更自由的形式选择。

图像保护功能应当具有良好的用户自由度，用户可指定图像的保护方式、保护效果、保护强度，体现在算法上则对应了不同的保护算法，设定算法迭代次数、数据量等条件可对算法效果进行自由控制，更好的算法可能需要更长的运行时间。除此之外，经过保护的图像所生成的换脸图像结果也应具有良好的自定义性，用户可指定换脸结果的损坏形式，包括全面损坏、部分属性变化、局部区域出现标记等情况。

系统不仅需要提供人脸保护功能，还需对结果进行有效性验证，因此系统还需实现换脸算法，既可作为系统内部验证也可使用户自行验证。换脸功能需由用户提供两张人脸图像，在目前大多数换脸系统中都采取一张用户个人照，另一张为用户自选名人明星照的做法，但使用名人照存在侵犯肖像权的问题，因此需对此类行为加以限制。对此，在系统功能方面包括以下方面：使用目前公开的 FaceShifter 换脸模型生成人脸供用户选择待换脸的图像；实现人脸交换功能。用户选择两张人脸图像上传，由服务器合成以其中一张图像为背景，另一张图像为面部特征的图像并返回；实现人脸特征修改功能。用户上传一张人脸图像，服务器返回一批修改了不同面部特征的图像，由用户在多个结果中自行选择，或选择重新生成新的批次；实现人脸图像保护功能，使得换脸后的图像损坏或出现明显辨识特征，避免同类模型对其进行篡改。用户上传一张人脸

图像，服务器针对换脸模型进行攻击，在人脸图像上添加肉眼不可见噪声，使得该图像经过换脸后所得图像包括以下几种改变方式：完全或部分损坏；面部某一属性或多种属性发生明显变化；面部出现明显标记。同时，要使得其改变方式可人为进行控制和调整，即可由用户或管理者指定图像破坏的程度、形式、面部发生变化的属性种类、面部出现标记的样式等。对于人脸交换算法，还要求算法具有高质量性、高鲁棒性，能适应多种场景下的任务，具有较好的泛化性能，经过算法交换的图像要求不能出现明显拼接痕迹。

### 3.2.2 系统功能需求

为使得该系统被更多广泛用户使用，系统应设计成 Web 应用形式，系统整体由前后端构成，其中前端完成用户在浏览器中进行的交互操作，以及访问后端所提供的算法服务功能。后端包含身份验证模块、人脸交换模块、攻击模块和任务管理模块。

系统应将所实现算法以服务形式对外开放，包括人脸交换服务与人脸保护服务，可实现个人图像与他人图像的面部交换功能，以及对指定人脸添加扰动进行保护，并且可以选择保护后换脸图像破坏的形式。需包含用户图像的上传、参数选择、后台任务处理以及结果储存与用户对结果的查询。用户使用人脸保护服务可实现对个人图像的保护，防止他人使用同类换脸模型进行交换，同时向用户展示可靠的保护效果。用户可自由根据报告选择是否进行进一步保护。用户可根据需要自行指定人脸保护的迭代次数、效果等，同时系统应能处理要求计算较长时间的任务，将任务集中管理并由用户查询任务执行结果。

除主要算法功能外，系统还应具备接入用户所需运营功能，包括用户身份认证机制、数据管理、多任务请求管理机制。用户身份认证机制用于实现用户注册、登入、登出功能，实现用户权限管理功能，实现账号管理机制，通过维护用户级别以实现用户访问控制，增强系统安全性。只有经过身份验证的用户才能进行系统资源、服务的访问和使用，以及个人信息的管理。对于管理员权限的用户可以对用户、系统资源和系统数据进行管理和修改。用户可以访问、管理历史个人数据及访问任务提交记录。用户拥有其个人数据、任务提交历史以及任务查询功能。

系统需要具有一定的安全机制，包括用户身份验证和用户访问控制，使得用户所上传的个人数据对外界及其他用户隔离，系统功能拥有访问限制。因此，基本的系统设计需包括一套完整的人脸交换流程、基于攻击模型实现人脸

图片的保护功能、用户管理机制。其次，系统还需以服务的形式对外提供算法功能，并能支持多数据并发处理，来满足大量用户使用需求。

身份和访问管理是一个分别处理身份验证和授权的概念。身份验证用于识别用户身份，而授权确保用户有权限执行特定操作。对于用户权限管理功能，要求实现基于角色的访问控制，对所有功能实行一定颗粒度的访问权限等级划分，对管理员身份可提供所有权限，对普通用户则限制于对部分资源的访问，包括个人信息管理、基本任务的提交与查询等普通级别权限。在该系统中，为最大限度为用户提供服务，并实现管理功能，用户在使用软件时需首先进行注册与登录，并将用户身份根据不同管理权限级别进行划分，同时允许匿名用户对服务的使用。对于注册用户，将支持个人照的上传及公开发布，对于公开的照片将自动进行图像保护处理。由于匿名用户无法上传个人资料，因此对其不提供图像保护服务。

用户的登录状态应在一段时间内保持，以使得当前活动的用户暂时性离开或刷新页面后所进行的操作不会丢失，会话不中断。而离开较长时间后的用户登录状态应失效，需重新登录，以保证系统的安全性。

## 3.3 非功能性需求分析

系统的非功能性需求包括系统的运行性能、安全性、健壮性、可用性、可扩展性、可维护性等方面。

（1）系统性能

在系统性能方面，该系统在实现人脸交换等基础服务时需保证算法的即时反馈性，保证基本功能算法的计算效率。由于用户端算力有限，且模型较大，因此需在服务端完成计算。基础换脸算法需保证及时反馈性，用户需在较短时间内获得输出结果。图像保护算法时间则可根据保护程度来决定，较高的保护程度可计算更长的时间，并可由用户控制。对于保护算法，则无需保证性能，但需要保证保护算法的可靠性。

由于换脸算法及保护算法需要深度学习框架支持，且运算消耗大，因此应当建立任务管理机制，根据算法执行效率不同，执行快的算法应优先执行，时间较长的算法应设置较低的优先级，推迟执行。应减少跨系统的交互，系统只对自己业务域的数据库进行更新，不跨域操作其他应用的数据。减少硬盘 IO 和网络的访问，将多次的调用整合在一次操作中完成，减少 IO 资源的浪费。

避免在操作数据库或者外部接口时候放在循环里面，尽量做成批量接口调用。系统间使用只读调用，系统间的修改用事件或者消息来实现。使用增加形式对系统的配置文件、数据库字段修改，或其他逻辑进行操作。单个算法任务执行出错应不影响其他任务执行，任务之间应互相隔离。

对于系统响应时间，普通的操作例如用户注册、登录、任务状态查询、页面导航等操作应控制在 1 秒内完成，换脸算法包括完成算法运行、结果返回和在页面中显示等步骤应控制在 10 秒内完成，对于耗时较长的图像保护算法则没有时间要求。

（2）可扩展性

系统需保证其可扩展性，当会话数量或计算数量增加时能根据需要增加相应计算节点数以满足服务需求。系统结构清晰，使用命名规范，并将参数配置进行集中化。系统应采取模块化设计，每个系统事件应分别由某一特定模块完成，每个模块应具有良好的输入输出数据定义以及功能定义，所定义的数据对模块运行应当是必要的。设计时应尽可能抽象化实现细节，以最小化其他模块的调整对这一模块的影响。系统中不同算法分别使用不同的模块，应做到灵活加入新的算法作为新的模块，为系统提供更高的扩展性。

（3）健壮性

系统的健壮性通常由异常处理机制来完成，需要使得系统能在包含无效输入的场景或严苛运行环境条件下仍然能正常运行，包括系统资源不足、用户高并发场景、服务器宕机等情况，当请求出错或发生异常时系统应具备异常处理能力，模块单元在检测到无法处理的异常时应抛出异常，由上层模块处理，使得进程不发生崩溃。对于无效的用户请求也需根据不同的错误情况分别返回不同的 HTTP 错误数值。最后，还需对系统进行大量测试，测试样例需包含各种易出错的情形和高并发等场景，从而使得系统能应对最大限度的不同场景并能正常运行不崩溃，以保证系统的健壮性。

（4）可维护性

要增加系统的可维护性，在设计模块功能时需要保证各模块的高内聚性和低耦合性。模块的内聚性表示模块内部实现的功能的单一性，模块耦合性表示模块之间联系的密切性。一个易于维护的系统需要通过松耦合来减少模块或类之间的互连性，使得在修改一个模块时最大限度减小对其他模块的修改。同时还需保证系统各模块的高内聚性，一个设计良好的模块应该有一定的用途，所有元素都应与单个任务相关联，使得模块具有高内聚性。

在系统模块化设计中，每个模块具备逻辑上单一与独立的功能，并具备良好定义的接口。设计时需保持每个模块的大小适中，易于维护。所有模块构成有向无环图，形成底层模块相互不具有依赖关系，上层模块依赖于下层模块的关系。应尽可能减少其下游模块调用的深度，对顶层模块保持高扇出，即尽可能多的直接下属模块，下游模块保持低扇出以及高扇入。每个模块功能应可预测，应具有单一入口和出口，每个模块的作用域仅限于该模块中。

## 3.4 本章小结

本章首先从功能可行性、经济可行性和技术可行性三个角度对系统可行性进行分析，接着分析了系统各组成部分的功能性需求、非功能性需求对系统的需求进行综合性的分析，从多角度描述了系统整体的设计目标及注意点，为系统的后续设计与开发工作奠定基础。

# 第四章　系统设计

本章分别从攻击算法设计、系统总体设计、前后端设计、各模块设计进行系统设计，在攻击算法设计中系统将从基于特征的攻击和基于结果的攻击两方面设计攻击方案，在系统总体设计中定义系统架构及各组成部分的功能。在前后端设计中分别介绍前端与后端的设计思路与数据流程，在模块设计中根据系统需求分析所提出的各项要求，分别对身份验证模块、换脸模块、攻击模块以及任务管理模块进行详细设计。

## 4.1　攻击算法设计

目前多数基于对抗攻击的工作是围绕对模型输出图像的破坏程度高、原图中添加的扰动少展开，而对于从其他角度设计攻击目标的工作较少。本文基于此，在对换脸模型进行攻击时，将不局限于传统的以破坏换脸图像为目标的攻击，而是对模型不同层面进行攻击，根据攻击的目标不同分别提出了基于特征攻击和基于结果攻击两类方法，实现了不破坏换脸结果但从特征上对人脸的保护，以及多种形式破坏换脸结果的人脸保护方案。下面将对两种方法作进一步说明。

### 4.1.1　基于特征的攻击

基于特征的攻击方法不直接攻击整个模型，而是对模型中的特征提取部分进行攻击，以使得模型的对人物的特征提取失败。当提取了错误的特征后，后续采用任意结构的模型进行换脸，所得的换脸图像都将呈现出另一个人物特征的效果，虽然没有破坏换脸结果，但使得原图像人物的特征得到了保护。设模型可分为 $F$、$G$ 两部分，其中由 $F$ 提取图像特征，由 $G$ 根据特征生成换脸图像，则换脸图像 $Y_{s,t}$ 表示如下：

$$Y_{s,t} = G\left(F(X_s), X_t\right), \tag{4-1}$$

以模型 $F$ 提取的特征 $F(X_s)$ 作为攻击目标对 $F$ 进行攻击，使得图像 $X_s$ 提取所得特征发生变化，则对任意模型 $G$，换脸图像 $Y_{s,t}$ 都不会包含 $X_s$ 原本的特征，因为换脸过程的特征层就已经被破坏。以 FaceShifter 模型为例，在该模型中，上述 $F$ 为模型中对源图像进行身份特征提取的身份编码器，即 arcface 模型。攻击该模型，在原图像中添加扰动后，能破坏图像所提取的身份特征。而当提取了错误的身份特征之后，使用该身份特征进行换脸后的结果必然呈现出另一个人物身份的效果，从而很好地保护了被换脸人物的身份，提升了图片的保密性。

系统针对 FaceShifter 特征层进行攻击，通过攻击 FaceShifter 中的身份编码模型，能达到保护源图像 $X_s$ 人物身份的目的，其攻击目标是使得将添加扰动后的源图像 $X_s + r_{adv}$ 经过身份编码模型所得特征 $z'_{id}$ 与原始身份特征差异最大化。使用 PGD 攻击的训练目标为：

$$\mathbb{E}_{(X_s,X_t)\sim\mathcal{D}}[\max_{r_{adv}\in S} L(\theta, X_s + r_{adv}, X_t)], \tag{4-2}$$

本系统中 FaceShifter 模型采用预训练 arcface 模型提取身份编码。PGD 训练损失设置为源图像攻击前后身份编码均方差损失，表示如下：

$$L(\theta, X_s + r_{adv}, X_t) = \frac{1}{n}\sum_{k=1}^{n} \left\| z_{id} - z'_{id} \right\|_2^2 . \tag{4-3}$$

使用 PGD 攻击的过程为：初始扰动为 0，使用 arcface 模型提取源图像 $X_s$ 身份编码 $z_{id}$ 以及添加扰动后的源图像 $X_{adv}$ 的身份编码 $z'_i d$。计算损失函数 $L(\theta, X_s + r_{adv}, X_t)$，根据反向传播梯度与攻击公式计算扰动 $r_{adv}$，叠加至源图像得到攻击后图像 $X_{adv}$，其中叠加扰动形式为：

$$X_{adv} = X_s + r_{adv} \tag{4-4}$$

基于特征攻击的优势在于，与一般的方法相比，该方法通过攻击身份特征，使得经过特征提取所得人物身份特征发生改变，能保证攻击所得换脸图片中人物身份发生变化，从而无法识别被换脸人身份。攻击过程依赖的模型权重相对更少，因此具有更快的攻击速度，同时具有更高的对抗鲁棒性，能适应其他类型的包含提取特征过程的换脸模型。

### 4.1.2 基于结果的攻击

基于特征的攻击方法不以破坏换脸为目标，而是修改所提取的特征。基于结果的攻击方法将以破坏换脸结果为目标，直接使得换脸失败。但与传统攻击方法不同的是，传统方法以攻击前后生成图像的差异或其他指标作为攻击目标，而该方法以生成图像在语义层面或其他层面产生特定破坏形式为目标，且能改变对换脸图像人物的认知，以实现稳定、特定形式的攻击。传统方法能使得被破坏的图片中出现随机性的斑块，但这些斑块的大小、质量均无法控制。该方法通过在模型后串联一算法，由算法提取所需攻击的特征，并对整体进行攻击，可以实现对换脸破坏结果的控制。设模型为 $G$，模型后添加的目标函数为 $H$，则攻击目标表示为：

$$\arg\min_{\eta} \|H(G(X_s+\eta, X_t)) - H(G(X_s, X_t))\|_2^2 \tag{4-5}$$

其中 $\eta$ 为向源图像 $X_s$ 添加的扰动。经过攻击后添加扰动的图像再次生成的换脸图片 $G(X_s+\eta, X_t)$ 所出现的破坏形式可由目标函数 $H$ 进行控制。设定不同的目标函数 $H$ 可以获得不同的攻击效果，下面分别设计了三种不同的目标函数作为本系统的攻击方案，并基于 PGD 方法进行攻击，其生成图像的破坏形式分别如下：

- 降低换脸所得图像真实度；
- 使图像中央出现黑色区域；
- 改变换脸图像语义特征，如出现胡须、性别变化等。

在以降低换脸所得图像真实度为攻击目标的方法中，通过使得添加扰动后源图像 $X_s + r_{adv}$ 与其他目标图像 $X_t$ 换脸所得图像 $Y_{adv,t}$ 的真实度最小化进行攻击。其训练目标表示如下：

$$\mathbb{E}_{(X_s,X_t)\sim\mathcal{D}}[\max_{r_{adv}\in\mathcal{S}} L(\theta, X_s + r_{adv}, X_t)], \tag{4-6}$$

其中，PGD 训练损失为换脸所得图像 $Y_{adv,t}$ 的负真实度，本系统中使用 StarGAN 判别器获取图像真实度，即 $D_{src}(Y_{adv})$：

$$L(\theta, X_s + r_{adv}, X_t) = -D_{src}(G(X_s + r_{adv}, X_t)). \tag{4-7}$$

在攻击的过程中，根据输入源图像 $X_s$ 与数据集中随机图像 $X_t$ 计算换脸后

图像 $Y_{s,t}$，使用 StarGAN 判别器计算损失函数 $L(\theta, X_s + r_{adv}, X_t)$，根据反向传播梯度与攻击公式计算扰动 $r_{adv}$，叠加至源图像得到攻击后图像 $X_{adv}$：

$$X_{adv} = X_s + r_{adv}. \tag{4-8}$$

除此之外，还可以指定图像特定部分出现黑色区域，通过使得添加扰动后源图像 $X_s + r_{adv}$ 与其他目标图像 $X_t$ 换脸所得图像 $Y_{adv,t}$ 中指定区域出现色块进行攻击，以达到破坏换脸图像的目的。其训练目标为：

$$\mathbb{E}_{(X_s,X_t)\sim\mathcal{D}}[\max_{r_{adv}\in\mathcal{S}} L(\theta, X_s + r_{adv}, X_t)], \tag{4-9}$$

其中，PGD 训练损失描述了 $Y_{adv,t}$ 在指定区域出现色块的程度，根据攻击后所希望产生的效果构造一个 mask，记为 $M$，该 mask 标识了 $Y_{adv,t}$ 中需要发生变化的像素及变化大小，作用于 $Y_{adv,t}$ 以得到损失函数。本系统以产生黑色块为例，展示一种 mask 表示方式：

$$M = \frac{1}{N}\begin{bmatrix} 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & 1 & 1 & \dots & 0 \\ 0 & \dots & 1 & 1 & 1 & \dots & 0 \\ 0 & \dots & 1 & 1 & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \tag{4-10}$$

其中 $N$ 为 mask 中标识为 1 的像素数量，在损失最小化过程中，被标识的像素的优化目标为像素数值最小值，即黑色。PGD 所需优化的损失计算如下：

$$L(\theta, X_s + r_{adv}, X_t) = M \circ G(X_s + r_{adv}). \tag{4-11}$$

使用 PGD 攻击的过程为：根据输入源图像 $X_s$ 与数据集中随机图像 $X_t$ 计算换脸后图像 $Y_{s,t}$，基于 mask 计算损失函数 $L(\theta, X_s + r_{adv}, X_t)$，根据反向传播梯度与攻击公式计算扰动 $r_{adv}$，叠加至源图像得到攻击后图像 $X_{adv}$：

$$X_{adv} = X_s + r_{adv}. \tag{4-12}$$

上述攻击方法在一般的破坏换脸图像基础上，控制了其破坏的形式，使得

破坏痕迹不再表现为随机性较强的形式，且用户能分别选择不同的攻击算法得到不同的具有确定性的攻击结果。通过提取图像的语义，能从图像的语义层面进一步控制攻击结果，使得换脸图中人的外观相貌等语义特征发生改变，且发生改变的部分可以人为指定，是一种可由用户指定编辑形式的攻击方法。

这种方法利用图像的语义可分解性，通过串联被攻击模型与语义判别器，基于 PGD 方法基于语义分类标签损失进行攻击，实现生成图像语义层面上的改变。在对图像语义的攻击中，所攻击的损失为语义判别器输出与目标语义标签的距离，且语义判别器训练时所设定的分类域种类为所要修改的语义类别，以达到控制图像保护效果的目的。用户可从年龄、性别、发色等多种特征中选取需要改变的属性，并提交需进行保护的图片，算法根据用户选取的特征有针对性地进行攻击，并在源图像中添加扰动，使得换脸图像中相应的属性发生变化。其攻击流程如图4-1所示。

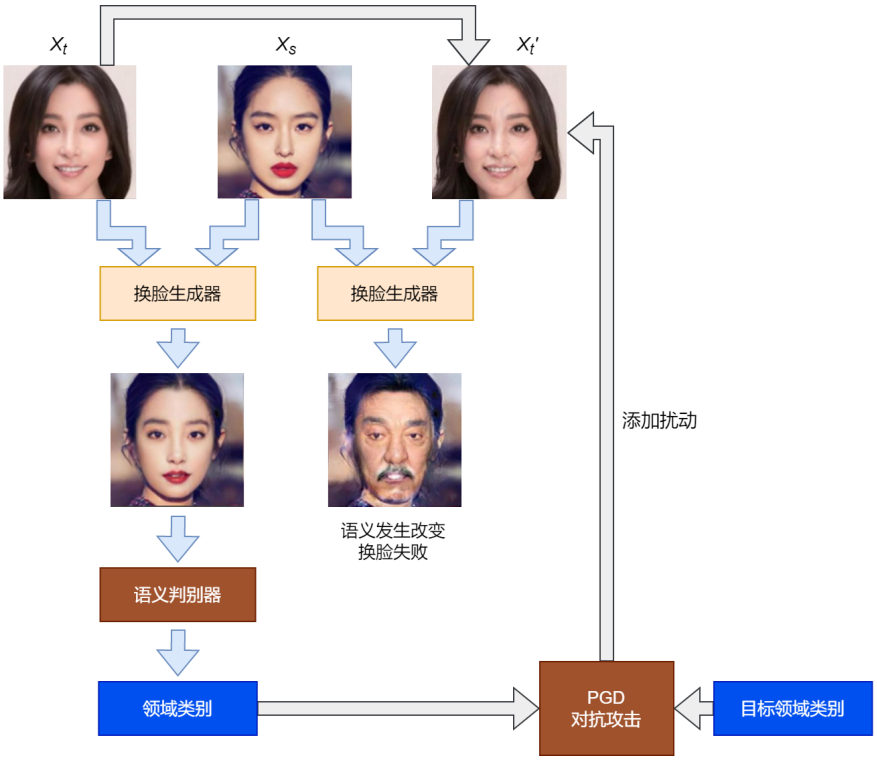



图 4-1: 针对图像语义的攻击示意图

使用 PGD 攻击的训练目标为：

$$\mathbb{E}_{(X_s,X_t)\sim\mathcal{D}}[\max_{r_{adv}\in S} L(\theta, X_s + r_{adv}, X_t)], \tag{4-13}$$

其中 PGD 损失函数定义为换脸后图像经判别器 $D$ 提取的领域标签 $D_{cls}(Y_{adv})$

与目标领域标签 $c$ 的交叉熵损失，在本系统中使用 StarGAN 判别器提取领域标签：

$$L(\theta, X_s + r_{adv}, X_t) = H(D_{cls}(Y_{adv}), c) \tag{4-14}$$

其中 $H$ 为交叉熵损失函数。

上述攻击方法从结果层面进行攻击，分别从降低换脸所得图像真实度、使图像中央出现黑色区域、改变换脸图像语义特征三个角度实现了对换脸结果不同形式的攻击效果。

## 4.2 系统总体设计

根据系统需求，系统使用 B/S 架构实现，基于前后端分离原则，由于系统中算法模块涉及深度学习框架的调用，使用其他语言作为后端语言将对 python 调用做一层封装，尤其不利于模型的一次性加载与多次使用，因此后端框架基于轻量级 Flask 框架开发，对外提供算法服务接口、身份验证 API 及查询 API 等；前端基于 React 框架开发页面，并以 NodeJS 辅助作为前端运行环境，负责对接后端所提供的服务，通过调用后端服务器的算法接口实现用户对算法功能的使用。系统在启动后端时将一次性加载算法模块所有模型，每次收到算法 API 请求时对预加载的模型进行调用，以减少后续调用的开销。前端为 Web 应用，基于 Nodejs 框架处理页面请求，并使用 React 框架进行前端页面设计与开发。收到的用户请求分为两类处理，一类是 API 请求，这类请求涉及调用算法、模型及访问数据库，发送至 Flask 后端进行处理，另一类是用于加载和渲染页面的网页文件请求，由 Nodejs 处理。

后端对外开放的 API 接口包括两种形式：基于 gRPC 的服务与基于 HTTP 的 RestFul API，其中基于 gRPC 的服务采用了 Protobuf 协议，拥有高速序列化能力，可应用于桌面端应用程序、移动端应用中基于视频数据流或图片数据的服务，或一些带宽受限场景等。基于 HTTP 的 RestFul API 则更为常用，可应用于网页前端、桌面端应用程序等更广泛的场景。后端的模块组成包括身份验证模块、人脸交换模块、攻击模块、任务管理模块。身份验证模块用于用户登录、登出、访问 API 时的权限检测；算法模块负责实现系统的主要功能，提供内部调用的算法接口，当对 API 发起调用时首先进行用户权限检查，当满足调用条件后再进行调用，保证系统内部的访问安全性。算法模块包括人脸交换模块与攻击模块，分别实现人脸交换与人脸保护算法。任务管理模块负责合理规

划系统资源完成在一定规模下用户算法的正常与高效率执行，对用户提交的算法任务进行调度、管理、分配计算资源，并记录返回结果，最终返回结果同步到数据库中，并提供任务记录查询接口。

若将 API 接口与网页静态文件分别使用两个框架部署，会存在由于浏览器同源策略导致的无法跨域访问资源的问题，即用户首先连接 Nodejs 服务器，通过所返回的 Javascript 脚本访问 Flask 服务器中的 API，由于两者不处于同一域，出于安全性考虑浏览器将拒绝访问 Flask 服务器。为解决跨域问题，系统将使用 NginX 对 Nodejs 服务器和 Flask 服务器进行反向代理，用户通过访问 NginX 端口，由 NginX 转发流量来达到对不同域的访问，既减少对外暴露的端口与地址数量，又解决同源策略引起的跨域无法访问问题，用户通过访问同一端口来分别完成页面文件访问和 API 调用。用户在前端提交请求时，将由 Nginx 转发至 Flask 后端对请求进行处理，而对数据库的数据交互也由后端完成。

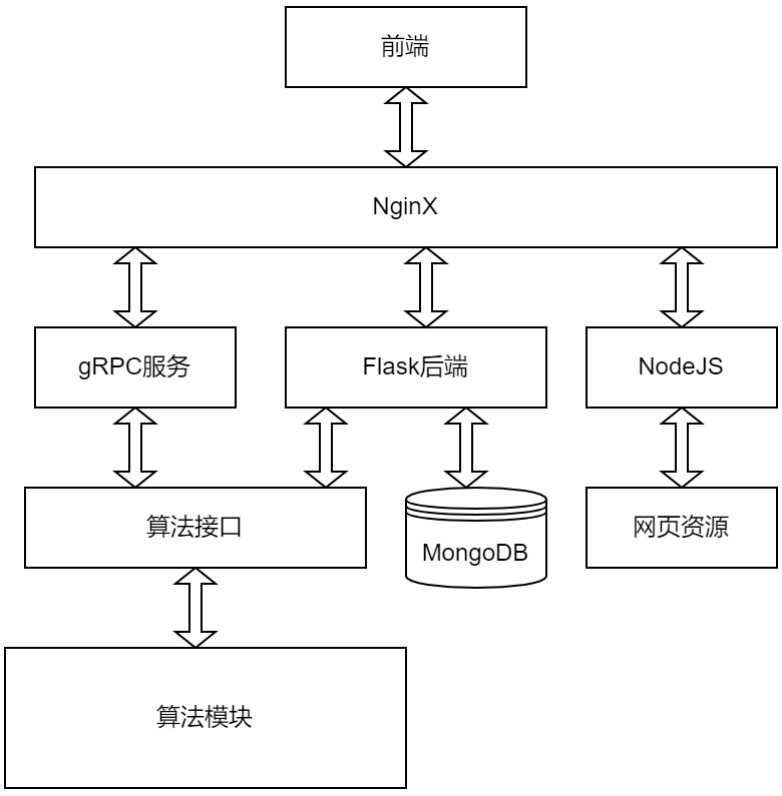



图 4-2: 系统架构设计

据此，系统整体架构设计如图4-2所示，前端在访问后端服务时经过 NginX 代理，根据不同的请求分别转发给相应的组件。请求页面资源文件转发

至 NodeJS，后端 API 转发至 Flask，基于 gRPC 的请求转发至提供 gRPC 服务的后端。Flask 后端与数据库进行交互，并适配换脸、攻击等算法接口，进行算法模块的调用。下面举一例说明用户使用算法时经历的过程：用户进入换脸或人脸保护页面，该页面经过 NginX 转发，由 NodeJS 返回。用户在页面中上传需要换脸或需要进行保护的图片，选择任务类型、算法参数并提交请求。该请求首先发送至 NginX 端口，并由 NginX 转发至 Flask 后端，调用相应的算法 API。调用时后端首先根据请求中携带的 token 验证其有效性，然后由任务管理模块创建相应任务，任务数据写入数据库中。当任务执行完成后调用回调函数，将任务运行结果同步到数据库中，图片数据暂存在服务端。用户通过查询任务列表得到该任务的状态，访问任务所提供的链接获取算法运行结果。

用户在使用 Web 应用时涉及到页面切换、页面跳转的功能，需要考虑路由方式的选择。常用的路由方式包括页内路由和后端路由，页内路由即不同的页面内容包含在同一个页面文件中，由页内编写的代码逻辑决定渲染哪个页面，用户只需发送一次页面请求。后端路由即不同页面分别使用多个页面文件，用户访问其他页面时需向后端发送请求，由后端决定返回哪个页面内容。在本系统中，API 请求使用后端路由并由 Flask 框架完成，在/api 路径下的所有请求均通过 Flask 框架令牌验证后返回，使系统核心内容拥有较高的安全性。Web 应用包含多个页面，虽然后端路由的安全性高，但由于使用后端路由将不能使用 React 热更新机制，即开发过程中不必重新启动 Nodejs 框架即可将代码变动呈现给浏览器的机制，因此基于页内路由设计前端页面框架，其优点为开发灵活，模块耦合度低，在 React 框架中对页内路由的封装由 react-router-dom 实现。

## 4.3 基于 React 的前端页面设计

网页前端为用户提供和系统功能交互的界面，通过设计 web 页面，将应用界面更好地呈现给用户，并为用户提供与后端服务的交互接口。通过前端可进行用户身份验证、系统功能调用、系统结果查询、任务提交历史查询等操作。

由于 React 框架的简洁、易用性，以及其特性使得开发者易于对复杂组件进行分解，并且其虚拟 DOM 使其具有快速的渲染性能，因此在系统前端页面基于 React 框架开发。系统前端页面总体上分为两部分：用户身份验证页面和系统功能页面。用户身份验证需要实现用户的注册与登录机制，其中包含 token

的储存和使用；系统功能页面通过导航栏的形式列举功能版块，并分别对不同版块展示所对应的页面，这种形式的访问可基于 react-router-dom 的页内路由机制实现，将不同版块分别获取后台资源的形式变为单次访问获取所有资源，其实质是切换页面由服务端完成路由变为由用户端完成路由，可节省大量网络开销，使得切换不同模块效率更高。

在用户身份验证部分，页面分为注册与登录两个版块，注册与登录功能合并在同一页面的不同标签栏内，通过点击标签进行注册与登录的切换，简化操作流程。用户注册后若注册请求成功，则自动跳转到登录标签页。用户注册时前端将自动检查表单各项的合法性，并显示不合法提示信息，当表单内容不符合要求时将不予提交。用户登录后跳转至系统主界面，系统主界面使用侧边栏菜单列表形式列举系统各项功能标签，包括首页、人脸交换功能页面、人脸保护功能页面、任务管理界面，在主界面中用户可选择登出账户，从而退出会话。

前端所实现的远程服务调用基于 token 验证机制，前端通过调用服务端后端接口以完成人脸交换、人脸保护等算法的执行和结果的返回，同时在请求中包含了用户登录时所获取的 token，使得后端通过 token 检查用户的合法性和访问权限。在人脸交换版块和人脸保护版块中实现用户图像上传功能，用户可以浏览本地文件的形式或拖拽上传两种形式进行。在任务管理界面实现任务列表展示的功能，通过调用后端接口得到该用户所创建的所有任务列表，将任务列表分页呈现给用户，实现翻页、查询等基本浏览操作，以及任务结果查看、任务中断等交互操作。

除此之外，前端页面还需管理用户登录状态、从服务端获取的访问令牌与刷新令牌，令牌可使用内存储存和 cookie 携带储存两种形式，其中 cookie 将储存在硬盘上，当用户登录时需将访问令牌在内存中，以及一个用于刷新访问令牌的刷新令牌，该令牌需储存在仅限于 HTTP 的 cookie 中，以使其防范攻击。前端页面需使用访问令牌对服务端访问受限的资源进行访问，以及任务的创建与查询。

在人脸交换、人脸保护算法部分中，需由用户上传图片执行服务器操作，前端实现图片上传组件来完成，该组件包括本地文件选择、图片拖拽上传、图像数据传输功能，用户点击组件或拖拽文件以进行本地文件选择，由组件记录本地文件路径，并在提交任务操作时将图像数据流附在请求内容中。

## 4.4 基于 Flask 的后端设计

根据系统需求，后端主要提供四方面功能：身份验证、算法服务、算法实现模块，以及与数据库进行交互。身份验证功能采用 flask_jwt_extended 库进行实现，该库提供用户访问令牌的封装，包括令牌的生成和验证工作及根据刷新令牌生成一组新的令牌的功能。对系统中由算法模块提供的服务，设计成 RESTful 形式的 API 接口，对该类型的 API 需要保证每一个 URI 代表一种资源，客户端和服务器之间，传递这种资源的某种表现层，以及客户端通过 HTTP 的常见操作方式，即 GET、POST、PUT、PATCH、DELETE 等来完成对服务器端资源的操作，从而实现 RESTful 架构。在该架构下基于 HTTPS 通信协议，在 URI 路径中包含 API 版本号，对于查询任务历史这类返回记录条目数量较多的请求，需由 API 提供筛选参数，例如页数或条目数，限制请求返回的大小，保证前后端通信的效率。服务端所返回的结果使用 JSON 格式。服务端对不同操作应遵循下列规范：对于 GET 请求，服务端返回资源对象的列表或单个对象；对 POST 请求，服务端返回新创建的资源对象；对 PUT 请求，服务端返回完整的资源对象；对 DELETE 请求，返回空文档。

表 4-1定义了后端的 API 接口：

表 4-1: 后端 API 接口

| api 名 | 请求类型 | 功能 | 参数 |
|---|---|---|---|
| login | POST | 提交表单进行用户登录 | 用户名、密码 |
| logout | POST | 用户登出 | |
| register | POST | 提交表单进行用户注册 | 用户名、密码及其他注册信息 |
| fs-custom-target | POST | 人脸交换 | 文件数据流 |
| fs-protect | POST | 对图像进行保护 | 文件数据流、算法参数 |
| get-img | GET | 获取图像 | 图像 ID |
| get-tasks | GET | 获取任务列表 | |

在身份验证模块中，用户身份验证机制基于访问令牌与刷新令牌实现，使用令牌可防止 CSRF 攻击及同源策略的限制，并可在多个服务之间共享。用户

登录时由前端发送用户填写的表单，后端验证请求有效性后创建一组访问与刷新令牌并包含在 cookie 中响应用户，用于后续的授权操作。后端通过查询数据库来验证用户信息的有效性，当用户认证成功时后端根据用户名创建令牌并返回给前端，每个令牌分别拥有一定的有效期。系统在令牌中包含用户名信息的目的是在用户请求资源过程中用于校验用户身份，确认用户有权限访问资源。令牌可携带于请求头中或 cookie 中，若令牌包含于请求头中则攻击者可使用 Javascript 获得令牌明文，给系统带来风险，因此系统中令牌携带于 cookie 中，并设定 cookie 仅限于 HTTP 请求，这种类型的 cookie 将无法由 Javascript 读取。当用户登出时浏览器丢弃该令牌。用户注册时后端查询数据库以检查用户所提交表单信息的合法性、有效性，当检验通过时将用户数据写入数据库，并提示前端用户注册成功。除此之外，后端还提供了在访问令牌失效后的刷新机制，当用户访问令牌过期后调用后端刷新接口，后端获取请求中储存在 cookies 中的刷新令牌，在检查刷新令牌的有效性后重新生成一对新的访问与刷新令牌，并通过 cookies 返回给用户；若用户刷新令牌也过期时则请求失败，用户将重新登录，后端生成新的令牌为用户提供访问权限。在系统其他功能的 API 中均需要检查访问令牌，根据令牌获得用户身份并检查该身份的访问权限来得知用户是否有权限访问资源，从而对系统资源的访问进行控制。

后端与数据库的交互由 Flask-PyMongo 库完成，它对 PyMongo 所包含的客户端、数据库、集合类进行封装，起到后端与 MongoDB 数据库之间的桥梁作用。系统在 Flask 启动时创建一个 PyMongo 实例，并绑定该网页应用进行初始化，实现后端对数据库的各项操作。根据系统需求，与数据库的交互主要围绕用户注册信息的增删改查操作、创建任务时对任务信息的记录、储存用户所发送的图像数据，以及对用户图像的修改操作。

由于后端各类算法耗时较长，且对于预估耗时较长的请求在高并发下即时等待会消耗大量用户时间，因此后端将支持任务管理与查询机制。基于此后端将包含以下内容：

- 换脸算法，将一张人脸照中的人脸替换为另一张人脸。
- 人脸图像保护，为用户上传的图像提供保护功能，用户选择保护的算法类型、算法参数，由后端调用算法模块执行人脸保护算法。
- 任务管理机制，对于用户提交的不论是换脸算法或图像保护算法，均为此新建一个任务，由任务管理模块执行任务，并收集任务结果。
- 任务查询机制，用户通过访问后端查询 API 对用户名下创建的任务进行查

询，并能获取任务执行结果。

任务管理机制对多用户并发请求采用任务队列形式处理，为每个用户提交的算法执行请求创建新任务，任务管理模块通过维护一个任务队列，依据任务优先级使用空闲的系统资源执行任务。在任务调度中多个需要调用相同模型的任务可以合并成同一批次输入模型，以减少多次新建进程、申请资源所带来的开销。

算法的接口适配与资源分配分别由 ServiceAdapter 和 ResourceManager 两个类完成，其中 ServiceAdapter 负责算法与后端 API 的对接，所有 Flask 后端对算法的调用首先调用 ServiceAdapter 相应接口，由 ServiceAdapter 调用任务管理器创建相应的算法任务，并指定 ResourceManager 中相应算法函数作为任务执行入口函数。当任务执行时会运行 ResourceManager 中指定的函数，对系统资源占用进行统计并分配计算资源执行任务。

## 4.5 人脸交换模块设计

人脸交换模块的主要任务是完成对用户上传图像和指定图像中人脸部分进行替换，其具体行为如下：定义输入图像为源图像 $X_s$ 与目标图像 $X_t$，从源图像 $X_s$ 中提取身份特征，从目标图像 $X_t$ 中提取外观属性特征，所生成的换脸图像 $Y_{s,t}$ 中的人物既包含了源图像的身份信息，又包含目标图像的外观信息。从图像结构上描述，其整体上与目标图像一致，只有面部身份信息被替换为源图像的人脸身份信息，从而实现换脸的效果。该模块最后将模型输出的换脸后图片返回给用户，结合任务管理模块后其返回结果将附加在用户所创建任务的结果中，用户通过查询任务结果获取交换后图像。

人脸交换模块由图像预处理、换脸算法和图像后处理构成，其中预处理部分的目标是根据用户输入图像裁剪、调整以得到利于换脸模型的人脸图像，预处理过程首先通过人脸检测模型 MTCNN 框取图像中人脸，将其中拥有权重最高的人脸认定为需进行交换的人脸，裁剪并重新调整大小后通过检测的面部特征点修正其人脸朝向，得到预处理后的人脸图像。换脸算法将两张经裁剪并重新调整大小后的人脸输入换脸模型进行交换，最后经过后处理过程，根据先前修正人脸朝向所使用的变换矩阵，将交换后的人脸逆变换后嵌回原图中，得到最终换脸图像。

为减小模块耦合性，该模块图像接口不直接使用图像数据作为输入和输

出，而采用图像地址的形式，模块输入为用户上传的图像在服务器中存储的地址，返回生成的图片对应的存储地址，模块对图像的输入输出均由资源管理模块接口完成，输出图像储存到服务器中，并通过任务管理模块的接口将输出图像所对应的结果信息传至任务管理模块，并由任务管理模块持久化至数据库中。

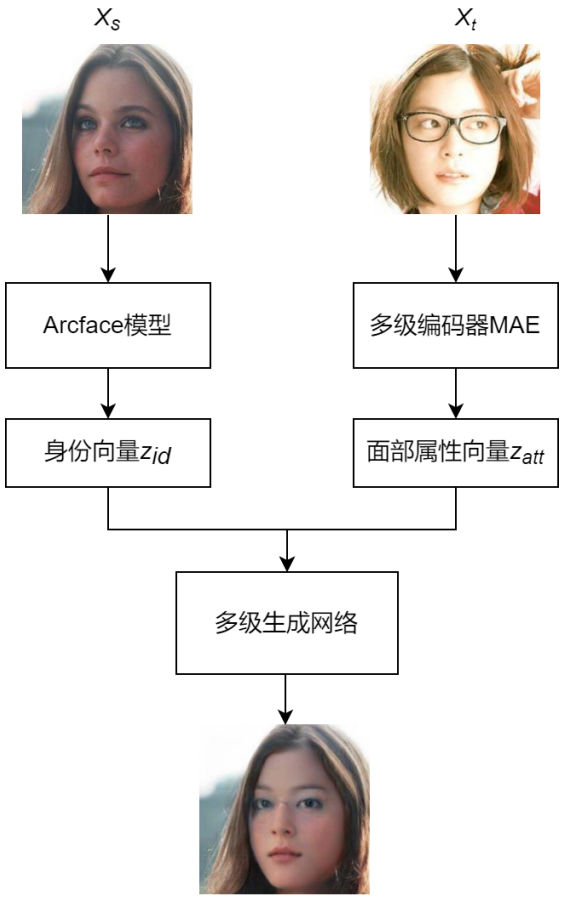



图 4-3: FaceShifter 人脸交换算法

在图像预处理过程中，首先用户输入的图像经由资源管理模块存放至图像暂存区，由预处理模块读入图像，使用 MTCNN 模型标注出人脸图像中的所有人脸位置，对图像中人脸部分进行框取，模型将返回所有人脸对应的边界框、人脸面部特征点以及每个框取的人脸对应的权重，选取权重最大的人脸，表示图像中最显著的人脸作为交换目标，基于人脸边界框裁剪并拉伸至模型输入所需大小。为适应多姿态场景下换脸的有效性，希望使得倾斜的人脸经过换脸模型也能得到良好的效果，因此将人脸所在区域作变换处理，使其脸部不倾斜。计算使得面部特征点转换为正脸面部特征点的仿射变换，得到变换矩阵与所

对应的逆矩阵。将框取的人脸区域经过仿射变换后即得到预处理完成的面部图像。

人脸交换算法使用目前已公开且效果较好的 FaceShifter 换脸模型。在换脸过程中首先将预处理变换后的图像作为源图像 $X_s$，将源图像经过预训练的 Arcface 模型提取身份特征向量，目标图像经过多层编码器模型提取外观特征，多层编码器的每一层输出一个外观属性特征向量，并将上述所得身份属性与每一层的外观属性特征作为 AAD 生成器的输入，最后经过生成器多级放大后得到换脸后的图像。FaceShifter 算法过程如图 4-3所示。

在图像后处理过程中，将预处理过程所得逆变换作用于换脸模型所生成的图像上，由于图像经 OpenCV 库变换所得图像边界倾斜，并且只计算包含在图像内部的像素对应的色彩值，图像外部的像素为背景色，经过变换后的图像边界未经插值计算而呈现锯齿状边界。后处理将变换后的图像通过添加透明度分量对图像边缘作平滑处理，其中完全在图像内部的像素赋予透明度 0，完全在外部的像素赋予透明度 1，边界处的像素根据其在图像内部的比例确定其透明度，将边缘透明度平滑的图像叠加到原图上，从而消除边界锯齿效应，得到最终换脸图像。

## 4.6 攻击模块设计

攻击模块主要完成人脸保护算法的实现，为防止被换脸人的图像作为源图像进行换脸，攻击的对象为换脸过程中提取身份特征向量的源图像，主要使用上文中基于特征的攻击和基于结果的攻击算法，对用户上传的源图片添加扰动以进行保护。记源图像为 $X_s$，目标图像为 $X_t$，攻击前源图像与目标图像换脸所得图像为 $Y_{s,t}$，攻击后的源图像为 $X_{adv}$，攻击后源图像与任意目标图像经过换脸所得图像为 $Y_{adv,t}$。四种攻击均基于 PGD 方法攻击 FaceShifter 换脸模型，通过在待保护的源图像 $X_s$ 上添加扰动，使得攻击后换脸图像 $Y_{adv,t}$ 出现不同形式的明显改变痕迹，以破坏换脸结果。根据攻击方式的不同，攻击算法分为对身份特征的攻击、使换脸图像真实性降低、出现黑色区域、人脸语义特征发生改变。其中，对身份特征的攻击使得源图像 $X_s$ 经过 arcface 所生成的身份编码发生改变，从而使得换脸图像 $Y_{adv,t}$ 丢失源图像身份；针对语义特征攻击的算法将使得换脸图像 $Y_{adv,t}$ 在语义特征层面上发生改变，例如发色、年龄、性别等，且可由用户选择语义改变的类型。由于针对语义特征的攻击方案最为复杂，因此

下面以该算法为例说明其攻击过程。

针对语义进行攻击的实现过程分为三个部分：换脸模型训练、语义判别器训练、模型攻击。换脸模型以两张人脸图作为输入，分别称为源图像 $X_s$ 与目标图像 $X_t$，最终输出的换脸结果中同时包含源图像 $X_s$ 的身份特征和目标图像 $X_t$ 的面部属性特征，实现将源图像身份嵌入到目标图像中，并保持面部身份特征以外区域基本不变的效果。换脸模型的训练数据集需要经过人脸中心化、大小归一化，构建数据集需使用人脸检测模型对原始数据集进行人脸标注并调整为同一大小，其中人脸检测使用现有的多任务级联卷积网络 (MTCNN)，该模型能在一张包含人脸的图片中提取人脸位置与面部特征点，并返回所有检出的人脸外框及 5 个特征点位置。每个检出人脸另外附加一个权重，表示该人脸的显著程度，所有权重中最大的是图片中最突出的人脸。

为实现对换脸图像语义特征进行攻击的目的，需训练一个识别人物语义特征的模型，在本系统中使用 StarGAN 模型判别器作为语义识别。StarGAN 生成器是一个多领域的图像到图像转换模型，而判别器用于识别人物外观特征及图像真实度，以人脸图像作为输入，输出训练模型时所用图像域分布，例如，可基于指定的几项人脸面部特征对模型进行训练，包括发色、性别、年龄、戴眼镜等特征，输出特征为一向量，其中每一维表示一种特征，多种特征可以同时存在。StarGAN 判别器被用作 PGD 攻击的一部分，在攻击阶段用于提取换脸后的人物外观属性。为使得攻击之后换脸图像中出现某些特定特征，需要在训练 StarGAN 模型时使用包含这些特征的图像进行训练，以所指定特征作为类别判定的领域。

攻击流程与其余三种攻击方法类似，首先设定目标领域标签 $c$，在 PGD 攻击的每一步中从数据集中随机选取一批次图像最为目标图像 $X_t$，并与用户上传的源图像 $X_s$ 进行换脸，得到换脸后图像 $Y_{s,t}$；将 $Y_{s,t}$ 输入 StarGAN 判别器，计算 $Y_{s,t}$ 所对应的图像域与目标域标签计算损失，并基于 PGD 攻击算法计算扰动值，将扰动值叠加到源图像中。经过多步迭代攻击得到最终语义攻击的结果。分别对不同目标领域标签 $c$ 进行攻击，并将攻击结果与数据集中的随机人脸图进行换脸，返回添加扰动的图像及换脸测试结果。

## 4.7 任务管理模块设计

在 python 环境下，多算法任务在后台的异步执行机制通常使用多进程来实现。虽然现有的 Celery 并行分布式框架可以实现多任务的并行化，但由于其采用 fork 而非 spawn 的形式创建子进程，不能兼容 CUDA 在多进程下的运行，因此不作为本系统任务管理模块中多进程的实现，而基于 python 多进程框架 multiprocessing 设计。该框架支持手动设置子进程的创建模式为 spawn，能很好地兼容多任务在 GPU 设备中的计算。通过将用户对算法的调用封装成任务形式，在不同进程下异步执行每个任务，并在任务执行完成后调用回调函数，返回任务结果。任务的执行成功与执行失败分别对应两个回调函数。任务创建与完成时将把任务数据同步到数据库中。用户可在前端调用后端所提供的 api 接口查询任务，后端返回该用户所创建的任务列表，每个任务展示其运行状态、创建时间、完成时间、任务结果等信息。

对创建的任务使用状态机制进行管理，每个任务定义以下状态：等待、正在运行、运行成功、运行失败、已撤回。新创建的任务为等待状态，当系统拥有空闲资源时将分配新进程执行任务，并使其进入运行状态；任务执行成功后调用相应的成功回调函数，将任务状态更新为运行成功状态，同时更新任务结果；任务执行失败后调用相应的失败回调函数，并更新状态为运行失败状态。如在任务执行前由用户或管理员手动撤回则取消任务执行，任务进入撤回状态。任务状态或结果更新时会将任务信息同步写入数据库中，以便用户查询与系统管理。

多进程框架所包含的进程池类 Pool 能提供一种便捷的并行化多个函数执行的方式，并通过在模块中定义任务状态来跟踪记录每个任务的执行情况。所有任务储存在字典型数据结构中，每个任务在创建时会产生一个唯一识别码作为字典的键，用于任务的查询与更新。任务管理模块的上游模块为所对应的各算法模块，负责接收算法模块分配的各算法任务，下游为 MongoDB 数据库，需要将所创建和更新的任务信息写入数据库，并在任务执行完毕后对任务状态及执行结果进行更新。模块还提供任务查询接口，根据任务标识码查询任务执行状态以及执行结果。查询结果使用超时机制，对正在执行的任务反复进行查询并等待一定时长，时长随查询次数增加而延长，等待任务执行完毕后返回执行结果，当循环查询超过一定次数或超时则查询失败。

## 4.8 本章小结

本章分别从系统整体架构设计、前后端设计、各模块设计等方面对系统整体架构与组成部分进行规划、方案设计，为后续系统实现打下基础。在系统总体设计中讨论了系统前后端架构、各框架的选择、路由方案选择等；在前端设计中讨论了如何将所需功能组装呈现给用户；在后端设计中讨论了身份验证机制实现方案、API 设计、算法执行机制及其余后端模块组成；各模块设计包括人脸交换模块、攻击模块与任务管理模块等主要模块的设计方案。

# 第五章　模块实现

本章首先介绍系统整体框架的实现方案，再依次介绍系统各模块的实现方案与细节，包括身份验证模块、人脸交换模块、攻击模块及任务管理模块。

系统部署在一台或多台服务器中，所需要的技术框架包括 Flask、NodeJS、MongoDB 及 NginX，基于 NodeJS 和 React 框架及其他所需组件开发实现前端页面，包括用户注册、登录、系统功能等，其中基于 react-router-dom 和页内路由机制实现导航栏菜单，使得在多页面间切换时不需要向前端发送请求。基于 Flask 框架实现后端身份验证模块、人脸交换模块、攻击模块及任务管理模块，其中基于 flask_jwt_extended 库实现后端的双 token 验证机制，并实现对算法 API 服务的响应以及与 MongoDB 数据库的交互。最后，由 NginX 对前后端进行反向代理，将请求前端页面与请求后端 API 的流量分别转发给相应服务器端口，设置防火墙只开放 NginX 端口，完成系统的总体框架实现。

下面介绍各模块的实现细节。

## 5.1　身份验证模块实现

身份验证模块基于双令牌机制实现用户身份验证及访问控制，包括用户注册、登录、登出、权限管理以及访问令牌与刷新令牌的管理。后端基于 Flask 框架中的 flask_jwt_extended 库实现令牌的创建与验证，在后端应用初始化时设置访问令牌的有效期为一小时。在用户登录时创建发送给前端，由前端管理令牌，并在需要调用访问受限接口时通过令牌验证用户身份。身份验证 API 包括四个，其路径与功能分别为：/api/v1/register/用于用户注册，/api/v1/login/用于用户登录，api/v1/logout/用于用户登出，api/v1/refresh/用于令牌失效后刷新。

后端于路径/api/v1/register/下实现的用户注册数据流中，由前端创建表单并调用后端接口发送表单信息，后端验证信息有效性并查询数据库。如果出现用户已存在、邮箱已被使用或表单内容不符合要求，则返回相应的创建失败信息，否则创建成功，将用户信息写入数据库，返回创建成功信息。在路径/api/v1/login/下实现的用户登录数据流中，由前端将表单发送至后端接口，后

端查询数据库并验证密码正确性，如果用户名和密码可在数据库中未找到匹配项，则登录失败，返回失败相关信息，否则登录成功，后端会根据用户身份创建访问令牌和刷新令牌，分别设定其有效期，其中访问令牌用于用于对其他算法接口的调用权限验证，刷新令牌用于在访问令牌过期后重新获取新的令牌，并以包含在 cookie 中的形式发送给前端。在路径/api/v1/logout/下实现的用户登出数据流中，由前端将用户登录状态置为未登录，跳转页面至登录页，并调用后端注销登录 API。后端注销用户的登录信息，修改响应以删除包含访问或刷新 JWT 的 cookie，同时删除相应的 CSRF cookie。在路径/api/v1/refresh/下实现的令牌刷新，其刷新机制基于 Flask 的 after_request 装饰器实现，其实现机制为前端对每一次其他 api 调用之后都会自动调用一次该 api，当用户持续性访问页面时会话不会中断，而用户停止活动一段时间后会话中断，需重新登录。该接口以前一个请求的响应类作为输入，获取响应中的访问令牌，对比该令牌与当前时间的时间戳，若两者相差 30 分钟以内则更新令牌，根据用户身份重新创建令牌，并写入响应类中，最后返回该响应类以完成令牌的刷新。

在前端令牌以两种形式储存：访问令牌储存在内存中，刷新令牌储存在 HTTP-only 的 cookie 中，并记录用户的登录状态。前端基于身份验证机制实现 authProvier()，其中储存了访问令牌与刷新令牌、登录状态，并提供以下接口：

- login()：传入访问令牌，储存令牌并更新登录状态；
- logout()：丢弃令牌并更新登录状态；
- isLoggedIn()：返回登录状态；
- updateToken()：使用内存中刷新令牌调用后端接口获取一组新的令；
- authFetch()：使用明文令牌访问后端需要身份验证的 API 接口，使用 Bearer 验证方式；
- authFetchByCookie()：基于 Cookie 携带令牌的方式访问后端需要身份验证的 API 接口，当使用 POST 方法时将令牌包含在请求头中，否则使用 JWT 验证方式。

登录时后端首先根据用户身份生成访问令牌与刷新令牌，并将其作为 cookie 字段附加在返回内容中。前端从收到的后端响应中解析出访问令牌，储存在内存中，并更新用户登录状态。当前端需要调用后端需要身份验证的 api 时首先检查登录状态是否有效，若登录状态无效，则跳转至登录页面；若登录状态有效，则将内存中储存的访问令牌作为请求头部的 X-CSRF-TOKEN 字段并发送 HTTP 请求。每次调用请求后自动调用刷新 api 接口，后端根据附带的

刷新令牌以及有效时间决定是否重新生成一组新的访问令牌和刷新令牌。当失效时间在 30 分钟以内时重新根据用户身份创建一组新的令牌，以延长令牌有效期，并将新生成的令牌写入响应中的 cookie，同时需保证该 cookie 仅限于 HTTP。当启用了跨站请求伪造攻击保护时，根据相应的经过编码的令牌解析出 CSRF 令牌并返回。

## 5.2 人脸交换模块实现

人脸交换模块使用目前公开的 FaceShifter 换脸模型，该模型中换脸部分由三部分组成：身份编码器、多级属性编码器、多层生成器，其中身份编码器从输入源图像 $X_s$ 中提取身份特征 $z_{id}$，多级属性编码器从输入目标图像 $X_t$ 中提取多层的外观属性特征 $z_{att}$，多层生成器使用身份特征 $z_{id}$ 和外观属性特征 $z_{att}$ 经多级放大形式逐层增加输出图片的分辨率以得到结合两者特征的人脸图片。身份编码器使用预训练 arcface 模型，多级属性编码器与多层生成器需结合判别器进行生成对抗式训练得到。

在人脸交换模型的训练过程中，首先对数据集预处理。由于人脸数据集图像大小不统一，因此使用预训练 MTCNN 模型获得每张图中所包含的人脸位置，若无人脸则跳过该图片，否则在所有框取的人脸中选取权重最大的人脸，截取人脸部分并缩放为长、宽均为 256 像素的图像作为数据集。训练过程每一轮迭代中首先从数据集中选取源图像和目标图像，裁剪源图像中心区域并使用双线性插值法将图像进行降采样，使用 arcface 提取源图像身份特征。将目标图像输入多级属性编码器中提取目标图像的外观属性特征，并与源图像身份特征输入多层生成器中得到换脸生成图像。将生成图像输入判别器得到判别结果，并计算身份损失、外观属性损失、生成器与判别器分别的对抗损失。当源图像与目标图像为同一张图像时，则计算重构损失，否则重构损失为 0。将计算所得损失分别训练生成网络与判别网络，迭代直到模型收敛即得到预训练的换脸模型。

人脸交换算法主要包含以下流程：计算资源分配、模型加载、图像数据预处理、换脸算法执行。计算资源分配由资源管理类 ResourceManager 实现，在任务管理模块执行任务入口函数时调用 ResourceManager，通过查看计算资源使用情况，分配合适的计算资源执行算法。模型加载涉及任务管理模块的多进程实现机制，由于多进程共享内存有限，因此不同进程需动态加载各自的模型

资源。在图像数据预处理中需要提取图片中的人脸部分并调整至标准大小，首先使用预训练 MTCNN 模型提取图像中的人脸的外框位置、面部特征点坐标，选取最显著的人脸，即模型返回所有人脸位置中权重最大的一个，将该人脸外框裁剪并缩放至长、宽均为 256 像素的图像，将所得人脸的所有面部特征点中选取前三个面部特征点，并根据预设正脸所对应的面部特征点坐标，计算得到框取人脸特征点到预设特征点的仿射变换矩阵。将仿射变换作用于源图像人脸区域得到朝向矫正后的标准化大小的人脸。将标准化大小的源图像输入预训练 arcface 模型、目标图像输入 FaceShifter 中的多级属性编码器并将两者结果输入多层生成器，输出换脸后的图像。在后处理部分中根据之前用于矫正人脸朝向所使用的仿射变换对该人脸部分做仿射变换的逆变换得到还原姿态的人脸，同时对图像边缘作抗锯齿处理，即对图像增加透明度一项。将含透明度的图像嵌入原图中得到边界处无明显痕迹的最终换脸结果。

换脸模型使用预训练的 arcface 模型作为编码源图像身份的编码器模型，并依照 FaceShifter 构造用于提取编码目标图像面部属性特征的多级编码器模型，和多级放大图像的生成器模型。在训练前首先将数据集预处理，对所有人脸数据集图像使用 MTCNN 模型标注人脸位置，将每幅图像中权重最大的人脸裁剪并调整为模型输入大小保存，作为新数据集。训练过程中，对每张输入源图像 $X_s$ 截取中心区域并进行降采样，输入 arcface 模型得到身份编码向量 $z_{id}$。在系统实现中选取数据集预处理后的图像边长为 256 像素，裁剪的中心区域大小为 216 像素，将中心区域降采样至边长为 112 像素的图像。对每张输入目标图像 $X_t$ 输入多级属性编码器，其中多级属性编码器由 8 层全连接卷积层构成，每层卷积层使用 4x4 的卷积核，每层均输出一个属性向量，共得到 8 个属性向量 $z_{att}$ 将得到的身份编码向量与属性向量输入至多层生成器中，得到生成图像 $Y_{s,t}$。将生成图像 $Y_{s,t}$ 输入判别器 $D$，得到判别结果 $D_{src}$。由下式分别计算对抗损失、身份损失、属性损失、重构损失，并加权累加得到生成器训练损失，反向传播优化生成器网络参数。最后计算判别器损失，反向传播优化判别器网络参数，并训练直到模型收敛。

$$\mathcal{L}_{id} = 1 - \cos\left(z_{id}\left(Y_{s,t}\right), z_{id}\left(X_s\right)\right) \tag{5-1}$$

$$\mathcal{L}_{att} = \frac{1}{2}\sum_{k=1}^{n}\left\|z_{att}^{k}\left(Y_{s,t}\right) - z_{att}^{k}\left(X_t\right)\right\|_2^2 \tag{5-2}$$

$$\mathcal{L}_{rec} = \begin{cases} \frac{1}{2}\left\|Y_{s,t} - X_t\right\|_2^2 & \text{if } X_t = X_s \\ 0 & \text{otherwise} \end{cases} \tag{5-3}$$

$$\mathcal{L}_{\text{AEI-Net}} = \mathcal{L}_{adv} + \lambda_{att}\mathcal{L}_{att} + \lambda_{id}\mathcal{L}_{id} + \lambda_{rec}\mathcal{L}_{rec} \tag{5-4}$$

人脸交换模块在后端对外提供一个 api 接口 fs-custom-target，需验证身份进行访问。用户在浏览器中上传两张图片并使用本地储存的访问令牌调用接口以完成操作。后端在请求处理中首先根据令牌获取用户身份，同时获得用户上传的两张图片，将图片储存到本地后使用两张图片的本地地址调用人脸交换过程，进行人脸交换，算法向任务管理模块创建一个新任务用于运行该算法，并分配一个回调函数用于保存算法结果。

## 5.3 攻击模块实现

攻击模块主要完成对用户上传图像的保护，通过使用 PGD 攻击算法对换脸模型进行攻击，计算一个扰动叠加到源图像上，使得攻击后的源图像与其他目标图像换脸所得图片被破坏。由于攻击模块处理单张图片的时间可能较长，因此需以异步任务的形式对任务进行管理、分配计算资源。攻击模块初始化时将初始化用于加载数据集的 Dataloader 并加载 arcface 模型、FaceShifter 生成器模型 $G$、StarGAN 生成器与判别器模型 $D$，以便后续算法调用时直接使用所需对象。攻击模块接收以下参数：源图像 $X_s$、算法类型、算法参数，其中算法类型为所设计的四种攻击类型，算法参数主要包括 PGD 攻击参数中的图像扰动上界 $\epsilon$、步长、迭代次数。攻击时根据算法类型分别调用不同攻击算法，并传入预加载的 Dataloader、所需模型、算法参数等。

攻击模块分别实现了基于特征的攻击和基于结果的攻击，换脸模型使用目前公开且效果最好的 FaceShifter 模型，其中基于特征的攻击通过攻击 FaceShifter 中用于提取身份特征的 arcface 身份编码器，实现了对源图像身份特征的攻击，基于结果的攻击实现了使换脸图像真实性降低、图像出现黑色区域、语义特征发生改变。由于攻击算法需使用换脸模型以及语义判别器，因此在攻击之前首先进行换脸模型训练和语义判别器训练。换脸模型使用两张人脸图像作为输入，分别称为源图像与目标图像，最终输出的换脸结果中需包含源图像的身份特征和目标图像的面部属性特征，实现在目标图像脸部以外区域不变的情况下将源图像身份转移嵌入到目标图像的效果。在训练之前需要构建以人脸为中心、大小归一化的人脸图像数据集，构建该数据集需使用人脸检测模

型，对原始数据集进行人脸标注并调整为同一大小。人脸检测基于目前公开的多任务级联卷积网络 (MTCNN) 实现，MTCNN 模型用于在一张包含人脸的图片中检出人脸位置并标注面部特征点，通过级联 P-网络、R-网络、O-网络在包含人脸的图像中检测并框取一个或多个人脸，每个检测出的人脸均包含一个权重表示该人脸的显著程度，同时对每个人脸的面部特征点进行标注。

本系统中针对图像语义的攻击使用目前公开的 StarGAN 模型中的判别器，用于提取图像语义特征，训练之前需要选择训练用的语义标签，根据语义攻击所要达成的效果选择若干图像域标签，在人脸任务中为多种面部外观特征，例如年龄、性别、发色等显性语义特征，最后根据所选取的标签训练语义判别器。StarGAN 模型由生成器和判别器构成，生成器输入为单张人脸图像以及多个领域外观特征标签，输出具有标签所描述的特征的人脸图片；判别器分别判定该图片是否为真实图片，以及对图像所属领域进行判定，一张图像可使用多个领域的分布来代表。在语义判别器训练过程中，通过提供所选择的语义特征，判别器对每个领域分别判定是否为真。使用该判别器可对换脸模型生成器合成的图像在多个特征维度上提取面部属性特征等语义信息。

训练上述模型后，完成攻击的准备工作。根据攻击功能的技术与流程，可进一步分为模型攻击、攻击效果测试。攻击中使用的投影梯度下降法 (PGD) 是一种简单有效的白盒攻击方法，被认为是实现对抗鲁棒性最有效的一阶攻击方法之一，即在对抗攻击过程中只利用相对输出结果的损失函数的梯度，只包含模型局部的一阶信息，同时试图找到在特定输入上最大化模型损失的扰动，并保持扰动的大小小于指定阈值。算法从样本点附近的一个随机扰动点出发，通过多次迭代的方式，每次迭代沿梯度方向下降一步，直到算法收敛。在攻击阶段，使用训练完成的判别器实现对不同 GAN 模型的基于添加噪声的隐式编辑。每种攻击算法分别定义不同的用于攻击的模型类：在降低图像真实性的攻击中定义为串联换脸模型与 StarGAN 判别器模型得到对真实度的输出 $D_{src}$；在使图像出现黑色区域的攻击中为串联换脸模型与定义的掩膜得到期望篡改区域的权值；在基于身份特征的攻击中为源图像经过换脸模型中身份编码器所得身份向量 $z_{id}$，在本文系统中为 arcface 模型；在针对图像语义的攻击中为串联换脸模型与 StarGAN 判别器模型得到对图像域的类别判定 $D_{cls}$。分别对四种攻击算法基于攻击目标和模型类输出定义损失函数，并使用投影梯度下降法对所定义的模型类进行白盒攻击，计算扰动并叠加到原始图像中，经过多次迭代后得到最终结果。在人脸攻击流程中，首先进行图像数据预处理，将人脸数据归

一化并重采样为模型训练所用图片大小，在本文实现中使用长、宽均为 256 的图片，然后使用上述方法对待攻击的换脸模型进行迭代攻击，并得到最终攻击图像。

经过攻击的图片需要完成攻击效果测试，分别将攻击前图像 $X_s$ 与攻击后图像 $X_{att}$ 与数据加载器中随机图片进行换脸，使用多张图片进行测试，将攻击前后换脸效果图以及攻击后图像作为结果一并返回。所生成的图片储存在服务器中，并返回图片地址。任务管理模块中回调函数收到返回结果后更新任务信息，并同步到数据库中，完成攻击算法的执行。

下面进一步对 FaceShifter 模型、StarGAN 模型的训练，以及攻击换脸的过程作详细的说明。

### 5.3.1 换脸模型训练

作为待攻击的换脸模型，一般基于 GAN 实现，但其内部结构不唯一。在系统中使用 FaceShifter 作为待攻击的换脸模型。FaceShifter 换脸模型分为三个部分：对源图像提取身份特征的身份编码器、对目标图像提取外观属性特征的多级属性编码器和结合上述两个特征生成换脸图片的多层生成器。其中身份编码器用于提取代表源图像人脸身份的特征；多级属性编码器用于提取代表目标图像外观属性的特征，其中编码器每一层均输出一个属性特征向量；多层生成器采用逐层增加生成图像分辨率的形式，结合上述源图像的身份特征与目标图像外观属性特征生成同时具有两者特征的换脸图片。

FaceShifter 模型需输入中心化的人脸图像，为此需首先对数据集预处理，由于人脸数据集图像大小不统一，人脸未中心化，因此需要识别人脸位置后重新调整大小。使用预训练 MTCNN 模型标注出每张图中所包含的人脸位置，将所检测的权重最大的人脸标注框进行裁剪，并缩放为标准大小作为预处理后的数据集。使用处理后的数据集训练换脸模型，每一轮训练中，首先从数据集中随机选取源图像 $X_s$ 与目标图像 $X_t$，裁剪源图像 $X_s$ 中心区域 218×218 大小，使用双线性插值法将裁剪区域图像降采样至 112×112 大小，使用 arcface 预训练模型对降采样后的图提取源图像身份特征 $z_{id}(X_s)$，将目标图像 $X_t$ 输入多级属性编码器提取目标图像多级属性特征 $z_{att}$，并将所得源图像身份特征 $z_{id}(X_s)$ 与目标图像多级属性特征 $z_{att}(X_t)$ 输入多层生成器 $G$ 中，得到生成图像 $Y_{s,t}$。将生成图像 $Y_{s,t}$ 输入 arcface 模型提取生成图像身份特征 $z_{id}(Y_{s,t})$，计算源图像身份特征与生

成图像身份特征的余弦相似度作为身份损失 $\mathcal{L}_{id}$:

$$\mathcal{L}_{id} = 1 - \cos\left(z_{id}\left(Y_{s,t}\right), z_{id}\left(X_s\right)\right). \tag{5-5}$$

将所得生成图像 $Y_{s,t}$ 输入多级属性编码器得到生成图像的属性特征 $z_{att}^{k}(Y_{s,t})$，$k$ 表示编码器级数，计算属性损失 $\mathcal{L}_{att}$，属性损失为源图像属性特征与生成图像属性特征的均方差：

$$\mathcal{L}_{att} = \frac{1}{2}\sum_{k=1}^{n}\left\|z_{att}^{k}\left(Y_{s,t}\right) - z_{att}^{k}\left(X_t\right)\right\|_2^2. \tag{5-6}$$

若源图像与目标图像为同一人时，根据生成图像与源图像的差异计算重构损失 $\mathcal{L}_{rec}$，否则重构损失为 0：

$$\mathcal{L}_{rec} = \begin{cases} \frac{1}{2}\left\|Y_{s,t} - X_t\right\|_2^2 & \text{if } X_t = X_s \\ 0 & \text{otherwise} \end{cases} \tag{5-7}$$

根据判别器 D 的输出结果计算对抗损失，其中对抗损失使用铰链损失函数，尽可能使得判别器误判为真实图片：

$$\mathcal{L}_{adv} = f_{hinge}(D(Y), 1). \tag{5-8}$$

将上述计算所得损失加权求和得到生成器 $G$ 的总损失函数：

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{att}\mathcal{L}_{att} + \lambda_{id}\mathcal{L}_{id} + \lambda_{rec}\mathcal{L}_{rec}. \tag{5-9}$$

使用 Adam 优化器根据损失优化生成器 $G$，其中优化器超参数 $\beta_1, \beta_2$ 分别为 0、0.999，分别使用真实图像与生成器 G 的生成图像作为判别器 $D$ 的输入，其优化目标为使得判别器正确辨别图像真伪。使用 Adam 优化器根据 $D$ 的判别损失优化判别器，其中优化器超参数 $\beta_1, \beta_2$ 分别为 0、0.999。

### 5.3.2 语义判别器训练

语义判别器使用目前公开的 StarGAN 一到多领域图像转换模型，用于提取图像的语义类别特征。在使用上述方法训练生成模型后，继续训练 StarGAN 语义判别模型，该模型的主要作用为从图像中提取具有高区分性、可辨识、可描述的特征，作为后续针对图像生成特征进行可控制攻击的基础。本文所使用

的判别器以人脸图像作为输入，输出结果为对给定面部外观属性特征，例如年龄、性别、发色等属性的 0/1 编码，表示人脸是否存在该属性特征。语义判别器模型构造如图 5-1所示。

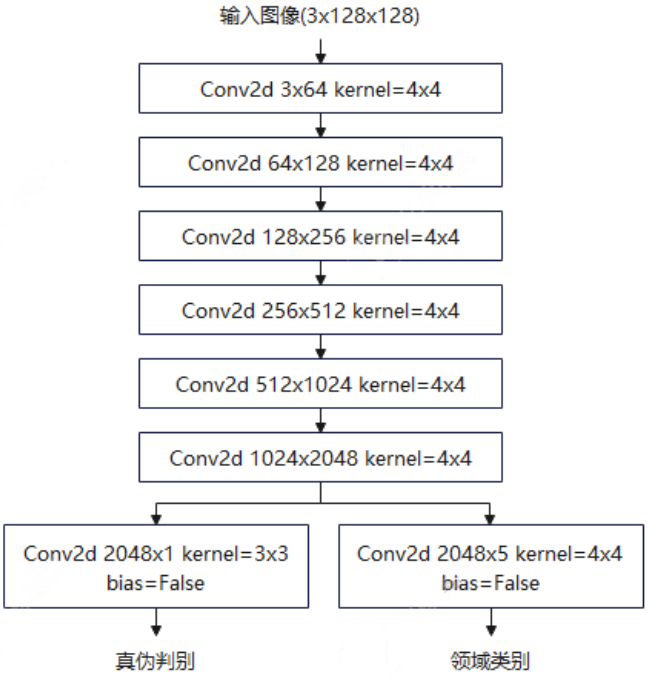



图 5-1: 语义判别器模型示意图

其中训练判别器时将根据真实图片判别结果 $D_{src}(x)$、伪造图片判别结果 $D_{src}(G(x,c)$、真实图片类别 $D_{cls}(x)$、梯度乘法系数 $\lambda_{gp}$ 和梯度惩罚 $GP$ 计算损失如下：

$$\mathcal{L}_{adv}^{D} = -\mathbb{E}_{x}\left[D_{src}(x)\right] + \mathbb{E}_{x,c}\left[D_{src}(G(x,c))\right] + \lambda_{gp}GP \tag{5-10}$$

$$\mathcal{L}_{cls}^{D} = H\left(D_{cls}(x), c'\right) \tag{5-11}$$

$$\mathcal{L}^{D} = \mathcal{L}_{adv}^{D} + \lambda_{cls}\mathcal{L}_{cls}^{D} \tag{5-12}$$

其中 $H$ 为交叉熵损失。对语义判别器的生成器优化中，重构损失 $\mathcal{L}_{\text{rec}}^{G}$、对抗损失 $\mathcal{L}_{\text{adv}}^{G}$、分类损失 $\mathcal{L}_{\text{cls}}^{G}$ 与总损失 $\mathcal{L}^{G}$ 计算如下：

$$\mathcal{L}_{\text{rec}}^{G} = \left\|G\left(G(x,c), c'\right), x\right\|_{1} \tag{5-13}$$

$$\mathcal{L}_{\text{adv}}^{G} = -\mathbb{E}_{x,c}\left[D_{\text{src}}\left(G(x,c)\right)\right] \tag{5-14}$$

$$\mathcal{L}_{\text{cls}}^{G} = H\left(D_{cls}(x), c\right) \tag{5-15}$$

$$\mathcal{L}^G = \mathcal{L}^G_{\text{adv}} + \lambda_{\text{cls}} \mathcal{L}^G_{\text{cls}} + \lambda_{\text{rec}} \mathcal{L}^G_{\text{rec}} \tag{5-16}$$

训练语义判别器使用 CelebA 人脸数据集，其中每张图片对人脸外观标签进行标注，例如年龄、性别、发色等外观属性。训练需指定若干领域标签作为 StarGAN 一对多领域的转换目标。训练过程的每一步，从数据集中获取一批次图像与图像所对应的类别标签，同时在所设定的图像分类领域空间中随机生成一组外观属性向量，使用生成器根据上述图像与随机外观属性生成一组伪造图片。使用真实图片与所生成的假图片输入判别器，判别器将真实图片和生成器所生成的伪造图片区分真伪，真实图片对其所属领域进行分类。将真实图片与伪造图片进行插值，将插值图片输入判别器，并求得其判别结果对插值图片的梯度，得到梯度惩罚。根据真实图片判别结果、伪造图片判别结果以及梯度惩罚计算对抗损失并优化判别器网络参数。根据训练判别器的次数周期性训练一次生成器，训练时使用真实图片与随机外观属性使用生成器生成伪造图片，使用所生成的伪造图片与真实图片标签再次经过生成器生成重构图片，使用重构图片与真实图片差异使用 $L_1$ 范数计算重构损失。将伪造图片输入判别器，根据判别结果计算对抗损失，根据判别器所生成领域类别标签与生成伪造图片所使用的随机外观属性标签计算差异得到分类损失。将上述重构损失、对抗损失、分类损失加权求和得到生成器总损失，优化生成器网络参数。

### 5.3.3 攻击换脸实现

攻击算法包括基于特征的攻击和基于结果的攻击，其中基于特征的攻击实现为对所提取的身份特征进行攻击，基于结果的攻击又分为使图像真实性降低的攻击、使图像出现黑色区域的攻击、使图像语义改变的攻击。在初始化攻击模块时将首先初始化加载所需模型参数以及数据集，包括大小归一化后的人脸数据集 Dataloader、FaceShifter 身份编码模型 arcface、生成器模型 $G$、StarGAN 生成器与判别器模型。当后端收到图像攻击的 API 请求后，将图像地址、算法参数、攻击类型等参数传入 ServiceAdapter，经 ServiceAdapter 创建任务、任务管理器执行任务时调用任务绑定的入口函数，由 ResourceManager 分配资源后执行攻击主函数。攻击模块根据攻击类型参数分别调用不同的攻击主函数，每种攻击方式又单独绑定一个模型类及损失函数，用于 PGD 算法输入。由于图像真实性攻击、使图像出现黑色区域的攻击、基于身份特征的攻击过程较为相似，下面首先对这三种攻击方式的模型类实现方式及攻击主函数实现过程作详

细解释。

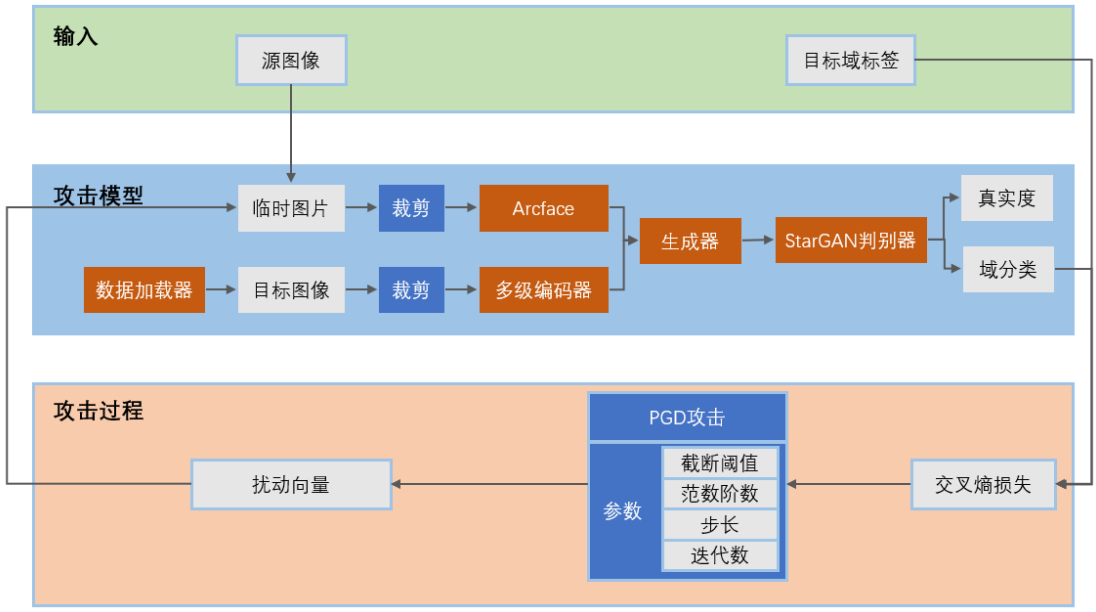



图 5-2: 图像语义层面攻击流程图

每种攻击方法所单独实现的模型类是 PGD 攻击的对象，该类基于 pytorch 框架并继承 pytorch.nn.Module 以实现前向传播时对梯度的计算，其内部通过组合换脸模型及其他模型或算法来提取攻击所需的关键特征量，例如图像真实性、mask 加权等。在图像真实性攻击方法中，基于 StarGAN 判别器对所输出换脸图 $Y_{s,t}$ 判别真伪，所需实现的模型类在初始化时获取数据加载器 Dataloader 及各模型，在前向传播时为使得攻击效果具有泛化性，将从数据集中抽取新的目标图像 $X_t$，并与所输入的 $X_{adv}$ 进行换脸后经过 starGAN 判别器提取其真实度 $D_{src}(Y_{adv,t})$，返回真实度数值。图像真实性攻击的实验效果如图 5-3所示，其中每两行为一组，每组对应 PGD 参数 $\epsilon$ 分别为 $0.3, 1, 3, 10$，每组图片中第一列分别为 $X_s$ 与 $X_t$，第二列上下均为 $Y_{s,t}$，后续列中第一行为攻击后图片 $X_{adv}$，第二行为攻击后图片与 $X_t$ 换脸得到 $Y_{adv,t}$。第三列使用第一列的 $X_s$ 进行攻击，后续列均为数据集中新图片。观察可发现，除个别图像出现明显变形之外其他图像效果并不明显，这是由于判别器对图像中微小的篡改痕迹也能识别为假，而破坏程度对识别结果并没有直接的影响，因此攻击后不足以使其出现较大的破坏痕迹。 在使图像出现黑色区域的攻击方法中，通过定义一个 mask 对换脸图像进行加权累加，以最小化该值作为攻击目标可使得换脸图像呈现 mask 所定义的效果，例如图像中央出现黑块等。所需实现的模型类在初始化时获取数据加载器及各模型，在前向传播时首先抽取新的目标图像与源图像进行换脸，并将

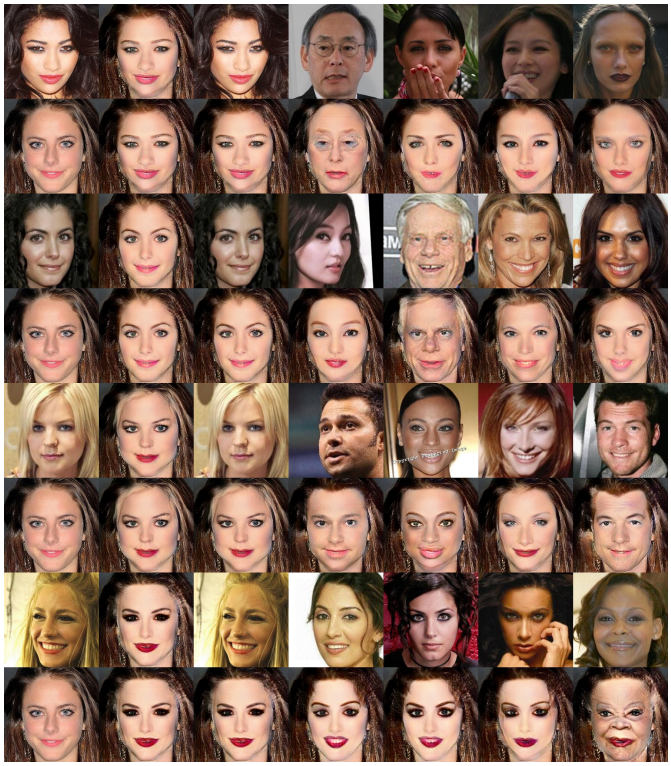


图 5-3: 图像真实性攻击效果图

换脸图像 $Y_{adv,t}$ 与预先定义的 mask 进行加权累加，返回累加值。其中为使图像中心出现黑色块，mask 中每个通道中心区域定义为 1，周围区域定义为 0。该攻击方法的实验效果如图所示，其中没两行为一组，其中每两行为一组，每组对应 PGD 参数 $\epsilon$ 分别为 $0.3, 1, 3, 10$，图像排列形式与图 5-3相同，观察可见，当 $\epsilon$ 较大时受攻击的图片经换脸所得图片出现明显攻击痕迹，不仅出现了明显扭曲变化，而且中心区域变黑，但这种变黑的痕迹需要使用较大的 $\epsilon$ 来得到。一般来说较大的 $\epsilon$ 值会使得受保护的图像添加的扰动较大，使图像质量受损。在基于身份特征的攻击方法中，将对换脸模型中的身份编码提取器进行攻击，在本系统中使用 arcface，其攻击目标为使得攻击后图像身份编码 $z_{id}(X_{att})$ 最大限度地偏离源图像身份编码 $z_{id}(X_s)$，从而改变换脸后图像的身份特征。基于身份特征的攻击所需实现的模型类初始化只需提供 arcface 模型，不需要数据加载器与其他模型，在前向传播时将源图像输入 arcface 模型，将身份编码进行返回。基于身份特征的攻击的实验效果如图所示，其中每三行为一组，参数分别为 $\epsilon = 3, 10, 30$，每组第一行为添加扰动的攻击后图像 $X_{att}$，第二行为未经攻击的换脸图片 $Y_{s,t}$，第三行为攻击后换脸图片 $Y_{adv,t}$，根据攻击前后图像对比，可见当 $\epsilon > 3$ 时图像身份受到明显变化，因此能较好保护源图像人物身份，且由

图 5-4: 使图像出现黑色区域的攻击效果图

于不需要经过换脸生成器，因此攻击速度较快。

在对图像语义的攻击中，通过设置换脸图片的目标语义标签，基于目标语义标签与原语义标签间的损失函数进行攻击，在原始图片中添加扰动，可使得攻击所得图片再次经过换脸模型所生成的换脸图片发生语义改变，且设置不同的目标语义标签能控制不同的语义改变方式。在对图像语义的攻击中，首先构造由投影梯度下降法攻击的模型，将先前训练完成的换脸模型和语义判别器串联作为被攻击模型，即以两张人脸图像作为模型输入，分别称为源图像 $X_s$ 与目标图像 $X_t$，首先将源图像 $X_s$ 通过换脸模型中的身份编码器得到身份特征，再将目标图像 $X_t$ 通过多级属性编码器得到目标图像的多级属性特征，将所得身份特征与多级属性特征输入至生成器得到换脸图像 $Y$，将换脸图像 $Y$ 经过语义判别器得到换脸图像的语义层面外观特征作为整个被攻击模型的输出。每一轮迭代基于 PGD 攻击计算对源图像添加的扰动，经过若干次迭代后该图像经过换脸所得图像将选择性地呈现出标签中被改动部分的语义信息。

换脸模型对图像语义的攻击首先构造如上所述投影梯度下降法所需要攻击的模型，该模型将换脸模型与语义判别器进行串联得到。记待编辑的源图像为 $X_s$，从数据集中随机选取一张人脸图像作为目标图像 $X_t$，每次迭代分别从数

图 5-5: 基于身份特征的攻击效果图

据集中抽取不同的目标图像。将源图像裁剪出中心 218×218 区域，并从语义判别器训练所使用的领域类别中选取一个或多个目标领域，设定或改变这些目标领域的值，例如：变为黑发、性别反转、年龄反转，目的是使得攻击后的图像再次经过换脸模型所得换脸图像具有上述特征。将换脸后的图像经过语义判别器得到原图像的属性标签，使用所选取的特征对原标签进行操作，例如：原图像的特征为为金发、男性、年轻、无眼镜，经过操作得到目标特征为黑发、女性、衰老、有眼镜。使用投影梯度下降法对所构造的模型进行攻击，设定投影梯度下降法超参数，将损失函数设为模型输出标签与目标标签的交叉熵损失。使用调整后的源图像及数据集中随机选取的目标图像作为输入，经过上述构造的由换脸模型与判别器构成的待攻击模型，输出图像领域标签，并将所得标签与修改后的目标领域标签计算损失，基于投影梯度下降法和损失进行迭代，得到待编辑图像的扰动大小。重复迭代直至到达设定的迭代轮数，经过多步梯度下降攻击后得到添加扰动后的图像。最后将该图像替代原图中心部分得到最终攻击图像。攻击流程示意图如图 5-2所示。

图 5-6: 源图像 $L_2$PGD 语义攻击效果图，$\epsilon = 0.04$

图 5-7: 源图像 $L_{inf}$PGD 语义攻击效果图，$\epsilon = 0.04$

图 5-6展示了基于 $L_2$ 范数的 PGD 攻击对源图像进行对图像语义的攻击的效果，其中每两行为一组，每组中左数第一列上下两张图分别为换脸的源图像 $X_s$ 与目标图像 $X_t$，第二列为同一张图，是源图像与目标图像经过换脸后的图像 $Y_{s,t}$。第三列至第五列分别表示修改不同语义标签对换脸模型进行攻击后的效果图，如男性、大鼻子、衰老、戴眼镜等，其中上图为对源图像添加噪声，下图为使用上图与目标图像进行换脸所得到的语义发生改变的图像 $Y'$。从图中可看出经过编辑的图片再次换脸将使得输出图片中人物呈现出所选特征的语义

变化，从而导致换脸结果被破坏。

图 5-8: 目标图像 $L_{inf}$PGD 语义攻击效果图，$\epsilon = 0.06$

图 5-7测试了使用 $L_{inf}$PGD 攻击对源图像进行编辑的效果，其中第三列起对应的语义特征修改分别为胡须、厚嘴唇、眼镜、秃头反转、鼻子大小反转。

这种语义攻击方法不仅可以用于对目标图像进行攻击，还可对源图像进行攻击，图 5-8测试了使用 $L_{inf}$PGD 攻击对目标图像进行编辑的效果，其中第三列起对应的语义特征修改分别为黑发、金发、棕发、性别反转、年龄反转。从以上图中可以看出，不同范数及不同参数下的攻击均能起到明显的攻击效果，攻击后所产生的换脸图片人物面部属性也与变化预期基本一致。

## 5.4 任务管理模块实现

任务管理模块用于有序、高效地管理并执行用户所提交的任务，包括创建任务、执行任务、跟踪任务状态、同步任务数据、任务结果回调及任务查询等功能。任务管理模块基于 python 的多进程框架 multiprocessing 实现，对每个新创建的任务分别新建一个进程执行，其目的是使得多项任务的非阻塞式高效率并行执行，同时支持任务的错误处理和用户对任务的新建与撤回操作。

在任务类的设计中，每个任务对象包含以下成员：任务名、任务创建时间、任务结束时间、任务优先级、任务状态、任务结果，如表 5-1所示。

表 5-1: 任务类成员

| 名称 | 成员变量名 | 成员类型 | 说明 |
|---|---|---|---|
| 任务 ID | task_id | String | 无碰撞唯一标识符 |
| 任务创建人 | username | String | |
| 任务类型 | task_type | String | 任务对应的算法名称 |
| 任务创建时间 | create_time | datetime.datetime | 日期类型 |
| 任务结束时间 | finish_time | datetime.datetime | 日期类型 |
| 任务优先级 | priority | Integer | 整数 1-10 之间 |
| 任务状态 | status | TaskStatusEnum | 任务执行状态，枚举类表示 |
| 任务结果 | result | String | 若结果包含较多内容，返回指向结果的链接 |

其中任务状态由枚举类 TaskStatusEnum 实现，包含以下五种状态：待执行、正在执行、运行成功、运行失败、已撤回，用于反映任务执行情况，对应表 5-2所示。

表 5-2: TaskStatusEnum 枚举类成员

| 名称 | 成员命名 | 成员数值 | 说明 |
|---|---|---|---|
| 待执行 | PENDING | 0 | 任务创建时的状态 |
| 正在执行 | STARTED | 1 | 任务执行时的状态 |
| 执行成功 | SUCCESS | 2 | 任务执行完毕，未抛出异常 |
| 执行失败 | FAILED | 3 | 任务运行出错 |
| 已撤回 | REVOKED | 4 | 人工撤销任务执行 |

任务在创建时进入待执行状态，优先级高的任务先执行，进入正在执行状态。若任务运行出错则进入运行失败状态，若正常运行至结束则进入运行成功状态。任务管理模块包含以下供外部调用的接口：任务创建、任务状态查询、任务结果查询、任务撤回。任务创建需要提供任务相关信息、任务运行主函

数、函数参数。任务管理模块将使用多进程框架中的成功回调函数与失败回调函数对任务的运行成功或失败进行处理，实现任务的异常处理、当触发回调函数时任务管理模块将任务运行结果、任务状态、任务完成时间等信息进行更新，并调用数据库接口将数据库中的任务信息进行同步。为使用户能够查询任务状态，在后端对外开放查询接口，用户通过该接口可以对数据库中所有该用户创建的任务进行查询。除此之外，用户还可对待执行状态或正在执行状态的任务进行撤回操作，被撤回的任务将中断其执行。

## 5.5 本章小结

本章对系统各模块具体实现方法作详细的阐述，包括系统身份验证模块、算法部分的人脸交换模块、人脸保护模块以及任务管理模块的实现，其中系统身份验证模块实现包括用户注册、登录的实现细节以及双 token 验证机制流程。算法模块实现从模型构造、模型训练以及攻击算法等方面完整阐述了算法在系统中的实现方案。

# 第六章　系统测试

本章分别从系统各模块测试、集成测试及非功能性测试对系统功能完整性、系统性能、系统质量等方面进行测试，并展示系统实际运行效果。系统测试是在软件开发过程结束时对完整实现的集成系统上进行测试，以验证系统是否符合需求分析所列出的实现要素，达到系统功能确认、系统风险控制的目的。系统测试分别针对功能性需求与非功能性需求进行测试，分为模块测试和集成测试，其中模块测试对系统每一子模块分别编写测试样例进行功能性测试；集成测试检测集成在一起的模块的协同行为是否存在缺陷，同时对集成单元性能负载、可靠性等非功能性需求进行测试。

## 6.1　测试环境

测试环境为一台服务器主机与一台个人电脑，其中服务端所包含的Nginx、NodeJS、MongoDB、Flask 等组件框架集中部署在同一台服务器主机上，服务器配置如表6-1所示。

表 6-1: 服务器硬件配置

| 硬件名 | 硬件信息 |
| --- | --- |
| 操作系统 | Ubuntu 16.04.6 LTS |
| 处理器 | Intel Xeon E5-2678 v3@2.50GHz |
| 内存 | 128G |
| 显卡 | NVIDIA GeForce RTX 2080 Ti |
| 网卡 | Intel I350 |

服务器主机中设置 gRPC 服务监听端口 41400，Flask 后端监听端口 41401，MongoDB 监听端口 41402，NodeJS 监听端口 41403，NginX 监听端口 41404，防火墙设置对外开放 41404 端口，NginX 反向代理规则中将/api 转发至端口 41401，其余流量转发至端口 41403。

个人主机测试环境如表6-2所示。

表 6-2: 个人主机测试环境

| 硬件名 | 硬件信息 |
| --- | --- |
| 操作系统 | Windows 10 |
| 处理器 | Intel Core i5-9300H@2.40GHz |
| 内存 | 16G |
| 显卡 | NVIDIA GeForce GTX 1650 |
| 浏览器 | Google Chrome, Firefox |

## 6.2 模块测试

模块测试用于验证构成系统的每个模块是否能正确执行其各个功能，模块测试中又包含更细粒度的单元测试，用于验证组成程序的相对较小的单元是否正确执行其各个功能。本节分别对系统中身份验证模块和算法模块进行测试，分别测试每个模块的功能完整性、检验每个功能的正确性以及相关非功能性测试，保证模块的正常运行，其中身份验证模块测试包括用户登录登出等功能实现是否正常，以及双 token 验证机制是否正常完成；算法模块测试换脸功能以及四种攻击方法的正确性及有效性。

### 6.2.1 身份验证模块测试

身份验证模块测试内容包括用户访问控制功能、用户访问权限测试、字段合法性检测。测试方法为使用多种表单信息创建用户，并使用创建的用户测试其访问权限、登录登出使用情况。当用户注册与登录时，需要根据用户表单信息有效性返回注册或登录成功或失败。当用户登出后，自动删除本地 cookie、token，并自动进入登录页面。对于不同的用户角色，系统分别根据角色类型给用户提供不同的访问权限及功能。后端在检查用户提交的表单时，当表单中信息包含非法字符时请求返回失败。当用户请求服务端数据、资源时，系统需要根据用户 token 有效性及用户访问权限，返回请求成功或失败。测试结果如表6-3所示。

表 6-3: 身份验证模块测试结果

| 测试内容 | 说明 | 测试结果 |
| --- | --- | --- |
| 访问控制 | 用户注册、登录功能 | 通过 |
| 用户登出 | 用户登出功能 | 通过 |
| 用户角色 | 包含管理员和普通用户 | 通过 |
| 表单字段 | 根据表单字段合法性处理请求 | 通过 |
| 数据请求 | 用户请求服务端数据、资源 | 通过 |

对 token 机制的测试包括四部分：创建 token、验证 token、刷新 token 及登出删除 token。通过浏览器可查看请求所携带的 cookie，登录时所返回的 cookie 如图6-1所示，其中包含了后端所创建并返回的经加密的 access_token 与 refresh_token，且所包含的 cookie 类型为 HTTP-Only，表示无法通过 javascript 读取。同时由于开启了 CSRF 保护设置，因此返回 csrf_access_token 及 csrf_refresh_token，共四个 token。 调用后端 API 时，从浏览器中查看请求

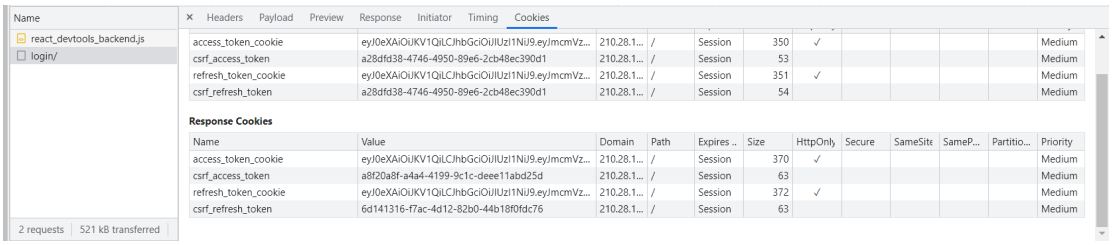



图 6-1: 登录时携带于 cookie 中的 token 示意图

头，可看到验证方式一栏显示使用 JWT 方式，且所包含的 token 为登录时创建的 csrf_access_token。当访问 cookie 超过有效期失效时，将调用后端刷新 API，后端收到一个包含在 HTTP-Only 类型 cookie 中的 refresh_token，并返回一组新的 token。在登出时，本地所储存的 cookie 将被丢弃，因此再次登录时请求中不再包含 cookie。经测试，身份验证模块运作正常，双 token 验证机制有效。

### 6.2.2 换脸模块与攻击模块测试

本节测试换脸模块与攻击模块正确性，其中包括所设计的四种攻击方式：图像真实性攻击、使图像出现黑色区域的攻击、基于身份特征的攻击、对图像语义的攻击，分别测试每个任务耗时、运行结果以及在多任务并行环境下的运

行情况。以下测试了人脸交换以及默认参数下 ($\epsilon = 10$, 步数 $= 200$) 的四种攻击方法，测试结果如表 6-4所示。

表 6-4: 人脸交换模块测试结果

| 测试内容 | 说明 | 耗时 | 测试结果 |
|---|---|---|---|
| 人脸交换 | 将所给两图人脸进行交换 | 21s | 通过 |
| 图像真实性攻击 | 降低攻击后换脸真实度 | 50s | 通过 |
| 出现黑色区域的攻击 | 使攻击后换脸图中央出现黑块 | 49s | 通过 |
| 基于身份特征的攻击 | 使攻击后换脸图像身份发生变化 | 23s | 通过 |
| 对图像语义的攻击 | 使攻击后换脸图像所指定的语义属性发生变化 | 100s | 通过 |

图 6-2: 使图像出现黑色区域的攻击测试

图 6-2展示了系统返回的使图像出现黑色区域的攻击测试结果图，算法参数分别为 $\epsilon = 30$, 步数 $= 200$，其中第一列分别为源图像 $X_s$ 和添加扰动后图像 $X_{att}$，其后图片中第一行为源图像针对不同目标图像换脸所得结果 $Y_{s,t}$，第二行为攻击后 $X_{att}$ 换脸结果 $Y_{att}$，观察可见图中央出现明显黑色块，换脸结果被破坏。

图 6-3展示了系统返回的对图像语义的攻击结果图，其中每两行为一组，每组第一列分别为源图像 $X_s$ 和目标图像 $X_t$，第二列为换脸后图像 $Y_{s,t}$，其后图片中第一行为源图像对不同语义修改进行攻击的添加扰动后的图像 $X_{att}$，第二行为不同语义下攻击后 $X_{att}$ 的换脸结果 $Y_{att}$。左起第 3 至第 7 幅图分别代表黑发、金发、棕发、性别反转、年龄反转，可见对原为黑发的图片，金发及棕发效果较明显，对黑发与性别反转以及对部分人像攻击效果不是特别显著，但面部仍会出现一些斑纹状特征。

图 6-3: 对图像语义的攻击测试

## 6.3 集成测试

集成测试将各个软件模块组合起来并作为一个整体进行测试，执行集成测试以评估系统和组件是否符合指定的功能需求。本节对系统架构运作进行集成测试，包括前后端架构测试、任务管理模块中多任务并行机制测试。

### 6.3.1 前后端架构测试

前后端架构中 Flask 与 Nodejs 分别由 NginX 作反向代理进行转发，系统运行时占用端口如图 6-4所示，图中分别显示 Flask 监听 41401 端口、MongoDB 监听 41402 端口、NodeJS 监听 41403 端口、NginX 监听 41404 端口、gRPC 服务监听 41400 端口，各组件运行正常。 对系统路由功能进行测试，路由功能分为 NginX 反向代理和页内路由，分别向服务器提交 API 请求与网页请求，预期为 Flask 后端和 NodeJS 分别收到请求，以测试 NginX 反向代理功能是否正常执行；通过发送其他路径请求可测试相关路由机制，执行结果如表 6-5所示。

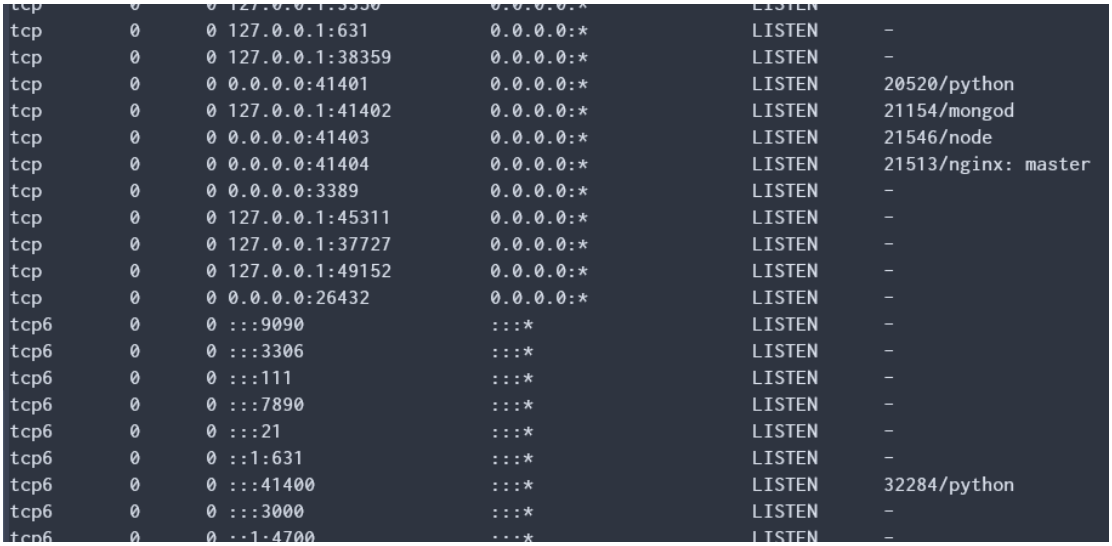



图 6-4: 系统运行端口情况

表 6-5: 系统路由功能测试

| 端口号 | 请求路径或操作 | 测试结果 | 预期结果 |
|---|---|---|---|
| 41404 | /api/v1/dummy-get | 通过 | Flask 收到请求，前端收到响应 |
| 41404 | /login | 通过 | NodeJS 收到请求，前端渲染登录页面 |
| 41404 | 使用导航栏导航页面 | 通过 | 分别导向指定页面 |
| 41404 | /api/v1/false-api | 通过 | 404 页面不存在 |
| 41404 | /false-page | 通过 | NodeJS 收到请求，自动重定向至/login 页面 |
| 41400-41403 | 任意路径 | 通过 | 响应超时或拒绝访问 |

在系统防火墙中对 NginX 端口流量放行后，当从 NginX 端口对后端服务发起请求时，NginX 将自动重定向至服务所对应的端口下的所指路径，NginX 根据所编写的规则根据访问路径的不同分别重定向至 NodeJS、Flask API 以及 gRPC API，而当向其他端口或直接向后端服务端口发起请求时将被防火墙拦截致使响应超时或拒绝访问。因此系统前后端架构测试无异常。

### 6.3.2 任务机制测试

系统采用任务机制为用户创建的任务分配资源计算。用户通过换脸模块或攻击模块创建的算法任务会以队列形式储存在后端，并同步至数据库，在任务管理版块中可查询该用户所创建的任务。每个任务显示其执行状态，用户可对任务进行撤回、删除等操作。表 6-6展示了系统任务机制测试结果。

表 6-6: 系统任务机制测试

| 操作 | 测试结果 | 预期结果 |
|---|---|---|
| 同一用户新建多任务 | 通过 | 列表中显示所创建的任务，并能查看结果 |
| 不同用户新建多任务 | 通过 | 每个用户只看到自己创建的任务 |
| 任务执行出错 | 通过 | 任务执行中断，状态更新为失败 |
| 任务撤回 | 通过 | 任务中止运行，状态更新为撤回状态 |
| 任务删除 | 通过 | 任务不再显示，并从数据库中删除 |

图 6-5为任务管理面板中新建的任务执行完毕情况，其中状态由正在运行变为已完成，并显示结果链接。 经测试，任务管理功能完整，能实现多任务异


图 6-5: 任务管理机制测试

步并行机制，在经过完整的任务执行流程后用户可正常查询结果。

## 6.4 非功能性测试

非功能性测试用于测试系统是否满足各项非功能性需求，包括系统是否具有良好的扩展性、系统数据安全性、是否易受攻击、系统数据吞吐量大小，在高并发请求下的表现等多项非功能性评价标准。下面分别对后端进行负载压力测试，以及对攻击模块并行化的性能进行测试，以反映系统整体的抗压能力与计算能力。

### 6.4.1 负载测试

负载测试通过所用多个线程并发地向服务器发送大量请求，能了解服务器的吞吐量、最大请求数等信息。这里使用 Apache JMeter 对服务器进行负载测试，设置线程数为 100，每个线程循环发送请求 10 次，从首个线程到最后一个线程启动的时间间隔为 100 秒，单次请求向服务器 41404 端口发送 HTTP GET 请求，设置跟随重定向、持久连接，并获取所有嵌入资源，根据每次请求的响应时间绘制散点图，并计算平均响应时间。吞吐量等性能指标。个人主机发出的请求将经过若干次跳转后对服务端登录页面进行请求，其测试结果如图 6-6所示。 图中数据显示服务器吞吐量为每分钟 320 次请求，单次请求最低

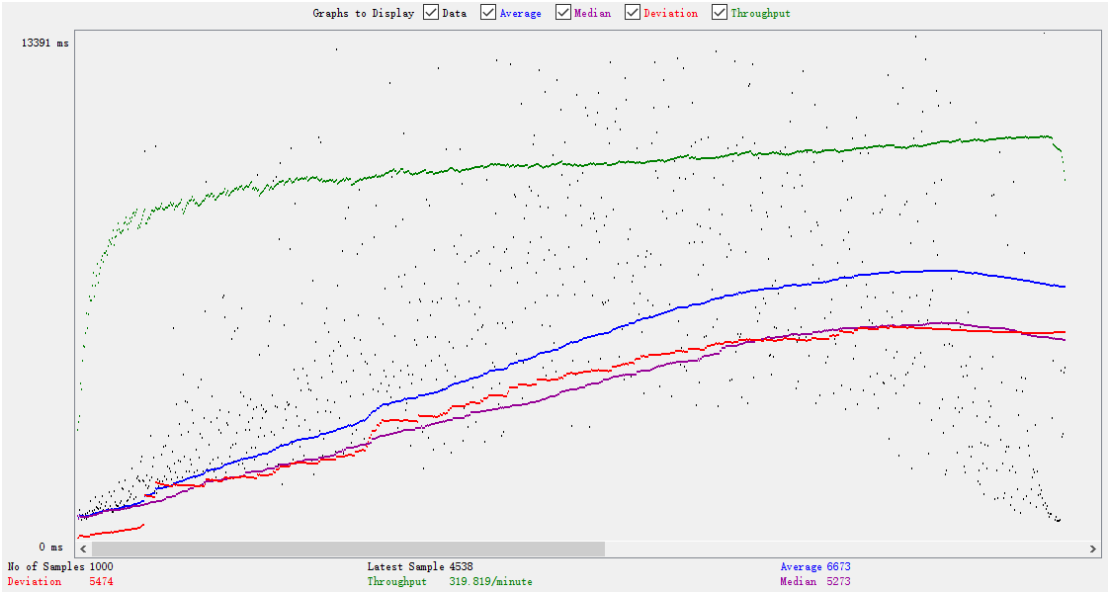



图 6-6: JMeter 负载测试

时延约 400ms，高负载时平均时延 6673ms，偏差为 5474ms。这表明系统在用户并发数量少的情况下有较高的响应速度和较低的延迟，这随用户并发数量增多而呈近似线性增长的趋势，即在用户并发数量超过系统所能承受的吞吐量情况下请求具有较低的响应速度和较高的延迟。

### 6.4.2 性能测试

本节对攻击模块进行性能测试，通过向后端同时发送多个人脸保护请求，测试攻击算法在多任务并行条件下执行的效率，以反映不同设备数情况下系统的运行效率以及加速情况。测试中使用 1 至 8 个 GPU，分别使用单类型任务和

混合类型任务进行测试，其中使用基于身份特征的攻击作为单类型任务，轮流使用四种攻击作为混合类型任务，测试不同设备数量下的耗时情况，测试结果如表 6-7所示。

表 6-7: 攻击模块性能测试结果

| 设备数 | 任务数 | 总耗时 | 平均任务耗时 |
|---|---|---|---|
| 1 | 单类型, 10 | 229s | 22.9s |
| 2 | 单类型, 20 | 260s | 13.0s |
| 8 | 单类型, 80 | 302s | 3.8s |
| 1 | 混合类型, 10 | 630s | 63.0s |
| 2 | 混合类型, 20 | 808s | 40.4s |
| 8 | 混合类型, 80 | 1007s | 12.6s |

由数据可知，系统中平均任务耗时随设备数增多而降低，设备数为 8 时单类型加速比达到 6.03，混合类型加速比为 5.0，显著地提高了任务执行效率，其加速比的不同是由于混合类型中不同算法所消耗的时间与资源不同所产生的差异。随着设备数的增多，系统并行产生的增益越小，但平均任务耗时也在显著降低，因此整体上算法的并行化能带来可观的收益，是一种有效提升系统性能的手段。

## 6.5 运行效果

本节展示系统的运行效果。

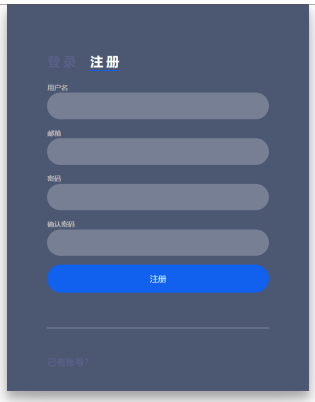



图 6-7: 系统注册界面

图6-7为新用户访问时的系统注册与登录页面，用户可以便捷地在注册与登录版块之间切换，实现对系统的访问。

图6-8为登录系统后进行换脸攻击操作的版块，用户可以上传本地图像，从四种攻击类型中选择一种，并设定攻击参数，提交后便可在任务管理页面查询到任务进度及攻击运行结果。


图 6-8: 系统主界面

图6-9为任务管理界面，所有该用户创建的任务均在该版块下列出。任务记录了算法执行类型、提交时间、运行结束时间、任务执行状态以及访问任务结果的链接，对运行成功的任务，用户点击右侧链接可查看执行结果。

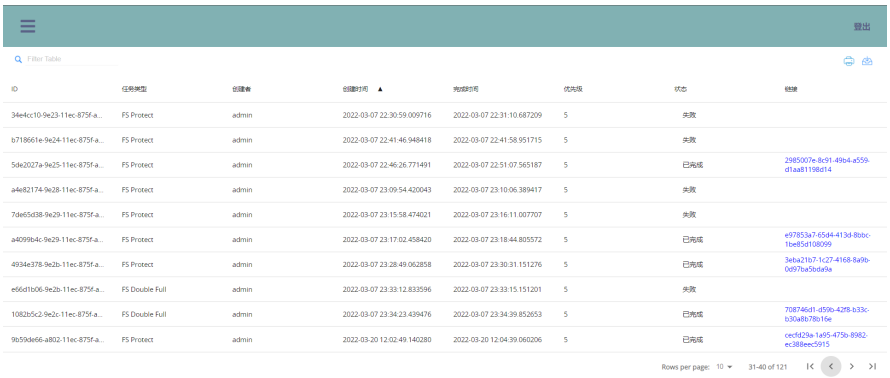



图 6-9: 系统任务界面

图6-10展示了人脸交换效果图，其中左图为源图像 $X_s$，中图为目标图像 $X_t$，右图为交换后图像 $Y_{s,t}$，可看到 $Y_{s,t}$ 中既包含了源图像的身份特征，又包含目标图像的外观属性特征。

图 6-10: 人脸交换效果图

图 6-11: 目标图像语义攻击

图6-11展示了使用目标图像 $X_t$ 进行攻击的效果图。图中第一行为原图经过添加噪声后的结果，后三行分别为从 CelebA 数据集中随机选取的三张源图像 $X_{s1}, X_{s2}, X_{s3}$ 与添加噪声后的目标图换脸所得结果。从图中可见，换脸结果中面部属性均发生了改变，其中第四列所代表的语义特征变化最明显。

## 6.6 本章小节

本章对系统分别进行模块测试、接口测试、集成测试、负载测试，系统功能完整，实现了换脸算法与可控攻击功能，各模块均正常运行，在压力环境下系统可容纳一定的并发请求数，但在请求数过多时会出现明显的延迟，可见在系统架构设计上对吞吐量的增加有改进空间。

# 第七章　结语

## 7.1　全文总结

本文分别从攻击特征层面和攻击结果层面两种角度，基于对抗攻击设计了人脸保护方法，并将其应用于人脸保护系统，系统的开发经过需求分析、总体设计、模块设计、模块实现、系统测试等步骤完成。经过验证，该系统可以提供有效的人脸保护服务，对于经过保护的人脸，再次经过换脸后图像中会出现不同形式的破坏，从而一定程度上避免了对被换脸人权益的侵害，有效实现了人脸保护功能。

结合现有系统及需求，系统的总体设计采用 B/S 架构，并基于前后端分离原则，技术架构采用 Flask+NodeJS+React 的方案，使用 NginX 解决跨域问题并提升系统性能，对外开放 RESTful API 和基于 gRPC 传输的 API。系统基于双令牌验证机制实现了身份验证模块，基于已公开 FaceShifter、MTCNN 模型实现了换脸功能，基于已公开的 StarGAN 模型、PGD 攻击实现了包括四种攻击方法的攻击模块，并基于 python 多进程框架实现了任务管理模块。最后，在系统测试阶段通过模块测试、集成测试对系统整体功能实现与系统性能进行验证，得出了系统在正常访问情况下系统响应时间满足正常访问需求的结论。

系统中身份验证模块基于双 token 验证机制，使用一组访问 token 与刷新 token 进行访问控制与用户授权，当访问 token 失效后使用刷新 token 重新获取一组新的 token，延长会话长度。使用 token 也在用户请求服务时为系统提供了安全可靠的权限验证功能。

系统中攻击模块基于 PGD 攻击实现了对换脸模型从特征层面及从结果层面进行攻击的方法，分别对模型所提取的身份特征和换脸结果进行不同形式的攻击，为用户提供了丰富的选择空间。

系统中任务管理模块基于多进程框架，需较长时间执行的任务不必由用户等待任务执行完成，而是将任务信息持久化至数据库中，当任务执行完成后用户对任务结果进行查询，实现了多任务异步执行机制。

本文的主要工作包括：

1. 提出了从特征层面对换脸模型进行对抗攻击的人脸保护方法，在传统的攻击方案基础上，通过攻击换脸模型中的身份特征提取部分，使得模型提取的特征发生变化，使其使用错误的特征进行换脸，能较好地隐藏换脸图片中被换脸者的身份信息。同时，由于在特征提取层面特征就已经发生变化，因此无论后续采用何种模型，换脸都将无效，这使得攻击方法具有更强的鲁棒性和稳定性，能在一定程度上解决换脸技术存在的隐私风险。

2. 提出了从换脸结果层面对换脸模型进行对抗攻击的人脸保护方法，以换脸结果的破坏为目的，对完整的模型进行攻击，且通过设计不同的攻击目标，能产生不同的改变换脸图像人物认知的破坏形式，其中包括真实性降低、特定位置出现黑块、语义特征发生改变等攻击方案，从而实现多种类、稳定、可靠的人脸保护。

3. 设计并实现了完整的基于对抗攻击进行人脸保护的系统，其中分别基于特征和基于结果进行攻击，设计了四种攻击形式，并对外提供可靠的人脸保护服务。系统架构以 Flask 为后端，NodeJS 与 React 为前端，结合 MongoDB 数据库与 NginX 反向代理实现，基于双 token 机制实现了身份验证模块，并设计了基于任务队列的单服务器多任务并发调度模式，使其具有异步任务处理能力，提升系统运行效率。

## 7.2 未来展望

本文所设计的人脸保护方法仍然具有一定局限性，其保护措施仍可通过一定手段进行规避，例如可以对图像进行降噪、重采样等方法减少或改变攻击所添加的扰动。另外，这种保护方法是针对成本低廉的基于 GAN 的换脸系统进行攻击，而对于其他少数类型的换脸算法，例如基于特征点将人脸剪切、拼接的算法，或是借助 Photoshop 等工具人工进行人脸编辑的方法则无法起到很好的保护效果。因此，在未来进一步的保护工作中，针对上述存在的问题，可以尝试对其他保护方法展开研究，例如添加隐藏水印、隐藏标记等方法，以进一步扩大保护适用范围。由于本文所使用的数据集有限，未在更广泛的场景下测试其有效性，因此在未来工作中可针对更多环境条件下的人物图像测试并优化其攻击性能。除此之外，可基于集成攻击方法提升算法在不同模型参数情况下或其他 GAN 架构模型上的泛化应用能力，还可增加针对人脸特定区域进行编辑、水印编辑等功能。在系统层面，可针对系统性能、可用性、可维护性作进

一步优化，例如引入中间件 Redix 优化系统在高并发下的响应速度，并行框架可作进一步改进和优化，还可基于 Jenkins 和 Kubernetes 等框架实现系统的自动部署，提升系统的维护性与持续集成能力。

# 参考文献


[1] BREGLER C, COVELL M, SLANEY M. Video Rewrite: Driving Visual Speech with Audio[J], 1997 : 353 – 360.

[2] THIES J, ZOLLHOFER M, STAMMINGER M, et al. Face2Face: Real-Time Face Capture and Reenactment of RGB Videos[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[3] Face Swap: an app for Android and iOS[EB/OL]. [2022-03-15]. https://www.microsoft.com/en-us/garage/profiles/face-swap.

[4] HUANG H, LI Z, HE R, et al. IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis[C] // BENGIO S, WALLACH H, LAROCHELLE H, et al. Advances in Neural Information Processing Systems : Vol 31. : Curran Associates, Inc., 2018.

[5] CHOI Y, CHOI M, KIM M, et al. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018.

[6] Face Swap Live: Switch faces with your friends and photos in real-time[EB/OL]. [2022-03-15]. http://faceswaplive.com.

[7] Snapchat now lets you face swap with pictures from your camera roll[EB/OL]. [2022-03-15]. https://www.theverge.com/2016/4/22/11486630/snapchat-update-free-replays-face-swap-photos.

[8] RUIZ N, BARGAL S A, SCLAROFF S. Disrupting Deepfakes: Adversarial Attacks Against Conditional Image Translation Networks and Facial Manipulation Systems[J]. CoRR, 2020, abs/2003.01279.

[9] Most Popular Selfie Editor – Selfie magic with just one tap[EB/OL]. [2022-03-15].
https://www.faceapp.com/.

[10] MAHAJAN S, CHEN L-J, TSAI T-C. SwapItUp: A Face Swap Application for Privacy Protection[C] // 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA). 2017 : 46 – 50.

[11] BLANZ V, SCHERBAUM K, VETTER T, et al. Exchanging Faces in Images[J]. Computer Graphics Forum, 2004, 23(3) : 669 – 676.

[12] ZENO B, KALINOVSKIY I, MATVEEV Y, et al. CtrlFaceNet: Framework for Geometric-driven Face Image Synthesis[J]. Pattern Recognition Letters, 2020, 138.

[13] ZHAO J, XIONG L, KARLEKAR J, et al. Dual-Agent GANs for Photorealistic and Identity Preserving Profile Face Synthesis (updated)[C] // . 2017.

[14] KARRAS T, AILA T, LAINE S, et al. Progressive Growing of GANs for Improved Quality, Stability, and Variation[J]. CoRR, 2017, abs/1710.10196.

[15] KARRAS T, LAINE S, AILA T. A Style-Based Generator Architecture for Generative Adversarial Networks[C] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019.

[16] PUMAROLA A, AGUDO A, MARTINEZ A M, et al. GANimation: Anatomically-aware Facial Animation from a Single Image[C] // Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[17] KORSHUNOVA I, SHI W, DAMBRE J, et al. Fast Face-Swap Using Convolutional Neural Networks[C] // Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2017.

[18] PENG B, FAN H, WANG W, et al. A Unified Framework for High Fidelity Face Swap and Expression Reenactment[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021 : 1 – 1.

[19] BHATTACHARJEE S, MARCEL S. What You Can't See Can Help You - Extended-Range Imaging for 3D-Mask Presentation Attack Detection[C] // 2017 International Conference of the Biometrics Special Interest Group (BIOSIG). 2017 : 1 – 7.

[20] KHODABAKHSH A, RAMACHANDRA R, RAJA K, et al. Fake Face Detection Methods: Can They Be Generalized?[C] // 2018 International Conference of the Biometrics Special Interest Group (BIOSIG). 2018 : 1 – 6.

[21] ZHOU P, HAN X, MORARIU V I, et al. Two-Stream Neural Networks for Tampered Face Detection[C] // 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). 2017 : 1831 – 1839.

[22] RÖSSLER A, COZZOLINO D, VERDOLIVA L, et al. FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces[J]. CoRR, 2018, abs/1803.09179.

[23] GUO Z, YANG G, CHEN J, et al. Fake face detection via adaptive manipulation traces extraction network[J]. Computer Vision and Image Understanding, 2021, 204 : 103170.

[24] FRIDRICH J, KODOVSKY J. Rich Models for Steganalysis of Digital Images[J]. IEEE Transactions on Information Forensics and Security, 2012, 7(3) : 868 – 882.

[25] LI H, LI B, TAN S, et al. Identification of deep network generated images using disparities in color components[J]. Signal Processing, 2020, 174 : 107616.

[26] AGARWAL S, FARID H, GU Y, et al. Protecting World Leaders Against Deep Fakes[C] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 2019.

[27] MATERN F, RIESS C, STAMMINGER M. Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations[C] // 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW). 2019 : 83 – 92.

[28] LI Y, CHANG M-C, LYU S. In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking[C] // 2018 IEEE International Workshop on Information Forensics and Security (WIFS). 2018 : 1 – 7.

[29] CIFTCI U A, DEMIR I, YIN L. FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020 : 1 – 1.

[30] WU H-Y, RUBINSTEIN M, SHIH E, et al. Eulerian Video Magnification for Revealing Subtle Changes in the World[J]. ACM Transactions on Graphics - TOG, 2012, 31.

[31] YEH C-Y, CHEN H-W, TSAI S-L, et al. Disrupting Image-Translation-Based DeepFake Algorithms with Adversarial Attacks[C] // Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops. 2020.

[32] SEGALIS E. Disrupting Deepfakes with an Adversarial Attack that Survives Training[J]. CoRR, 2020, abs/2006.12247.

[33] WILLETTS M, CAMUTO A, RAINFORTH T, et al. Improving VAEs' Robustness to Adversarial Attack[J]. arXiv: Machine Learning, 2021.

[34] LI D, WANG W, FAN H, et al. Exploring Adversarial Fake Images on Face Manifold[C] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021 : 5789 – 5798.

[35] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[J]. Advances in neural information processing systems, 2014, 27.

[36] LI L, BAO J, YANG H, et al. FaceShifter: Towards High Fidelity And Occlusion Aware Face Swapping[J]. CoRR, 2019, abs/1912.13457.

[37] GOODFELLOW I, SHLENS J, SZEGEDY C. Explaining and Harnessing Adversarial Examples[J]. arXiv 1412.6572, 2014.

[38] KURAKIN A, GOODFELLOW I J, BENGIO S. Adversarial examples in the physical world[J]. CoRR, 2016, abs/1607.02533.

[39] DONG Y, LIAO F, PANG T, et al. Discovering Adversarial Examples with Momentum[J]. CoRR, 2017, abs/1710.06081.

[40] BUBECK S. Convex Optimization: Algorithms and Complexity[J]. Found. Trends Mach. Learn., 2015, 8(3–4): 231–357.

[41] MADRY A, MAKELOV A, SCHMIDT L, et al. Towards deep learning models resistant to adversarial attacks[J]. arXiv preprint arXiv:1706.06083, 2017.

[42] GHIMIRE D. Comparative study on Python web frameworks: Flask and Django[J], 2020.

[43] IDRIS N, FOOZY C F M, SHAMALA P. A generic review of web technology: Django and flask[J]. International Journal of Advanced Science Computing and Engineering, 2020, 2(1): 34–40.

[44] DEJONGHE D. Nginx CookBook[M]. O'Reilly Media, 2020.

[45] NEDELCU C. Nginx HTTP Server[M]. Packt Publishing Ltd, 2015.

[46] SONI R. Nginx[M]. Springer, 2016.

[47] GACKENHEIMER C, PAUL A. Introduction to React: Vol 52[M]. Springer, 2015.

[48] AGGARWAL S. Modern web-development using reactjs[J]. International Journal of Recent Research Aspects, 2018, 5(1): 133–137.

[49] GYŐRÖDI C, GYŐRÖDI R, PECHERLE G, et al. A comparative study: MongoDB vs. MySQL[C] // 2015 13th International Conference on Engineering of Modern Electric Systems (EMES). 2015: 1–6.

[50] TOLOSANA R, VERA-RODRIGUEZ R, FIERREZ J, et al. Deepfakes and beyond: A Survey of face manipulation and fake detection[J]. Information Fusion, 2020, 64: 131–148.

# 致　　谢

时光荏苒，三年的研究生涯即将结束，回想过去三年里自己在南京大学的收获和成长，感慨万分。在此论文完成之际，向所有为我提供帮助、关心和支持的人表示感谢。

首先，我向我的导师申富饶教授表示由衷的感谢，申老师严谨的治学态度和求实的科研精神对我产生了深远的影响，三年来在科研方面给予我极大的支持，不断支持和激励着我前行，每周一次与组内同学的一对一交流以及讨论班研究内容的分享也是坚持进行，不仅给予我研究上的指导，还提供了人生建议。此次学位论文也是在导师的悉心指导下完成，在研究过程与论文写作中为我提出了诸多有价值的指导意见，能遇见这样的好导师，我感到十分庆幸。

其次，我要感谢我的同组同学，感谢你们陪伴我度过了三年研究生生活，知识的交流与分享也让我收获颇丰，充实了我三年的生活。感谢和我一起打比赛的同学，与我携手共同进步。感谢学长为我提供了诸多帮助和支持。此外，还要感谢我的室友，带给我诸多美好回忆的同时也建立了深厚友谊，我将终身难忘。

最后，我要感谢我的家人，父母是我成长道路背后默默付出的人，为我提供了诸多鼓励与关怀，也为我提供了良好的生活保障，使我能够集中精力投入到学习之中，今日取得的成绩离不开父母的付出。

# 简历与科研成果

## 基本信息

迟宇翔，男，汉族，1997 年 12 月出生，江苏省苏州人。

## 教育背景

**2019 年 9 月 — 2022 年 6 月** 南京大学人工智能学院 硕士

**2015 年 9 月 — 2019 年 6 月** 南京大学匡亚明学院 本科

## 攻读硕士学位期间的发明专利

1. 范彧, 迟宇翔, 俞扬, 一种基于对抗学习的数据隐私保护方法 (202210372873.X).
2. 迟宇翔, 范彧, 基于强化学习的多目标复杂交通场景下自动驾驶解决方法 (202210370991.7).

## 攻读硕士学位期间参与的科研课题

1. 国家自然科学基金“基于深度感知增量式联想记忆神经网络的信息融合系统研究”（项目编号：61876076，课题年限 2019 年 1 月-2022 年 12 月），负责神经网络模型相关研究。

# 《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：迟宇翔

2022年 5月 25日

| | | | | | |
|---|---|---|---|---|---|
| 论文题名 | 基于对抗攻击换脸的人脸保护系统设计与实现 | | | | |
| 研究生学号 | MG1937004 | 所在院系 | 人工智能学院 | 学位年度 | 2022 |
| 论文级别 | ☑学术学位硕士 ☐专业学位硕士<br>☐学术学位博士 ☐专业学位博士<br>(请在方框内画钩) | | | | |
| 作者 Email | szfxcyx@126.com | | | | |
| 导师姓名 | 申富饶 教授 | | | | |

论文涉密情况：

☑ 不保密

☐ 保密，保密期（_____年_____月_____日 至 _____年_____月_____日）

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。