研 究 生 毕 业 论 文
（申请工程硕士学位）

论 文 题 目 对抗样本检测和防御系统的设计与实现

作 者 姓 名 _____

学科、专业名称 _____

研 究 方 向 _____

指 导 教 师 _____

2021 年 **5** 月 **18** 日

学　　号：

论文答辩日期：**xxxx** 年 **xx** 月 **xx** 日

指 导 教 师：　　（签字）

The Design and Implementation of the System for Detecting and Defending against Adversarial Examples

by

Supervised by

A dissertation submitted to
the graduate school of
in partial fulfilment of the requirements for the degree of
Master of Engineering
in

May 18, 2021

研究生毕业论文中文摘要首页用纸

毕业论文题目：对抗样本检测和防御系统的设计与实现

专业 级硕士生姓名：

指导教师（姓名、职称）：

摘　要

随着深度学习技术在各种领域的应用越来越多，深度学习鲁棒性变得至关重要。对抗样本的提出给深度学习鲁棒性带来了极大的挑战。对抗样本通过对图像添加细微的扰动，肉眼无法察觉，却可以轻松使得深度学习模型出现错误预测。因此研究者们提出了许多针对对抗样本的防御方法。现有的对抗防御方法，如特征压缩和由高级特征主导的去噪器（HGD），在面对不同的攻击方法时表现各有优劣，无法保证防御的有效性。同时对防御效果的评估只关注模型预测准确率，缺乏对经过防御优化的数据集的度量。

本文从数据集的层面防御对抗样本，构建一个基于对抗检测与防御的深度学习鲁棒性优化系统。系统提供对抗样本的检测与防御服务，对不同的对抗样本类型提供针对性的防御。系统针对对抗防御效果因样本类型而异的特点，提供对抗样本检测模型，检测对抗样本类型，并结合系统维护的对抗防御效果表推荐最佳的方法。针对防御优化产生的数据集的评估，引入结构相似性（SSIM）和扰动敏感距离（PSD）作为图像质量的评估标准，判断防御所得的数据集是否出现图像的失真。本系统的功能分为数据流处理、对抗检测、对抗防御和任务报告模块，通过模块之间的协作来确保系统的可用性。

本系统在部署时覆盖 9 种对抗攻击方法，检测对抗样本类型的成功率在 90% 以上。检测出对抗样本类型后，进一步选择对抗防御方法。以面对由 BIM 生成的对抗样本为例，模型预测准确率低于 2%，应用系统推荐的特征压缩方法后，能够将模型预测准确率提升至 93% 左右。对比随机选择的防御方法如 HGD，准确率仅为 81%，前者效果更佳。通过 SSIM 和 PSD 的评估，对抗防御生成的样本与原样本结构基本一致，扰动敏感距离在允许范围内，样本的质量符合预期。综上，系统能够为用户提供良好的对抗样本检测和防御服务。

关键词： 深度学习，对抗样本检测，对抗防御，鲁棒性，数据集质量评估

研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of the System for Detecting and Defending against Adversarial Examples

SPECIALIZATION:

POSTGRADUATE:

MENTOR:

Abstract

With the increasing application of deep learning technology in various fields, deep learning testing becomes crucial. The proposal of adversarial examples brings great challenges to the robustness of deep learning. By adding subtle perturbations to the image that are invisible to the naked eye, the adversarial example can easily make the deep learning model make false predictions. The researchers therefore propose a number of defences against the adversarial examples. Existing adversarial defense methods, such as feature squeezing and HGD, have their own advantages and disadvantages in the face of different attack methods, and cannot guarantee the effectiveness of defense. At the same time, the evaluation of defense effect only focuses on the accuracy of model prediction and lacks the measurement of the data set optimized by defense.

In this thesis, a DL robustness optimization system based on AE detection and defense is constructed based on adversarial defenses at the level of data sets. The system provides detection and defense services for adversarial examples, and provides targeted defense for different adversarial examples. Considering that the effect of adversarial defenses varies with the type of AEs, the system providess the detection model of AEs and recommends the best method based on the table of adversarial defense effects maintained by the system. In order to evaluate the data set generated by defense, Structural Similarity (SSIM) and Perturbation Sensitivity Distance (PSD) are introduced as the evaluation criteria of image quality to judge whether the image distortion occurs in the data set.

The system covers 9 adversarial attack methods when deployed, and the prediction success rate was more than 90% after the training. After detecting the type of adversarial example, the adversarial defense method will be applied. For example, when faced

with the adversarial examples generated by BIM, the prediction accuracy of the model is less than 2%, After applying the feature squeezing method recommended by the system, the prediction accuracy of the model can be improved to about 93%. The former is more effective than a randomly chosen defense method such as HGD, which has an accuracy rate of only 81%. Compared to a randomly selected defense method such as HGD, its accuracy was only 81%. Through the evaluation of SSIM and PSD, the structure of the samples generated by the system is consistent with the original samples, and the disturbance sensitivity distance is within the allowable range of the experiment, which meets the requirements of the system. In conclusion, the system can provide users with good adverarial example detection and defense services.

keywords: Deep Learning, AE Detection, Adversarial Defense, Robustness, Dataset Quality Assessment

目　　录

插图清单

附表清单

第一章　引言

。

1.1　选题的背景和意义

人工智能 (AI) 时代的到来，使得深度神经网络在许多领域都显示出了巨大的优越性。然而深度神经网络十分脆弱，容易遭受对抗样本 [1] 的攻击，这些攻击针对数据集进行细微的扰动，从而愚弄深度学习模型。因此，神经网络难以被广泛应用于安全至关的领域。为此，研究者提出了多种构建对抗样本的方法 [2] 来测试神经网络，目的是提高神经网络的鲁棒性。

在对抗性深度学习中，对抗样本的目标是使模型对输出进行错误分类。根据扰动对分类器的不同影响，我们将对抗目标分为四种类型: 置信度降低: 降低输出分类的置信度。无目标性误分类: 将输出分类改变为与原分类不同的任何类。有针对性的误分类: 将输出分类强制为攻击者特定的目标类。源/目标误分类: 攻击者选择特定的输入，生成特定于攻击者的目标类。

同时，对于对抗样本的防御 [3] 也因此诞生。不管是模型层面的防御还是数据层面的防御，都有一些共同的目标：对模型体系结构的低影响: 当构建任何形式的对抗防御时，主要考虑是对模型体系结构进行最小的修改。保持模型速度: 运行时间对深度学习模型的可用性非常重要。在测试过程中不应该受到影响。随着防御系统的部署，深度学习模型仍应在大型数据集上保持高性能。保持准确性: 防御对模型分类精度的影响应尽可能小。防御应该有针对性: 防御应该在对抗样本相对接近训练集的地方工作。因为远离数据集的例子相对安全，分类器很容易检测到这些例子的扰动。

在这样的背景下，我们需要为模型准备防御手段，这样模型在面对对抗样本时，仍能够保持较高的准确率。本文的目标是建立一个对抗样本检测和防御服务，从模型和数据集的准备，到对抗样本的检测和对抗防御的实现。这样测试人员才能够在系统中完成一个完整的深度学习鲁棒性 [4] 优化流程，评估对抗样本对模型带来的威胁，了解对抗防御方法起到的作用。对抗防御方法并不

能适用于所有的攻击方法，不同的防御方法有其独特的优点，系统通过对抗样本检测功能帮助用户选择更合适的防御方法。

1.2 国内外研究现状及分析

在人工智能时代，深度神经网络在图像/语音识别 [5]、自然语言处理 (NLP)[6]、自动驾驶 [7]、机器人 [8]、网络安全 [9] 等领域都提供了巨大的支持。在人工智能发展的初期，人们更加关注其基础理论和应用研究。随着人工智能的快速发展，安全问题引起了人们的广泛关注。例如，世界上第一个机器人受伤事件，机器人查比突然失控撞到展台玻璃，导致行人受伤。不久之后，由硅谷机器人公司制造的杀人机器人 Knightscope[10] 在硅谷购物中心撞倒并打伤了一名 16 个月大的男孩。再有特斯拉自动驾驶汽车事故的频繁发生，推动着深度学习安全性测试的前进。

在人工智能安全领域，对抗样本 (AE) 已经成为人工智能系统的一种新的攻击手段。对抗样本最早在图像识别领域的应用是于 2013 年由 Szegedy[11] 等人提出的。文中提到，神经网络是很脆弱的，提供一个细微的、难以察觉的扰动，就可以使得神经网络产生错误的分类，指鹿为马，或者是无中生有。即使不同的模型有不同的架构和训练数据，同一套对抗样本也可以用来攻击所有相关的模型。

出于对抗样本的反直觉特点，研究人员被其吸引而开始投入到对抗样本的检测与防御之中。对抗样本的成因有不同的解释。首先是模型的过度拟合或正则化不足，使得深度学习模型在面对对未知数据时，缺乏泛化能力。然而，Goodfellow 等人 [12] 发现通过在正则化模型中添加扰动对对抗样本的有效性并没有显著提高。其他研究人员 [13] 怀疑对抗样本是由深度神经网络的极端非线性引起的。对抗样本的研究，为难以解释的深度学习模型鲁棒性问题提供了研究方向。

对抗样本防御是深度学习安全性的迫切要求，其研究如今主要沿着三个方向发展: 在深度学习模型的学习过程中使用扰动过的训练集，或者在测试的过程中使用扰动过的输入。修改网络，对深度学习模型添加更多的层或者子网络，以及修改损失，激活函数等。当对未见过的示例进行分类时，使用外部模型作为网络插件。

在实际操作中，对抗样本的防御手段并不能有效覆盖所有的攻击手段，不

同的攻防组合可能会带来不同的效果，防御手段是有局限性的。例如，防御蒸馏 [14] 能够在不改变神经网络结构的前提下大大降低对扰动的敏感性。因此，防御蒸馏在训练和测试中所需要的开销很低。然而，防御蒸馏要求添加蒸馏温度和修改目标函数，使得防御模型的设计变得很复杂性。此外，攻击方法可以通过以下三种策略来降低防御蒸馏的影响力:1) 选择更合适的目标函数;2) 计算最后一层梯度，而不是倒数第二层梯度;3) 攻击较为脆弱的模型，然后迁移至蒸馏模型。

为更好地进行对抗样本测试，Rauber 等人提出了 Foolbox[15]，以 Python 库的形式为测试人员提供对抗样本攻击方法的参考实现，支持主流的深度学习框架。Adversarial Robustness 360 Toolbox（ART）[16] 为开发人员和研究人员提供防御机器学习模型，以抵御对抗性威胁，并有助于使 AI 系统更加安全和值得信赖。此外，百度安全实验室也推出了 Advbox[17]，以百度 PaddlePaddle 平台为基础，提供高效构建对抗样本数据集的方法，为测试人员在其平台上进行模型安全的研究提供支持。

1.3 主要研究内容

随着人工智能的快速发展，其应用已经遍布各个领域，对于人工智能安全性的研究已是大势所趋。人工智能在图像方面的应用是重要的一部分，人工智能可以为图像识别提供高效的支持，这也使得其安全性的考量成为关键。对抗样本的出现最早就是在图像领域，它揭露了深度学习模型的脆弱性，基于对抗样本来测试深度学习的鲁棒性是可行的。用对抗样本防御方法来优化深度学习模型，能够提供可观有效的辅助。

本文的主要内容是整合如今主流的对抗样本攻击方法和防御手段，对比分析各方法的有效性，帮助测试人员评估基于对抗样本的深度学习鲁棒性优化技术的可靠性和合理性，并对深度学习模型的改进和数据集的优化有一定的指导性意义。具体来讲，本文的设计在集成现有的对抗攻击和防御方法的基础上，增加对抗样本检测功能，目的是检测数据集中是否存在对抗样本以及是由何种攻击方法产生，应对不同的攻击方法针对性地采用更为合理有效的防御手段，以期更高的神经网络鲁棒性。本项目是深度学习安全性研究的一个尝试，帮助测试人员实际接触对抗样本，对了解对抗样本的特性与效果有重要的意义。

1.4 本文的工作

本文的研究始于图像识别的广泛应用下产生的安全性问题，旨在帮助测试人员更好地了解对抗样本，并实际应用对抗样本的攻防方法来体验对抗样本对深度学习模型的影响。目前，对抗样本是深度学习安全性领域的重要研究方向，对深度学习模型的数据集生成对抗样本能够轻易地使得模型做出错误的预测，因此需要针对对抗样本研究其成因和防御方法。此外，对抗样本攻击方法有许多种类，单一的攻防效果具有片面性，需要通过多种组合来分析防御方法的性能。

基于上述的分析，本文的工作重心在于建立一个自动化检测对抗样本类别并进行针对性防御的优化体系。对于本文来讲，分别从对抗样本攻击导致模型预测出错和对抗防御修正模型预测成功率两个方面来建立实现方案：通过选用现有对抗攻击方法，在选定的模型和数据集上使用并获得对应的对抗样本，以模型对于对抗样本的预测成功率来展现其效果；在对抗样本检测功能的支持下，推荐最为有效的对抗防御方法，实现针对性的防御，评估其效果。引入结构相似性 [18] 和扰动敏感距离 [19]，评估系统处理后的样本数据集质量，保证图像不会出现失真情况。基于效果评估工作来设计系统功能，展开描述系统每个部分的设计与实现，从而构成一个可运行的完整的系统。另外，通过测试来检验系统各功能的可用性，保证用户的使用体验。

1.5 本文的组织结构

图像识别是人工智能的重要应用，然而，安全性问题给这一应用带来了难题，深度学习的测试显得十分重要。在实际的测试中，深度学习模型由于其难以解释的特点，测试方法与传统的软件工程测试相差很大。本文以对抗样本为主要研究方向来开展深度学习测试，从对抗样本攻击和对抗样本防御两个方面，并结合使用以进行深度学习模型鲁棒性的评估，体现攻防两方面的方法对深度学习模型带来的影响，以此衍生成可用的系统。本文的组织结构如下：

第一章引言，阐述了基于对抗样本的深度学习优化技术的项目背景和意义，国内外的相关研究现状，本文的主要研究内容和主要工作。

第二章技术综述，主要介绍本文涉及的技术和框架，包括项目采用的 Vue，Django 框架和主要研究的对抗样本技术。

第三章系统需求分析与概要设计，首先提出本项目的基本需求，围绕需求展开并描述项目整体的设计思路。从整体到细节，将设计以模块形式划分，并从不同的角度来阐述设计的思路。

第四章系统详细设计与实现，以需求分析为基础，介绍项目核心模块的详细设计以及实现方法。

第五章系统测试与分析，将根据项目需求对系统的功能从功能性和非功能性两个方面进行测试，以保证系统的可用性和实用性。

第六章总结与展望，总结描述本项目，提出项目开发中遇到的问题，存在的不足，描述系统未来的改进方向。

第二章　技术综述

2.1　Django 框架

Django[20] 是一个用 Python 开发的开源 Web 框架，内置开发服务器和调试器，使得构建高性能 Web 系统变得简单快捷。它具有强大的后台功能，可以通过简单的代码和命令轻松管理系统内容。Django 基于 MVC[21] 架构，但与 MVC 不同的是，Django 更注重模型、模板和视图之间的关系，即 MTV 模式。模型（Model）指数据存取层，主要负责处理数据的存取，数据的有效性检验等；模板（Template）指展示层，其职责是将页面上的内容显示出来；与 MVC 不同，视图（View）不再与 HTML 相关，而是专注于业务逻辑，并充当前两者之间的桥梁。

2.2　Vue 技术

Vue.js[22] 是一款渐进式的 JavaScript[23] 框架，渐进式代表着由浅入深地，由简单到复杂的方式去使用 Vue.js。Vue.js 的优势是体积小，且运行效率更高。它基于虚拟 DOM[24] 技术，该技术通过 Javascript 进行预处理操作，提前计算出最终的 DOM 操作并进行优化，并不进行真实的 DOM 操作，所以为成为虚拟 DOM。Vue.js 实现了双向的数据绑定，开发者无需对 DOM 对象进行操作，只需要把重心放在业务逻辑的开发中。如今的市场上拥有大量成熟、稳定的基于 Vue.js 的框架以及组件，生态丰富，易于实现快速开发。

2.3　对抗样本

对抗样本在图像分类、语音识别、恶意软件检测和图像字幕等领域显示出强大的攻击能力。例如，停车交通标志图像经过对抗样本处理后，分类器可能误将图像分类为其它标志，从而导致严重的交通事故 [25]。随着语音识别在移动设备和嵌入式设备中的广泛应用，许多业务和数据都是通过语音识别系统转

录的。这种趋势允许攻击者在用户不知情的情况下使用对抗样本控制设备，从而导致不可预测的风险。例如，玩隐藏语音命令可能会导致智能手机访问恶意网页，下载恶意软件 [26]。在恶意软件分类中，对抗样本限制了它们潜在的应用程序设置。攻击者可以稍微修改恶意软件的某些属性，保留恶意属性，但恶意软件检测系统 [27] 仍将其归类为良性。在图像字幕系统中，以一幅图像作为输入，生成描述被攻击者干扰的图像的一些字幕，从而生成一些与图像无关、完全相反甚至是恶意的字幕 [28]。

对抗样本的成因如今还没有一个公认的答案，许多研究人员都对其成因进行了猜想与实验，而对对抗样本成因的研究也能促进深度学习模型脆弱点的解决。在 Goodfellow[13] 等人的研究中，对对抗样本的成因进行了以下的探讨与实验：一些学者认为是模型的过度拟合或正则化不足，深度学习模型尽管对极其庞大的数据集进行训练，但仍然会存在未被考虑的数据，而对抗样本就是其中比较特殊的数据。但将对抗样本加入训练集进行训练后，却没有使得深度学习模型对对抗样本的防御能力获得显著提升。还有的学者认为成因是深度学习模型的极端非线性，但在高维的线性模型中，也找到了对抗样本。因此 Goodfellow 等人提出了这样的结论，过高的输入特征维度以及模型的线性性质导致了对抗样本的产生。在高维的空间，每个像素只需要一点微小的扰动，经过线性模型的参数点乘就为累积成为一个明显的变动，比如在图像识别领域，每一个图像都有着极高的维度，微小的扰动在高维度的图像下就跨越了决策层，产生了错误的分类，输出了错误的结果。

2.3.1 对抗样本攻击

1）L-BFGS

Szegedy[11] 等人提出了 L-BFGS 来构建对抗样本。该方法对给定的一个图像 x，采用 L2 范数构造一个类似于 x 的图像 x’, x’在模型的预测中将被标记为与原样本不同的类。其优化问题为:

$$minimize \quad \|x - x'\|_2^2 \tag{2-1}$$

其中 $\|x - x'\|_2^2$ 是 L2 范数。这个攻击的目标是使得 $f(x') = l, x' \in [0,1]^n$, 其中的 l 是目标分类，$f(x') = l$ 是难以直接求解的非线性非凸函数。因此，这个问题

可以用盒约束的 L-BFGS 近似求解。

$$minimize\ \ c \cdot ||x - x'||_2^2 + loss_{F,l}(x') \tag{2-2}$$

其中 c 是一个随机初始化的超参数，由行搜索决定；$loss_{F,l}$ 为损失函数。$f(x') = l$ 通过最小化损失函数来近似。该方法虽然稳定有效，但计算过程较为复杂。

2)FGSM

Goodfellow[13] 等人提出了快速梯度符号法 (FGSM)，该方法能够简单快速地构建对抗样本。通过添加扰动并在梯度方向上线性化代价函数，对生成的图像进行错误分类。给定原始图像 X，可以用:

$$X_{adv} = X + \varepsilon sign(\nabla_x J(x, y_{true})) \tag{2-3}$$

其中 X_{adv} 表示来自 X 的对抗样本，ε 是一个随机初始化的超参数，$sign(*)$ 为一个 sign 函数，y_{true} 是 X 所对应的真实标签，$J(*)$ 是用于训练神经网络的代价函数，∇_x 表示 X 的梯度。

FGSM 和 LBFGS 之间有两个主要的区别。首先，利用 $L\infty$ 范数对 FGSM 进行优化。其次，FGSM 是一种快速构建对抗样本的方法，因为它不需要迭代过程来进行计算，因此比其他方法具有更低的计算成本。但是，FGSM 容易产生标签泄漏效应。

3)JSMA

Papernot[29] 等人提出了基于雅可比矩阵的显著映射攻击 (JSMA)，该攻击基于 L_0 距离范数。其基本思想是用梯度构造显著图，然后根据每个像素的影响对梯度进行建模。梯度与图像被分类为目标类的概率成正比，即改变较大的值会显著增加模型将图像分类为目标类的可能性。JSMA 允许我们根据显著性映射选择最重要的像素 (最大梯度)，然后对像素进行扰动，以增加将图像标记为目标类的可能性。更具体地说，JSMA 包括以下步骤:

a) 向前计算导数 $\nabla F(X)$

$$\nabla F(X) = \frac{dF(X)}{dD} = [\frac{dF_j(X)}{dX_i}]_{i\in 1...M, j\in 1...N} \tag{2-4}$$

b) 构造一个基于正向导数的显著映射。

c) 根据显著性图修改最重要的像素，重复此过程，直到输出为目标类或得

到最大扰动。

在模型对输入变化非常敏感的情况下，与其他攻击方法相比，该方法计算的最小扰动值更容易被误分类，并且使扰动实例难以被人眼感知。虽然 JSMA 的效率较低，但它更隐蔽，成功率和转移率也相对较高。

4)DeepFool

Mohsen[30] 等人提出了一种基于 L_2 距离范数的非目标攻击方法，称为 DeepFool。假设神经网络是完全线性的，在不同的分类之间将会存在超平面将各分类分隔开。文章在超平民假设的基础上，分析了该问题的最优解并构造了对抗样本。相应的优化问题如下:

$$r_*(x_0) = argmin \|r\|_2 \tag{2-5}$$

服从于 $sign(f(x_0 + r)) \neq sign(f(x_0)))$，其中 r 表示扰动。

与 L-BFGS 相比，DeepFool 更加高效和强大。其基本思想是找到图像空间中最接近 x 的决策边界，然后利用该边界来误导分类器作出错误的分类。在高维非线性空间的神经网络中，直接解决这一问题是非常困难的。因此，采用线性化逼近法迭代求解该问题。近似方法是在每次迭代中将中间的 x_0 分类器线性化，并在线性化后的模型上获得最优更新方向。然后，x_0 在这个方向上迭代地更新一小步，重复线性更新过程，直到 x_0 跨越决策边界。最后，整个过程仅会产生微小的扰动，却能成功构造对抗样本。

5)Carlini Attack(C&W)

Carlini 和 Wagner[31] 提出了一种基于 L-BFGS 的强大攻击方法。具有 L_0,L_2,L_∞ 距离范数的攻击可以是有针对性的，也可以是非针对性的，这里以非针对性的 L_2 范数为例。对应的优化问题为:

$$minimize_\delta \|\delta\|_2 + c \cdot f(x + \delta) \tag{2-6}$$

其中 $x + \delta \in [0, 1]^n$，c 是一个可以平衡这两项的超参数，δ 是一个小扰动。目标函数 f(x) 的定义如下:

$$f(x) = max(max\{Z(x')_i : i \neq t\} - Z(x')_t, -l) \tag{2-7}$$

其中 $Z(x')$ 是最后的隐藏层，t 是目标标签，l 是超参数，用来控制模型误分类的置信度。通过调整 l 的值，对抗样本 x' 将被分类为具有较高置信度。一般

来说，高可信攻击具有较大的扰动和高的传递速率，基于 L_0,L_2,L_∞ 距离度量的 Carlini 攻击可以成功地击败防御性蒸馏。基于 L-BFGS 的攻击有三个改进: a) 在模型中使用实际输出的梯度，而不是 softmax 的梯度。b) 应用不同的距离度量 (L_0,L_2,L_∞)。c) 应用不同的目标函数 $f(x)$。

2.3.2 对抗样本防御

1）特征压缩 Feature Squeezing

特征压缩 [32] 的目的是压缩特征之间的输入空间，减少可利用的攻击空间。一般在正常的数据中，特征输入空间较大，使得对抗样本攻击方法可以在很大的输入空间内进行样本构建。该方法的算法提取并合并原始空间中不同特征向量所对应的样本，成为单个样本，从而压缩输入空间，使得在空间内构建对抗样本变得困难。其核心思想是将模型对原始样品的预测与挤压后样品的预测进行比较，如图 2-1所示。

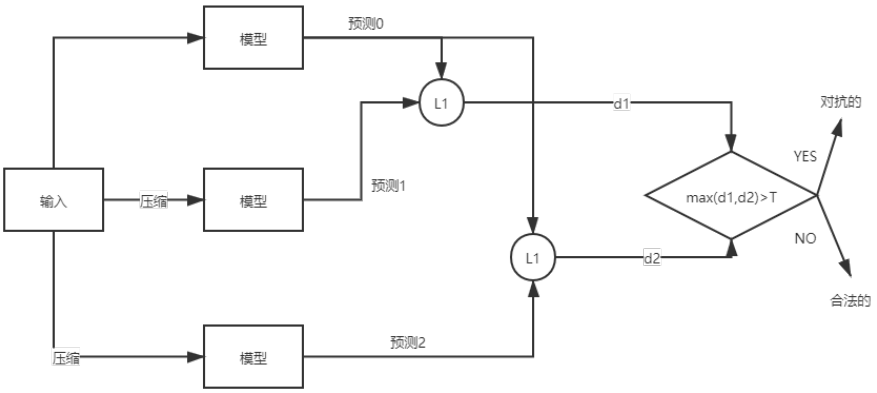

图 2-1: 特征压缩框架图

如果原始输入和压缩输入产生的输出与模型有很大不同，那么输入很可能是敌对的。通过比较预测与选择的阈值之间的差异，该方法为合法的例子输出正确的预测并拒绝敌对的输入。本方法的优点是需求的计算资源低，可以与其他防御方法相兼容。

2) 由高级特征主导的去噪器 HGD

传统的去噪器 PGD 针对对抗样本的像素点进行去噪，从而获得去噪样本。该方法的损失函数基于像素点之间的距离，一般计算原始样本和去噪样本之间的 L1 距离。PGD 的一个潜在问题是对抗性噪声的放大效应。对抗样本与干净

的图片有微不足道的差异。然而，这种微小的扰动会被深层神经网络逐渐放大，从而产生错误的预测。即使去噪器可以显著地抑制像素级噪声，剩余的噪声仍然可能扭曲目标模型的高阶响应。

为了克服这个问题，HGD[33] 从高层次的视角定义损失函数，将像素级损失函数替换为干净图像和去噪图像在目标模型上产生的输出的差值。具体来说，给定一个目标神经网络，我们提取其在第 l 层由 x 和 $\widehat{x}$ 激活的表示，并将损失函数定义为其差值的 L_1 范数:

$$L = \left\| f_l(\widehat{x}) - f_l(x) \right\| \tag{2-8}$$

HGD 具有以下优势：(a)HGD 能够使目标模型在面对对抗攻击时更具有鲁棒性 (b)HGD 具有较好的泛化性，并对其他图像和未知分类表现良好 (c)HGD 效率更高，在小批量图像上进行训练，且需要的训练时间短

3) 像素防御 Pixel Defend

Song[34] 等人提出，对抗样本主要存在于训练分布的低概率区域，无论攻击类型和目标模型。使用统计假设检验，发现现代神经密度模型在检测难以察觉的图像扰动方面非常出色。基于这一发现，设计了 Pixel Defend，这是一种新的方法，通过将恶意扰动的图像移回训练数据中的分布来净化它。然后，纯净的图像通过一个未修改的分类器运行，使得该方法对分类器和攻击方法都是不可知的。因此，Pixel Defend 可以用来保护已经部署的模型，并与其他特定于模型的防御相结合。

4) JPEG 压缩

JPEG 是一种广泛使用的编码技术，许多图像已经存储在 JPEG 格式中。大多数操作系统还内置了对 JPEG 图像编码和解码的支持，JPEG 压缩方案 [35] 分为以下几个阶段:

- 变换图像到一个最佳的颜色空间。通过将一组像素平均在一起来降低色度分量的采样。
- 为消除冗余的图像数据，对像素块应用离散余弦变换 (DCT)。
- 使用加权函数量化 DCT 系数的每个块，使人眼难以察觉。
- 使用 Huffman 可变字长算法对产生的系数 (图像数据) 进行编码，以消除系数中的冗余。

即使非专业用户也可以很容易地应用这个预处理步骤。该方法不需要关于模型和攻击的知识，可以应用于广泛的图像数据集。

2.4 结构相似性 SSIM

结构相似性（SSIM）[18] 是评价图片质量的一个经典算法。当对一个图片作出缩放，压缩等操作后，图像可能会出现失真。为了评价这些操作后图片的质量，便引入了该评价指标。结构相似性分别从图像的亮度，对比度与结构三个方面来计算相似性。其中亮度通过计算各像素的均值得到，将图像像素数据与刚得到的亮度测量结合能够计算得到对比度测量，即通过计算标准差得到。接着应用亮度测量和对比度测量的数据得到结构对比，即协方差。最后综合三者得到相似度的测量

2.5 扰动敏感距离 PSD

扰动敏感距离（PSD）[19] 的提出是为了评估人类对扰动的感知，即是否能够察觉对抗样本。当人看一张图片时，我们更容易发现对比度低的一片区域内的扰动，而不是对比度很高的一片区域中的扰动，杂乱的颜色能够掩盖对样本作出的扰动，难以被人眼察觉。因此为了评估生成的对抗样本图片是否会被人眼察觉，引入扰动敏感距离。通过一个像素周围像素的对比度以及扰动大小来计算被人眼察觉的可能性大小。

2.6 本章小结

本章介绍系统进行 web 开发时使用到的前后端框架以及系统涉及的深度学习技术。前端框架采用 Vue.js，后端框架采用 Django。深度学习技术指对抗样本的攻防方法，作为深度学习测试的有力手段。结构相似性和扰动敏感距离为防御优化得到的数据集图像进行质量度量。本章的主要目标是为项目提供技术支持和理论依据。

第三章　对抗样本检测和防御系统的需求分析与概要设计

3.1　总体规划

本系统的设计是面向对对抗样本在神经网络之上的影响有测试需求的测试人员，从对抗样本攻击和防御两个角度来开展测试。如图3-1所示为对抗样本防御优化技术的流程图。系统在进行测试任务之外，提供了数据集预处理，模型搭建和样本识别操作这些辅助模块。对于对抗样本生成，系统提供了主流的对抗样本攻击方法。同样的，系统在防御优化方面也集成了大量防御手段。由于对抗防御的效果在面对不同的攻击方法时效果有差异，本系统旨在为不同攻击方法产生的对抗样本使用不同的防御手段。因此，提供了对抗样本检测模块，该功能将在对抗防御任务启动前识别数据集内存在的对抗样本种类，并分类使用防御手段。综上，本系统通过对抗样本的检测和防御两个方面来测试对抗样本对模型和数据集带来的影响。

本系统划分为数据流处理模块，对抗检测模块，对抗防御模块和任务报告模块四部分。数据流处理模块对用户上传的深度学习模型和数据集负责，为用户构建专属的深度学习资源库，用户可以对自己上传的模型和数据集做各种操作，也为对抗任务模块提供资源。对抗检测模块对对抗防御任务提供支持，主要的工作是训练模型检测对抗样本类型。对抗防御模块负责对抗样本生成任务和对抗防御优化任务的管理，包括新建，查看详情，删除，下载报告等操作，是本系统的核心模块。任务报告模块负责评估测试任务的有效性，并引入结构相似性 SSIM 和扰动敏感距离 PSD，对任务生成的样本质量进行评估。在多模块的协作下，给用户良好的使用体验。本章节从需求分析出发，从多个角度对系统的设计进行规划。

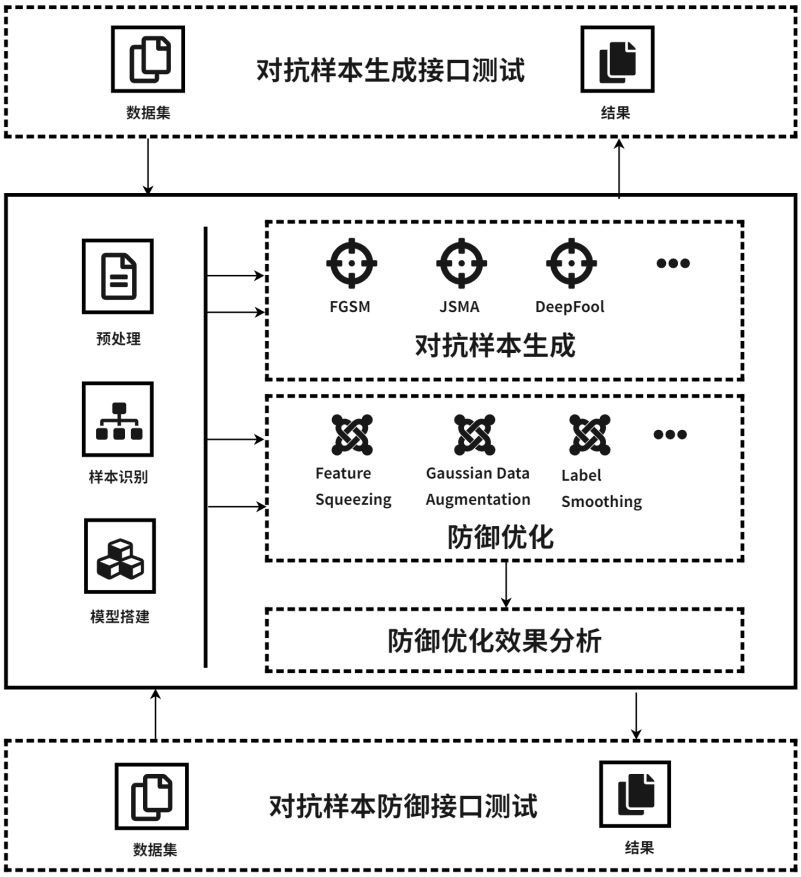

图 3-1: 总体设计图

3.2 需求分析

3.2.1 功能需求

基于对抗防御的深度学习鲁棒性优化技术，以对抗样本为核心，此技术依托于对抗样本的生成技术和防御方法，并围绕此实现一个系统。该系统为深度学习测试人员提供了面向数据集的鲁棒性优化功能，对包含对抗样本的数据集进行对抗样本检测，并根据对抗样本的类型使用合适的对抗防御方法。为此，系统首先提供了生成对抗样本的方法，对给定的数据集进行攻击，获取的对抗样本可用于后续的测试。在使用对抗防御方法进行数据集的优化时引入对抗样本检测，训练模型检测属于哪种对抗样本，并根据记录的对抗防御效果匹配合适的防御方法。系统功能成效体现在模型面对对抗样本时，由于使用了系统提供的数据集检测与防御预处理，能保持较高的准确率。

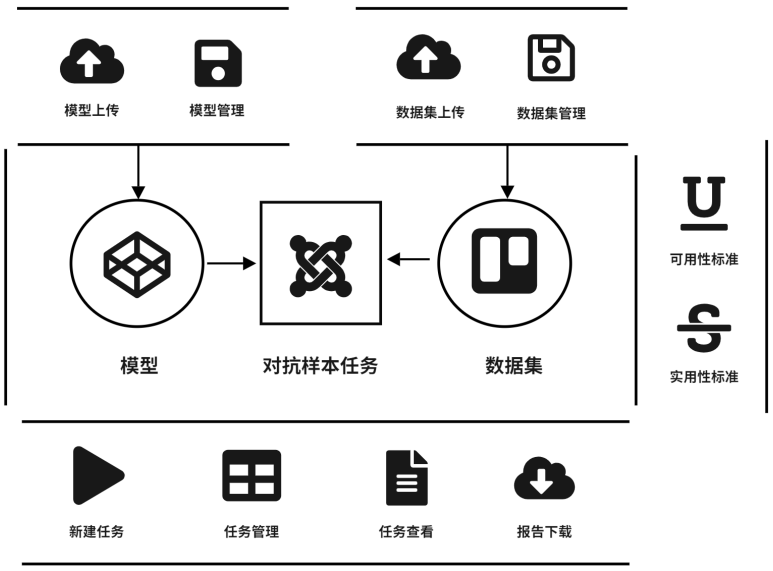

图 3-2: 流程示意图

如图3-2所示，对抗样本任务的实现与所面对的模型和原始样本，即数据集有关。用户在创建任务前，需要维护自己的模型库和数据集库，可以通过上传的形式将文件上传到系统服务器中。用户可以先通过对抗样本攻击任务生成对抗样本，用于后续对抗防御任务，也可以直接使用自己上传的带有对抗样本的数据集进行对抗防御任务。任务完成后用户可以查看报告，也可以下载报告和生成的结果文件，即新数据集。

系统的功能性需求列表整理如表3-1所示，包含模型上传，数据集上传，上传管理，对抗样本生成，对抗样本检测，基于对抗样本的防御优化，任务状态查看，任务历史管理和任务报告管理。

3.2.2 非功能需求

系统的非功能性需求如表3-2所示，主要从可靠性、时间特性、安全性、可用性、可扩展性和并发性这几个方面来制定，具体内容如表所示。系统的非功能性需求是为了保障软件的质量，加强用户体验以及提高开发效率。

表 3-1: 系统功能性需求列表

需求 ID	需求名称	需求描述
R1	模型上传	用户通过用户界面进行深度学习模型的搭建上传，完成后后台会对模型类型进行格式的统一并存储。系统支持基于 Tensorflow 和 Keras 框架的神经网络模型文件。
R2	数据集上传	用户在界面上传相关数据集用于对抗样本生成以及防御优化，系统会将其存储于数据库。
R3	上传管理	用户能够查看已经搭建上传的模型和数据集记录，且支持对已存在的模型和数据集的下载、修改以及删除操作。
R4	对抗样本生成	系统支持对上传的数据集生成对抗样本，用户可以选择不同的攻击方法来生成，生成的对抗样本也将存储在对应的位置。
R5	对抗样本识别	系统支持对上传的混有对抗样本的数据集进行对抗样本识别，系统将使用模型分类并识别数据是否是对抗样本，以及受到了何种攻击方法的扰动。
R6	基于对抗样本的防御优化	系统支持对上传的混有对抗样本的数据集进行防御优化，系统将根据对抗样本识别模块产生的结果采取不同的防御优化方法。用户可以自定义能够使用的对抗样本防御方法。
R7	任务状态查看	系统的任务将在后台自动运行，用户可以查看任务状态。
R8	任务历史管理	用户能查看已存在的任务的历史记录，并能够查看其结果详情。
R9	任务报告管理	任务结束后需要自动化生成任务报告并存储至数据库，用户能够查看和下载。系统将提供包括样本质量度量等的评估。

表 3-2: 系统非功能性需求

类型	非功能性需求描述
可靠性	系统功能相互独立，发生故障时不影响其他功能的使用 提供错误日志，方便排查错误
时间特性	用户执行页面的刷新或跳转操作时，响应时间不超过 2s
安全性	系统服务仅对系统注册用户开放，严格权限访问控制 密码通过加密保护，数据存储在数据库中
可扩展性	系统支持新增功能，且可复用已有功能
并发性	系统可支持 10 名用户同时访问某一功能

3.2.3 用例设计

本系统的主要用户是希望了解对抗样本在深度学习模型上的影响的测试人员，且能够利用本系统测试不同的对抗样本防御方法所能起到的效果，为用户上传的模型和数据集选择最佳的防御方法，使得模型在面对对抗样本时仍能够保持较高的预测正确率。根据上文的总体设计与功能需求分析，可以得到本系统的用例设计图如图3-3所示。本系统包含以下用例：模型上传、模型处理、数据集上传、数据集预处理、上传管理、下载、新建对抗样本生成任务、新建对抗样本防御任务、对抗样本检测、任务管理和查看报告。

在用户登录系统后，可以在模型上传界面上传本地的模型文件，并配置相关信息，系统将存储模型文件以备后续使用。模型文件需要符合一定的标准，系统将检测文件的格式保证模型文件符合要求。同样的，数据集也需要通过上传存储。用户能够对已上传的模型和数据集进行管理，包括按条件筛选列表，查看详情，删除记录以及下载操作。系统用户可以在任务创建界面新建任务，包含对抗样本攻击和对抗防御优化。在创建任务时，需要选择数据集和模型，此时可以在已上传的数据集和模型中进行选择。任务在创建完成后将在后台运行，此时的任务处于运行中，当任务完成后，可以查看任务结果详情。对于已经完成的任务历史，用户可以在任务管理界面进行管理，同样的，包含按条件筛选列表，查看详情，删除记录以及下载报告操作。

表3-3描述了模型搭建用例的详情。系统用户首先需要进入到模型搭建界面，在该界面中，用户可以通过上传按钮上传本地的模型文件。额外的，用户需要填写模型的其他相关信息，包括名称，类型等，以便后续调用。当用户完成模型搭建后，后台将解析并存储模型，已存储的模型可以在上传管理中查看

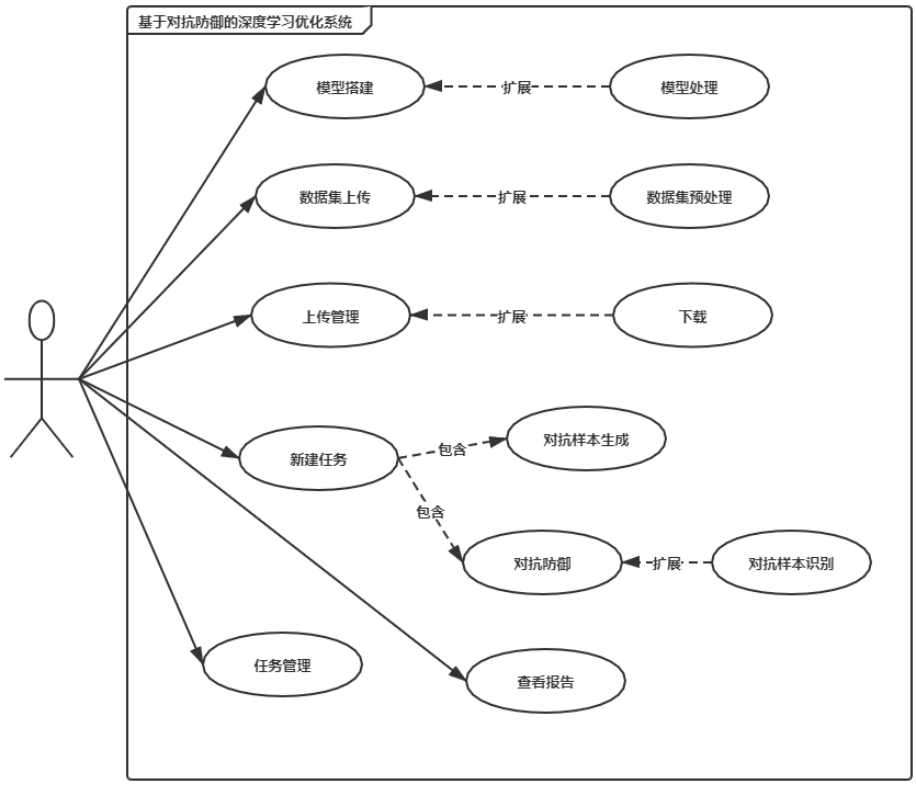

图 3-3: 用例图

与管理。上传的模型文件名称需要唯一，这是为了用户创建任务时方便辨认不同的模型，如果系统已经存在同名的模型，将提示该文件名称重复，并取消提交，避免文件出现冲突。同名文件的检测和模型文件格式的检测都将使用正则表达式进行匹配，如果不符合则需要发出提示。对于深度学习模型的格式，以 Tensorflow 和 Keras 为例，前者一般保存为.ckpt 格式，而后者一般保存为.h5 格式，系统将支持一些主流的文件格式。

表 3-4描述了数据集上传用例的详情。当系统用户进入到数据集上传界面，在该界面中，用户可以将本地的数据集文件上传到服务器中。在上传之前，用户需要填写数据集的其他相关信息，包括名称，训练集、测试集的占比等，这些数据集将在创建任务时被使用。当用户完成数据集上传后，数据集将被存储与数据库中，已存储的数据集可以在上传管理中查看与管理。当用户上传的数据集文件格式不符合要求时，系统将发出错误警告。若系统已经存储了相同名称的数据集，将提示已经存在同名文件，取消上传，这是为了使用户在后续数据集的调用时不产生混淆。文件格式的检查和同名文件的检查都将使用正则表达式。

表 3-3: 模型上传用例描述

ID	UC01
名称	模型上传
描述	用户执行深度学习模型上传操作
参与者	系统用户
触发条件	系统跳转到深度学习模型上传界面
前置条件	用户需要上传新的深度学习模型，并存储记录
后置条件	系统反馈深度学习模型上传的结果
优先级	高
正常流程	1. 用户进入到深度学习模型上传界面 2. 用户点击或拖拽上传模型文件 3. 用户填写模型的其他信息，如名称，类型等 4. 用户点击完成按钮，提交上传深度学习模型
扩展流程	2a. 用户选择上传已有模型 2b. 用户上传模型格式不符合
特殊需求	若系统已存储同名的模型信息，提示已存在并取消提交，避免冲突

表 3-4: 数据集上传用例描述

ID	UC02
名称	数据集上传
描述	用户进行数据集的上传操作
参与者	系统用户
触发条件	系统跳转到上传数据集界面
前置条件	用户需要上传数据集，为后续任务提供数据集支持
后置条件	系统反馈上传数据集结果
优先级	高
正常流程	1. 用户进入上传数据集的用户界面 2. 用户选择或拖拽本地数据集文件 3. 用户填写数据集的其他信息，如名称，占比等 4. 用户点击完成按钮，提交并上传数据集文件
扩展流程	2a. 数据集上传失败，系统发出错误提示 2b. 数据集文件格式不符
特殊需求	若系统已存储同名的数据集信息，提示已存在并选中 已存在的数据集，避免重复上传

表3-5描述了上传管理用例的详情。用户在登录系统后可以管理自己已上传的模型文件和数据集文件。这些信息首先需要在上传管理界面以列表形式展示，用户可以通过条件筛选来找到自己所需的某条历史记录。提供上传时间和类型的筛选。对于这些记录，用户可以查看详情，也可以进行删除和下载操作。上传管理功能是为了给用户维护一个测试任务的资源库，方便用户创建任务时选用，而不用每次创建任务时都进行上传操作，增加系统可用性，增强用户使用体验。

表 3-5: 上传管理用例描述

ID	UC03
名称	上传管理
描述	用户对已上传的文件进行管理
参与者	系统用户
触发条件	用户进入到模型或数据集管理界面
前置条件	用户需要对已上传的模型和数据集进行管理，如查看，删除等操作
后置条件	系统反馈管理操作所产生的结果
优先级	高
正常流程	1. 用户进入到模型或数据集的管理界面 2. 用户对已上传的模型或数据集进行筛选，获取对应列表 3. 用户选择查看某条记录，系统显示该记录的详情 4. 用户选择删除某条记录，系统提示删除成功 5. 用户选择下载某条记录，系统建立下载任务
扩展流程	3a. 记录信息丢失，系统发出错误提示 4a. 删除失败，系统发出错误提示 5a. 下载出错，系统发出错误提示

表3-6介绍了新建对抗样本生成任务这一用例。用户可以通过本系统来自己生成对抗样本。在新建任务之初，用户需要先确定模型和数据集，依此来进行对抗样本攻击，模型和数据集可以调用已上传的文件。同样的，用户还需要规定使用的对抗样本攻击方法，系统将提供主流的攻击方法，支持多选，另外也需要配置任务的相关信息，比如生成的数量。在任务新建完成后，任务将在后台运行。用户可以在任务历史管理中查看已建立的任务。对于已经完成的任务，用户可以查看结果详情。新建的任务由于运行时间较长，且可能有同时进行多个任务的需求，为此使用线程池，在创建任务的时候系统将在线程池中创

表 3-6: 新建对抗样本生成任务用例描述

ID	UC04
名称	新建对抗样本生成任务
描述	用户新建对抗样本生成任务
参与者	系统用户
触发条件	用户进入到对抗样本生成界面
前置条件	用户需要对已有的数据集生成对抗样本
后置条件	系统提示任务创建成功，之后跳转结果展示界面
优先级	高
正常流程	1. 用户进入对抗样本生成界面 2. 用户从已有模型列表中选择模型 3. 用户上传新数据集或从已有数据集列表选择数据集 4. 用户配置任务的具体信息，如选用的对抗样本攻击方法 5. 用户点击确认，提交对抗样本生成任务 6. 系统后台执行任务并跳转任务历史界面
扩展流程	5a. 用户点击取消按钮，系统返回首页

建一个新线程来异步执行任务。用户在任务执行期间可以进行其他任意操作，而任务完成时线程将自动结束并更新数据库，包括更新任务状态和写入任务结果。

表3-7介绍了新建对抗防御任务这一用例。用户可以通过本系统来对包含对抗样本的数据集进行防御优化。同样的，在新建任务之初，用户需要先确定模型和数据集，依此来进行对抗防御，模型和数据集可以调用已上传的文件。用户还需要确定使用哪些防御手段。防御手段的选用关乎防御优化的效果，本系统也将为此实验总结各种攻防手段的适用效果，并通过对抗样本检测模块来进行针对性优化。对抗样本检测功能将训练一个多分类模型，识别数据集主要由何种对抗样本组成，并基于实验数据推荐效果最佳的对抗防御方法。在任务新建完成后，任务将在后台运行。用户可以在任务历史管理中查看已建立的任务。对于已经完成的任务，用户可以查看结果详情。与对抗攻击任务类似，新建的任务由于运行时间较长，且可能有同时进行多个任务的需求，使用线程池创建异步任务线程，任务完成时线程将自动结束并更新数据库。

表3-8描述了任务管理用例的详情。用户在登录系统后可以管理自己已建立的对抗样本生成任务和对抗防御优化任务。任务历史可以在任务管理界面查

表 3-7: 新建对抗防御用例描述

ID	UC05
名称	新建对抗防御任务
描述	用户新建对抗防御任务
参与者	系统用户
触发条件	用户进入到对抗防御界面
前置条件	用户需要对数据集进行对抗防御
后置条件	系统提示任务创建成功，之后跳转结果展示界面
优先级	高
正常流程	1. 用户进入对抗防御界面 2. 用户从已有模型列表中选择模型 3. 用户上传新数据集或从已有数据集列表选择数据集 4. 用户配置任务的具体信息，如选用的对抗防御方法 5. 用户点击确认，提交对抗防御任务 6. 系统后台执行任务并跳转任务历史界面
扩展流程	5a. 用户点击取消按钮，系统返回首页

看，用户可以通过搜索或条件筛选来找到自己所需的某条历史记录。对于这些任务记录，用户可以查看详情，也可以进行删除和下载操作。由于任务是异步运行的，在任务管理界面需要展示包括正在运行的任务，无法进行查看详情和下载操作。任务的删除采用任务记录的逻辑删除，避免物理删除，以便数据恢复。

用户进入任务历史界面后可以通过查看详情按钮查看已完成任务的结果分析。对抗样本生成任务需要展示用户所选的攻击方法所产生的对抗样本，并通过样本示例显示对抗攻击的视觉效果，通过成功率显示攻击效果。而对抗防御优化任务需着重于展示防御手段对对抗样本的针对性优化，通过对比应用前后的预测成功率直观体现其效果。为了增加防御效果的丰富性，引入结构相似性 SSIM 和扰动敏感距离 PSD，从数据集角度评估样本质量。查看任务报告用例如表 3-9所示。

用户进入任务报告详情页面后，可以对任务详情进行下载操作，将对应的任务结果保存至本地，以供离线查看。除此以外，对于对抗样本生成任务，用户可以下载生成的对抗样本数据集；对于对抗防御任务，用户可以下载使用对抗防御方法优化后的数据集和任务运行中产生的对抗样本检测模型。

表 3-8: 任务管理用例描述

ID	UC06
名称	任务管理
描述	用户进行任务管理操作
参与者	系统用户
触发条件	用户进入到任务历史记录界面
前置条件	用户需要查看任务历史列表
后置条件	系统显示已有的任务历史列表
优先级	高
正常流程	1. 用户进入任务历史记录界面 2. 用户筛选已存在的任务历史，获得对应的任务列表 3. 用户选择查看某条任务记录 4. 系统进入任务详情界面，显示该任务的详情 5. 用户选择某条任务记录，进行删除操作 6. 系统反馈删除操作成功
扩展流程	4a. 记录信息丢失，系统发出错误提示 6a. 删除失败，系统发出错误提示

表 3-9: 查看任务报告用例描述

ID	UC07
名称	查看任务报告
描述	用户查看任务报告
参与者	系统用户
触发条件	用户进入到任务历史界面
前置条件	用户需要查看任务结果报告
后置条件	系统跳转结果展示界面，显示详情
优先级	高
正常流程	1. 用户进入任务历史界面 2. 用户筛选已存在的任务历史，获得对应的任务列表 3. 用户选择查看某条任务记录 4. 系统进入任务详情界面，显示该任务的详情
扩展流程	4a. 记录信息丢失，系统发出错误提示 4b. 任务仍处于进行中，系统发出提示

3.3 概要设计

3.3.1 架构设计

系统采用前后端分离，实现高内聚低耦合，提高开发效率。系统的前端采用 Vue.js，Vue.js 提供了多样化的组件，为前端的开发提供了便利。同时通过 Vuex 来帮助管理共享状态，系统应用所包含的所有组件将被集中式存储并进行管理，组件的状态变化依照一定的规则，其变化将会是可预测的。系统的后端使用 Django 框架，为前端提供接口，其计算和统计功能，为机器学习和 AI 领域的系统开发提供了便利。后端的功能划分为多个模块，协同为系统提供稳定的功能。前后端之间的交互使用 Nginx 来进行负载均衡，为访问效率提供保障。Django 还提供了 ORM 功能，其作用是应用对类属性和方法的操作，替换对数据库的操作，避免各类数据库 SQL 语句的编写，使得系统数据库的搭建和操作变得简单。系统框架图图如 3-4所示。

系统的后端按功能分为数据流处理，对抗样本类型检测，对抗防御任务和测试任务报告生成，除此以外，还包含公共模块。数据流处理功能对深度学习模型和框架进行管理，当用户上传文件后可以对其进行查看，删除，下载等操作。对抗样本检测功能通过训练深度学习模型检测对抗样本类别，结合实验数据推荐最佳的对抗防御方法。对抗防御任务功能同时包含任务的创建和任务记录的管理，为系统提供核心的功能。该功能通过系统接口与任务资源管理，对抗检测和任务报告生成功能进行交互。通过数据流处理功能，可以调用用户已上传的模型和数据集，基于对抗检测，检测对抗样本类型并选择合适的防御方法，最后进行样本质量度量。对抗任务功能同样支持对已创建任务的查看，删除和下载报告操作。测试任务报告生成功能评估任务效果，计算原始样本与目标样本的结构相似性和扰动敏感距离，为任务有效性的评估添加数据集角度的标准。

后端的公共模块涉及异常处理，日志管理，常量和工具类等，这些功能贯穿于整个系统中，以公共模块的形式实现使得系统的可复用性提升。对于异常处理和一些工具类的实现，包括用户权限的检测，使用 Django 的中间件。中间件以插件的形式，影响系统的请求和响应处理过程，从而改变系统的输入和输出，实现十分轻量级。

系统的数据计算得益于 Python 在深度学习和数学分析方面的优势，作为

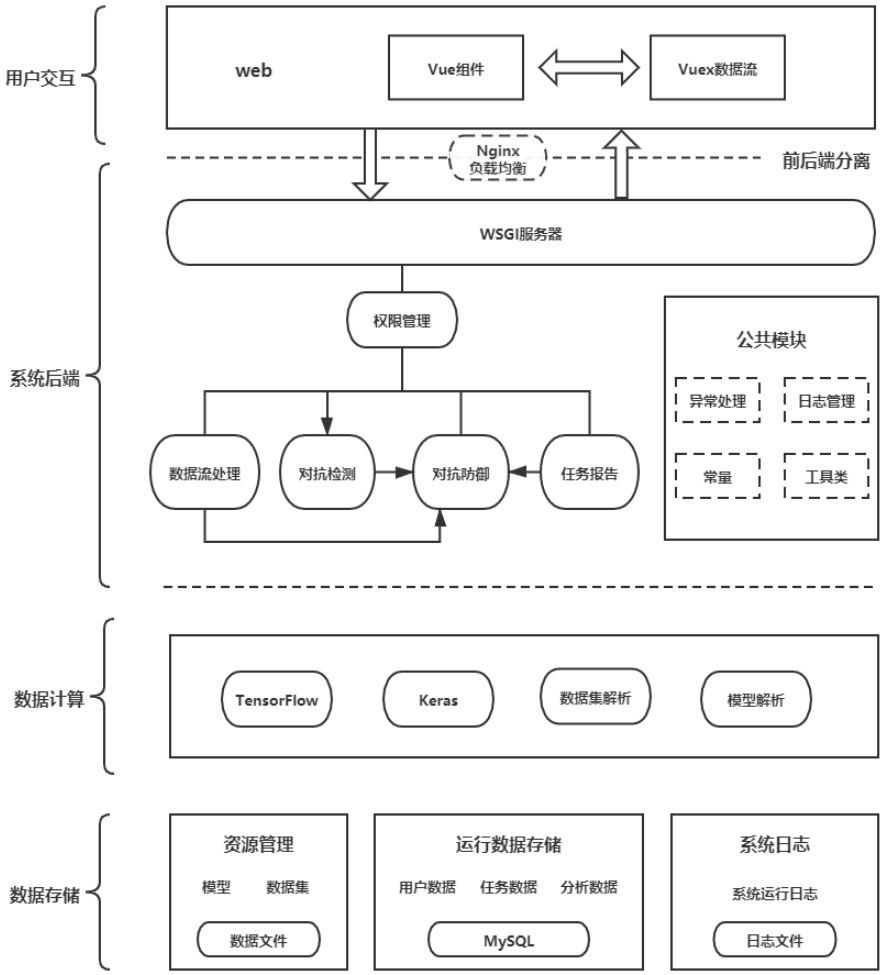

图 3-4: 系统框架图

一个深度学习方向的系统，不可缺少的是深度学习的框架，模型和数据集的解析。此外，系统还集成了主流的对抗样本攻防方法，为任务的创建提供基础。数据的存储按照不同的类型选择不同的存储方式，对于模型和数据集，上传的记录存储于数据库中，而本身的文件以文件形式存储，并在记录中添加路径标识。运行数据如用户数据，任务记录等统一存储于 MySQL 数据库，而系统的运行日志以日志文件的形式存储，方便维护和排查。

3.3.2 逻辑设计

系统的逻辑设计参考前文的用例设计和系统架构设计，本节进行具体描述。逻辑视图关注系统提供的功能以及功能之间的交互，不考虑系统的输出和部署，从抽象的角度来描述系统的结构。如图3-5所示为系统的逻辑视图，以

面向对象的思想将系统模块化，差分模块的功能，明确各功能的职责，以及确定系统内部各功能的协作关系。

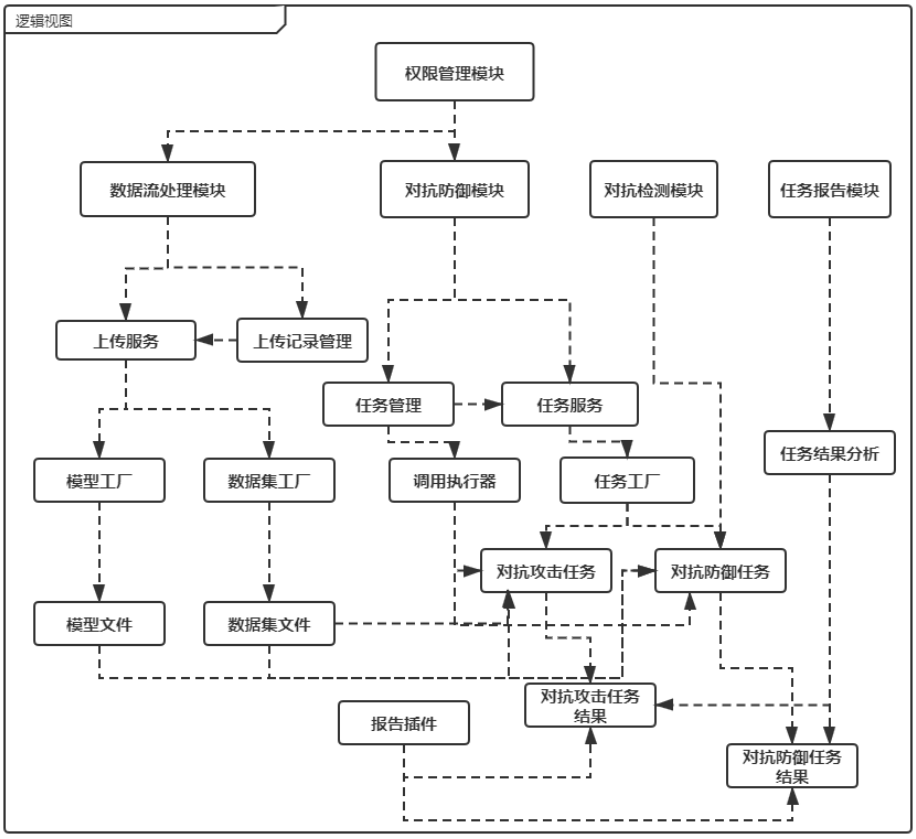

图 3-5: 系统逻辑视图

系统按照逻辑层面划分成数据流处理模块，对抗检测模块，对抗防御模块和任务报告模块。数据流处理模块负责系统文件的上传，其中的文件包含深度学习模型文件和数据集文件。在上传的过程中通过工厂产生对应的实例，并持久化地存入数据库，为任务模块提供资源支持。此外，还负责管理已上传的文件，支持基础的增删查等操作。对抗检测模块负责对抗防御任务中对抗样本类型的检测，通过深度学习模型训练实现，为防御方法的选择提供依据。对抗防御模块的主要功能是创建对抗防御任务，对应的任务在执行后将记录存储于数据库中，产生的结果分析也将一并存储。为方便用户获取包含对抗样本的数据集，提供对抗样本生成任务，用户可使用系统提供的攻击方法生成对抗样本。除此以外，该模块还包含对已完成任务的增删查看操作。任务报告模块为对抗任务结果进行评估，为模块生成的样本提供质量度量功能。

3.3.3 开发设计

本系统的前端采用 Vue.js，如3-6所示，主要由浏览器，JS 模块，webpack，页面容器，util，Service，Network，API 这几个部分组成。Vue.js 体积小，且拥有更高的运行效率，且相比传统的页面通过超链接实现页面的切换和跳转，Vue 使用路由不会刷新页面，适合作为一个轻量级测试平台前端框架。

- 浏览器是系统的入口，系统功能可视化的平台，UI 交互使得项目的可用性大大提升。
- 生成各页面的 JS 模块包含各页面的样式和函数，从而构成各个系统界面。
- webpack 提供前端资源加载和打包功能，首先静态分析模块之间的依赖关系，然后依照指定规则对各模块生成对应的静态资源，从而减少页面的请求次数，大大提高效率。
- 页面容器包含 Vue 组件，Vue 路由，Vuex 状态管理和 UI 组件库。组件是组成页面结构的基础，如菜单，导航栏等。路由管理 URL 实现不同组件之间的跳转和切换。
- util 包含各类工具函数，能够提供给各模块使用，减少冗余代码，以此来提高代码的可复用性。
- Service，Network 和 API 都是为了系统的前后端交互而服务。

系统的后端使用 Django 框架，Django 采用 MTV 的模式，其本质上与 MVC 相似，都是为了各组件之间保持松耦合关系，以期开发快捷，部署方便。Python 加 Django 的组合使得系统的开发与设计变得迅速，且方便部署。系统后端的结构图如图3-7所示。

- 由于是前后端分离，Django 后端的将系统功能封装成 API，将接口参数暴露给前端，供前端调用。这样做大大降低了项目的耦合度，前端只需要知道接口如何调用，而不需要理解是如何实现的。同时也提升了代码的复用性，无需为相似的功能编写多个方法。
- Request/Response 是 Web 服务器收到客户端的 http 请求时，分别创建的针对请求的一个用于代表请求的 request 对象和代表响应的 response 对象。前后端通过这种形式交互。
- URLs 和 Router 为前端的 HTTP 请求实现拦截、转发和解析，从而实现页面跳转和函数调用。每个 Django 项目至少需要有一个 urls.py，用于存储 URL 和视图函数的映射模式。Url 映射基于正则表达式实现。

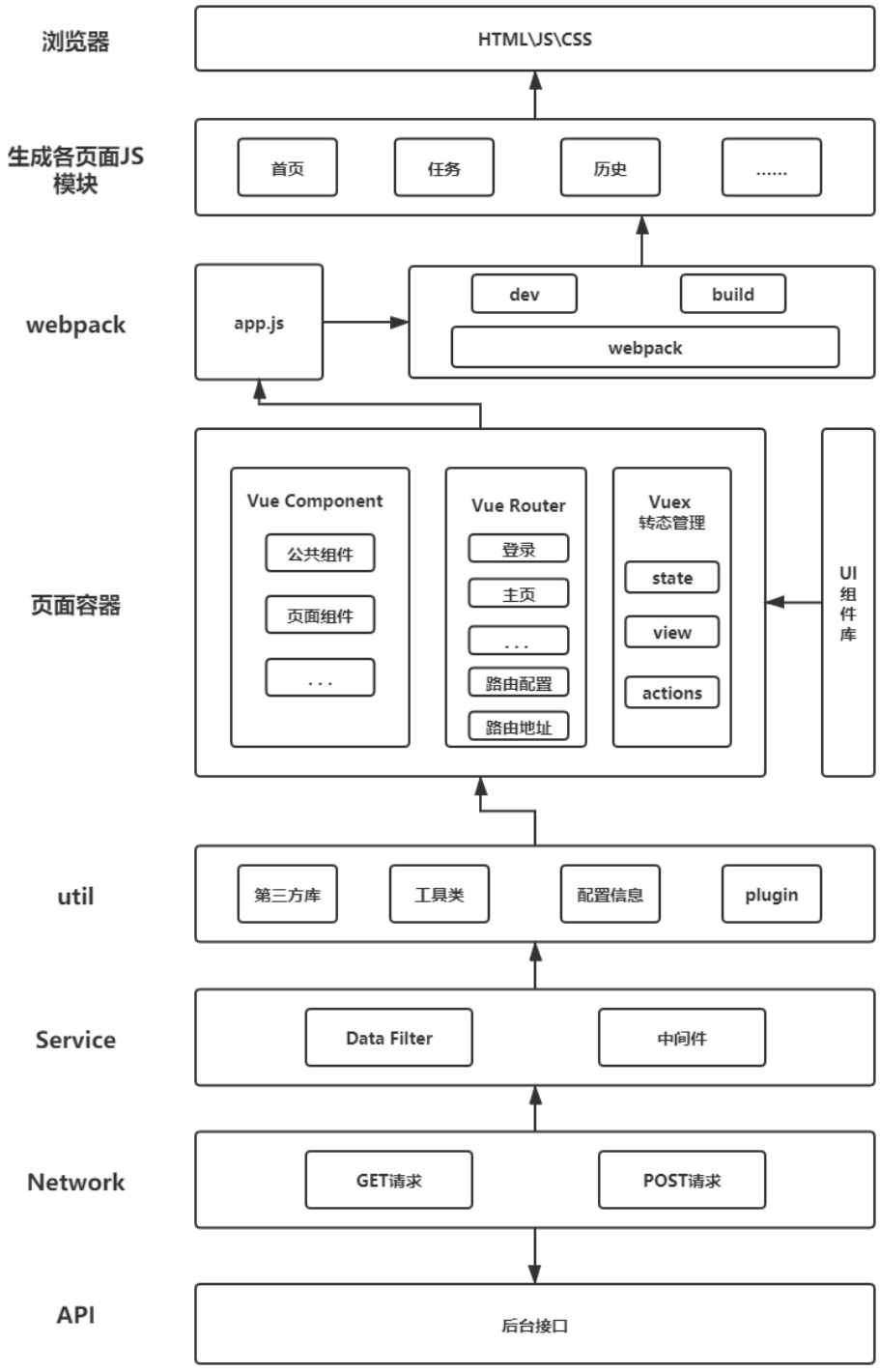
浏览器
HTML\JS\CSS
生成各页面JS模块
首页
任务
历史
.....
webpack
app.js
dev
build
webpack
页面容器
Vue Component
公共组件
页面组件
. . .
Vue Router
登录
主页
. . .
路由配置
路由地址
Vuex 转态管理
state
view
actions
UI组件库
util
第三方库
工具类
配置信息
plugin
Service
Data Filter
中间件
Network
GET请求
POST请求
API
后台接口

图 3-6: 系统前端结构图

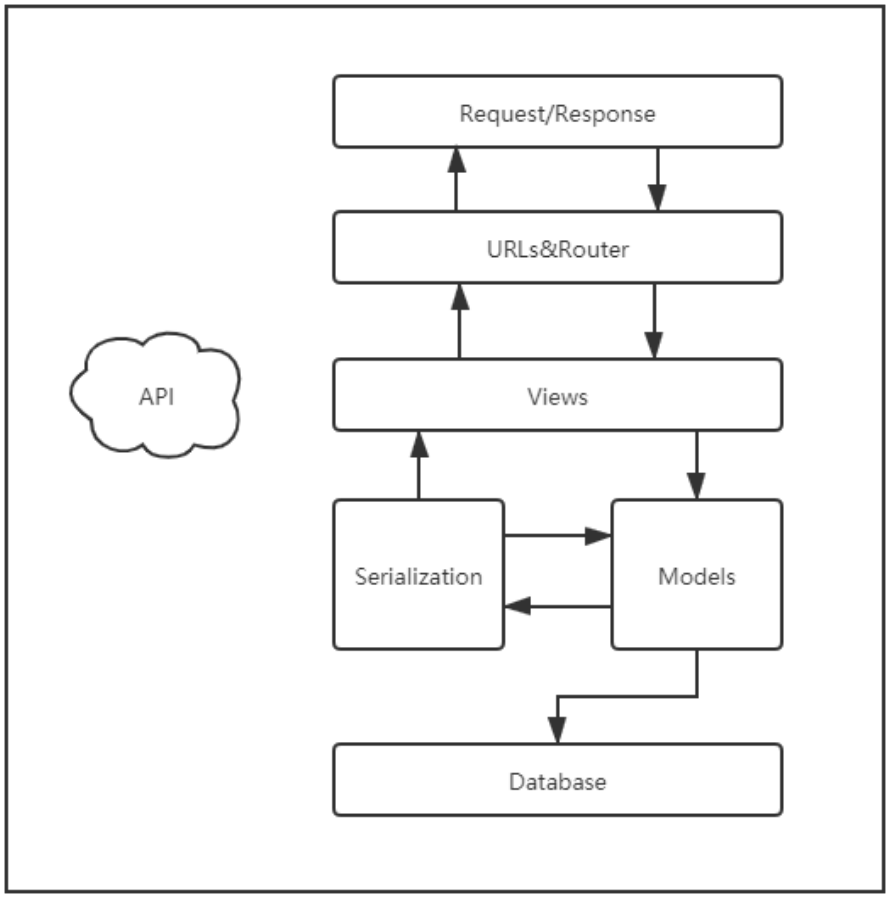

图 3-7: 系统后端结构图

- Views 是 Django 框架的视图层，所谓的视图层是进行业务处理的，也包括接收和响应 HTTP 请求，此处主要指 views.py。当 Web 服务器接受到用户发出的 HTTP Request 后，Django 会执行 url 映射，然后把后续的解析过程交给视图函数来处理。视图函数通常会调用模型层的数据，从而渲染 Template 网页，最终以 HTTP Response 的形式返回给用户。
- Serialization 在 Django 框架下会将 Django 模型翻译成其他格式，一般为 xml，json 和 yaml 格式，主要应用是将数据库中检索的数据返回给客户端用户。
- Models 里定义了系统的模型，每个模型都是一个 Python 的类，继承于 django.db.models.Model，而模型类的属性则代表数据库的字段。模型是关于系统数据的单一、确认的信息来源。
- Database 在 Django 的 settings.py 中配置，本项目采用 MySQL。

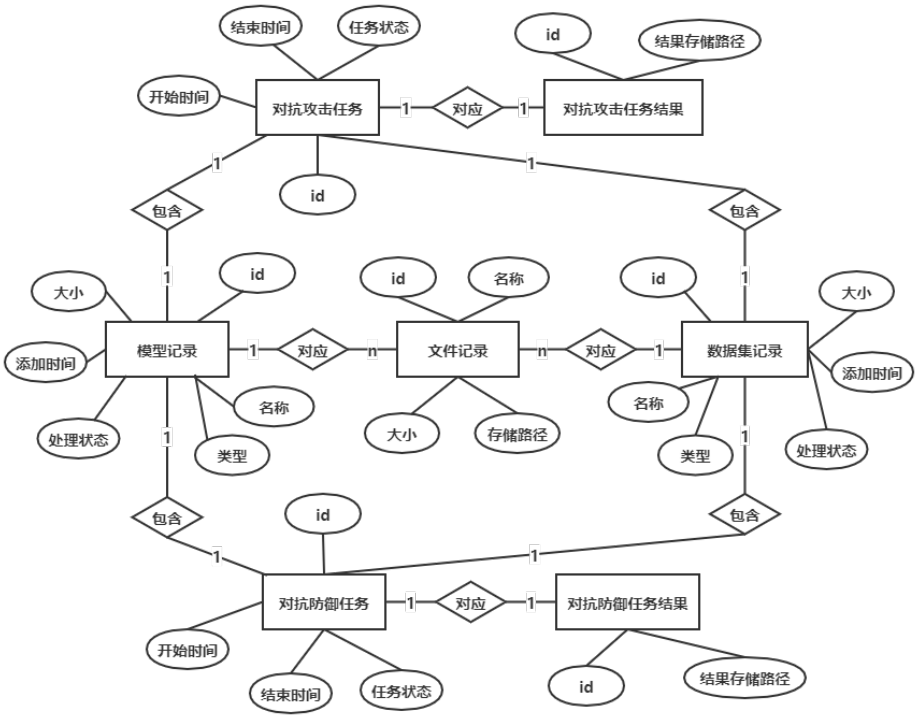

图 3-8: 实体关系图

本系统的实体关系图如图3-8所示，实体关系图表现了数据对象的各属性和实体之间的联系，为数据库的设计提供一个整体的方向。文件记录作为一个实体，对应模型记录和数据集记录，包含文件本身的基本参数和存储位置，为将文件本身与上传记录拆分而设计。当文件上传成功后，数据库中将新增上传模型和数据集的记录，记录包含添加时间，上传状态等。任务实体分为对抗攻击任务和对抗防御任务。任务包含模型记录和数据集记录，是因为任务需要调用用户上传的模型和数据集，同时任务由于需要一定的运行时间，将记录开始时间，结束时间和任务执行状态，当任务执行完成后，线程将自动更新任务执行状态。一个任务对应一个任务结果，任务结果包含 json 文件，记录存储路径，为任务结果的下载提供支持。

依据前文的模块划分，设计系统涉及的实体表，包含字段，含义，类型和描述。关于用户管理的用户实体较为简单，仅存储用户的基础信息，如账号密码，不在此描述。表 file，表 model 和表 dataset 支持文件的上传；表 attack，表 defense，表 atkresult 和表 defresult 支持任务的创建执行。管理模块与上述的文件和任务记录实体都有对应。由于本系统的后端使用 Django 框架，数据库表在 model 中定义且通过 ORM 生成，面向项目实体对象进行编程，避免了数据库语

句的编写。因此，在面对数据库的差异时，不会出现兼容性问题。实现了数据模型与数据库的解耦。

表 3-10: file 表字段详细设计

字段	含义	类型	描述
id	文件 id	int	上传文件的主键
user_id	用户 id	int	用户主键
file_name	文件名称	varchar	上传文件的名称
file_size	文件大小	int	上传文件的大小
file_path	文件路径	varchar	上传文件的存储路径
md5	文件 md5 值	varchar	文件 md5 值，检验唯一性
is_delete	文件是否删除	tinyint	逻辑删除标志

表3-10描述了上传文件实体表的详细设计。文件在上传完成后将记录文件的基本信息，包括名称和大小，同时保存文件所在路径以备读取使用。md5 值用来确保文件的唯一性。

表3-11和表3-12分别展示了深度学习模型的上传记录和模型上传记录的字段详细设计。模型和数据集是专属于上传者的，因此与用户 id 相关联，用户不能访问别的用户上传的文件，保护用户的数据安全。

表 3-11: model 表字段详细设计

字段	含义	类型	描述
id	模型记录 id	int	上传模型记录的主键
user_id	用户 id	int	用户主键
model_name	模型名称	varchar	模型名称
file_id	模型文件 id	int	对应存储文件的 id
size	模型大小	int	模型文件的大小
input_dim	输入维度	varchar	模型的输入维度
output_dim	输出维度	varchar	模型的输出维度
type	模型类型	tinyint	模型类型
add_time	添加时间	varchar	模型添加时间
status	处理状态	tinyint	模型处理状态
is_deleted	模型删除标记	tinyint	模型是否已删除

上传记录与文件本身分为两个实体，是为了区分两者的职责，前者注重上

传这一动作，而后者注重文件本身的存储，且模型和数据集都是文件实体，提取相同的部分作为一个单独的实体，减少冗余设计。对于模型记录和数据集记录，存储用户上传时的配置信息，包括维度、类型等。两种实体都存储了添加时间，处理状态和是否删除的信息，作为查询条件，方便用户查找历史记录。

表 3-12: dataset 表字段详细设计

字段	含义	类型	描述
id	数据集记录 id	int	上传数据集记录的主键
user_id	用户 id	int	用户主键
dataset_name	数据集名称	varchar	数据集名称
file_id	数据集文件 id	int	对应存储文件的 id
size	数据集文件大小	int	数据集大小
data_dim	数据维度	varchar	数据维度
label_dim	标签维度	varchar	标签维度
type	数据集类型	tinyint	数据集类型
add_time	添加时间	varchar	数据集添加时间
is_deleted	数据集删除标记	tinyint	数据集是否已删除

表 3-13和表 3-14描述了用户创建的任务记录，包括对抗样本攻击任务和防御任务。对抗样本生成任务根据用户选择的攻击方法列表 atk_list 生成不同的对抗样本; 而对抗防御任务在用户选择的防御方法列表 def_list 中选择效果最好的方法来优化数据集，构造模型能够正确预测的数据集。

表 3-13: attack 表字段详细设计

字段	含义	类型	描述
id	对抗攻击任务 id	int	对抗攻击任务的主键
user_id	用户 id	int	用户主键
start_time	开始时间	datetime	任务开始时间
finish_time	结束时间	datetime	任务结束时间
model_id	模型 id	int	模型主键
dataset_id	数据集 id	int	数据集主键
atk_list	攻击方法列表	varchar	选用的对抗攻击方法列表
status	任务状态	tinyint	任务执行状态
is_deleted	是否删除	tinyint	删除标志

任务记录开始时间，结束时间和任务状态，作为查询条件，方便用户筛选任务记录。模型 id 和数据集 id 对应模型和数据集的数据库表，表示任务对该模型和数据集的资源调用。两种任务分别会选择所需要使用的对抗样本攻击/防御方法，方法可能有多个，因此该字段存储一个列表。添加逻辑删除标记是为了避免误删除，方便数据恢复。

表 3-14: defense 表字段详细设计

字段	含义	类型	描述
id	对抗防御任务 id	int	对抗防御任务的主键
user_id	用户 id	int	用户主键
start_time	开始时间	datetime	任务开始时间
finish_time	结束时间	datetime	任务结束时间
model_id	模型 id	int	模型主键
dataset_id	原始数据集 id	int	原始数据集主键
dataset_adv_id	对抗数据集 id	int	对抗数据集主键
def_list	防御方法列表	varchar	选用的对抗防御方法列表
status	任务状态	tinyint	任务执行状态
is_deleted	是否删除	tinyint	删除标志

表 3-15: atkresult 表字段详细设计

字段	含义	类型	描述
id	对抗攻击任务结果记录 id	int	对抗攻击任务结果记录的主键
user_id	用户 id	int	用户主键
attack_task_id	对抗攻击任务 id	int	对抗攻击任务主键
result_json	结果存储路径	varchar	任务结果 json 的存储路径
result_file_id	结果文件 id	int	生成的对抗样本文件主键
is_deleted	是否删除	tinyint	删除标志

表 3-15和表 3-16分别描述了对抗攻击任务的结果和对抗防御任务的结果。当任务执行完成后，atkresult 表或 defresult 表中会生成一条任务结果记录，任务结果和任务记录一一对应。任务的结果一般以 json 格式存储，方便前端调用并展示。用户可以根据数据库存储的结果路径下载查看任务结果。对于对抗样

本生成任务，用户可以下载生成的对抗样本数据集；对于对抗样本防御任务，用户可以下载使用防御方法优化后的数据集和任务执行过程产生的对抗样本检测模型。

表 3-16: defresult 表字段详细设计

字段	含义	类型	描述
id	对抗防御任务结果记录 id	int	对抗防御任务结果记录的主键
user_id	用户 id	int	用户主键
defence_task_id	对抗防御任务 id	int	对抗防御任务主键
result_json	结果存储路径	varchar	任务结果 json 的存储路径
result_file_id	结果文件 id	int	优化后的数据集文件主键
model_detect_id	检测模型 id	int	对抗样本检测模型
is_deleted	是否删除	tinyint	删除标志

3.3.4 进程设计

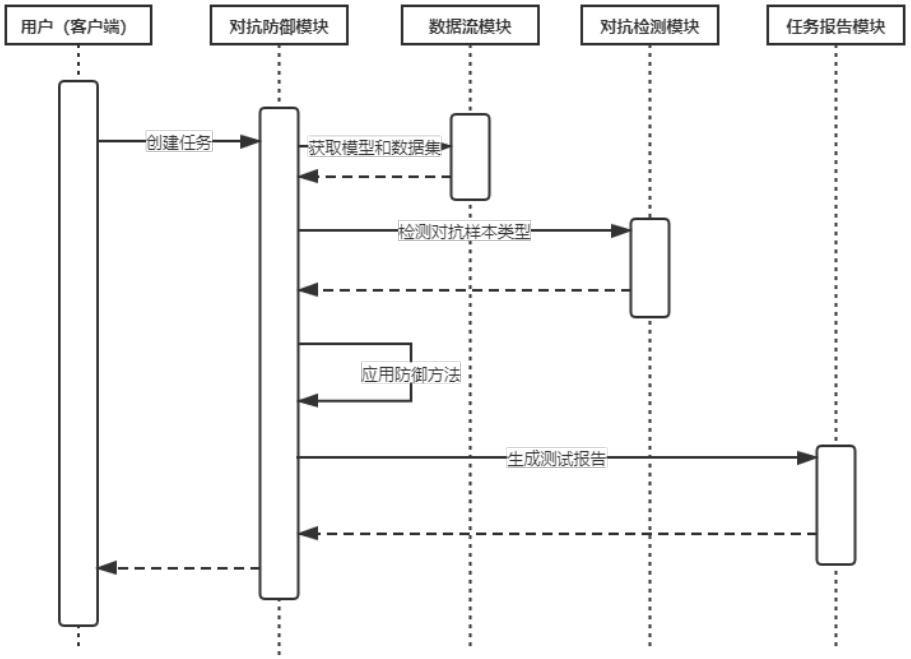

图 3-9: 进程视图

本节描述系统的进程设计，进程视图关注系统的运行流程和特性，描述系统各功能是如何在系统中执行的。本节主要从数据流处理，对抗防御任务，对抗样本检测和任务报告这四个方面来分析描述系统的进程设计，描述对象之间的交互，关注消息序列，即消息在对象之间的发送和接收过程。

系统进程视图如图3-9所示。当用户选择创建任务，任务进程开始运行。对抗防御模块将调用数据流处理模块获取用户选择的深度学习模型和对应的数据集，数据流处理模块负责模型和数据集的解析。接着，将调用对抗检测模块，通过深度学习模型识别对抗样本类型。根据目标对抗样本类型，对抗防御模块将通过系统维护的对抗防御方法效果表查询效果最佳的防御方法并应用。对于测试任务结果，调用任务报告模块，生成任务报告，包括预测成功率，样本结构相似性和扰动敏感距离。在模块之间的协同下，执行一个完整的对抗防御任务流程，帮助用户在给定的模型和数据集上，防御对抗样本的攻击。

3.3.5 部署设计

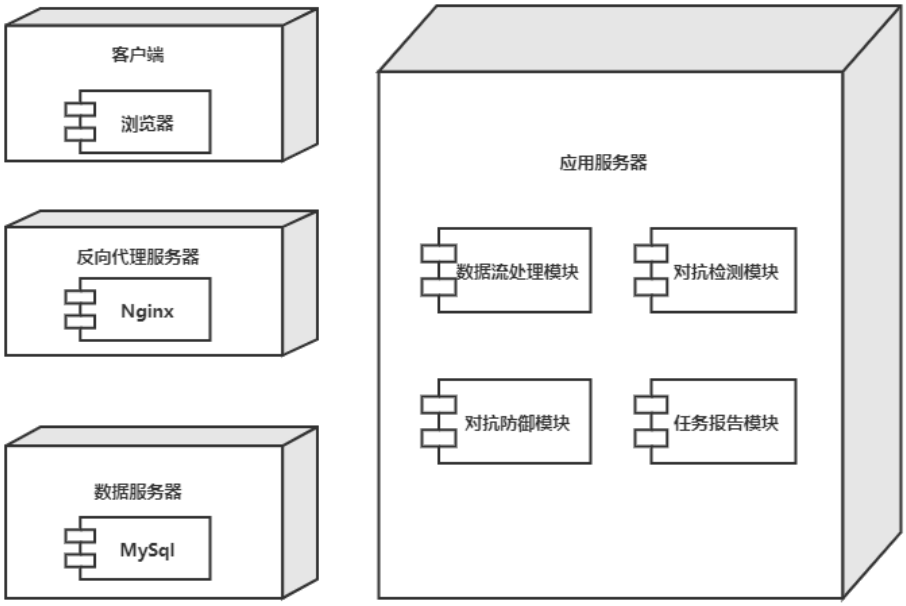

图 3-10: 物理视图

系统的物理视图如图3-10所示，描述了系统软硬件之间的对应关系，解决系统安装、通讯和拓扑结构等问题，其作用是把软件的不同部分映射到不同的硬件节点中。用户可以通过浏览器来访问本系统，使用本系统的功能。用户

的操作请求将被发送到 Nginx 反向代理服务器中，再通过负载均衡将请求交付给应用服务器。为了增强系统的可靠性，提高故障处理能力，可以部署多个 Nginx 实例。系统的各个功能模块统一部署在应用服务器上，且服务之间可交互。为了降低应用服务器的复杂度，并不在应用服务器中存储各模块的内部状态，而是将数据存储至数据服务器。本系统的数据库使用 MySQL。

3.4 本章小结

本章主要介绍系统的需求分析和概要设计。需求分析从功能性需求和非功能性需求两个方面来展开，并对系统的功能归纳用例。概要设计包含架构设计，逻辑设计，开发设计，进程设计和部署设计，通过图表的形式展开描述，之后将系统划分为数据流处理模块，对抗检测模块，对抗防御模块和任务报告模块。

第四章　对抗样本检测和防御系统的详细设计与实现

本章主要介绍项目的详细设计与实现，按模块的角度来分别描述。从第三章的需求分析和概要设计中，总结出系统的四个模块，分别为数据流处理模块，对抗检测模块，对抗防御模块和任务报告模块。

4.1　数据流处理模块

数据流处理模块实现深度学习模型和数据集的上传，并对已上传的文件进行查看，删除和下载操作。该模块主要涉及 Model，Dataset 和 File 三个模型类，定义在 models.py 中，并映射到数据库表。

4.1.1　时序图

系统对任务资源的数据流时序图如图4-1所示，分为上传文件和文件管理两个部分。用户在上传文件或数据集时，通过用户界面完成上传操作，将文件上传到系统服务器上。系统首先会生成上传记录，在数据库生成一条数据。同时，系统异步进行文件的存储，生成在文件表中生成条目记录文件的存储路径。当存储完成后，数据库会进行存储反馈，告知系统已经存储完毕，系统也将反馈给用户上传操作成功。此时系统会更新存储状态，即将文件上传记录中的处理状态设为完成。当数据库完成更新，则表示上传文件操作完全结束。用户在查询上传记录时，首先进行条件筛选，系统对数据库进行查询操作。当数据库查询完成后，进行数据反馈，系统获取数据并反馈给前端，以不同的形式展现在用户界面上。除此以外，用户还可以删除和下载已上传的文件，时序图流程与查询上传记录相似。删除操作采用逻辑删除，即将数据库表中的 is_deleted 字段设为真，不采用物理删除是为了方便数据恢复。

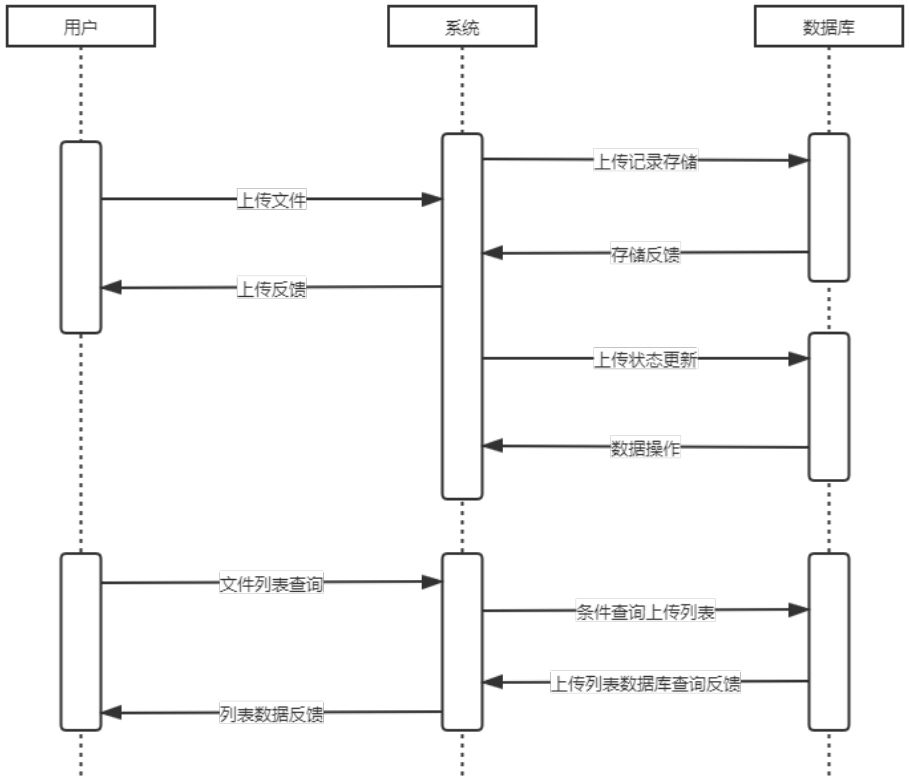

图 4-1: 数据流处理时序图

4.1.2 关键代码

文件上传功能的核心代码实现如图4-2所示。以数据集的上传为例，首先，从 HTTP Request 中获取上传文件的名称，与系统配置的数据集存储路径组成目标存储位置。由于数据集一般为大文件，使用 chunks() 方法代替 read() 方法，chunks() 方法在上传大文件时使用分块方法，默认分成多个 2.5M 的块，值可以调节。对于分块的文件循环调用写操作从而保证大文件的上传不会大量占用系统的内存，优化上传功能的性能。

当文件上传完毕后，将存储文件记录和上传记录。文件记录对应 File 模型类，存储文件的存储路径，为后续的调用和下载提供依据。当在数据库表中添加记录完成后，返回生成的记录 id，即文件主键，作为上传记录的参数。

在 Django 框架下，上传记录管理使用 ORM 的对象关系映射，通过模型类的筛选来进行数据库查询，无需 SQL 语句。根据用户 id 和逻辑删除标志 is_deleted 来筛选上传记录，如果该条记录的逻辑删除标志为真，则代表已删除，不再进行展示，用户不可再对其做包括查询和下载的操作。在数据库查询

```
def upload_dataset(request):
    dataset_name=request.FILES["upload_file"].name
    path=DATASET_URL
    with open(path+dataset_name,"wb") as dest:
        for i in request.FILES["upload_file"].chunks():
            #chunks()打开大文件
            dest.write(i)
    #存储文件记录
    file_id=upload.save_file(request,path+dataset_name)
    #存储上传记录
    upload.save_dataset(request,file_id)

    return HttpResponse("上传成功")
```

图 4-2: 文件上传

完成后，系统将结果包装成 json 格式，以 JsonResponse 的格式返回给前端，在界面中进行展示。

文件下载的代码实现如图4-3所示。为了处理大文件的下载，添加 file_iterator 方法，分块读取文件，不占用大量系统的内存，优化下载功能的性能，然后将该迭代器以参数的形式传给 StreaminghttpResponse 对象。StreamingHttpResponse 对象的作用是将文件流发送给浏览器，与 HttpResponse 相比，对大文件的支持更好，而 HttpResponse 会直接使用迭代器对象，将其内容存储成字符串，当文件变大时会出现耗费时间和内存的缺点。文件的名称即指定的路径 path，通过 HttpRequest 获取。

```
def download(request):
    def file_iterator(file_name, chunk_size=512):
        with open(file_name) as f:
            while True:
                c = f.read(chunk_size)
                if c:
                    yield c
                else:
                    break
    #使用文件流
    the_file_name = request.POST.get('path','')
    response = StreamingHttpResponse(file_iterator(the_file_name))
    #将文件流写入硬盘
    response['Content-Type'] = 'application/octet-stream'
    response['Content-Disposition'] = 'attachment;filename="{0}"'
                                      .format(the_file_name)

    return response
```

图 4-3: 文件下载

通过上述的代码，可以实现将服务器上的文件，以文件流的形式传输到浏览器，但这样实现带来的问题是文件流会在浏览器中显示，且通常以乱码的形式，而不是将文件直接下载到硬盘中，不符合需求。因此，在 StreamingHttpResponse 中添加 Content-Type 和 Content-Disposition 字段，其作用是指定文件流写入硬盘。Content-Type 指示资源的 MIME 类型，指示传输内容的类型；Content-Disposition 是 MIME 协议的扩展，表示服务端指示客户端响应的两种形式，而此处的参数表示以附件的形式下载或保存。

4.1.3 界面演示

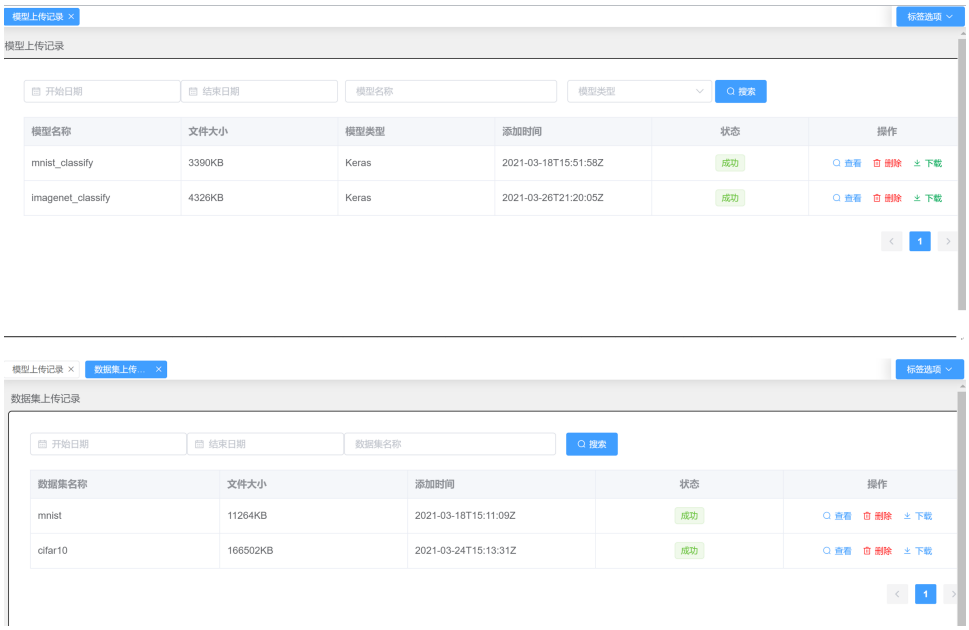

图 4-4: 上传记录管理界面

上传记录管理的用户界面如图4-4所示，上图为模型上传记录，下图为数据集上传记录。用户进入上传记录界面后，可以查看上传记录的列表信息。此外，用户还可以通过设定添加时间的范围，模型的名称和模型类型来进行筛选，获取指定的上传记录，筛选操作在 Vue 前端实现。对于其中的任意一条上传记录，用户可以进行查看，删除和下载操作，查看和下载操作仅限已完成的上传记录。

4.2 对抗检测模块

4.2.1 时序图

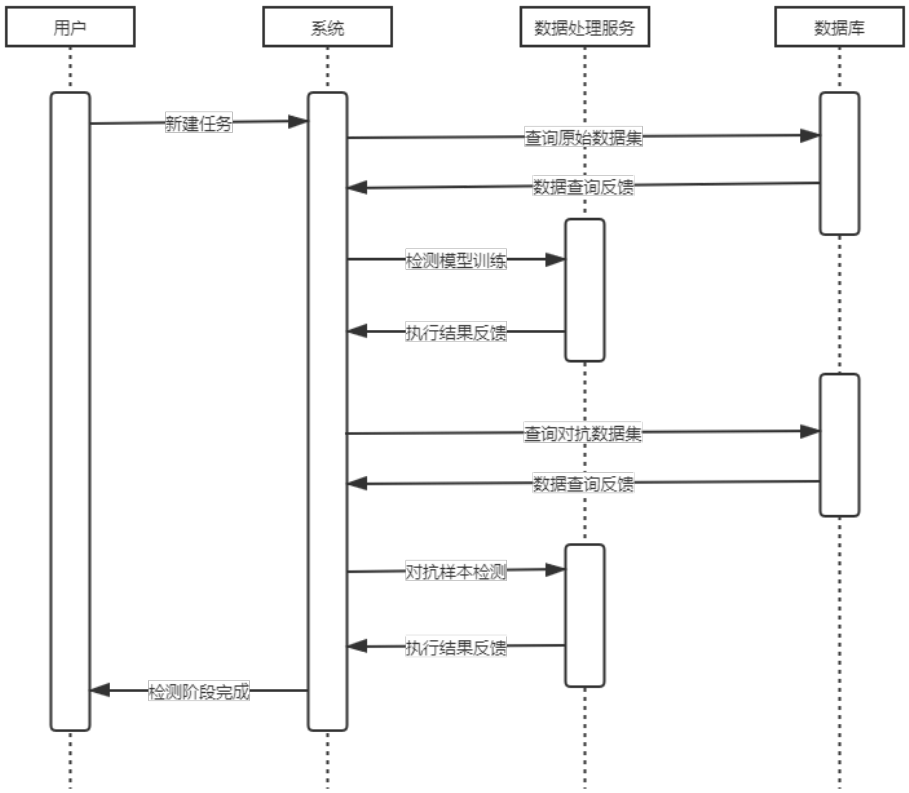

图 4-5: 对抗样本检测时序图

系统的对抗样本检测时序图如图4-5所示。该功能的作用是检测对抗样本，并根据其生成所用的攻击方法，选择合适的防御方法。当用户创建对抗防御任务时，系统会向数据库查询原始数据集，并对数据集生成多种类型的对抗样本，作为模型的训练集，训练一个对抗样本检测模型。当模型训练完成后，系统再次向数据库查询对抗训练集，并检测该数据集是由哪种对抗样本攻击方法产生，检测结果是后续对抗样本防御方法选用的依据。

4.2.2 关键代码

对抗样本检测模型的数据准备代码实现如图4-6所示。对于训练数据集，检测其数值范围，并规范至 [0,1] 的取值范围。接着从指定的目录中产生批量的

```
    if(check_scale(train_dir)):
        train_datagen = ImageDataGenerator(rescale=1. / 255)
        test_datagen = ImageDataGenerator(rescale=1. / 255)
    ......
    train_generator = train_datagen.flow_from_directory(train_dir,
                          target_size=(shape,shape),batch_size=BATCH_SIZE,
                                  class_mode='categorical')
    validation_generator=test_datagen.flow_from_directory(validation_dir,
                      target_size=(shape, shape), batch_size=BATCH_SIZE,
                                class_mode='categorical')
    ......
```

图 4-6: 数据准备

格式化数据，其中 target_size 是所有图片经过处理后的尺寸，通过处理该生产器生产模型训练所需的张量和类型标签。

```
def train(train_dir,validation_dir,test_dir,shape,class_num):
    ......
    model = generate_model(class_num)
    model.compile(loss='categorical_crossentropy',
             optimizer=optimizers.RMSprop(lr=1e-4),metrics=['acc'])
    history = model.fit_generator(train_generator,steps_per_epoch=100,
                epochs=30,validation_data=validation_generator,
                                validation_steps=50)
    test_generator = test_datagen.flow_from_directory(test_dir,
                target_size=(shape, shape),batch_size=BATCH_SIZE,
                    class_mode='categorical')
    test_loss, test_acc = model.evaluate_generator(test_generator, steps=50)
```

图 4-7: 模型训练

如图4-7为模型训练的代码实现。首先构建分类模型，目标将数据集分类为原始样本和各类对抗样本，分类数量为 class_num。采用交叉熵损失函数，其作用是计算实际概率与期望概率之间的距离，交叉熵值越小，概率分布越接近。RMSprop 优化算法的使用是为了优化损失函数在更新期间可能存在摆动幅度过大的问题。在模型训练完毕后应用测试集评估模型的预测准确率。

训练分类模型来识别对抗样本类型经实验认证是可行的，但当对抗样本类型变多时，效果容易下降。且单个检测模型容易受到二次攻击的影响，因此将多分类模型与多个二分类模型相结合。首先对待测数据集应用多分类模型进行第一轮检测，当识别为对抗样本时，直接反馈结果。当图像被识别为干净样本时，进入第二轮的检测，使用多个二分类模型来验证其是否仍为干净样本。二分类模型在预测成功率上能够接近 100%，因此作为第二轮的检测能够为对抗

样本检测的可靠性提供保障。

该模型为对抗防御任务提供对抗样本检测，识别用户上传的包含对抗样本的数据集属于哪种类别。针对不同类型的对抗样本，推荐效果最佳的对抗防御方法。对抗防御方法的推荐基于实验数据，将攻防效果的数据保存为一个表，通过查表比较作出方法选择。这样做的好处是方便扩展，在添加新的对抗样本攻防方法时只需更新表即可实现。

4.3 对抗防御模块

对抗防御模块实现任务创建，任务记录管理，任务详情查看和任务结果下载的功能。系统支持对抗样本防御任务和对抗样本生成任务，前者对指定的对抗样本数据集进行对抗样本检测，并选用最优的对抗样本防御方法来进行优化，使得该数据集在进行预测时仍能保持较高的正确率。后者在给定的模型和数据集上应用对抗样本攻击方法，生成对抗样本数据集，其目的是为用户提供用于测试的对抗样本数据集。任务记录管理功能以列表形式展示任务历史，用户通过筛选查询目标记录并查看任务结果。任务结果以图表形式展示，通过对比方法应用前后的模型预测成功率表现任务的效果。

4.3.1 时序图

系统的对抗任务时序图如图4-8所示，对抗任务是系统的核心功能，由新建任务，查询任务历史列表和查看任务报告三部分组成。当用户创建对抗样本防御或对抗样本攻击任务任务时，首先配置任务的各项参数，例如选用的模型，数据集和对抗样本相关方法，然后向系统提交任务。系统会在任务创建时在数据库存储一个对应的任务记录，数据库进行操作反馈。由于任务的运行需要一定的时间，初始的任务记录中任务状态为运行中，系统会在反馈用户任务创建成功的同时，在后台异步执行任务。异步执行操作采用线程池，线程池能帮助我们管理线程，它维护多个线程，能够使得资源的消耗降低，从而提高系统的性能，这也使得系统具有并发性。当任务执行完毕后，数据处理服务执行结果反馈，系统得知任务执行完毕并对数据库提出数据更新请求。数据库更新完毕则表示任务创建的流程完结。用户可以在历史记录页面查询任务历史，通过条件筛选找到符合要求的列表。当用户执行筛选时，系统对数据库进行查询请求，数据库反馈查询结果列表，系统在获取结果列表后在用户界面以列表形

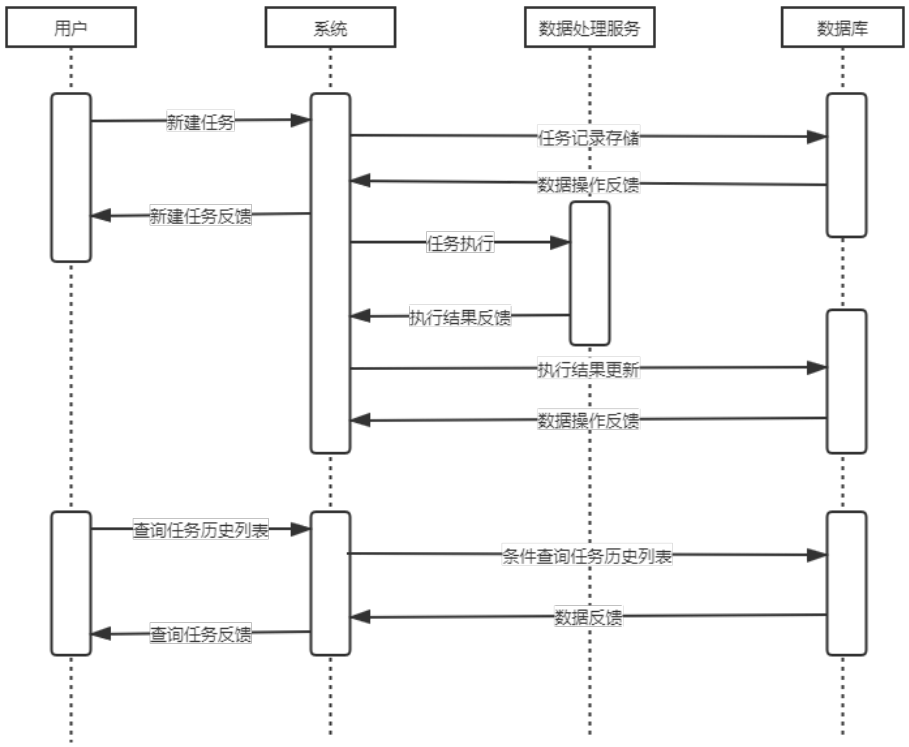

图 4-8: 对抗任务时序图

式展示。用户也能够查询任务结果，与任务记录不同，系统此时进行单个记录的查询，获取数据库的结果反馈后，解析任务结果的 json 文件，并在用户界面以图表等形式展示。用户删除任务记录和下载任务结果的流程与查询类似，未在时序图中指出。为避免数据库出错，且方便数据的恢复，使用逻辑删除，通过数据库表中的 is_deleted 字段控制。任务结果的下载因任务而已，两种任务都存在 json 格式的分析结果，此外，对抗样本生成任务可以下载生成的对抗样本数据集，而防御任务提供优化后的数据集和对抗样本检测模型的下载。

4.3.2 关键代码

本系统目前包含的对抗防御方法包括 Feature Squeezing，HGD，Pixel Defend，Gaussian Augmentation，Inverse GAN 和 JPEG Compression。以 Feature Squeezing 为例，其核心代码如图 4-9所示。

特征压缩（Feature Squeezing）的策略是通过“挤压”出不必要的输入特性来减少对手可用的自由度。函数的输入参数 x 是需要进行压缩的样本，x 值应该在 clip_values 提供的数据范围内。Clip_values 表示特征允许的最小值和最

```
def __call__(self, x: np.ndarray, y: Optional[np.ndarray] = None) ->
            Tuple[np.ndarray, Optional[np.ndarray]]:
    x_normalized = x - self.clip_values[0]
    x_normalized = x_normalized / (self.clip_values[1] - self.clip_values[0])
    max_value = np.rint(2 ** self.bit_depth - 1)
    res = np.rint(x_normalized * max_value) / max_value
    res = res * (self.clip_values[1] - self.clip_values[0])
    res = res + self.clip_values[0]
    return res, y
```

图 4-9: Feature Squeezing 核心代码

大值，一般为（0,1）或（0,255）。Bit_depth 表示每个通道用于编码数据的位数，一般为 8。算法的目标是减少每个像素的位深度（bit depth），位深度的压缩不会降低图像可识别性，实验表明当位深度压缩到 4 时可以有效防御大多对抗样本。

当用户发起创建任务的请求时，系统将在线程池中创建一个新的线程来执行任务，具体的代码实现如图4-10所示，以对抗样本防御任务为例。后端通过 form 来接收任务配置参数，并从 session 中获取 userid，完善 Defense 模块类的其他属性，从而在数据库表中添加该任务记录。此时的任务记录状态为运行中，同时异步运行任务。通过线程池的管理来实现异步执行任务，在线程池中使用 submit 方法提交一个新的 DefenseTask 线程。线程创建成功后通过 HttpResponse 反馈创建成功的信息。

当线程创建后，系统需要为防御任务准备资源，代码实现如图4-11所示。首先根据模型和数据集的 id 获取对应的记录，进一步查询文件存储的路径，并依此训练对抗样本检测模型。对抗样本检测模型的训练基于给定的数据集所生产的各类对抗样本，系统会使用各类对抗攻击方法在数据集上生成一定量的对抗样本作为检测模型的训练集，从而训练出一个可以检测对抗样本的分类模型。

在实际的使用中，该模型在面对不超过 10 种对抗样本类型时，能够训练到 90% 以上的预测成功率。当对抗样本类型增加后，成功率会略有下降，需要在多分类模型的基础上增加由多个二分类模型组成的第二轮检测，来维持预测成功率。在检测模型的支持下，系统可以检测用户上传的对抗数据集主要由哪种对抗攻击方法生成，并根据实验数据的比较分析和用户规定的对抗防御方法列表获取推荐的对抗样本防御方法，从而为任务的运行提供防御方法的选择，该防御方法在此应用场景具有最好的表现。

```
def run_defense(request):
    ......
    if form.is_valid():  # 进行校验
        data = form.cleaned_data
        data['userid'] = request.session['userid']
        data['status'] = STATUS_RUNNING
        data['start_time'] = time.strftime("%Y-%m-%d %H:%M:%S",
                                    time.localtime())
        models.Defense.objects.create(**data)
        defense_response = models.Defense.objects.get(userid=data.
                            get('userid'),start_time=data.get('start_time'))
        # 通过线程池管理线程数
        try:
            pool.submit(DefenseTask, defense_response.id,
                    data.get("model_id"),data.get("dataset_id",
                    data.get("dataset_adv_id")),data.get("def_list"),
                    data.get("userid"))
            response['msg'] = 'success'
            response['error_num'] = 0
        except:
            ......
        return JsonResponse(response)
    else:  # 校验失败
        ......
        return JsonResponse(response)
```

图 4-10: 对抗防御任务创建

```
def DefenseTask(taskid, model_id, dataset_id, dataset_adv_id, def_list, userid):
    #获取资源文件
    model=models.Model.objects.filter(id=model_id)
    dataset=models.Dataset.objects.filter(id=dataset_id)
    dataset_adv=models.Dataset.objects.filter(id=dataset_adv_id)

    detect_model=train_detect_model(model,dataset)
    #获取推荐的防御方法
    defense=get_defense(def_list,detect_model,dataset_adv)
    #运行对抗样本防御的实现
    Defense(taskid, model, dataset_adv, defense, userid).run_defense()
```

图 4-11: 任务资源准备

对抗样本防御任务定义为Defense类，当类初始化后可以调用其run_defense方法来运行，代码实现如图4-12所示。首先加载用户选择的数据集属性，包含数据和标签，接着加载模型。由于加载方法所获取的参数为文件记录，需要通过该记录去数据库查询文件的路径，获取用户上传的文件。根据以上信息可以构建分类器，项目实现的时候采用 KerasClassifier，并调用 fit 功能在原始数据

集上重新训练模型，目的是对比原始数据集，对抗数据集和优化后数据集的预测正确率差距，从而分析防御效果。由于对抗防御方法的选择在任务资源准备步骤中已决定，此处直接获取对应的接口，并对对抗数据集进行防御优化，优化完成后通过分类器进行分类。

```
def run_defense(self):
    # 获取数据集的各项属性
    (x_train, y_train), (x_test, y_test), min_pixel_value, max_pixel_value
                = self.load_dataset(dataset=self.dataset_adv)
    (x_train_adv, y_train_adv), (x_test_adv, y_test_adv),
                min_pixel_value_adv, max_pixel_value_adv =
                self.load_dataset(dataset=self.dataset_adv)
    # 获取模型
    model=self.load_model(self.model)
    #构建分类器
    classifier = KerasClassifier(model=model, clip_values=(min_pixel_value,
             max_pixel_value), use_logits=False)
    classifier.fit(x_train, y_train, batch_size=64, nb_epochs=3)
    #运行对抗防御任务
    defense = self.get_defense(defense=self.defense)
    (x_test_fs, y_test) = defense(x_test_adv, y_test)
    predictions = classifier.predict(x_test_fs)
    accuracy = np.sum(np.argmax(predictions, axis=1) ==
                                np.argmax(y_test, axis=1)) / len(y_test)
```

图 4-12: 任务执行

任务线程执行完毕后，将在任务结果表中新增一条记录，记录任务的运行结果，任务结果主要以 json 文件形式存储，其中包含模型对不同数据集的预测成功率。此外，由于任务的完成，需要对任务记录数据库中对应的记录进行更新，通过 update 方法将任务状态设为已完成。

任务记录管理的实现使用 ORM 的对象关系映射，通过模型类的筛选来进行数据库查询，获取 Defense 列表。此处根据用户 id 和逻辑删除标志 is_deleted 来筛选，如果该条记录的逻辑删除标志为真，则代表已删除，不再进行展示。在数据库查询完成后，系统将结果包装成 json 格式，以 JsonResponse 的格式返回给前端，在界面中进行展示。

4.3.3 界面演示

对抗防御任务创建功能的实现是项目的核心功能，如图 4-13是用户创建对抗样本防御任务的界面。用户需要在界面中进行任务配置，包含模型选择，原

始数据集选择和对抗数据集选择，用户的目标是对对抗数据集进行优化，使得模型面对该数据集时能够保持预测正确率。此外，用户需要在系统提供的对抗样本防御方法池中选择此次任务可以选用的方法，系统将从防御方法池中根据效果选用最佳的方法。任务配置将以 form 表单的形式传递给后端，后端获取参数执行相关方法。对抗样本攻击任务的创建与此类似，用户需要选择对抗样本攻击方法，系统会对每种攻击方法都生成对抗样本。

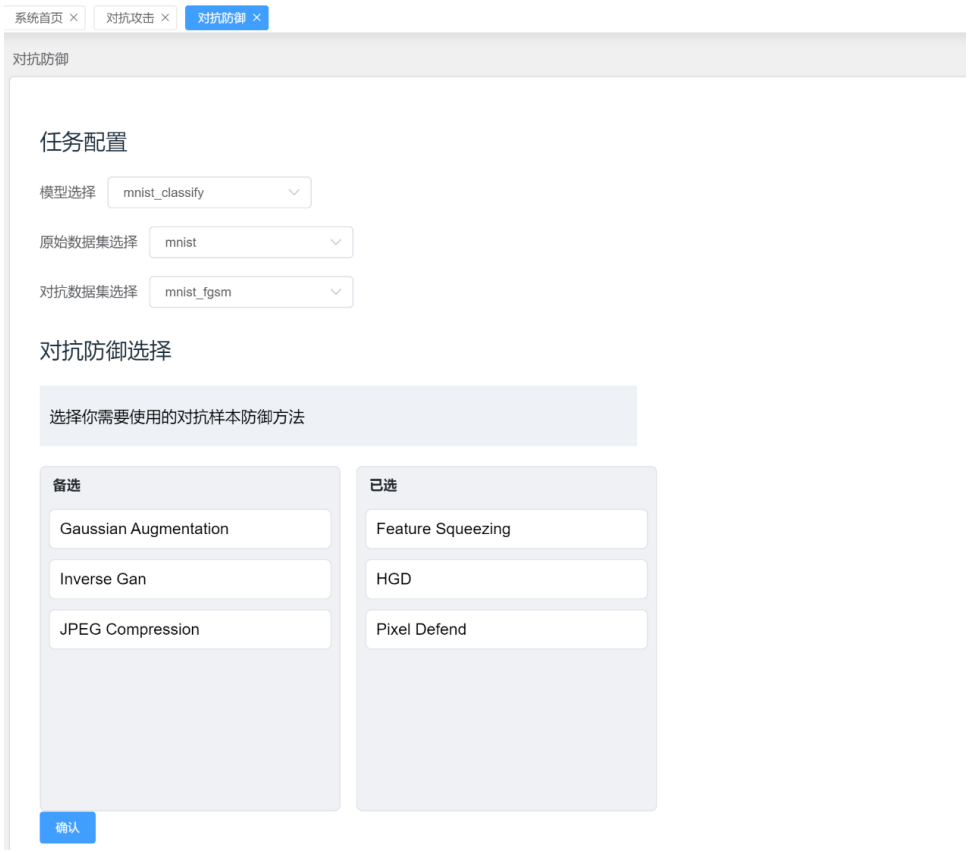

图 4-13: 对抗防御任务创建界面

任务管理界面如图4-14所示。上图为攻击任务记录，下图为防御任务记录。用户进入任务记录界面后，可以查看任务记录的列表信息。此外，用户还可以通过设定添加时间的范围，和任务状态来进行筛选，获取指定的任务记录，筛选操作在 Vue 前端实现。对于其中的任意一条任务记录，用户可以进行查看和下载操作，查看和下载操作仅限已完成的任务记录。用户点击查看按钮将跳转至任务详情界面。

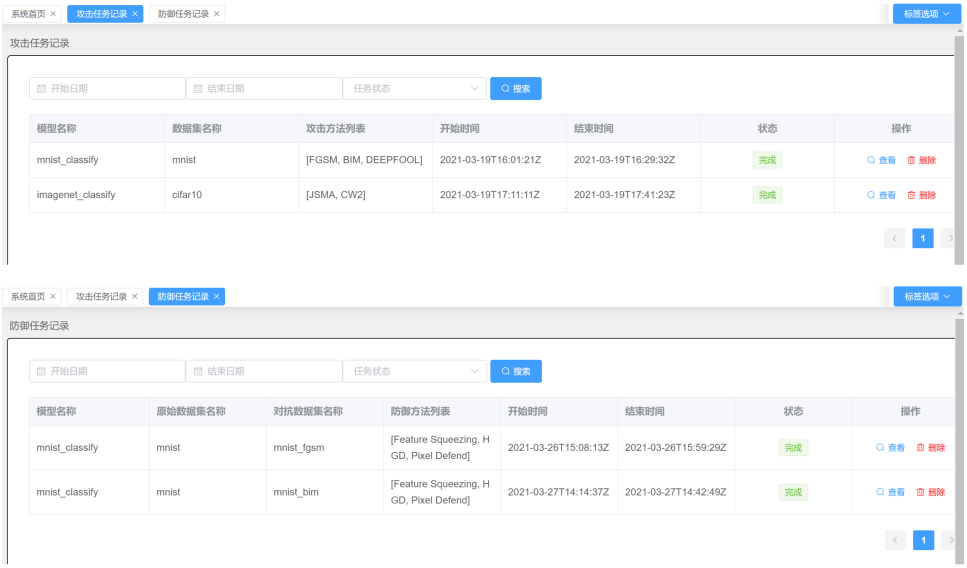

图 4-14: 任务记录管理界面

4.4 任务报告模块

4.4.1 时序图

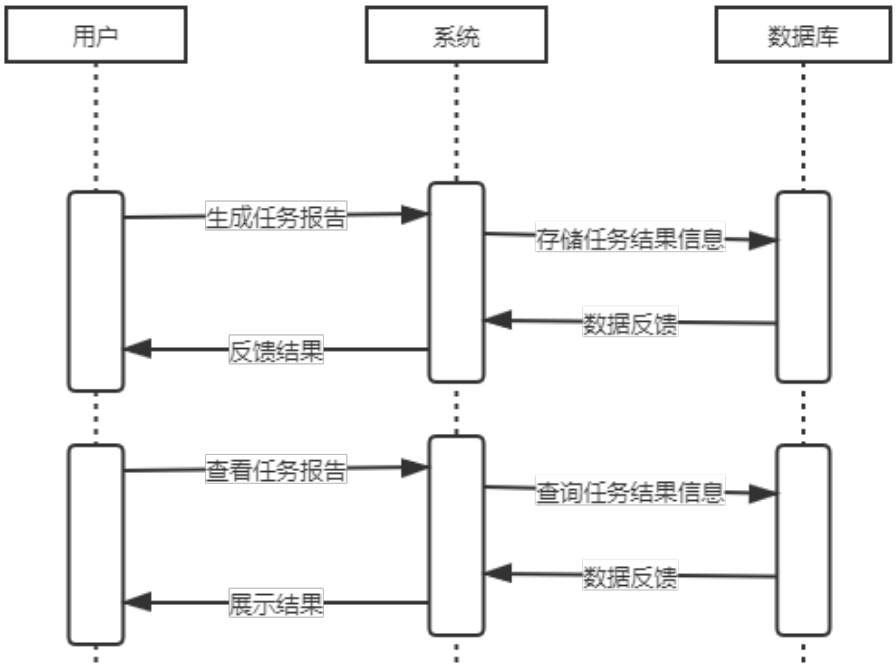

图 4-15: 任务报告时序图

任务报告模块的时序图如图4-15所示。当对抗任务在系统后台执行完毕

后，将生成测试任务报告。对于任务生成的样本数据集，测试模型的预测成功率。此外，对比原始样本和生成的样本，应用结构相似性和扰动敏感距离进行度量。系统将任务报告存储进数据库，供用户后续查看。任务报告在用户界面上以图表形式表现。

4.4.2 关键代码

系统为所实现的功能目前面向图像领域，而应用的对抗样本攻击和防御方法都对图像作出了扰动。为了保证图像的扰动并不会使得其出现失真，人眼无法察觉其变化，引入样本质量评估方法 SSIM 和 PSD。

```
def compute_ssim(image1, image2, k1=0.01, k2=0.03, size=11, L=255):
    C1 = (k1 * L) ** 2
    C2 = (k2 * L) ** 2
    filter = gauss2D(shape=(size, size), sigma=1.5)
    filter = filter / np.sum(np.sum(filter))
    if image1.dtype == np.uint8:
        image1 = np.double(image1)
    if image2.dtype == np.uint8:
        image2 = np.double(image2)
    # 获取公式需要的数据
    mu_x = filter2(image1, filter)
    mu_y = filter2(image2, filter)
    mu_x_sq = mu_x * mu_x
    mu_y_sq = mu_y * mu_y
    mu_x_mu_y = mu_x * mu_y
    sigma_x_sq = filter2(image1 * image1, filter) - mu_x_sq
    sigma_y_sq = filter2(image2 * image2, filter) - mu_y_sq
    sigma_xy = filter2(image1 * image2, filter) - mu_x_mu_y
    # 根据公式计算SSIM
    ssim = ((2 * mu_x_mu_y + C1) * (2 * sigma_xy + C2)) / ((mu_x_sq + mu_y_sq + C1)
               * (sigma_x_sq + sigma_y_sq + C2))
    return np.mean(np.mean(ssim))
```

图 4-16: SSIM

如图4-16所示是计算结构相似性（SSIM）的代码实现。其中 gauss2D 函数为二维高斯滤波器，模仿 matlab 实现，为了建立预定义的滤波器 filter，并为后续的卷积做准备。Filter2 函数是为了实现卷积操作，从而计算所需要的均值，标准差和协方差。filter2 函数传入两个参数，并调用 scipy.signal 库中的 convolve2d 函数进行以模式决定的输出大小进行卷积。通过卷积能够更快速地计算我们所需要的均值、标准差和协方差。Gauss2D 方法的参数 size 默认值为 3×3，作用是指定过滤器的大小。参数 sigma 是过滤器的标准差，默认值为

0.5，它决定了高斯模糊的模糊程度。当取较小的 sigma 值时，数值的分布趋于集中，导致模糊程度减小。而选取较大的 sigma 值时，数值便会显得很分散，导致模糊程度变大。

```
def get_psd(original, adversarial):
    ......
    # 对每个像素计算样本差别与对比度的商
    for channel in range(0, channels):
        for row in range(1, rows - 1):
            for col in range(1, cols - 1):
                square = np.array([[adversarial[row - 1][col - 1][channel],
                                    adversarial[row - 1][col][channel],
                                    adversarial[row - 1][col + 1][channel]],
                                   [adversarial[row][col][channel],
                                    adversarial[row][col][channel],
                                    adversarial[row][col + 1][channel]],
                                   [adversarial[row + 1][col - 1][channel],
                                    adversarial[row + 1][col][channel],
                                    adversarial[row + 1][col + 1][channel]]])
                psd_score = psd_score + abs(difference[row][col][channel]) /
                                    math.exp(square.std())
    ......
    return psd_score
```

图 4-17: PSD

如图4-17是计算扰动敏感距离（PSD）的代码实现。该函数接收原始样本与对抗样本的三维数组形式，代码中的 difference 计算原始样本与对抗样本之间的差距。对于每一个像素，都构成一个新的 3×3 矩阵，来计算一个像素与其周边 8 个像素构成区域的对比度。对比度通过标准差来计算，标准差越高，对比度越高，此时对其作出扰动则难以察觉。通过计算每个扰动除以该像素周围区域的对比度，累积得到扰动敏感距离得分。此时的 PSD 得分越高，则代表越容易被察觉，经实验统计在 400 以内都符合标准。

4.4.3 界面演示

如图4-18为对抗样本防御任务的详情界面，界面展示任务记录的基本信息和任务结果分析。由于在任务执行的过程中进行了对抗防御方法的选择，任务只会选用效果最佳的方法来进行数据集的优化。如任务记录所示，在使用 Feature Squeezing 方法后，预测成功率摆脱了对抗样本的影响，从 1.93% 提升至 93.02%，虽然不及原始样本，但也比较接近，从而体现出防御方法的有效

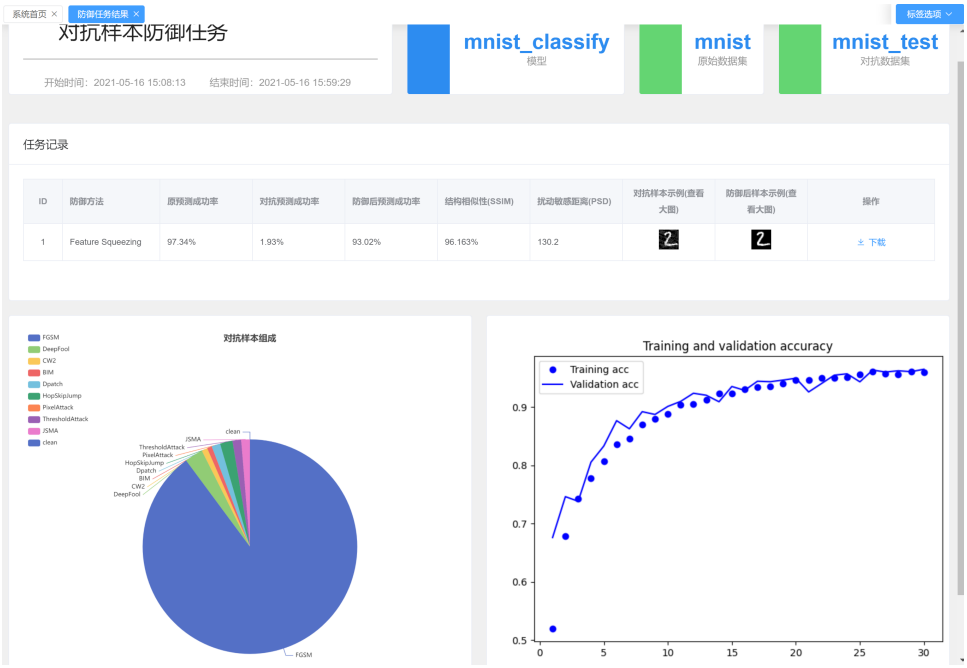

图 4-18: 对抗防御任务详情查看界面

性。界面下方的两张图表表示对抗样本检测模型的训练过程，能够以 90% 以上的准确率进行对抗样本分类。用户可以通过下载按钮下载生成的优化后的数据集和对抗样本检测模型。

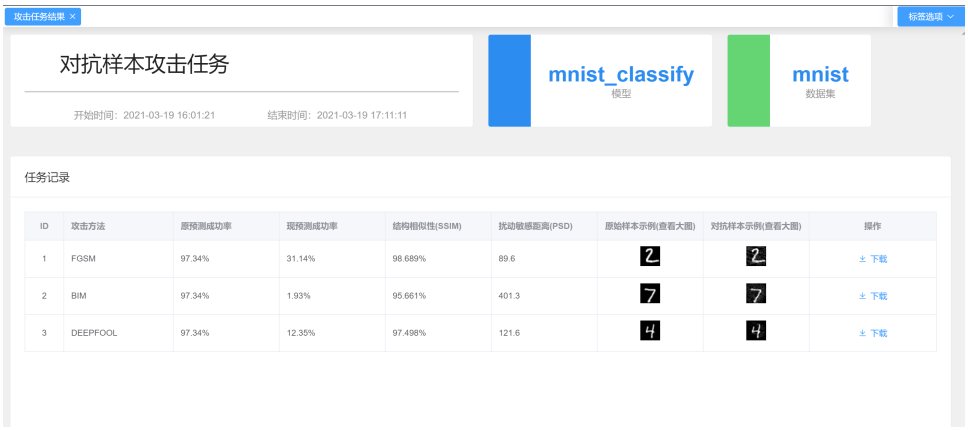

图 4-19: 对抗攻击任务详情查看界面

如图4-19为对抗样本攻击任务详情的界面。界面首先展示任务的基础信息，包括任务使用的模型和数据集，开始时间和结束时间。任务记录基于用户选用的攻击方法列表。对于每种攻击方法，计算其原预测成功率和生成的对抗样本的预测成功率，通过对比可以看到对抗样本能轻易干扰模型的预测结果。为直观展示，添加样本示例图。用户可以通过下载按钮下载生成的对抗样本数

据集。为度量生成的对抗样本的质量，引入结构相似性和扰动敏感距离。任务生成的三类对抗样本与原始样本结构基本一致，扰动敏感距离低于或接近实验总结的标准 400，基本符合要求。

4.5 本章小结

本章从系统的数据流处理，对抗检测，对抗防御和任务报告四大模块描述详细设计与实现。通过模块类的设计，关键代码的实现和相关界面截图的展示来阐述关键功能。

第五章　对抗样本检测和防御系统的测试与分析

5.1　测试准备

5.1.1　测试目标

在现有的测试环境下，基于需求设计功能测试用例，通过执行用例来测试系统功能的可用性和有效性，检验系统的各功能模块是否能满足用户的需求。功能性测试目标包括深度学习模型搭建，数据集上传，上传管理，新建对抗样本生成任务，新建对抗防御任务，历史任务管理，查看任务报告和下载任务报告，测试的目的是保障系统的质量。通过比较各操作输入的预期结果和实际结果，检验功能的可靠性。

5.1.2　测试环境

表 5-1: 系统测试环境

字段	测试环境
服务器	物理服务器 16G 内存，NVIDIA Geforce GTX 1060
操作系统	Windows 10
数据库	Mysql
Python 库	Python 3.6，Django 3.1.2，Tensorflow 1.14，Keras 2.3.1
测试软件	chrome 浏览器
其他软件	Node.js 14.15.4, Vue 2.6.10

如表5-1所示为系统的测试环境，配系统的功能测试配置如下环境。系统部署在 16G 内存的物理服务器上，带有 NVIDIA 显卡，为深度学习的运行提供计算资源。系统的后端采用 Django 框架，作为服务器提供 api 接口，项目通过

Nginx 进行反向代理。前端框架使用 Vue.js，打包后通过 Nginx 配置部署。系统的数据库采用 Mysql，测试时将更新数据库存储的信息。由于项目属于 Web 项目，使用 chrome 浏览器进行测试。

5.2 功能性测试

系统的功能性测试以项目的需求分析为依据来设计测试用例，测试用例包含用户的输入或操作，预期结果和系统功能产生的实际结果组成。通过对比预期结果和实际结果，可以得到测试是否通过。当遇到测试不通过的情况，可以排查出系统所存在的问题和缺陷，以保证系统的可用性。

表 5-2: 文件上传功能测试用例

测试项	操作/输入	预期结果	测试结果
模型搭建	1. 用户登陆系统 2. 点击侧边栏模型搭建按钮 3. 使用 UI 组件搭建模型 4. 点击上传按钮，选择本地模型上传 5. 点击上传按钮，选择同一模型文件再次上传 6. 填写模型信息，点击确认按钮	1. 登陆成功 2. 系统跳转至搭建模型界面 3. 搭建并上传完成后，系统提示成功 4. 上传完成后，系统提示成功 5. 系统提示该模型已上传，无需重复上传 6. 系统提示完成模型搭建，并跳转至模型管理界面	通过
上传数据集	1. 用户登陆系统 2. 点击侧边栏数据集上传按钮 3. 点击上传按钮，选择本地数据集上传 4. 点击上传按钮，选择同一数据集文件再次上传 5. 填写数据集信息，点击确认按钮	1. 登陆成功 2. 系统跳转至数据集上传界面 3. 上传完成后，系统提示成功 4. 系统提示该数据集已上传，无需重复上传 5. 系统提示完成数据集上传，并跳转至上传管理界面	通过

文件上传功能的测试用例如表5-2所示，分为深度学习模型的上传和数据集的上传，用户操作覆盖了正常上传，取消上传和重复上传这几种情况。正常上传采用异步上传，用户的上传操作结束后，系统将跳转至上传管理界面，用户可以在上传记录中找到上传的文件记录。如果此时已上传完成，该记录状态显示已完成，如果还未上传完毕，显示上传中。对于上传中的记录，不可以进行查看详情操作，但可以通过删除记录来取消上传。对于重复上传的情况，系统通过文件的 md5 值来进行检测。测试的实际结果和预期结果一致，设计的测试用例都成功通过。

表 5-3: 上传管理功能测试用例

测试项	操作/输入	预期结果	测试结果
查询模型记录	1. 点击侧边栏上传管理，选择模型 Tab 2. 选择条件筛选	1. 跳转至模型记录页面 2. 显示符合条件的模型记录	通过
查看模型详情	点击某一条模型记录的详情按钮	跳转至模型详情页面	通过
删除模型记录	点击某一条模型记录的删除按钮	系统提示删除成功	通过
下载模型文件	点击某一条模型记录的下载按钮	系统新建下载任务	通过
查询数据集记录	1. 点击侧边栏上传管理，选择数据集 Tab 2. 选择条件筛选	1. 跳转至数据集记录页面 2. 显示符合条件的数据集记录	通过
查看数据集详情	点击某一条数据集记录的详情按钮	跳转至数据集详情页面	通过
删除数据集记录	点击某一条数据集记录的删除按钮	系统提示删除成功	通过
下载数据集文件	点击某一条数据集记录的下载按钮	系统新建下载任务	通过

表5-3描述了上传管理功能的测试用例，从查询列表，查看详情，删除记录，下载文件这四个方面来设计，并验证其可用性。查询列表时可以通过文件的名称，记录添加的时间范围等来进行列表筛选，从而获取用户需要的上传记录。测试的实际结果和预期结果一致，设计的测试用例都成功通过。

新建对抗样本攻击任务的测试用例描述如5-4所示，用户进入对抗样本攻

表 5-4: 新建攻击任务功能测试用例

测试项	操作/输入	预期结果	测试结果
新建对抗样本生成任务	1. 点击侧边栏对抗攻击按钮 2. 点击模型输入框 3. 从已有模型列表中选择模型 4. 点击数据集输入框 5. 从已有数据集列表中选择数据集 6. 选择对抗攻击方法以及完善信息，点击确认按钮	1. 跳转至生成对抗攻击页面 2. 显示已上传模型列表供选择 3. 选择成功，输入框显示选中模型名称 4. 显示已上传数据集列表供选择 5. 选择成功，输入框显示选中数据集名称 6. 系统提示成功，进入任务管理界面	通过

表 5-5: 新建防御任务功能测试用例

测试项	操作/输入	预期结果	测试结果
新建对抗防御优化任务	1. 点击侧边栏对抗防御按钮 2. 点击模型输入框 3. 从已有模型列表中选择模型 4. 点击数据集输入框 5. 从已有数据集列表中选择数据集 6. 选择对抗防御方法以及完善信息，点击确认按钮	1. 跳转至生成对抗样本防御优化页面 2. 显示已上传模型列表供选择 3. 选择成功，输入框显示选中模型名称 4. 显示已上传数据集列表供选择 5. 选择成功，输入框显示选中数据集名称 6. 系统提示成功，进入任务管理界面	通过

击任务创建界面，首先进行各项参数设置。用户可以从已经上传的模型和数据集记录中选择创建任务所需的模型和数据集。由于是对抗样本生成任务，用户需要从系统提供的对抗样本攻击方法列表中选择使用的方法，支持多选。当用户点击确认按钮，任务创建，跳转至任务管理界面。测试的实际结果和预期结

果一致，设计的测试用例都成功通过。

新建对抗样本防御任务的测试用例描述如5-5所示，测试了系统的核心功能，用户进入对抗样本防御任务创建界面，通过配置各项参数来创建任务。用户需要指定防御优化任务面向的深度学习模型，原始的数据集和在该数据集上生成的对抗样本数据集。此外，用户还需要指定对抗样本防御方法的范围，供系统选用。当用户点击确认按钮，任务创建。由于任务异步运行，将跳转至任务管理界面。测试的实际结果和预期结果一致，设计的测试用例都成功通过。

表 5-6: 任务管理功能测试用例

测试项	操作/输入	预期结果	测试结果
查询对抗攻击任务记录	1. 点击侧边栏任务历史 2. 选择条件筛选	1. 跳转至任务历史记录页面 2. 显示符合条件的对抗攻击任务记录	通过
查看对抗攻击任务详情	点击某一条对抗攻击任务记录的详情按钮	跳转至任务详情页面	通过
删除对抗攻击任务记录	点击某一条对抗攻击任务记录的删除按钮	系统提示删除成功	通过
查询防御优化任务记录	1. 点击侧边栏任务历史 2. 选择条件筛选	1. 跳转至任务历史记录页面 2. 显示符合条件的防御优化任务记录	通过
查看防御优化任务详情	点击某一条防御优化任务记录的详情按钮	跳转至任务详情页面	通过
删除防御优化任务记录	点击某一条防御优化任务记录的删除按钮	系统提示删除成功	通过

表5-6描述了任务管理功能的测试用例，与上传管理类似，测试目标为查询任务列表，查看任务报告，删除记录，下载报告这四个功能，并验证其可用性。查看任务记录时，用户可以通过时间范围和任务状态来筛选列表，获取指定的任务记录。任务的状态分为已完成，运行中和错误，仅已完成的任务可以查看其结果。删除记录后界面将进行刷新，从而去除已删除的记录。测试的实际结果和预期结果一致，设计的测试用例都成功通过。

表5-7是任务报告管理功能的测试用例，这部分的测试用例面向单个的任

表 5-7: 任务报告管理功能测试用例

测试项	操作/输入	预期结果	测试结果
对抗攻击任务查看	进入任务历史界面，点击一条对抗攻击任务查看详情	系统跳转至详情界面，展示对抗攻击任务报告	通过
下载对抗攻击任务结果文件	点击某一条对抗攻击任务记录的下载按钮	系统新建下载任务	通过
对抗防御优化任务查看	进入任务历史界面，点击一条对抗防御任务查看详情	系统跳转至详情界面，展示防御优化任务报告	通过
下载防御优化任务结果文件	点击某一条防御优化任务记录的下载按钮	系统新建下载任务	通过

务报告，关注任务报告详情页面的查看和下载功能。测试的实际结果和预期结果一致，设计的测试用例都成功通过。

5.3 非功能性测试

非功能性测试根据非功能性需求设计测试用例。系统的可靠性，安全性和可扩展性体现在项目的设计与实现中，系统的前后端分离，日志记录在一定程度上保障了可靠性；安全性体现在系统对用户数据的保护；项目前端组件的复用，后端功能的切分，通用工具类的设计，都为项目的扩展提供了条件。本节主要描述系统的时间特性。

如表5-8所示，使用测试工具对系统各功能的响应时间进行测试，包含用户登录，上传新模型，上传新数据集，上传管理，新建对抗攻击任务，新建防御优化任务，任务历史查看，对抗攻击任务详情查看和防御优化任务详情查看。对以上功能各进行 100 次操作来获取每次的响应时间，并计算平均响应时间。从表中的数据可以得出，满足系统的时间特性非功能性需求。用户创建对抗攻击任务和防御任务时，表中记录创建任务消耗的时间，系统将在后台花费较长的时间运行任务。考虑用户的使用体验，当任务创建完成后，任务采用异步运行，用户无需等待任务运行完毕。

表 5-8: 非功能性测试用例

操作	平均相应时间	最大相应时间	最小相应时间	测试结果
用户登录	798ms	1256ms	618ms	通过
上传新模型	1459ms	1912ms	994ms	通过
上传新数据集	1617ms	1756ms	1485ms	通过
上传管理	937ms	1046ms	669ms	通过
新建对抗攻击任务	1575ms	1811ms	1331ms	通过
新建防御优化任务	1743ms	2044ms	1423ms	通过
任务历史查看	1048ms	1561ms	741ms	通过
对抗攻击任务详情查看	841ms	1053ms	672ms	通过
防御优化任务详情查看	851ms	947ms	599ms	通过

5.4 对抗样本攻防效果测试

为验证系统对深度学习鲁棒性的优化效果，设计以下实验：实验选用 MNIST 数据集，选取 5 种对抗样本攻击方法（FGSM, BIM, DEEPFOOL, JSMA, CW2）和三种对抗样本防御方法（Feature Squeezing, HGD, Pixel Defend），测试不同情况下的模型预测成功率。具体数据如表 5-9所示。

表 5-9: 对抗样本攻防效果记录

	无对抗防御方法	Feature Squeezing	HGD	Pixel Defend
无对抗攻击方法	97.34%	/	/	/
FGSM	31.14%	87.63%	61.35%	84.38%
BIM	1.93%	93.02%	81.35%	66.53%
DEEPFOOL	12.35%	25.97%	75.42%	61.98%
JSMA	0.30%	88.59%	83.13%	94.49%
CW2	65.50%	81.64%	51.08%	63.57%

从表中数据可以发现，原始样本在模型中的预测成功率为 97.34%，当对抗攻击方法对数据集作出扰动后，模型预测成功率都将受到极大的影响。其中 JSMA 尤其明显，使得成功率降低至 1% 以下。而对上述对抗样本应用对抗防御方法后，深度学习模型预测成功率总体上有了回升。然而，并非应用了对抗

防御方法就能起到鲁棒性优化效果，也会出现对抗防御在某些攻击方法上表现不佳的情况，甚至起到反作用。对抗防御的方法因对抗攻击而异，因此总结各组合之间的效果记录作为对抗防御推荐的基础。

表 5-10: 样本图像质量度量

		SSIM(%)	PSD
FGSM	无防御方法	98.689	89.6
	Feature Squeezing	96.163	130.2
	HGD	97.498	121.6
BIM	无防御方法	95.661	401.3
	Feature Squeezing	93.635	385.2
	HGD	95.387	278.7

如表5-10所示为样本图像质量度量的实验数据。实验在 MNIST 数据集上应用 FGSM 和 BIM 生成对抗样本，并应用 Feature Squeezing 和 HGD 从数据集的层面进行对抗防御。结构相似性（SSIM）代表原始样本和目标样本在亮度，对比度和结构三个方面综合计算得出的相似性，从图表数据可以发现都接近100%，效果很高。扰动敏感距离（PSD）评估人眼对扰动的感知，结合实验数据，设定 400 为标准，越低效果越好。图表中 BIM 在无防御方法情况下略高于标准，其它情况满足要求。对比 Feature Squeezing 和 HGD，前者生成的样本评估得分较低，但根据5-9的数据，在预测成功率上表现更好。对比 FGSM 和 BIM，前者的生成算法无迭代，仅一步便生成对抗样本。而后者通过多次迭代生成对抗样本，扰动更明显，因此在样本质量评估时表现较差。

5.5 本章小结

本章主要介绍了系统的测试与分析，从功能性测试和非功能性测试两个方面来进行描述。首先介绍测试目标和测试环境，接着根据需求分析与用例描述来设计测试用例。功能性测试用例的作用是验证系统功能的可用性和有效性，非功能性测试的作用是验证系统的可靠性。

第六章　总结与展望

6.1　总结

基于对抗防御的深度学习鲁棒性优化技术，以对抗样本为主要研究对象，评估对抗样本对模型预测作出的扰动效果，即准确率的下降，和对抗防御方法对对抗样本的防守效果。本文面向对抗样本的攻防方法进行任务设计与实现，实现一个可用的系统，用户可以在系统中选用攻击方法对数据集进行对抗样本生成操作，也可以对给定的模型添加数据集预处理的防御手段，以期在遇到对抗样本的攻击时能够保持鲁棒性，作出正确的预测。本文首先介绍了选题的背景和意义、深度学习安全性的测试和研究现状。接着介绍系统进行 web 开发时使用到的前后端框架以及系统涉及的深度学习技术，包含对抗样本的攻防方法。

随后，本文详细分析了系统系统的功能性需求和非功能性需求，辅以系统用例图，分解描述系统所需要实现的功能。之后从架构设计，逻辑设计，开发设计和进程设计四个方面来进行概要设计，引入系统框架图，前后端架构图，实体关系图，数据库表的设计和时序图等来直观描述系统的概要设计。通过概要设计，本文将系统划分为四个模块，分别为数据流处理模块，对抗检测模块，对抗防御模块和任务报告模块。各模块有自己的职责，但模块不是单独存在的，模块之间相辅相成，使得系统功能能够顺利实现。

本文接着介绍了系统各功能模块的详细设计与实现，通过核心模块的时序图和关键代码的展示来描述模块实现的功能，同时用系统界面截图来直观展示功能的实际效果。最后，本文根据需求和用例设计测试用例，从功能性和非功能性两个方面来检测系统的可用性和可靠性。

本系统为深度学习测人员提供一个进行对抗样本测试的平台，检测对抗样本的类型并应用对抗防御手段，评估其对神经网络鲁棒性的影响。本系统提供一个完整的测试流程，包含数据集和模型的准备，对抗样本的生成，到对抗防御执行。本人主要负责本系统的前端和后端的实现。

6.2 展望

本文初步实现了基于对抗防御的深度学习鲁棒性优化技术，实现了对抗样本攻击任务和对抗样本防御任务以及其效果的分析，但系统仍存在许多不足，在此分析系统可以从哪些方面进行改进。

首先，如今的攻击防御方法在不断的更新，新的方法层出不穷，本系统选用的对抗方法有限，具有局限性。为此，系统可以给用户提供上传新方法的功能，允许用户维护一个自己的攻防方法库，使得测试更具扩展性和灵活性。

其次，系统功能涉及深度学习任务，由于服务器资源有限，在运行速度上并不理想。后续可以从物理资源和代码优化两方面着手，提高运算效率和减少不必要的开销。

最后，防御任务分析结果较为简单，主要以模型预测成功率和样本质量度量作为评估标准。后续可以从方法的时间成本，可移植性等角度更全面地评估其有效性。

参考文献

[1] VAKHSHITEH F, RAMACHANDRA R, NICKABADI A. Threat of Adversarial Attacks on Face Recognition: A Comprehensive Survey[J], 2020.

[2] ZHANG J, LI C. Adversarial Examples: Opportunities and Challenges[J]. IEEE Transactions on Neural Networks and Learning Systems, 2019.

[3] JANG Y, ZHAO T, HONG S, et al. Adversarial Defense via Learning to Generate Diverse Attacks[C] // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019 : 2740 – 2749.

[4] FAWZI A, MOOSAVI-DEZFOOLI S-M, FROSSARD P. The Robustness of Deep Networks: A Geometrical Perspective[J]. IEEE Signal Processing Magazine, 2017, 34(6) : 50 – 62.

[5] CHIU C-C, SAINATH T N, WU Y, et al. State-of-the-art Speech Recognition with Sequence-to-sequence Models[C] // 2018 IEEE International Conference on Acoustics, Speech and Signal Processing. 2018 : 4774 – 4778.

[6] 李生. 自然语言处理的研究与发展 [J]. 燕山大学学报, 2013, 37(5) : 377 – 384.

[7] 杨明. 无人自动驾驶车辆研究综述与展望 [J]. 哈尔滨工业大学学报, 2006, 38(8) : 1259 – 1262.

[8] YANG P-C, SASAKI K, SUZUKI K, et al. Repeatable Folding Task by Humanoid Robot Worker using Deep Learning[J]. IEEE Robotics and Automation Letters, 2016, 2(2) : 397 – 403.

[9] JAVAID A, NIYAZ Q, SUN W, et al. A Deep Learning Approach for Network Intrusion Detection System[C] // Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies. 2016 : 21 – 26.

[10] LOKE S W. Are We Ready for the Internet of Robotic Things in Public Spaces?[C] // Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. 2018 : 891 – 900.

[11] SZEGEDY C, ZAREMBA W, SUTSKEVER I, et al. Intriguing Properties of Neural Networks[J]. arXiv preprint arXiv:1312.6199, 2013.

[12] COTTON N J, WILAMOWSKI B M. Compensation of Sensors Nonlinearity with Neural Networks[C] // IEEE. 2010.

[13] GOODFELLOW I J, SHLENS J, SZEGEDY C. Explaining and Harnessing Adversarial Examples[C] // ICML. 2015.

[14] PAPERNOT N, MCDANIEL P, WU X, et al. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks[C] // 2016 IEEE Symposium on Security and Privacy. 2016 : 582 – 597.

[15] RAUBER J, BRENDEL W, BETHGE M. Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models[J]. arXiv preprint arXiv:1707.04131, 2017.

[16] NICOLAE M-I, SINN M, MINH T N, et al. Adversarial Robustness Toolbox v0. 2.2[J], 2018.

[17] GOODMAN D, XIN H, YANG W, et al. Advbox: A Toolbox to Generate Adversarial Examples that Fool Neural Networks[J]. arXiv preprint arXiv:2001.05574, 2020.

[18] WANG Z, BOVIK A C, SHEIKH H R, et al. Image Quality Assessment: From Error Visibility to Structural Similarity[J]. IEEE transactions on image processing, 2004, 13(4) : 600 – 612.

[19] LUO B, LIU Y, WEI L, et al. Towards Imperceptible and Robust Adversarial Example Attacks against Neural Networks[C] // Proceedings of the AAAI Conference on Artificial Intelligence : Vol 32. 2018.

[20] FORCIER J, BISSEX P, CHUN W J. Python Web Development with Django[M]. [S.l.] : Addison-Wesley Professional, 2008.

[21] DEACON J. Model-view-controller (mvc) Architecture[J]. Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf, 2009.

[22] FILIPOVA O. Learning Vue. js 2[M]. [S.l.] : Packt Publishing Ltd, 2016.

[23] CROCKFORD D. JavaScript: The Good Parts: The Good Parts[M]. [S.l.] : " O'Reilly Media, Inc.", 2008.

[24] 戴志诚, 程劲草. 基于虚拟 DOM 的 Web 前端性能优化研究 [J]. 计算机应用与软件, 2017, 34(12) : 21 – 25.

[25] BOLOOR A, HE X, GILL C, et al. Simple Physical Adversarial Examples against End-to-end Autonomous Driving Models[C] // 2019 IEEE International Conference on Embedded Software and Systems. 2019 : 1 – 7.

[26] TAORI R, KAMSETTY A, CHU B, et al. Targeted Adversarial Examples for Black Box Audio Systems[C] // 2019 IEEE Security and Privacy Workshops. 2019 : 15 – 20.

[27] HU W, TAN Y. Generating Adversarial Malware Examples for Black-box Attacks based on GAN[J]. arXiv preprint arXiv:1702.05983, 2017.

[28] CHEN H, ZHANG H, CHEN P-Y, et al. Show-and-fool: Crafting Adversarial Examples for Neural Image Captioning[J]. arXiv preprint arXiv:1712.02051, 2017, 2.

[29] PAPERNOT N, MCDANIEL P, JHA S, et al. The Limitations of Deep Learning in Adversarial Settings[C] // 2016 IEEE European Symposium on Security and Privacy. 2016 : 372 – 387.

[30] MOOSAVI-DEZFOOLI S-M, FAWZI A, FROSSARD P. Deepfool: A Simple and Accurate Method to Fool Deep Neural Networks[C] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016 : 2574 – 2582.

[31] CARLINI N, WAGNER D. Audio Adversarial Examples: Targeted Attacks on Speech-to-text[C] // 2018 IEEE Security and Privacy Workshops. 2018 : 1 – 7.

[32] XU W, EVANS D, QI Y. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks[J]. arXiv preprint arXiv:1704.01155, 2017.

[33] LIAO F, LIANG M, DONG Y, et al. Defense against Adversarial Attacks using High-level Representation Guided Denoiser[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018 : 1778 – 1787.

[34] SONG Y, KIM T, NOWOZIN S, et al. Pixeldefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples[J]. arXiv preprint arXiv:1710.10766, 2017.

[35] DAS N, SHANBHOGUE M, CHEN S-T, et al. Keeping the Bad Guys out: Protecting and Vaccinating Deep Learning with JPEG Compression[J]. arXiv preprint arXiv:1705.02900, 2017.