南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论　文　题　目　基于深度学习的UGC短视频质量评审系统设计与实现

作　者　姓　名　王楠

学科、专业名称　工程硕士（软件工程领域）

研　究　方　向　软件工程

指　导　教　师　陈振宇 教授

2021 年 5 月 19 日

学　　号：**MF1932176**

论文答辩日期：**2021 年 5 月 20 日**

指 导 教 师：　　**（签字）**

The Design and Implementation of UGC Short Video Quality Assessment System Based on Deep Learning

By

Nan Wang

Supervised by

Professor **Zhenyu Chen**

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2021

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：基于深度学习的UGC短视频质量评审系统设计与实现

工程硕士（软件工程领域） 专业 2019 级硕士生姓名： 王楠

指导教师（姓名、职称）： 陈振宇 教授

摘 要

随着移动互联网的普及和多媒体技术的进步，短视频产品已发展成为最炙手可热的互联网产品之一。在各大短视频平台中，每天都有大量 UGC（User Generated Content）短视频被生产出来。完全依靠人工审核视频质量不仅成本巨大，而且效率低下，严重阻碍优质视频的分发和传播。为解决这个问题，本文提出了一种基于深度学习的 UGC 视频质量评估算法，并以此为基础，搭建了一个 UGC 短视频质量评审系统。

本文提出的算法分别使用 CNN 和 LSTM 网络捕获 UGC 视频的空域失真信息和时域失真信息。与以往的工作不同，为了更准确地提取出 UGC 视频中与空域失真相关的深度特征，我们基于大量具有不同类型、不同程度失真的图像训练了一个能够比较图像相对质量的 Siamese 网络，并将它的一个子网络作为预训练网络，在 UGC 视频质量数据集上对其进行微调。在提取出每个 UGC 视频的深度特征后，我们将其组成帧级深度特征序列输入到 LSTM 网络中，并对其进行训练，已使其能够预测 UGC 视频的最终质量。

本文搭建的系统创建了以机器审核为主、人工审核为辅的视频质量审核模式。其中，机器审核主要用于批量评估视频质量，快速过滤低质视频；人工审核主要用于评估机器审核结果，提高机器审核质量。在机器审核模式下，超级审核人员既可以基于上传的数据训练机器审核模型，又可以使用训练好的模型预测 UGC 短视频的质量分数。在人工审核模式下，超级审核人员可以将人工审核任务分发给普通审核人员，而普通审核人员可以对任务中的视频进行质量评分。系统基于 Vue 前端框架和 Flask 后端框架开发，通过 Docker 技术部署。

在 Konvid-1k 数据集上，算法的 PLCC 和 SROCC 指标分别达到了 0.788 和 0.789，较其他算法有明显提升。另外，消融实验和交叉实验的结果证明了算法中每一个步骤的必要性以及算法的泛化能力。测试结果表明，系统每天至少可以审核 7000 个 UGC 短视频。因此，通过这个系统，短视频平台不仅可以降低人工审核成本，而且可以提高视频审核效率，在更短的时间内为用户提供更多优质视频，进一步满足用户日益增长的优质视频需求。

关键词： UGC 短视频，视频质量评估，深度学习

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of UGC Short Video Quality Assessment System Based on Deep Learning

SPECIALIZATION: Software Engineering

POSTGRADUATE: Nan Wang

MENTOR: Professor Zhenyu Chen

Abstract

With the popularization of mobile Internet and the progress of multimedia technology, short video products have developed into one of the most popular Internet products. In major short video platforms, a large number of UGC (User Generated Content) short videos are produced every day. Relying entirely on manual review of video quality is not only costly, but also inefficient, which seriously hinders the distribution and dissemination of high-quality videos. To solve this problem, this thesis proposes a UGC video quality assessment algorithm based on deep learning, and builds a UGC short video quality assessment system based on the algorithm.

The algorithm proposed in this thesis uses CNN and LSTM network to capture spatial distortion information and temporal distortion information of UGC videos, respectively. Unlike previous work, in order to extract the deep features related to spatial distortion in UGC videos more accurately, we train a Siamese network that can compare the relative image quality based on a large number of images with different types and different degrees of distortion, and use one of its sub-networks as a pre-training network to fine-tune it on the UGC video quality dataset. After extracting the deep features of each UGC video, we input the sequences of frame-level deep features into the LSTM network and train it so that it can predict the final quality of UGC videos.

The system built in this thesis creates a video quality review mode with machine review as the main and manual review as the supplement. Among them, machine review is mainly used to evaluate videos quality in batches and quickly filter low-quality videos, manual review is mainly used to evaluate the results of machine review and improve the quality of machine review. In the machine review mode, super reviewers

can train the machine review models based on the uploaded data, and use the trained models to predict the quality scores of the UGC short videos. In the manual review mode, super reviewers can distribute manual review tasks to ordinary reviewers, and ordinary reviewers can rate the quality of the videos in the task. The system is developed based on the Vue front-end framework and Flask back-end framework, and is deployed through Docker technology.

On the Konvid-1k dataset, the algorithm achieves PLCC and SROCC metrics of 0.788 and 0.789 respectively, which is a significant improvement over other algorithms. In addition, the results of the ablation and cross dataset experiments demonstrate the necessity of each step in the algorithm and the generalization ability of the algorithm respectively. The test results show that the system can review at least 7,000 UGC short videos per day. Therefore, through this system, the short video platform can not only reduce the cost of manual review, but also improve the efficiency of video review, provide more high-quality videos to users in a shorter time, and further satisfy the growing demand for quality videos from users.

Keywords: UGC Short Video, Video Quality Assessment, Deep Learning

目录

表 目 录

插图

第一章　引言

1.1　项目背景和意义

短视频即短片视频，是一种继文字、图片、长视频之后的新兴互联网内容传播方式，通常是指在互联网新媒体上播放的、适合在碎片时间和移动状态下观看的、时长在 5 分钟以内的视频。不同于微电影和直播，短视频的创作并没有像微电影一样具有特定的表达形式和配置要求，具有生产流程简单、创作门槛低、参与性强等特点，同时又比直播更具有传播价值。

近几年来，随着移动互联网的普及和多媒体技术的进步，短视频信息流产品迅速发展成为仅次于即时通讯的互联网第二大产品类型，完全占领了用户的碎片时间。目前，我国短视频用户规模已达 8 亿，日均使用时长超过 110 分钟。

在各大短视频平台中，每天都有大量新的 UGC（User Generated Content）短视频被生产出来。与传统视频一般由专业的内容制作者制作不同，短视频以其极低的技术门槛和便捷的制作方式，被大量非专业用户创作与传播。由于缺乏专业的拍摄设备和特定的拍摄技巧，与 PGC（Professionally Generated Content）视频相比，UGC 短视频的质量往往更加参差不齐。另外，视频中特效的添加也为视频质量增加了很多不确定因素。因此，如何高效评估视频质量，快速过滤低质视频，满足用户呈指数式增长的优质视频需求，是各大 UGC 短视频平台都会面临的难题。

一般而言，UGC 短视频平台的视频审核步骤如图 1.1 所示。其中，红线过滤主要用于过滤违规内容，如涉黄、涉恐、涉暴等。规则过滤通常从时长、播控状态、分辨率、横竖版等方面对不合规的视频进行筛选、过滤。人工审核主要用于评估视频质量，筛选低质视频，是视频审核的最关键环节。生产出的 UGC 短视频在经过基础过滤后，仍有大量视频进入到人工审核环节。然而，人工审核视频质量不仅成本巨大，而且效率低下，这严重阻碍了视频审核进程，影响了优质视频的分发和传播。

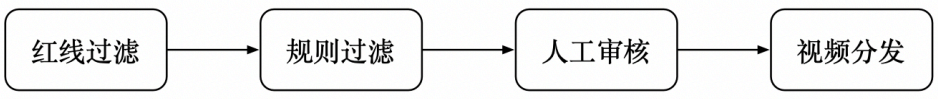


图 1.1: UGC 短视频平台的视频审核步骤

使用机器自动评估视频质量，不仅可以降低人工审核成本，而且可以极大

地提高视频审核效率，缩短视频审核时间。此外，使用机器自动评估视频质量，还可以在用户上传视频时，有针对性地为用户提供修改或优化建议，从而从源头提高视频的质量。因此，通过在视频的生产和消费阶段应用 UGC 短视频质量评估技术，UGC 短视频平台可以在更短的时间内为用户提供更多优质视频，进一步满足用户日益增长的优质视频需求。

1.2　研究现状

目前，国内外对 UGC 短视频的研究相对较少，且多集中于传播学领域和营销学领域。

近几年来，随着各大短视频平台的快速兴起与 UGC 短视频数量的爆发式增长，UGC 视频质量评估技术越来越受到学术界和工业界关注。与传统视频不同，UGC 视频的质量评估具有以下鲜明特点，这些特点使得 UGC 视频的质量评估成为了一项具有挑战性的工作。

（1）因为无法获取到原始的参考视频，这些视频的质量只能通过无参考视频质量评估方法度量与评估。

（2）所有的视频都由普通用户而非专业的内容生产者生产，因此这些视频都存在复杂且无法预知的失真。

（3）这些视频的失真是在真实的现实环境中产生的，所以通常由不同类型、不同程度的失真混合而成，失真类型包括但不限于捕获失真、处理失真、压缩失真、传输失真。此外，与传统的视频质量评估不同，压缩失真并不一定是影响视频质量的决定性因素。

1.2.1　UGC 视频质量数据集现状

最近，KoNViD-1k [1]、LIVE-VQC [2]、YouTube-UGC [3] 等开源 UGC 视频质量数据集的发布大大推动了 UGC 视频质量的研究。这些 UGC 视频质量数据集与传统的视频质量数据集有一个本质区别：在传统的视频质量数据集中，失真是人为合成的，而在 UGC 视频质量数据集中，失真是在现实环境中自然产生的。

传统的视频质量数据集往往以少量的高质量原始视频为基础，并针对每个原始视频人为地合成多个具有不同类型失真的视频，这些合成视频的失真类型通常较为局限，每个视频仅包含至多两种类型的失真。然而，对于真实环境而言，这些视频中的失真过于简单，远远不足以表示现实情况下的失真。在传统的视频质量数据集中，具有代表性且被广泛使用的数据集包括 LIVE-VQA [4]、CSIQ [5] 等。

UGC 视频质量数据集中的视频主要通过采集现实环境中的视频获得。第一个 UGC 相关的视频质量数据集是 CVD2014（Camera Video Database）[6]，这个数据集中的视频由 78 个不同的视频拍摄设备拍摄而成。之后的 LIVE-Qualcomm [7] 也与其类似，视频由 8 个不同的移动设备拍摄而成。然而，这两个数据集仅模拟了视频拍摄过程中产生的捕获失真，没有考虑其他类型的失真。

Hosu 等人 [1] 沿用了第一个成功的 UGC 图像质量数据集 [8] 的思路，使用在线众包的方式构建了 KoNViD-1k 视频质量数据集，这是第一个真正意义上的 UGC 视频质量数据集，包含了从 YFCC100M [9] 数据集中采样的 1200 个公开视频，由 642 个受测者进行主观评分。LIVE-VQC [2] 是另一个包含了 585 个视频的大型 UGC 视频质量数据集，它在 Amazon Mechanical Turk[1] 平台上收集了来自 4776 个受测者的主观评分。最近发布的 UGC 视频质量数据集 YouTube-UGC [3] 由 1380 个时长为 20 秒的视频组成，这些视频从几百万个 YouTube[2] 视频中抽样而来，并由 8000 多个受试者进行主观评分。

1.2.2　UGC 视频质量评估技术现状

传统视频质量数据集与 UGC 视频质量数据集的本质区别导致了以传统视频数据集为基础的视频质量评估算法难以推广到真实的视频质量评估场景。随着上述 UGC 视频质量数据集的发布，视频质量评估领域的研究重点逐渐向 UGC 视频质量数据集的视频质量评估倾斜。

Li 等人 [10] 将 HVS（Human Visual System，人类视觉系统）的两个显著方面纳入到自然失真视频的质量评估研究中，即内容依赖效应和时间记忆效应。内容依赖效应表明人类对视频质量的主观判断取决于视频的内容，人类会根据视频内容对同样质量的图片作出不同的评价。时间记忆效应表明人类对当前帧质量的判断不仅受到当前帧的影响，而且受到之前帧的影响。也就是说，人类会记得之前质量较差的帧，并且会因此降低对当前帧质量的判断。基于这两种效应，他们将在图像分类任务上预训练过的 CNN 作为深度特征提取器，并使用 GRU（Gated Recurrent Unit，门控循环单元） [11] 和主观激发的时域池化层整合帧级深度特征，最终在多个自然失真视频数据集上取得了领先的性能。

Tu 等人 [12] 首次提出了 UGC-VQA 问题，并对这个具有挑战性的问题进行了深入探讨，以启发和推动未来对 UGC-VQA 问题的研究。他们使用统一的框架较为全面地评估了当前领先的无参考视频质量评估方法，并提出了一种基于融合的 UGC 视频质量评估方法 VIDEVAL。鉴于由不同模型提取的特征可能代表

[1] https://www.mturk.com

[2] https://www.youtube.com

了视频不同层面的感知质量，他们从当前领先的无参考视频质量评估模型中抽取了 763 个特征，并从中选取了 60 个特征进行融合。实验结果表明，与其他领先的无参考视频质量评估方法相比，他们的方法以较低的计算成本获得了相当好的性能。同时，深度学习模型的良好实验结果也表明了迁移学习在 UGC-VQA 问题上的巨大潜力。

Li 等人 [13] 基于 TikTok[3]短视频平台，对编码后的 UGC 视频质量进行了研究。他们构造了一个 UGC 视频质量数据集，数据集的视频由 TikTok 用户上传到平台的 UGC 原始参考视频和相应的转码视频组成。同时，他们还提出了一种评估数据集中视频质量的方法。由于用户上传到平台的 UGC 原始参考视频本身存在多种失真，所以为了准确衡量数据集中转码视频的质量，他们不仅考虑了原始视频和转码视频之间的差异，还融合了原始视频的质量。他们的研究为 UGC 视频编码器的设计提供了有力依据。

1.3　本文主要工作

随着 UGC 短视频的大量涌现，人工审核视频质量已难以满足短视频平台用户日益增长的优质视频需求，使用机器批量评估视频质量、快速过滤低质视频，已成为 UGC 短视频质量审核的必然趋势。UGC 短视频质量评审系统以提出的 UGC 视频质量评估算法为基础，创建了以机器审核为主、人工审核为辅的视频质量审核模式。算法分别使用 CNN 和 LSTM 网络捕获 UGC 视频的空域失真信息和时域失真信息，共包括图像相对质量学习、UGC 视频深度特征提取、UGC 视频质量评估三个部分。在 Konvid-1k 数据集上，算法的 PLCC 和 SROCC 指标分别达到了 0.788 和 0.789，较其他算法有明显提升。

为解决 UGC 短视频平台视频审核效率低下的问题，本文提出了一种基于深度学习的 UGC 视频质量评估算法，并以此为基础，搭建了一个 UGC 短视频质量评审系统。这个系统主要面向 UGC 短视频平台的质量审核人员，旨在帮助他们高效评估视频质量，快速过滤低质视频。

本文提出的算法综合考虑了 UGC 视频的空域失真和时域失真。在这个算法中，我们使用 CNN 提取 UGC 视频帧的深度特征，从而将待评估的 UGC 视频建模为帧级深度特征序列。之后，我们将帧级深度特征序列输入到 LSTM 网络中，使用 LSTM 网络预测 UGC 视频的最终质量。与以往工作不同的是，为了更准确地提取出 UGC 视频中与空域失真相关的深度特征，我们没有使用在图像分类任务上预训练的 CNN 提取 UGC 视频帧的深度特征，而是训练了一个能够比较图像相对质量的 Siamese 网络，并使用微调后的网络提取 UGC 视频帧的深度

[3]https://www.tiktok.com

特征。KoNViD-1k 数据集上的实验结果表明，本文提出的算法取得了优于其他算法的性能。为验证算法中每一个步骤的必要性以及算法的泛化能力，我们设计了消融实验和交叉实验。

本文搭建的系统创建了以机器审核为主、人工审核为辅的视频质量审核模式。在机器审核模式下，超级审核人员既可以基于上传的数据训练机器审核模型，又可以创建机器审核任务，使用训练好的模型预测 UGC 短视频的质量分数。在人工审核模式下，超级审核人员可以创建人工审核任务，并将其分发给普通审核人员，而普通审核人员可以对任务中的 UGC 短视频进行质量评分。机器审核任务和人工审核任务完成后，超级审核人员可以对审核结果进行分析。系统按逻辑可以划分为四个模块：机器审核模型训练模块实现了训练数据集管理和机器审核模型训练功能，人工审核模块实现了人工审核任务管理、人工审核结果分析和视频质量评分功能，机器审核模块实现了机器审核任务管理和机器审核结果分析功能，账号管理模块实现了账号管理功能。系统基于 Vue 前端框架和 Flask 后端框架开发，通过 Docker 技术部署。另外，Gunicorn 和 Nginx 分别被用于进程管理和负载均衡。

1.4　本文组织结构

在本文中，我们将通过以下六章介绍提出的 UGC 视频质量评估算法和搭建的 UGC 短视频质量评审系统。

第一章，引言。在这一章中，我们阐述了项目的背景和意义，分析了 UGC 视频质量评估的研究现状，说明了本文的主要工作。

第二章，相关概念与技术综述。在这一章中，我们介绍了本文中涉及的重要概念与主要技术，包括视频质量评估算法、深度学习、前后端框架等。

第三章，UGC 视频质量评估算法设计。在这一章中，我们提出了一种基于深度学习的 UGC 视频质量评估算法，并详细讨论了算法的三个主要部分：图像相对质量学习、UGC 视频深度特征提取和 UGC 视频质量评估。

第四章，UGC 短视频质量评审系统搭建。在这一章中，我们按照系统需求分析、系统总体设计、系统详细设计、系统实现的基本顺序，详细描述了 UGC 短视频质量评审系统的搭建流程。

第五章，实验设计与系统测试。在这一章中，我们介绍了实验设计的配置和结果，描述了系统测试的环境、过程和结果。

第六章，总结与展望。在这一章中，我们对本文进行了总结，同时也指出了本文的不足和改进方向。

第二章　相关概念与技术综述

2.1　视频质量评估算法

视频质量评估算法用于量化一段视频通过视频处理系统或传输系统时画面质量的变化。视频质量评估有着丰富的应用场景：首先，通过比较编解码前后的视频质量，可以评估编解码算法的优劣；其次，通过比较经过传输系统前后的视频质量，可以衡量通信系统的优劣；最后，通过比较使用图像增强和重建算法前后的视频质量，可以评估图像增强和重建算法的优劣。同时，视频质量的评估结果可以进一步反馈给上述算法和系统，为下一步的参数优化和配置更改提供依据。在过去的几十年里，由于视频质量评估在视频压缩、视频编解码、视频监控等领域的广泛应用，视频质量评估成为了一个热门的研究领域，涌现出了大量的视频质量评估算法。日益增加的质量评估需求也对视频质量评估算法的准确性和实时性提出了更高的要求。

视频质量评估的方法通常可以分为两类: 主观质量评估和客观质量评估。

主观质量评估是选择一批非专家类型的受测者，让他们在一个特定的受控环境中对视频序列的质量进行评分。主观评估得分一般使用 MOS（Mean Opinion Score，即平均主观得分）和 DMOS（Difference Mean Opinion Score，即平均主观得分差）来表示。前者是将受测者对失真视频的评分做归一化处理，后者是将受测者对原始视频和失真视频的评分差异做归一化处理。主观质量评估中的受控因素包括观看环境、观看距离、视频序列的选择等，主观质量评估的进行一般需要专业的实验环境，以及大量经过培训的受测人员，因此成本较大。

虽然主观质量评估是最为准确的评估方法，但由于该方法相对复杂且其结果易受多种因素影响，在实际应用中通常使用更易于实现的客观质量评估方法，客观质量评估方法通过机器计算的方式评估视频质量，客观质量评估的结果应与主观质量评估的结果具有较好的相关性与一致性。根据对原始视频依赖程度的不同，客观质量评估又可以分为三种类型: 全参考、部分参考和无参考视频质量评估，这三种客观质量评估方法的对比如图 2.1 所示。

2.1.1　全参考视频质量评估算法

全参考视频质量评估算法通过对比原始视频和失真视频的方式评估视频质量，因此适用于原始视频已知的情况。与部分参考、无参考视频质量评估算法相

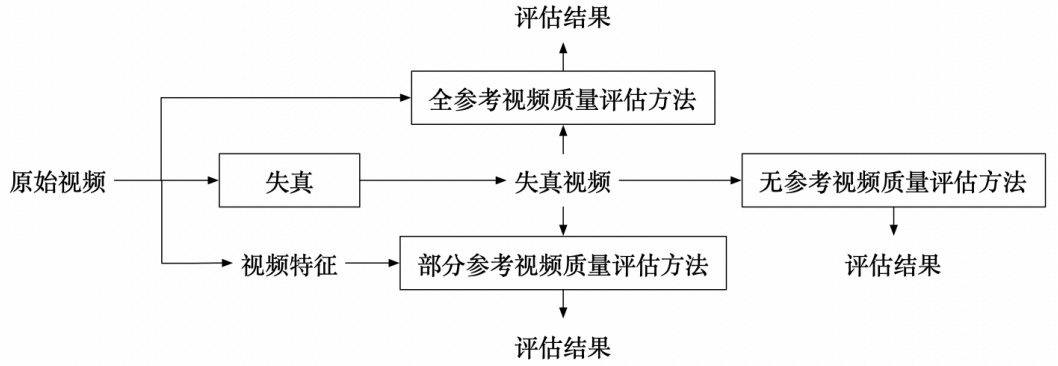


图 2.1: 三种客观质量评估方法的对比图

比，全参考视频质量评估算法已经较为成熟，大致可以分为三类：基于像素误差的评估方法、基于结构信息相似度的评估方法和基于机器学习的评估方法。

基于像素误差的评估方法是将原始视频帧和失真视频帧进行对比，计算两个视频帧像素级别的误差，MSE（Mean Square Error，均方误差）和 PSNR（Peak Signal Noise Ratio，峰值信噪比）是两种典型的基于像素误差的视频质量评估算法。MSE 和 PSNR 的计算方法分别如公式 2.1 和公式 2.2 所示。此外，MAE（Mean Absolute Error，平均绝对误差）、STD（Standard deviation，标准偏差）等方法也与这两种方法类似。基于像素误差的评估方法计算复杂度低，但与主观质量评估的结果有较大差距。目前 PSNR 仍然是编解码器中评估视频质量时使用的最多的方法，这种方法虽然不能很好地反应人类主观感受，但能够客观地衡量编码器带来的损失。

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[I(i,j) - K(i,j)]^2 \tag{2.1}$$

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{2.2}$$

其中，I 和 K 分别为原始视频和失真视频的对应帧，它们的大小为 $M \times N$，视频帧中像素点的最大可能取值为 MAX。

基于结构信息相似度的评估方法为了更好地与主观评估结果匹配，引入了 HVS 的概念，将人类对视频质量的感知建模为结构信息的变化，代表方法有 SSIM（Structure Similarity Index，结构相似度指标）[14]、MS-SSIM（Multi-scale Structural Similarity Index，多尺度结构相似度指标）[15]。SSIM 从亮度、对比度与结构信息三个方面对比原始视频和失真视频的相应帧，在实现上，SSIM 用均值作为亮度差异的度量，用标准差作为对比度差异的度量，用协方差作为结

构信息相似度的度量。MS-SSIM 是 SSIM 的扩展，将不同分辨率和观看条件下视频帧的细节结合到视频质量评估算法中。基于结构信息相似度的评估方法与基于像素误差的评估方法相比，能够更好地反应人类主观感受，但由于人类对视频质量的感知是高度非线性的，所以仍有很大提升空间。

基于机器学习的评估方法以 VMAF（Video Multi-method Assessment Fusion，视频多评估方法融合）[1]为代表性方法，VMAF 由 Netflix 公司开发，用来解决传统评估方法不能反映多种特征、多种场景的视频质量的问题。VMAF 通过机器学习算法将 VIF（Visual Information Fidelity，视觉信息保真度）[16]、DLM（Detail Loss Measure，细节损失指标）[17]、TI（Temporal Information，时域信息）三个基本指标融合为一个最终指标，并为每个基本指标分配一定的权重，这样最终获得的指标就可以保留每个指标的优势，从而获得更好的评估效果。基于机器学习的评估方法将多种评估指标进行融合，既能够把其他好的评估指标纳入到自己的评估指标中来，又能够通过替换各种机器学习算法提高评估准确性。

2.1.2　部分参考视频质量评估算法

部分参考视频质量评估算法通过参考部分原始视频或从原始视频中提取出的一些特征评估视频质量。虽然全参考视频质量评估算法取得了良好的效果，但在现实场景中，存在很多获取不到原始视频，只能获取到原始视频的一部分信息和特征的情况，这便发展出了部分参考视频质量评估算法。较为典型的部分参考视频质量评估方法包括基于特征提取的评估方法、基于谐波强度的评估方法和基于小波域统计模型的评估方法。

2.1.3　无参考视频质量评估算法

无参考视频质量评估算法在完全没有原始视频信息的条件下评估视频质量。近几年来，由于互联网视频类应用的爆发式增长及其原始视频难以获取的特性，无参考视频质量评估逐渐成为研究的热点。现实场景中，人们可以在没有原始参考视频的情况下无差错地判断视频质量，但对于计算机而言，这仍是一项具有挑战性的任务。大多数早期的无参考视频质量评估方法仅针对某种特定类型的失真进行检测，如模糊、噪声、块效应等。近几年，出现了很多通用的无参考评估方法，在失真类型未知的情况下检测视频质量。这些方法可以分为两类：基于传统特征的评估方法和基于深度学习的评估方法。

基于传统特征的评估方法一般使用 NSS（Natural Scene Statistics，自然场景统计）和 NVS（Natural Video Statistics，自然视频统计）模型评估视频质量，因

[1]https://github.com/Netflix/vmaf

为它们对各种失真进行感知。Saad 等人 [18] 提出的 VBLIINDS 无参考评估方法使用 NSS 模型和量化运动相关性的运动模型评估视频质量。Mittal 等人 [19] 提出的 VIIDEO 无参考评估方法根据 NVS 模型对由于失真引入的干扰进行量化。Zhu 等人 [20] 将视频内容分解为预测部分和不确定部分，并使用 NVS 模型分别评估两部分的质量，从而得到整体质量。Li 等人 [21] 针对 3D-DCT（3D Discrete Cosine Transform，三维离散余弦变换）域提出了一种基于 NVS 的无参考视频质量评估方法。

基于深度学习的评估算法将深度学习应用到视频质量评估上来。近几年来，深度学习在各计算机视觉任务上取得了巨大成功。在图像质量评估领域，也涌现出了大量基于深度学习的无参考图像质量评估算法。然而，大规模图像质量数据集的缺乏影响了深度学习在图像质量评估上的应用性能。为了解决这个问题，Kang 等人 [22] 使用由图像分割而成的小块而非整个图像作为 CNN 的输入，以增加训练数据集的数量。Kang 等人 [23] 还设计了一个多任务的 CNN，同时学习图像的失真类型和质量。Bianco 等人 [24] 则使用预训练的网络来缓解训练数据集缺乏的问题，他们使用图像质量评估数据集对预训练的模型进行微调，并从微调后的模型中提取特征，之后将提取的特征作为 SVR（Support Vector Regression，支持向量回归）模型的输入训练模型，完成从特征到质量分数的映射。Liu 等人 [25] 提出了 RankIQA 模型，他们首先从原始图像生成不同等级失真的图像，然后使用 Siamese 网络（孪生网络）学习图像质量等级，之后将 Siamese 网络中的知识迁移到图像质量评估网络中，最后使用微调后的图像质量评估网络评估图像的最终质量。由于视频是由一系列的视频帧组成的，所以这些基于深度学习的图像质量评估算法都可以应用到视频的质量评估中来。

在深度学习直接用于视频质量评估的应用中，Kim 等人 [26] 通过深度 CNN 和卷积聚合网络学习时域和空域的视觉敏感特征。Liu 等人 [27] 提出的 V-MEON 模型使用了一个多任务的 CNN 框架，这个框架同时优化了用于特征提取的 3D-CNN（3D Convolutional Neural Network，三维卷积神经网络）和用于视频质量预测的分类器。Zhang 等人 [28] 利用迁移学习建立了一个基于弱监督学习和重采样策略的无参考评估框架。

2.1.4　算法评价指标

客观视频质量评估方法尽可能地从人的主观视角评估视频质量。通常情况下，我们可以从准确性、单调性、一致性、稳定性几个方面衡量客观质量评分与主观质量评分的匹配程度。所谓准确性指的是客观质量评分与主观质量评分相似的程度；单调性指的是客观质量评分随主观质量评分增减而单调递增或单调

递减的程度；一致性指的是在评估不同类型视频质量时表现一致的程度；稳定性指的是在评估同一视频质量时评分稳定的程度。

在实际应用中，PLCC（Pearson's Linear Correlation Coefficient，Pearson 线性相关系数）和 SROCC（Spearman's Rank Order Correlation Coefficient，Spearman 秩序相关系数）被广泛用于衡量客观质量评估方法的优劣。PLCC 可以衡量算法的准确性，而 SROCC 可以衡量算法的单调性。假设有 N 个失真视频，这 N 个失真视频的主观质量评分均值和客观质量评分均值分别用 $\bar{y}$ 和 $\bar{\hat{y}}$ 表示，第 i 个视频的主观质量评分和客观质量评分分别用 y_i 和 $\hat{y}_i$ 表示，则 PLCC 的计算公式为：

$$PLCC = \frac{\sum_{i=1}^{N}(y_i - \bar{y})\left(\hat{y}_i - \bar{\hat{y}}\right)}{\sqrt{\sum_{i}^{N}(y_i - \bar{y})^2}\sqrt{\sum_{i}^{N}\left(\hat{y}_i - \bar{\hat{y}}\right)^2}} \tag{2.3}$$

第 i 个视频的主观质量评分等级和客观质量评分等级分别用 v_i 和 p_i 表示，则 SROCC 的计算公式为：

$$SROCC = 1 - \frac{6\sum_{i=1}^{N}(v_i - p_i)^2}{N(N^2 - 1)} \tag{2.4}$$

2.2　深度学习

深度学习是机器学习的一个分支，是一种以人工神经网络为架构，对大批量数据进行特征学习的算法 [29]。深度学习起源于对人脑的研究，为了模拟人类的学习方式，深度学习使用了类似人脑的底层架构，以神经元组成的人工神经网络对人脑的信息处理方式进行建模。常见的深度学习网络类型包括 CNN、RNN（Recurrent Neural Network，循环神经网络）、DBN（Deep Belief Network，深度置信网络）等。

在传统的机器学习中，特征提取的工作主要通过人工提取的方式进行，这被称为特征工程。然而，由于特征提取的优劣对算法的性能有至关重要的影响，特征提取者必须具备丰富的本领域专业知识。在深度学习中，特征提取的工作则是由机器自动完成的，完全不依赖于人工，这也使得深度学习的可解释性较差。深度学习运用了分层抽象的思想：在人工神经网络中，每一层的特征都由上一层的特征抽象得到，通过多层抽象，逐渐将初始的低级特征转化为最终的高级特征。一个典型的多层人工神经网络由输入层、隐藏层和输出层三个部分组成，如图 2.2 所示。

深度学习于 2006 年被 Hinton 等人 [30] 提出后，一直受到学术界和工业界的高度关注。最初，深度学习主要应用于图像和语音领域。在图像领域，深度学

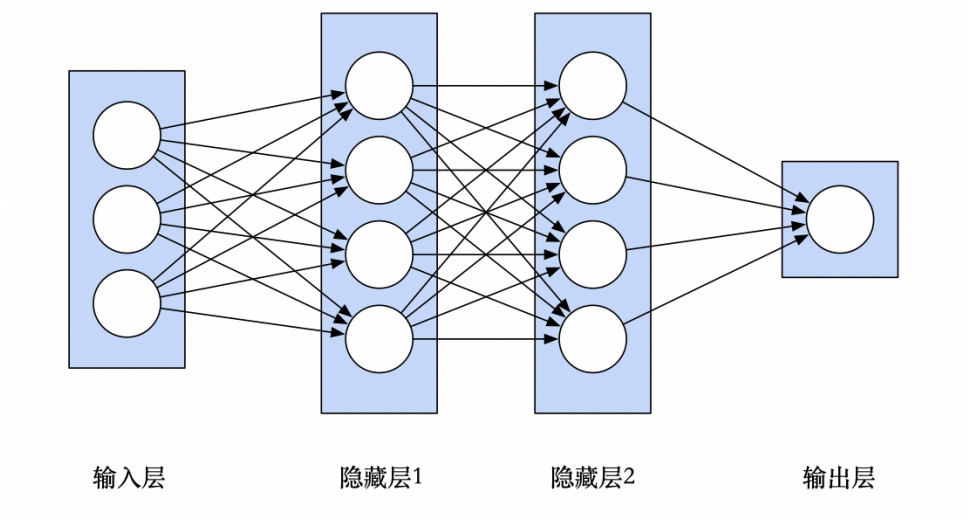


图 2.2: 多层人工神经网络的结构图

习经历了从数字识别到物体识别的重大转变，取得了较多的成果。如今，深度学习已经在计算机视觉、语音处理、自然语言处理、数据挖掘、搜索等领域获得了广泛应用。

2.2.1　CNN

CNN 是一类具有深度结构且包含卷积计算的前馈神经网络 [31]，是深度学习的代表性网络之一，被广泛应用于计算机视觉、自然语言处理等领域。

CNN 源于对人类视觉系统的研究，LeNet-5 [32] 是最早出现的 CNN 之一，用于解决 MNIST [32] 数据集上的数字分类问题。随着硬件能力的不断增强，一些大型的 CNN 开始在计算机视觉领域展现出巨大优势。Krizhevsky 等人 [33] 于 2012 年提出了 AlexNet 网络，并在 ImageNet[2] 图像分类比赛中取得了冠军。随后，一系列新的 CNN 网络结构被提出，并不断更新着 ImageNet 的记录，其中，比较经典的网络结构包括 VGG [34]、GoogLeNet [35] 和 ResNet [36] 等。在 CNN 被广泛用于图像分类问题的同时，一些研究者也尝试将 CNN 推广到计算机视觉的其他领域，并取得了不错的成绩，如语义分割、物体检测、行为识别等。

典型的 CNN 由卷积层、池化层和全连接层构成。卷积层用于提取输入数据的特征，其内部包含多个卷积核。每个卷积核都对应一组固定的权重系数，类似于前馈神经网络中的神经元。通过卷积运算，不同的卷积核可以提取到不同的特征，输出不同的特征图，这与人类视觉系统的特征提取方式类似。卷积层中的

[2] http://www.image-net.org

参数包括卷积核大小、步长和填充值，这些参数也是 CNN 的超参数，它们共同决定了卷积层输出的特征图的大小。在卷积层进行特征提取后，输出的特征图会被传递至池化层进行数据过滤。池化层可以通过非线性池化函数对输入数据进行降采样，其中，最大池化函数是最为常见的非线性池化函数。池化层与卷积层相比，可以更有效地降低数据维度。另外，池化层在大大减少运算量的同时，也可以避免过拟合。全连接层与常规神经网络中的网络层类似，用于综合卷积层和池化层的结果，推理得到最终的预测值。

卷积层通过参数共享和稀疏连接两大特性解决了常规神经网络在处理图像时处理数据量大、训练成本高、无法保留原有特征的难题。

2.2.2　Siamese 网络

Siamese 网络，即孪生网络，因包含了两个完全相同的子网络而得名，通常被用来衡量两个输入数据的相似程度 [37]。Siamese 网络的两个子网络不仅具有相同的超参数，其权重的更新也会同步进行。

Siamese 网络最著名的应用是人脸识别系统，如 DeepFace [38]。在人脸识别系统中，我们可以使用传统的神经网络对不同的人脸进行分类。然而，在人脸识别的场景下，每个类别的样本量往往很少，因此使用这种方式训练的网络难以有好的性能。同时，当我们需要增加或减少某个类别的时候，我们又需要在新的人脸数据集上重新训练网络。Siamese 网络则可以很好地解决上面的问题，Siamese 网络在其他类别的大批量样本上学习如何度量两个输入数据的相似性，并将其推广到待识别的样本类别，适用于待识别类别的样本量很少但其他类别的样本量很多的情况。

三元组损失函数通常用来训练 Siamese 网络，我们分别使用 A（Anchor）、P（Positive）和 N（Negative）表示基准样本、正样本和负样本，对于给定的三元组 (A, P, N)，其损失函数可以表示为：

$$\mathcal{L}(A, P, N) = \max\left(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0\right) \tag{2.5}$$

其中，$\|f(A) - f(P)\|^2$ 表示基准样本和正样本之间的距离，$\|f(A) - f(N)\|^2$ 表示基准样本和负样本之间的距离，α 是一个超参数，用来拉大两者之间的间隔。

2.2.3　LSTM 网络

LSTM 网络 [39] 是一种特殊的 RNN，由 Sepp Hochreiter 和 Jürgen Schmidhuber 于 1997 年首次提出，主要用于解决 RNN 的梯度消失问题。RNN 是深度学习

的代表性网络之一，与常规的神经网络相比，RNN 更善于处理序列数据。普通的 RNN 无法处理随着递归而产生的梯度消失的问题，于是，GRU [11] 和 LSTM 网络被提了出来。它们不仅可以有效地解决梯度消失问题，而且能够捕获长期的依赖。

LSTM 网络于 1997 年提出后，在多个领域得到了广泛应用。2007 年，LSTM 网络开始被大量用于语音识别领域，并在某些语音应用中打败了传统模型 [40]。2009 年，LSTM 网络在 ICDAR[3]手写识别比赛中取得了冠军。

公式 2.6 和图 2.3 展示了 LSTM 网络的前向传播过程：

$$
\begin{aligned}
\Gamma_f^{\langle t\rangle} &= \sigma\left(W_f\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_f\right) \\
\Gamma_u^{\langle t\rangle} &= \sigma\left(W_u\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_u\right) \\
\tilde{c}^{\langle t\rangle} &= \tanh\left(W_c\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_c\right) \\
c^{\langle t\rangle} &= \Gamma_f^{\langle t\rangle} * c^{\langle t-1\rangle} + \Gamma_u^{\langle t\rangle} * \tilde{c}^{\langle t\rangle} \\
\Gamma_o^{\langle t\rangle} &= \sigma\left(W_o\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_o\right) \\
a^{\langle t\rangle} &= \Gamma_o^{\langle t\rangle} * \tanh\left(c^{\langle t\rangle}\right)
\end{aligned}
\tag{2.6}
$$

其中，$\Gamma_f^{\langle t\rangle}$、$\Gamma_u^{\langle t\rangle}$ 和 $\Gamma_o^{\langle t\rangle}$ 分别代表时间 t 处的遗忘门、更新门和输出门，W_f、W_u 和 W_o 分别为它们对应的权重，b_f、b_u 和 b_o 分别为它们对应的偏差。$x^{\langle t\rangle}$ 代表时间 t 处的输入值，$c^{\langle t\rangle}$ 代表时间 t 处的记忆细胞值，$a^{\langle t\rangle}$ 代表时间 t 处的激活值，$\tilde{c}^{\langle t\rangle}$ 为 $c^{\langle t\rangle}$ 在时间 t 处的候选值，$\tilde{c}^{\langle t\rangle}$ 对应的权重和偏差分别为 W_c 和 b_c。从公式可以看出，LSTM 网络通过遗忘门和更新门控制当前记忆细胞值的更新与否，从而捕获长期的依赖。

2.2.4　迁移学习

迁移学习是机器学习中的一个重要分支，指的是将在一个任务中学习到的知识迁移到另一个任务中。 [41]

在深度学习中，迁移学习被广泛应用于计算机视觉、自然语言处理等领域。fine-tuning（微调）是深度学习中最简单的迁移学习技术。在实际应用中，从头训练一个神经网络的代价非常巨大，所以针对一个新的任务，我们通常不会从头训练一个网络，而是将在其他相似任务上已经训练好的网络迁移过来，再针对自己的任务对网络进行调整。这个调整的过程称为微调，被迁移过来的网络称为预训练网络。一般而言，当任务 A 的数据比任务 B 多得多，并且从任务 A

[3]https://rrc.cvc.uab.es

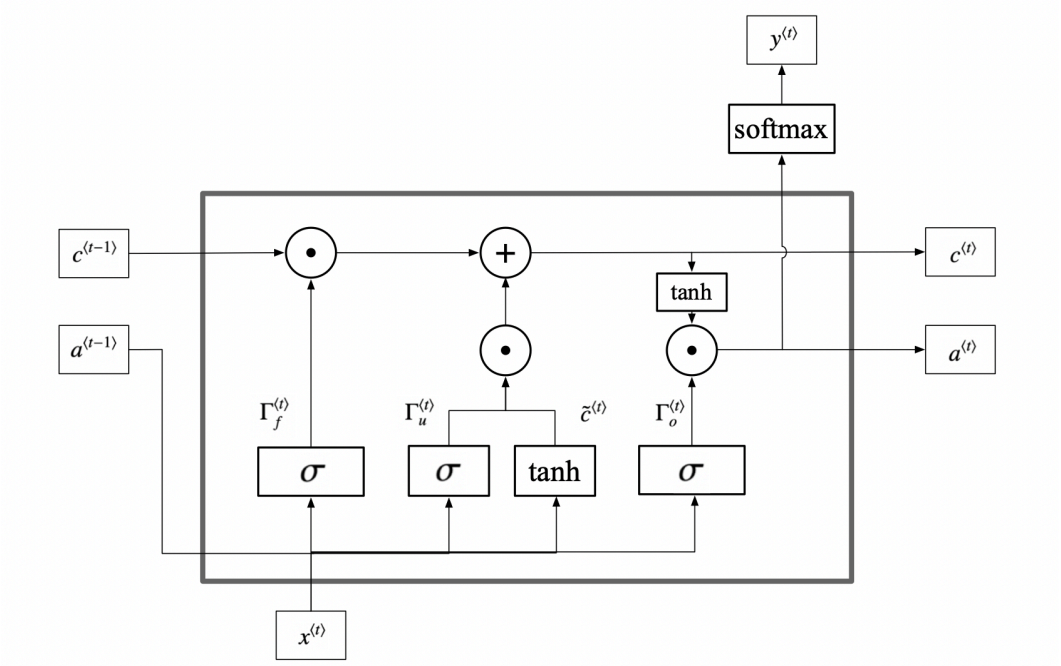


图 2.3: LSTM 网络的前向传播过程图

中学习到的低层特征可以帮助到任务 B 的学习时，我们就会将在任务 A 中训练得到的网络迁移到任务 B 中。

2.3　Docker 技术

Docker[4]是一个开源的应用容器引擎，于 2013 年被提出，主要用于实现应用的快速分发和部署。

在软件开发中，经常会出现一个应用可以在当前机器上正常运行，却无法在其他机器上运行的情况，这是因为应用的正常运行往往需要依赖于特定的操作系统和软件库。因此，在软件的部署过程中，软件开发和运维人员需要分别配置每个机器的环境，以确保每台机器上都安装了正确的操作系统和软件库，这会花费他们大量的时间和精力。容器技术则很好地解决了这个问题，它允许软件开发和运维人员将应用及应用的所有依赖打包到一个轻量级、可移植的容器文件里，之后在待部署的机器上运行这个文件，机器上就会生成一个虚拟容器，应用在这个虚拟容器里运行。这样，就实现了软件的快速部署。

容器作为一种轻量级的虚拟化方式，与传统的虚拟机相比，有着明显的优

[4]https://www.docker.com

势：

（1）传统的虚拟机是操作系统级别的程序，需要占用大量的内存和硬盘资源；容器则是进程级别的程序，只占用少量的资源，且多个容器可以共享资源。

（2）启动传统的虚拟机相当于启动一个操作系统，需要几分钟的时间；启动容器则相当于启动一个进程，只需要几秒钟的时间。

（3）在使用传统的虚拟机时，往往存在着一些无法跳过的冗余步骤，如用户登录；在使用容器时，则没有这些冗余步骤。

Docker 对 Linux 容器进行了封装，并提供了简单易用的容器使用接口，是目前使用最为广泛的 Linux 容器解决方案。通过使用 Docker，可以实现应用的快速迭代。开发人员可以构建一套标准的开发环境，而测试和运维人员也可以直接使用这套环境进行代码的测试和部署，节约大量的时间。通过使用 Docker，还可以组建微服务架构。我们可以在一台机器上同时运行多个 Docker 服务，从而轻松模拟出微服务架构。通过使用 Docker，也可以提供弹性的云服务。

image（镜像）和 container（容器）是 Docker 中的两个重要概念，image 可以看做是 container 的模板，Docker 把应用及应用的所有依赖打包到 image 文件里面，并根据 image 文件生成多个 container 实例。

2.4　Flask 框架

Flask[5]是一个使用 Python 开发的微型 Web 框架。与 Django[6]等其他同类型的框架相比，它更加灵活、轻便、安全。作为一个轻量级的 Web 框架，它只保留了两个核心的功能，即请求响应和模板渲染，这两个功能分别由 Werkzeug WSGI 工具库和 Jinja2 模板渲染引擎完成。另外，Flask 在保证核心功能简单易用的同时，又有很强的定制性，其丰富的扩展库可以使用户根据自己的需求添加相应的功能，开发出更加强大的网站。

Flask 会在每个程序里为每个 URL 分配一个视图函数，当收到用户对某个 URL 的请求时，Flask 就会执行为这个 URL 分配的视图函数，并将这个函数的返回值做为请求的响应返回给用户，这就是 Flask 的工作模式。

除了 Flask，目前使用较为广泛的 Python 框架还有 Django、Tornado[7]等。Django 作为一个重量级的框架，多被用于大型网站的开发，而对于大多数小型网站的开发，使用 Flask 即可满足。

[5]https://palletsprojects.com/p/flask

[6]https://www.djangoproject.com

[7]https://www.tornadoweb.org

2.5　Vue 框架

Vue[8]是一个用于构建用户界面的开源渐进式 JavaScript 框架，旨在更好地组织和简化前端的开发。Vue 采用自底向上增量开发的设计，它只关注 MVC（Model–View–Controller，模型-视图-控制器）模式中的视图层，同时，它也能通过内部特定的方法实现视图与模型的交互，方便地获取数据的更新。Vue 不仅容易上手，还非常便于与既有项目或第三方库整合。

组件是 Vue 中最为强大的一个特性。在 Vue 中，组件是基础 HTML 元素的扩展，我们可以方便地为其定义数据和行为。为了更好地管理应用程序，我们往往需要将应用切割为小而具有复用性的组件。

另外，Vue 还使用了基于 HTML 的模板语法，我们可以将 Vue 实例中的数据与 DOM 元素进行绑定。由于 Vue 模板是合法的 HTML，所以能够被遵循规范的浏览器解析。当应用状态发生改变时，Vue 能够智能地计算出重新渲染组件的最小代价并应用到 DOM 操作上。

Vue 是目前前端的主流框架之一，除此之外，还有 Angular[9]、React[10]等。

2.6　本章小结

本章我们对本文中涉及的重要概念与主要技术进行了概述。首先，我们介绍了三类视频质量评估算法：全参考视频质量评估算法、部分参考视频质量评估算法和无参考视频质量评估算法。其次，我们介绍了视频质量评估算法的评价指标。再次，我们介绍了深度学习的相关技术，包括 CNN、Siamese 网络、LSTM 网络和迁移学习。最后，我们介绍了 Docker 技术、Flask 框架和 Vue 框架。

[8]https://cn.vuejs.org

[9]https://angular.io

[10]https://reactjs.org/

第三章　UGC 视频质量评估算法设计

3.1　整体概述

本文提出了一种基于深度学习的 UGC 视频质量评估算法。在这个算法中，CNN 和 LSTM 网络被分别用于捕获 UGC 视频的空域失真信息和时域失真信息。我们使用这个算法，在包含了 1200 个失真视频的 KoNViD-1k [1]UGC 视频质量数据集上取得了优于其他算法的性能。

在深度学习领域，CNN 通常被用来提取图像特征。所以，在这个算法中，我们使用 CNN 提取 UGC 视频帧中与视频空域失真相关的深度特征，从而将待评估的 UGC 视频建模为帧级深度特征序列。之后，我们将帧级深度特征序列输入到 LSTM 网络中，并对其进行训练，以使其能够预测 UGC 视频的最终质量。LSTM 网络在处理序列数据时能够捕获序列之间的长期依赖关系，因此，我们使用 LSTM 网络学习 UGC 视频的时域失真。

此外，这个算法的独特之处在于，我们没有借助在图像分类任务上训练过的 CNN 提取 UGC 视频帧的深度特征，而是使用了一种基于图像相对质量的特征提取方法。在以往的工作中，有些研究者直接使用在图像分类任务上训练过的 CNN 提取 UGC 视频帧的深度特征。另外，还有一些研究者使用微调后的网络提取 UGC 视频帧的深度特征，他们将在图像分类任务上训练过的 CNN 迁移过来，并基于 UGC 视频质量数据集对模型进行微调。前者提取的深度特征中包含了大量与视频质量无关的信息，而后者虽然提取了与视频质量相关的深度特征，但由于缺乏大规模的视频质量数据集，提取效果往往较差。为解决数据集缺乏的问题，我们在提取 UGC 视频帧的深度特征时，基于大量具有不同类型、不同程度失真的图像训练了一个能够比较图像相对质量的 Siamese 网络，并将它的一个子网络作为预训练网络，在 UGC 视频质量数据集上进行微调。对于训练 Siamese 网络的失真图像，我们并不需要对其质量分数进行人工标注，而只需要知道它们的相对质量，这就使得我们可以生成足够的失真图像，在不产生过拟合的情况下训练一个层次更深、含有更多参数的网络。虽然我们训练的 Siamese 网络只能比较图像的相对质量，但 Siamese 网络中学习到的低层特征可以帮助我们更好地提取出 UGC 视频帧中与视频质量相关的深度特征。

综上所述，我们的方法可以分为图像相对质量学习、UGC 视频深度特征提取、UGC 视频质量评估三个部分，下面我们将详细地讨论每个部分的设计与实现细节。图 3.1 展示了我们这个方法的大致工作流程。

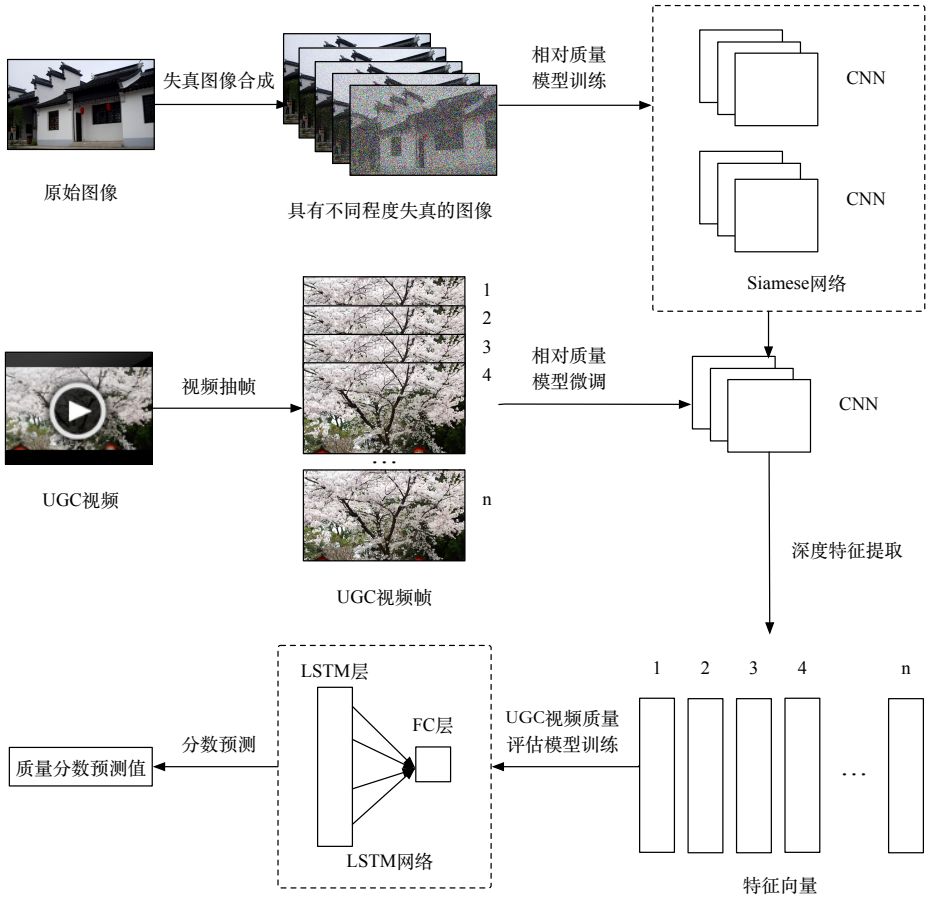


图 3.1: 算法的工作流程图

3.2　图像相对质量学习

由于人工标注视频质量成本巨大，目前的 UGC 视频质量数据集，规模普遍较小。如果我们仅使用 UGC 视频质量数据集训练神经网络，则无法获得很好的深度特征提取效果。于是，我们通过人工合成的方式构建了一个大规模的无标签失真图像数据集，并在这个数据集上训练了一个能够比较图像相对质量的 Siamese 网络。将这个 Siamese 网络的子网络作为预训练网络，我们就可以更好地提取 UGC 视频帧中与视频质量相关的深度特征。

在我们构造的无标签数据集中，每个图像都具有不同类型、不同程度的失真。虽然我们不能确定数据集中图像的绝对质量分数，但对于任意两个图像，我们可以知道它们的相对质量，即知道哪一个图像的质量更高，哪一个图像的质量更低。基于上述思想，我们在这个数据集上训练了一个 Siamese 网络，使其对图像的相对质量进行学习。因为数据集中的数据无需人工标注，所以我们可以构造出比现有数据集多得多的数据，这就使得我们可以在不产生过拟合的情况

下训练一个层次更深的网络，也因此在后续的深度特征提取过程中，相比其他算法获得更好的性能。

3.2.1　失真图像合成

为训练 Siamese 网络，我们基于 Waterloo [42] 数据集生成了大量具有不同类型、不同程度失真的图像，并由此构建了一个大规模的无标签失真图像数据集。由于我们并不需要对数据集中的失真图像进行人工标注，我们构建的数据集比其他已有的图像或视频质量数据集要大得多。

Waterloo 数据集由 Ma 等人构建，由 4744 个在网络上精心挑选的高质量自然图像组成，图像的内容非常丰富，涵盖了人、动物、植物、自然景观、城市景观、静物和交通。除高质量图像外，他们还给出了生成相应失真图像的方法。他们没有对图像质量进行人工标注，而是提供了三种替代的图像质量评估标准。

我们遵循 Waterloo 数据集的方法，以其中的高质量图像为原始图像，生成了四种类型的失真图像，这四种失真类型相比其他失真类型更为广泛和普遍，它们分别是高斯白噪声、高斯模糊、JPEG 压缩、JPEG2000 压缩。同时，对于每种失真，我们都生成了具有五种不同程度失真的图像。因此，我们共生成了 94880 个失真图像，其规模远远大于其他现有的图像或视频质量数据集。

所有的失真图像都通过 MATLAB[1]生成，每种失真的参数分别如下所示：

（1）高斯白噪声：噪声的方差分别被设为 0.001、0.006、0.022、0.088 和 1.000。

（2）高斯模糊：二维圆形对称高斯模糊核的标准偏差分别被设为 1.2、2.5、6.5、15.2 和 33.2。

（3）JPEG 压缩：参数化 DCT（Discrete Cosine Transform，离散余弦变换）量化矩阵的质量因子分别被设置为 43、12、7、4 和 0。

（4）JPEG2000 压缩：压缩比分别被设为 52、150、343、600 和 1200。

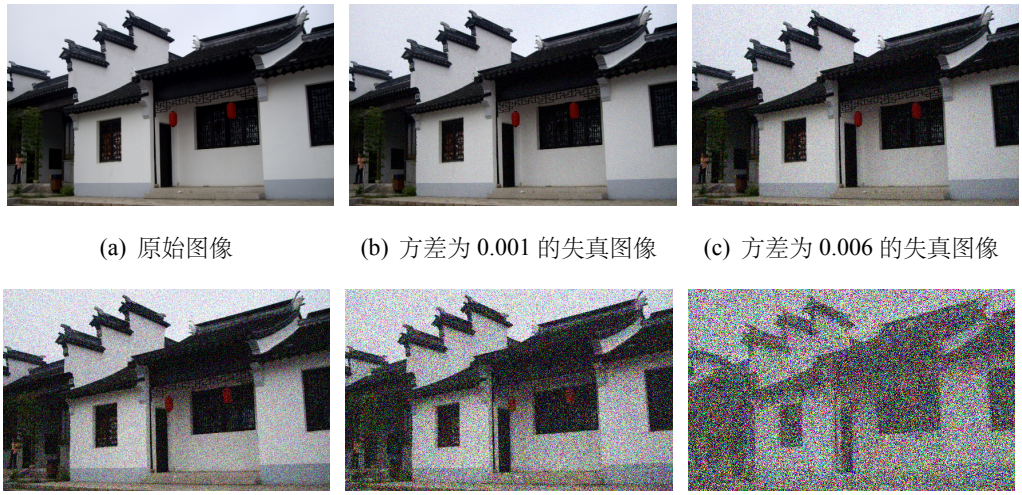
图 3.2 是一个原始图像与其对应的高斯白噪声失真图像的例子。图 3.2(a) 是一个高质量的原始图像，图 3.2(b) 、图 3.2(c) 、图 3.2(d) 、图 3.2(e) 和图 3.2(f) 分别在其基础上添加了方差为 0.001、0.006、0.022、0.088 和 1.000 的高斯白噪声。

3.2.2　图像相对质量模型训练

（1）总体方法

以图 3.2 中的原始图像和失真图像为例，虽然这些图像没有经过人工标注，我们无法确定它们的绝对质量分数，但我们可以知道它们的相对质量。对于任

[1]https://www.mathworks.com



(a) 原始图像　(b) 方差为 0.001 的失真图像　(c) 方差为 0.006 的失真图像

(d) 方差为 0.022 的失真图像　(e) 方差为 0.088 的失真图像　(f) 方差为 1.000 的失真图像

图 3.2: 一个原始图像与其对应的高斯白噪声失真图像

意的两个图像，我们可以知道哪一个图像的质量更高，哪一个图像的质量更低。图 3.2 中原始图像的质量一定高于其他失真图像的质量。同时，每一个失真图像的质量也一定高于其后面任意一个失真图像的质量，因为后面的图像被添加了更大方差的高斯白噪声，失真程度更高。

基于上述思想，我们以构建的大规模失真图像数据集为实验数据集，训练了一个能够判断图像相对质量的 Siamese 网络。这个 Siamese 网络由两个完全相同的 CNN 组成，因为在训练的过程中两个子网络的参数同步更新，所以这两个子网络始终保持一致。训练时成对的图像被输入到 Siamese 网络的这两个子网络中，我们分别用 x_i 和 x_j 表示输入到网络中的两个图像，分别用 $\hat{y}_i$ 和 $\hat{y}_j$ 表示这两个图像在子网络中的输出，也即它们的相对质量预测值。假设 x_i 的质量高于 x_j 的质量，为了使 Siamese 网络学习图像的相对质量，我们将其损失函数定义为公式 3.1 。

$$L(x_i, x_j) = g(\hat{y}_i, \hat{y}_j) = \max(0, \hat{y}_j - \hat{y}_i + \varepsilon) \tag{3.1}$$

其中 ε 是一个正常数，用来增大 $\hat{y}_i$ 和 $\hat{y}_j$ 的间隔。由上述公式可以看出，当 $\hat{y}_j - \hat{y}_i + \varepsilon \geq 0$ 时，损失为 $\hat{y}_j - \hat{y}_i + \varepsilon$ 自身，也即一个正数；当 $\hat{y}_j - \hat{y}_i + \varepsilon < 0$，损失为 0。这样的损失函数使得 Siamese 网络不断调整网络参数，以减小 $\hat{y}_j - \hat{y}_i + \varepsilon$ 的值，即增大 $\hat{y}_i$ 的值，减小 $\hat{y}_j$ 的值，直到 $\hat{y}_j - \hat{y}_i + \varepsilon < 0$。

需要注意的是，Siamese 网络并不能预测图像的绝对质量。当一个图像的预测值大于另一个图像时，仅表示这个图像的质量高于另一个图像，而不表示这

个图像的质量比另一个图像高对应的数值。另外，我们不会将两个不同失真类型的图像同时输入到 Siamese 网络中。因此，不同失真类型图像的预测值不具备可比性。

（2）具体实现

在具体搭建和训练 Siamese 网络时，我们使用了 Caffe[2]深度学习框架。Caffe 是一个使用 C++ 编写的高效深度学习框架，带有 Python 接口，它的一个重要特点是可以在不编写任何代码的情况下训练和部署模型。

a. Siamese 网络的实现

我们并没有使用 Caffe 官方提供的方法实现 Siamese 网络，而是使用了一种更为高效的方法。

官方的 Siamese 网络实现方法存在很多的冗余计算。假设每次迭代都有 n 个数据被输入到网络中，则对于其他的常规网络，总计算次数为 n，这是因为每个输入数据都只会被计算一次。而对于 Siamese 网络，总计算次数为 $n(n-1)$，这是因为输入到网络中的数据会以两两组合的形式进行计算。在 n 个输入数据中，每个输入数据都可以与其他 $n-1$ 个数据进行组合，所以每个输入数据都会被计算 $n-1$ 次，总的计算次数就为 $n(n-1)$。

我们使用了一种仅需要 n 次计算的 Siamese 网络实现方法。当数据被输入到网络中后，我们先不对其进行两两组合，而是同其他的常规网络一样，分别对每个输入数据进行一次计算。之后，在计算总的损失时，我们再对所有的输入数据进行两两组合，并分别计算每个组合产生的损失。这样，对于每次迭代，我们都将输入数据的计算次数减少为了之前的 $n-1$ 倍，从而大大缩短了 Siamese 网络的训练时间，提高了效率。

因此，在编写 Siamese 网络的配置文件时，我们只使用了一个 CNN，同时还加入了一个自定义的损失层。在每次迭代中，所有输入图像都会被输入到 CNN 中进行一次计算。之后，同种失真图像的计算结果，也即它们的相对质量预测值，会被输入到损失层中两两组合，并分别按照公式 3.1 计算损失。这些损失的平均值即每种失真图像的总损失。假设每次迭代同种失真的输入图像有 n 个，第 i 个失真图像的相对质量预测值为 $\hat{y}_i$，则每种失真图像的总损失就如公式 3.2 所示。

$$J = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j>i}^{n} g(\hat{y}_i, \hat{y}_j) \tag{3.2}$$

[2]https://caffe.berkeleyvision.org

在实现自定义的损失层时，不仅需要计算所有输入数据的总损失，而且需要计算总损失对于每个预测值的偏导数。在我们的损失层中，每种失真图像的总损失 J 对于每个相对质量预测值 $\hat{y}_i$ 的偏导数可以用公式 3.3 表示。

$$\frac{\partial J}{\partial \hat{y}_i} = \frac{2}{n(n-1)} \sum_{j=1}^{n} \frac{\partial g(\hat{y}_i, \hat{y}_j)}{\partial \hat{y}_i} \quad (j \neq i) \tag{3.3}$$

根据 $g(\hat{y}_i, \hat{y}_j)$ 在公式 3.1 中的定义，其对 $\hat{y}_i$ 的偏导数可以表示为公式 3.4 。当 x_i 的相对质量大于 x_j 时，a 为-1；当 x_i 的相对质量小于 x_j 时，a 为 1。

$$\frac{\partial g(\hat{y}_i, \hat{y}_j)}{\partial \hat{y}_i} = \begin{cases} 0 & \text{if } a(\hat{y}_i - \hat{y}_j) + \varepsilon < 0 \\ a & \text{otherwise} \end{cases} \quad (j \neq i) \tag{3.4}$$

综上所述，在每次迭代的前向传播中，自定义的损失层接收来自 CNN 的输入，并按照公式 3.2 计算出每种失真图像的总损失。在每次迭代的反向传播中，这个损失层又按照公式 3.3 计算出每种失真图像的总损失对于每一个输入的偏导数，并将这些计算出的偏导数传回到 CNN 中。

这个自定义的损失层使用 Python 语言实现，其前向传播过程如算法 1 所示。

Algorithm 1: 自定义损失层的前向传播过程

Data: *bottom* (data from last layer) , *top* (data to next layer)

```
loss = [];
count = 0;
for type = 0 ; type < type_num ; type++ do
    base = type * level_num;
    for high = 0 ; high < level_num ; high++ do
        for low = high + 1 ; low < level_num ; low++ do
            loss[count] =
              (bottom[0].data[base + high] − bottom[0].data[base + low]);
            count = count + 1;
        end
    end
end
loss = max(0, margin − loss);
top[0].data[...] = sum(loss)/count;
```

在算法 1 中，bottom 为与损失层输入数据相关的数据，top 为与损失层输出数据相关的数据。同时，损失层的输入数据即 CNN 的输出数据。在每次迭代中，我们只针对一个原始图像及其所有的失真图像进行处理。我们会先按照图像质量由高到低的顺序读取某个原始图像产生的一种失真类型的数据，之后再按照同样的顺序读取这个原始图像产生的其他失真类型的数据。这些输入数据会按照同样的顺序在网络中进行传播，因此，bottom 中的相对质量预测值也保持了与输入顺序相同的顺序。损失层基于 bottom 中的预测值计算出每次迭代的总损失，并将总损失保存在 top[0] 的 data 变量中。

自定义损失层的反向传播过程如算法 2 所示。在这个算法中，bottom 和 top 的含义与在算法 1 中相同。算法 2 的 loss 数组也是由算法 1 计算出来的，它按序存储了同种失真图像中每一个组合的损失。损失层基于这个数组计算出每种失真图像的总损失对于每个相对质量预测值的偏导数，并将其保存在 bottom[0] 的 diff 变量中。

Algorithm 2: 自定义损失层的反向传播过程

```
Data: bottom (data from last layer) , top (data to next layer)
count = 0;
derivative = zeros(bottom[0].num);
for type = 0 ; type < type_num ; type++ do
    base = type * level_num;
    for high = 0 ; high < level_num ; high++ do
        for low = high + 1 ; low < level_num ; low++ do
            if loss[count] ≥ 0 then
                derivative[base + high] = derivative[base + high] − 1;
                derivative[base + low] = derivative[base + low] + 1;
            end
            count = count + 1;
        end
    end
end
bottom[0].diff[...] = derivative * type_num/count;
```

b. CNN 的选择

因为实验数据集中的数据足以训练一个较深的网络，所以我们选用 VGG16 [34] 作为 Siamese 网络的子网络。VGG 是由 Simonyan 和 Zisserman 提出的深度神经

网络，在 2014 年的 ImageNet[3]图像定位与分类比赛中 VGG 分别取得了第一名和第二名的成绩。VGG 有 VGG16 和 VGG19 两种典型的网络结构，其中，VGG16 共由 13 个卷积层和 3 个全连接层组成。在用于图像分类任务的 VGG16 中，输出层包含了 1000 个神经元，这 1000 个神经元对应了图像分类的 1000 个类别。在我们的应用场景中，我们期望 Siamese 网络的子网络仅有一个输出，即输入图像的相对质量预测值。因此，我们将 VGG16 的最后一层删除，并使用一个仅含一个神经元的全连接层代替它。

在搭建和训练网络的过程中，我们发现使用在 ImageNet 图像分类任务上预训练的网络可以提高网络性能。所以，除最后一层外，我们将 Siamese 网络各层的参数都初始化为预训练网络各层的参数值。

c. 参数的设置

表 3.1 列出了训练 Siamese 网络时我们设置的参数。在训练 Siamese 网络时，我们使用了带有动量的 SGD（Stochastic Gradient Descent，随机梯度下降法）算法优化网络参数，动量的引入使我们的优化过程更加稳定和迅速。在迭代的过程中，我们还对学习率进行了调整。我们使用了等步长的学习率调整方法，每迭代 10^4 次，我们将学习率调整为之前的 10^{-1} 倍。另外，权重衰减项被用来减少过拟合。

表 3.1: 训练 Siamese 网络的参数表

参数	参数值
优化算法	SGD
基础学习率	10^{-5}
学习率调整策略	等步长
学习率调整因子	10^{-1}
学习率调整步长	10^4
动量	9×10^{-1}
权重衰减项	5×10^{-4}
总迭代次数	2×10^4

d. 实验结果的分析

在训练 Siamese 网络时，我们随机地将无标签失真图像数据集中的数据划分到训练数据集和测试数据集中，使得训练数据集和测试数据集的比例为 8:2。同时，由同一原始图像生成的失真图像不会同时出现在两个数据集中。

[3]http://www.image-net.org

因为 VGG 以分辨率为 224×224 的图像为输入数据，而 Waterloo 数据集中的图像又没有固定的分辨率，所以在输入数据时，我们将训练数据集和测试数据集中的图像剪裁为 224×224 的小块，每个小块在图像中的位置随机。为了保持图像的原有质量，不引入新的失真，我们没有对图像进行缩放。在训练 Siamese 网络的时候，我们只从每个训练数据中生成一个小块输入到网络中。在测试 Siamese 网络的时候，我们从每个测试数据中随机生成 30 个小块，这些小块的平均相对质量预测值即整个图像的相对质量预测值。

图 3.3 描绘了 Siamese 网络的训练过程中，训练数据集的总损失和测试数据集的总损失随迭代次数的变化情况，深蓝色曲线为训练数据集的损失变化曲线，浅蓝色曲线为测试数据集的损失变化曲线。我们共进行了 20000 次迭代，随着迭代次数的增加，训练数据集的总损失和测试数据集的总损失逐渐趋向于 0。

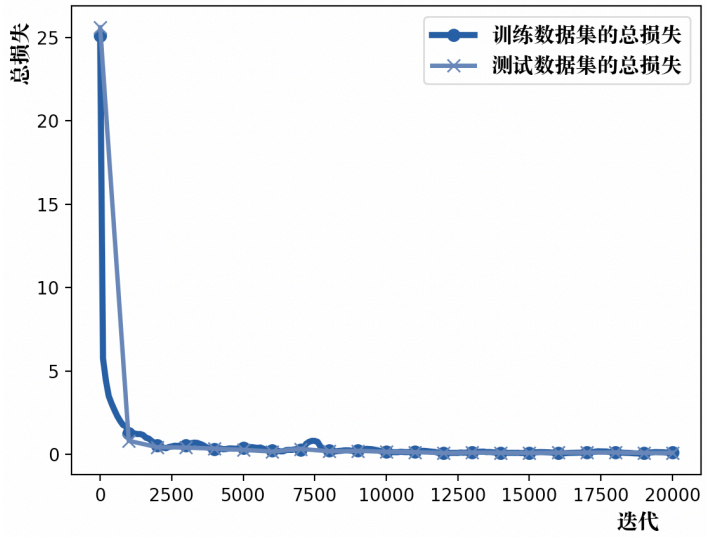


图 3.3: Siamese 网络训练过程中的损失变化曲线

在测试 Siamese 网络时，我们使用训练好的 Siamese 网络预测测试数据集中每个失真图像的相对质量。图 3.4 展示了 Siamese 网络在所有高斯白噪声失真图像上的预测结果。为方便表示，我们按照从低到高的顺序将失真图像的相对质量预测值划分为了 20 个等级，每个等级中包含的失真图像数量如图所示。除此之外，我们还使用不同的颜色对图像的失真程度进行了标注。图像的失真程度越高，则使用的颜色越深。从图中可以清晰地看出，失真程度较高的图像，其相对质量预测值也较低，而失真程度较低的图像，其相对质量预测值也较高。这表明了我们的 Siamese 网络可以很好地分辨图像的相对质量。

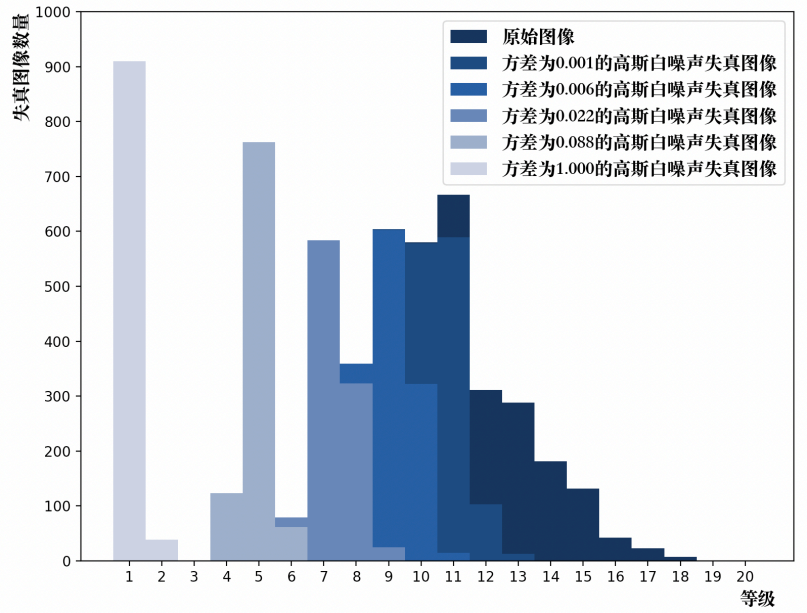


图 3.4: Siamese 网络在高斯白噪声失真图像上的预测结果

3.3　UGC 视频深度特征提取

在 3.2 中，我们基于大规模的无标签失真图像数据集，训练了一个能够分辨图像相对质量的 Siamese 网络。虽然这个 Siamese 网络不能预测图像的绝对质量分数，但这个 Siamese 网络学习到了很多与图像质量相关的特征，尤其是低层特征。因此，在提取 UGC 视频帧中与视频质量相关的深度特征时，我们将这个 Siamese 网络的一个子网络作为预训练网络，在 UGC 视频质量数据集上对其进行微调。我们先按照一定的采样率对 UGC 视频进行抽帧，再使用抽取到的视频帧对 Siamese 网络进行校正。我们认为，UGC 视频帧的质量与原始 UGC 视频的质量有一定的相关性。所以，在微调时，每一个 UGC 视频帧都继承了原始 UGC 视频的质量分数。微调完成后，我们就通过这个微调后的网络提取 UGC 视频帧中与视频质量相关的深度特征。

3.3.1　UGC 视频抽帧

在本文提出的 UGC 视频质量评估算法中，KoNViD-1k 数据集被作为实验数据集，训练和测试我们的网络。KoNViD-1k 数据集是一个包含了 1200 个失真视频的 UGC 视频质量数据集，其中的每个视频都是分辨率为 960×540 的公开视频。视频的时长大约为 8 秒，每秒有 30 帧。这个数据集通过在线众包的方式收集受测者对失真视频的主观质量评分，评分的分值为从 1 至 5 的连续值。

为使用 KoNViD-1k 数据集训练和测试我们的网络，我们将其中的 1200 个

失真视频随机地划分到训练数据集和测试数据集中，使得训练数据集和测试数据集的比例为 8:2。划分完成后，我们分别对训练数据集和测试数据集中的视频进行抽帧。

在抽帧时，我们使用了 FFmpeg[4]音视频处理工具。FFmpeg 是一个集编码、解码、转码、过滤、播放等功能为一体的多媒体框架，它几乎支持所有的音视频格式，功能强大，用途广泛。FFmpeg 是基于 Linux 平台开发的，包含许多库文件。通常，我们以命令行的形式使用它。下面是我们在抽帧时使用的 FFmpeg 命令。

```
> ffmpeg -i [ ].mp4 -vf fps=10 [ ]_%d.png
```

通过这条命令我们可以按照每秒 10 帧的频率从 mp4 格式的输入文件中抽取视频帧，保存到 png 格式的输出文件中。-i 参数指定了输入文件，-vf 参数指定了抽帧的频率。因为 KoNViD-1k 数据集中视频的时长大约为 8 秒，所以针对每个失真视频，我们都抽取了 80 帧。对于时长略小于 8 秒的失真视频，我们使用抽取的最后一帧补齐。

3.3.2　图像相对质量模型微调

（1）总体方法

前面我们训练了一个能够分辨图像相对质量的 Siamese 网络。虽然这个 Siamese 网络只能判断同种失真图像的相对质量，但在训练的过程中，这个网络很好地学习了与图像质量相关的特征。所以，借助于 Siamese 网络的子网络，我们可以更好地提取 UGC 视频帧中与视频质量相关的深度特征。不过，在最终提取每个 UGC 视频帧的深度特征之前，我们还需要使用抽取到的 UGC 视频帧对这个网络进行微调，使这个网络与我们的 UGC 视频质量数据集更加契合。因为在某种程度上每个 UGC 视频帧的质量与原始 UGC 视频的质量有关，所以在对这个网络进行微调时，我们将原始 UGC 视频的质量分数作为每个 UGC 视频帧的质量分数。

（2）具体实现

与训练 Siamese 网络时一样，我们也使用了 Caffe 深度学习框架。

a. 网络的搭建

在微调 Siamese 网络的子网络时，我们没有对原有的网络结构进行扩展或修改，而是完全继承了原有的网络结构。除了最后一层外，网络的其余各层都被

[4]https://ffmpeg.org

初始化为 Siamese 网络的参数值。在损失函数的选择上，我们使用了 Caffe 自带的欧式损失函数，也即平方和损失函数。假设每次迭代的输入数据中都有 n 个 UGC 视频帧，第 i 个 UGC 视频帧的质量分数真值和质量分数预测值分别为 y_i 和 $\hat{y}_i$，则每次迭代的总损失就如公式 3.5 所示。

$$J = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

b. 参数的设置

表 3.2 列出了微调 Siamese 网络的子网络时我们设置的参数。在微调 Siamese 网络的子网络时，我们同样也使用了带有动量的 SGD 算法优化网络参数，动量的引入使我们的优化过程更加稳定和迅速。在迭代的过程中，我们同样也对学习率进行了调整。我们使用了等步长的学习率调整方法，每迭代 5×10^3 次，我们将学习率调整为之前的 5×10^{-1} 倍。另外，权重衰减项被用来减少过拟合。

表 3.2: 微调 Siamese 网络的参数表

参数	参数值
优化算法	SGD
基础学习率	10^{-5}
学习率调整策略	等步长
学习率调整因子	5×10^{-1}
学习率调整步长	5×10^{3}
动量	9×10^{-1}
权重衰减项	5×10^{-4}
总迭代次数	10^{4}

c. 实验结果的分析

对于每个训练数据集中的失真视频，我们都从其抽取的 UGC 视频帧中等间隔地选取 8 个 UGC 视频帧。之后，我们将这些选取的 UGC 视频帧以及它们的质量分数输入到 Siamese 网络的一个子网络中，对其进行微调，每个输入的 UGC 视频帧都继承了原始 UGC 视频的质量分数。因为训练数据集共有 960 个失真视频，所以我们共使用了 7680 个 UGC 视频帧校正 Siamese 网络的子网络。

因为 VGG 以分辨率为 224×224 的图像为输入数据，所以在微调的时候，我们从每个 UGC 视频帧中生成一个小块输入到 Siamese 网络的子网络中。为了保持 UGC 视频帧的原有质量，不引入新的失真，我们同样也没有对训练数据集中

的 UGC 视频帧进行缩放，而是将其剪裁为 224×224 的小块，每个小块在 UGC 视频帧中的位置随机。

图 3.5 描绘了 Siamese 网络的微调过程中，训练数据集的总损失和测试数据集的总损失随迭代次数的变化情况，深蓝色曲线为训练数据集的损失变化曲线，浅蓝色曲线为测试数据集的损失变化曲线。我们共进行了 10000 次迭代，随着迭代次数的增加，训练数据集的总损失和测试数据集的总损失逐渐趋向于 0。

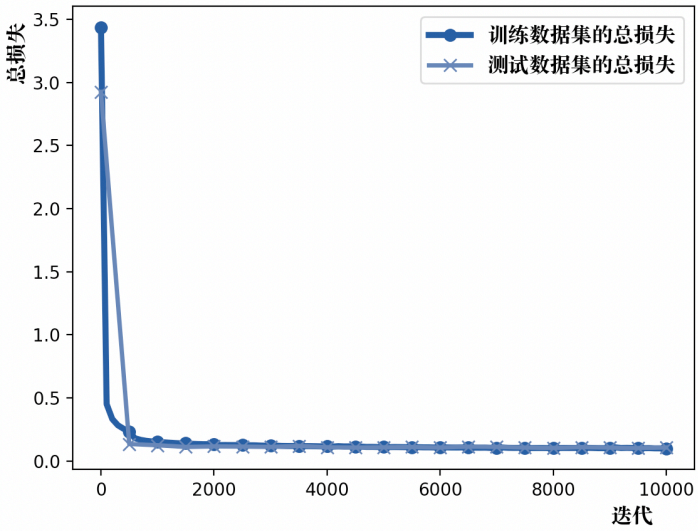


图 3.5: Siamese 网络微调过程中的损失变化曲线

3.3.3　UGC 视频深度特征提取

微调完成后，我们使用微调后的网络提取 UGC 视频帧中与视频质量相关的深度特征。通过提取每个 UGC 视频帧的深度特征，我们将 UGC 视频都建模为一个帧级深度特征序列。

对于训练数据集和测试数据集中的每一个失真视频，我们都提取了深度特征。在具体提取特征时，我们将抽取的 UGC 视频帧输入到微调后的网络中进行一次前向传播，UGC 视频帧在网络倒数第二层的输出被作为它的深度特征。因为 VGG16 的倒数第二层输出一个长度为 4096 的特征向量，所以对于每个 UGC 视频，我们都提取了 80 个长度为 4096 的深度特征。同时，由于 VGG 以分辨率为 224×224 的图像为输入数据，我们把 UGC 视频帧剪裁成了 224×224 的小块。对于每个 UGC 视频帧，我们随机地生成 30 个小块输入到网络中，这 30 个小块的平均深度特征被作为整个 UGC 视频帧的深度特征。

我们使用了 Caffe 的 Python 接口提取 UGC 视频帧的深度特征，其核心算法如算法 3 所示。

Algorithm 3: UGC 视频帧的深度特征提取

Input: $frame\_names$ (list of frame name)

Output: $features$ (list of deep feature)

```
features = zeros(frame_num, feature_length);
for f = 0 ; f < frame_num ; f++ do
    frame_name = frame_names[f];
    frame = read(frame_name);
    x = frame.shape[0];
    y = frame.shape[1];
    for p = 0 ; p < patch_num ; p++ do
        x_start = rand_int(x − 224);
        y_start = rand_int(y − 224);
        patch = frame[x_start : x_start + 224, y_start : y_start + 224, :];
        net.forward_all(patch);
        features[f] = features[f] + net.blobs['fc7'].data[0];
    end
    features[f] = features[f]/patch_num;
end
```

在算法 3 中，*net* 为微调后的网络。通过调用 *net* 的 $forward\_all$ 方法，我们可以对输入数据进行一次前向传播。*net* 的 *blobs* 变量记录了前向传播时网络中每层的输出数据，$fc7$ 表示网络的倒数第二层。首先，我们按照 $frame\_names$ 列表中的路径，依次读取 UGC 视频帧。之后，我们将读取的 UGC 视频帧输入到微调后的网络中，计算深度特征。最后，我们将计算出的深度特征按序存储在 $features$ 列表中。

3.4　UGC 视频质量评估

在以往视频质量评估领域的研究中，有一些视频质量评估方法直接将视频帧的平均质量分数作为原始视频的质量分数，这种做法实际上只考虑了视频的空域失真，而忽略了视频的时域失真。为了更准确地衡量 UGC 视频的质量，我们在考虑空域失真的同时，使用 LSTM 网络捕获 UGC 视频的时域失真信息。在 3.3 中，我们通过微调后的网络提取了 UGC 视频帧中与视频空域失真相关的深度特征。在本节中，我们将提取的深度特征组成帧级深度特征序列输入到 LSTM 网络中，使用 LSTM 网络预测 UGC 视频的最终质量。除此之外，我们还

将本文提出的 UGC 视频质量评估方法与其他领先的视频质量评估方法进行了对比，KoNViD-1k 数据集上的实验结果表明我们的算法取得了优于其他算法的性能。

3.4.1　UGC 视频质量评估模型训练

（1）总体方法

前面我们将 UGC 视频建模为了一个帧级深度特征序列。这个帧级深度特征序列虽然很好地记录了 UGC 视频的空域失真信息，但却未能包含 UGC 视频的时域失真信息。我们若想捕获 UGC 视频的时域失真信息，还需要借助于其他可以处理时间序列的网络。LSTM 网络是一种特殊的 RNN，在处理序列数据时，可以捕获序列之间的长期依赖关系。因此，为了全面地捕获 UGC 视频的失真信息，更准确地衡量 UGC 视频的质量，我们将帧级深度特征序列输入到 LSTM 网络中，并对其进行训练，以使其预测 UGC 视频的最终质量。

（2）具体实现

在具体搭建和训练 LSTM 网络时，我们使用了 Keras[5]深度学习框架。Keras 是一个用于快速构建模型的高层深度学习框架，由 Python 编写而成，能够在 TensorFlow[6]、CNTK[7]等框架之上运行。Keras 向用户提供了简单易用的接口，对用户极其友好。

a. 网络的搭建

在使用 Keras 搭建 LSTM 网络时，我们先添加了一个含有 512 个神经元的 LSTM 层。之后，我们又在 LSTM 层的后面添加了一个含有 1 个神经元的全连接层。LSTM 层接收这样的输入序列：输入序列的长度为 80，并且序列中每一个向量的维度为 4096。LSTM 层的输出也是一个 4096 维的向量，所以，后面的全连接层用于将 LSTM 层的多维输出转化为最终的一维质量分数。在损失函数的选择上，我们同样也使用了平方和损失函数。

b. 参数的设置

表 3.3 列出了训练 LSTM 网络时我们设置的参数。我们选用了 Adam [43] 算法优化 LSTM 网络的参数。Adam 算法是由 Kingma 等人提出的自适应性学习算法，综合了 AdaGrad [44] 算法和 RMSProp 算法的优点。Adam 算法通过计算梯度的一阶矩估计和二阶矩估计，为不同的参数设置不同的自适应性学习率。在

[5]https://keras.io

[6]https://www.tensorflow.org

[7]https://docs.microsoft.com/en-us/cognitive-toolkit

设置 Adam 算法的相关参数时，我们将它们都设置为了论文 [43] 中推荐的参数值。

表 3.3: 训练 LSTM 网络的参数表

参数	参数值
优化算法	Adam
基础学习率	10^{-3}
一阶矩估计的指数衰减率	9×10^{-1}
二阶矩估计的指数衰减率	9.99×10^{-1}
避免除零项	10^{-8}
总迭代次数	2×10^{3}

c. 实验结果的分析

前面我们提取了 UGC 视频帧的深度特征，从而使每个 UGC 视频都对应于一个长度为 80 的帧级深度特征序列。为使 LSTM 网络学习 UGC 视频与其质量分数的对应关系，我们将训练数据集中所有的深度特征序列及其对应的质量分数输入到 LSTM 网络中，并对其进行训练。

图 3.6 描绘了 LSTM 网络的训练过程中，训练数据集的总损失和测试数据集的总损失随 Epoch 的变化情况，深蓝色曲线为训练数据集的损失变化曲线，浅蓝色曲线为测试数据集的损失变化曲线。一个 Epoch 即将训练数据集中所有的数据都训练一次，我们共完成了 20 个 Epoch。随着 Epoch 的增加，训练数据集的总损失和测试数据集的总损失逐渐趋向于 0。

LSTM 网络训练完成后，我们又对其进行了测试。在测试时，我们将测试数据集中所有的深度特征序列输入到 LSTM 网络中进行一次前向传播，这些深度特征序列在网络中的输出即 UGC 视频的质量分数预测值。在得到这些预测值后，我们对这些预测值与它们的质量分数真值进行了比较，并将比较结果绘制到了图 3.7 中，这个图的横轴为 UGC 视频的质量分数真值，纵轴为 UGC 视频的质量分数预测值。每个红色圆点表示一个测试数据。从图中可以看出，这些红色圆点很好地拟合在了中间的蓝色直线上，这表明我们的 LSTM 网络在对 UGC 视频进行质量评估时可以较为准确地给出评估结果。

3.4.2　结果对比

为准确衡量算法的性能，我们根据测试数据集中 UGC 视频的质量分数真值和预测值计算出了算法的 PLCC 和 SROCC 指标，并与其他领先的视频质量评估

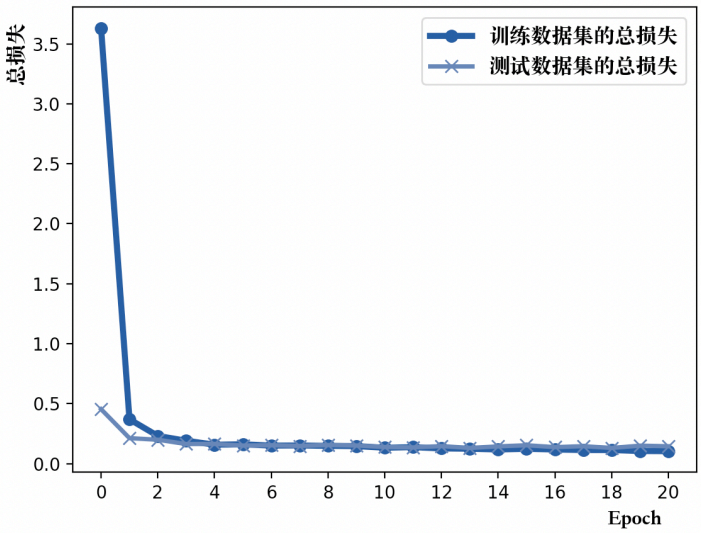


图 3.6: LSTM 网络训练过程中的损失变化曲线

算法进行了对比，对比结果如表 3.4 所示。从表中可以看出，我们提出的算法在 KoNViD-1k 数据集上表现出了优于其他算法的性能。根据我们的研究，Li 等人在 [10] 中提出的视频质量评估算法是目前最为领先的 UGC 视频质量评估算法，而我们的算法又在这个算法的基础上，分别将 PLCC 和 SROCC 指标大约提高了 0.04。另外，表中 BRISQUE [45]、NIQE [46]、CORNIA [47]、VIIDEO [19]、VBLIINDS [18] 的结果也来源于 Li 等人在 [10] 中的测量。

表 3.4: 与其他算法的对比结果

算法	PLCC	SROCC
BRISQUE	0.626	0.654
NIQE	0.546	0.544
CORNIA	0.608	0.610
VIIDEO	0.303	0.298
VBLIINDS	0.658	0.695
Li 等人的算法	0.744	0.755
我们的算法	**0.788**	**0.789**

3.5　本章小结

本章我们提出了一种基于深度学习的 UGC 视频质量评估算法，并详细介绍了算法的设计和实现细节。为便于理解，我们将算法分为了三个部分：图像相对

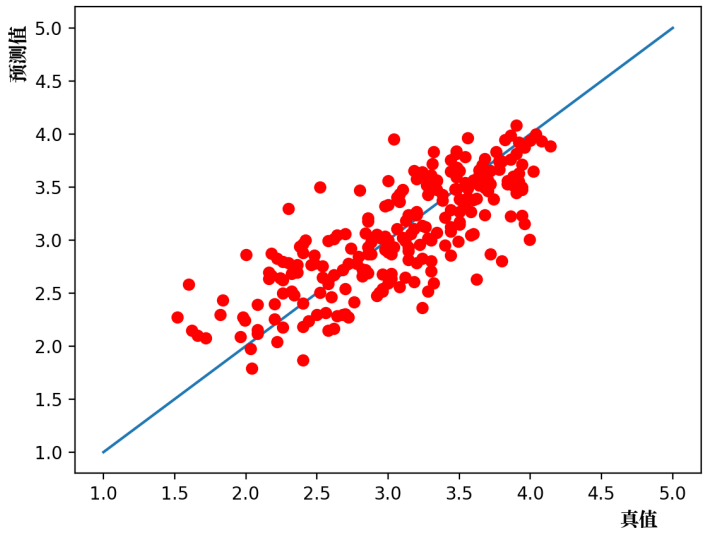


图 3.7: LSTM 网络在测试数据集上的预测结果

质量学习、UGC 视频深度特征提取、UGC 视频质量评估。在图像相对质量学习部分，我们描述了合成失真图像的过程，介绍了训练图像相对质量模型的总体方法和具体实现。在 UGC 视频深度特征提取部分，我们给出了 UGC 视频抽帧的方法，介绍了微调图像相对质量模型的总体方法和具体实现，描述了 UGC 视频提取深度特征的过程。在 UGC 视频质量评估部分，我们介绍了训练最终质量评估模型的总体方法和具体实现，列出了我们的模型与其他模型在 KoNViD-1k 数据集上的对比结果。

第四章　UGC 短视频质量评审系统搭建

4.1　系统整体概述

基于本文提出的 UGC 视频质量评估算法，我们搭建了一个 UGC 短视频质量评审系统。这个系统主要面向 UGC 短视频平台的质量审核人员，旨在帮助他们批量评估用户上传视频的质量，快速过滤低质视频。通过使用这个系统，UGC 短视频平台不仅可以提高视频审核效率，在更短的时间内为平台用户提供更多的优质视频，而且可以极大地节省人工审核成本。

UGC 短视频质量评审系统创建了以机器审核为主、人工审核为辅的视频质量审核模式，其总体流程图如图 4.1 所示。

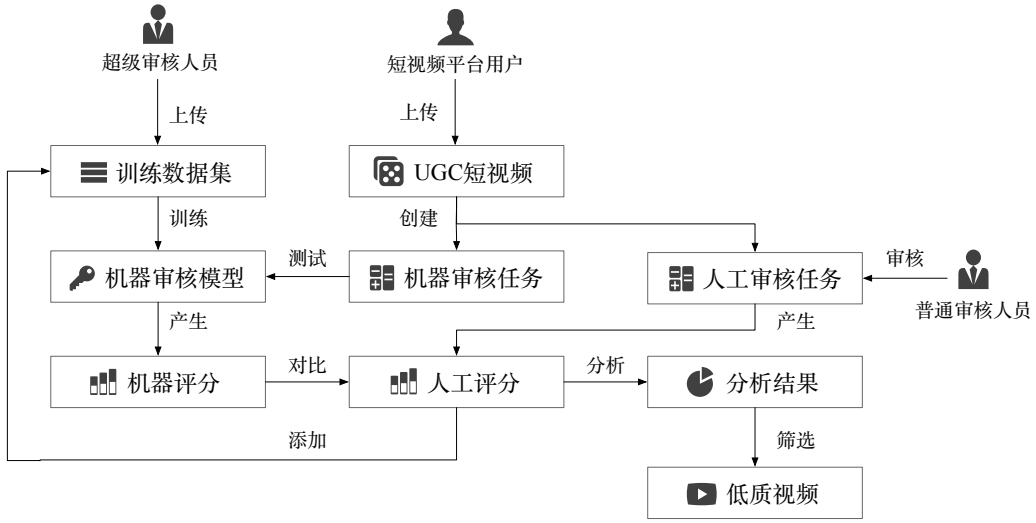


图 4.1: 系统总体流程图

在这个系统中，超级审核人员一方面可以创建机器审核任务，使用训练好的机器审核模型预测 UGC 短视频的质量分数，另一方面也可以创建人工审核任务，通过人工审核的方式审核 UGC 短视频的质量。在人工审核模式下，系统会将超级审核人员创建的人工审核任务分发给全部普通审核人员，而普通审核人员需要对任务中的 UGC 短视频进行质量评分。在人工审核任务和机器审核任务完成之后，超级审核人员可以分析审核结果，对比任务中相同 UGC 短视频的人工评分和机器评分。若人工评分与机器评分的匹配程度较高，则超级审核人员就可以通过机器审核模型大规模地审核 UGC 短视频。因此，机器审核主要用于快速过滤低质视频，人工审核主要用于评估机器审核结果，提高机器审核质量。

另外，超级审核人员也可以将人工审核的结果添加至训练数据集中，以扩充训练数据集，提高机器审核的效果。

UGC 短视频质量评审系统通过 Web 技术开发，实现了前后端的分离，其总体架构图如图 4.2 所示。在这个图中，我们将系统划分为四层，从上到下分别是用户交互层、业务逻辑层、算法能力层和数据存储层。

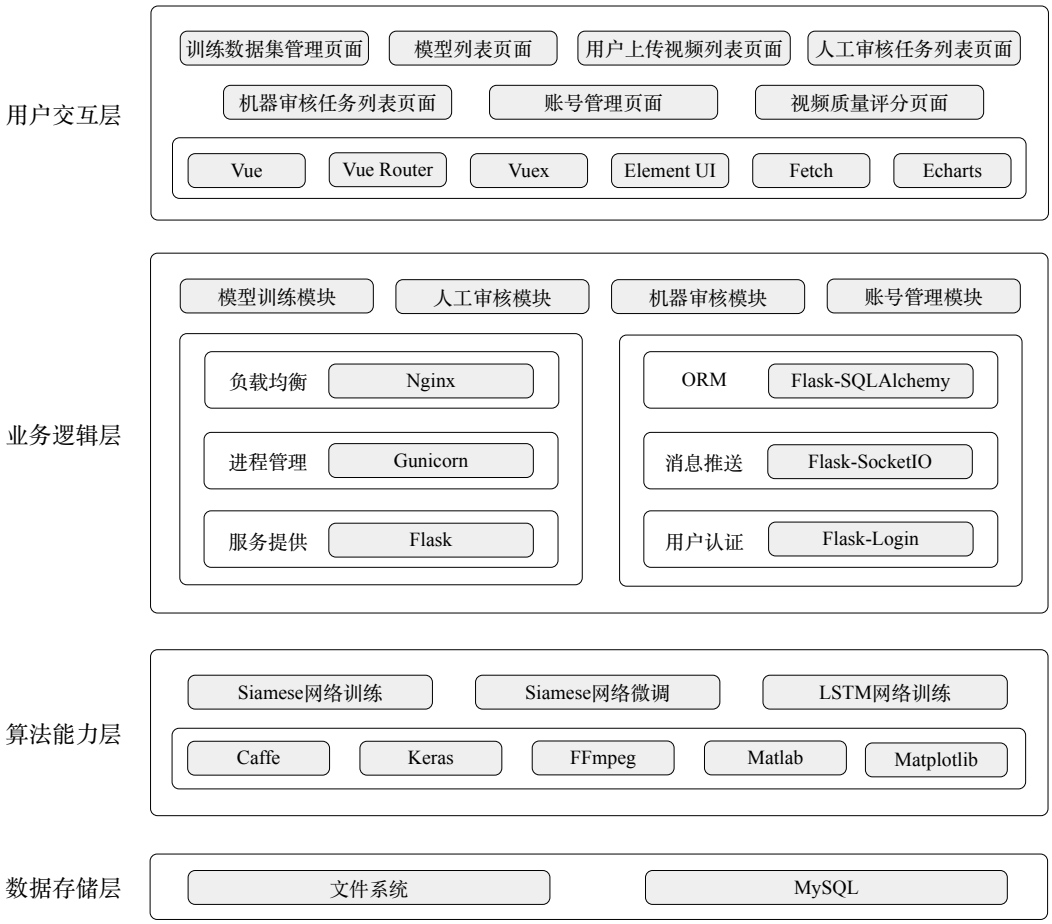


图 4.2: 系统总体架构图

用户交互层展示系统数据，接收用户行为。在这一层中，我们基于 Vue 前端框架搭建了多个系统页面，用于用户与系统间的交互。在搭建这些页面时，我们主要使用了 Vue 相关的技术，如 Vuex 和 Vue Router 等。除此之外，我们还引入了 Element UI 组件库、Echarts 图表库和 Axios 网络库。

业务逻辑层实现系统的核心业务逻辑。在这一层中，我们基于 Flask 后端框架构建了四个核心业务模块，即机器审核模型训练模块、人工审核模块、机器审核模块和账号管理模块，这些模块为系统前端提供服务。为实现它们，我们依赖了 Flask 框架中的多个扩展。Flask-SocketIO 扩展被用于建立与系统前端的长

连接，Flask-Login 扩展被用于实现系统的用户认证功能，Flask-SQLAlchemy 扩展被用于完成系统的 ORM（Object Relational Mapping，对象关系映射）。通过 Gunicorn，我们实现了 Flask 应用的多进程、自启动部署。Nginx 则具有请求转发和负载均衡的功能。

算法能力层提供机器审核模型训练和测试的能力。在这一层中，我们基于 Caffe 深度学习框架实现了 Siamese 网络的训练、微调和测试，基于 Keras 深度学习框架实现了 LSTM 网络的训练和测试。除此之外，我们还使用了 FFmpeg 音视频处理工具、Matlab 数值计算工具和 Matplotlib 图形绘制工具。

数据存储层为上面的几层提供数据支撑。在这一层中，我们通过文件系统存储训练产生的模型、系统用户上传的 UGC 视频及其产物，通过 MySQL 关系型数据库存储系统运行过程中生成的数据。

下面我们将按照系统需求分析、系统总体设计、系统详细设计、系统实现的基本顺序，详细描述 UGC 短视频质量评审系统的搭建流程。

4.2　系统需求分析

在本文中，我们分别从功能性需求和非功能性需求两个方面分析系统需求。

4.2.1　功能性需求

用例图以用例为基本单位展示系统的功能，是一种常用的需求分析手段。下面，我们将借助于用例图分析系统的功能性需求。

搭建 UGC 短视频质量评审系统的主要目的是对平台用户上传的 UGC 短视频进行高效的质量审核，降低人工审核成本。为实现这个目标，系统应该具有以下系统特性：提供人工审核和机器审核两种视频质量审核手段，机器审核用于快速过滤低质视频，人工审核用于评估机器审核结果，提高机器审核质量。

根据上面确定的系统目标和特性，我们发现 UGC 短视频质量评审系统的用户可以分为超级审核人员和普通审核人员两类。超级审核人员即负责创建人工审核任务和机器审核任务，推进 UGC 短视频质量审核总体进程的系统用户。普通审核人员即负责对 UGC 短视频进行人工质量评分的系统用户。通过进一步的分析，我们得出这个系统中超级审核人员参与的用例应包括训练数据集管理用例、机器审核模型训练用例、审核任务管理用例、审核结果分析用例和账号管理用例，普通审核人员参与的用例应包括视频质量评分用例。另外，人工审核任务管理用例和机器审核任务管理用例应继承审核任务管理用例，人工审核结果分析用例和机器审核结果分析用例应继承审核结果分析用例，视频质量评分用例应包含人工审核结果提交用例。

基于上述分析，我们绘制了如图 4.3所示的系统用例图。

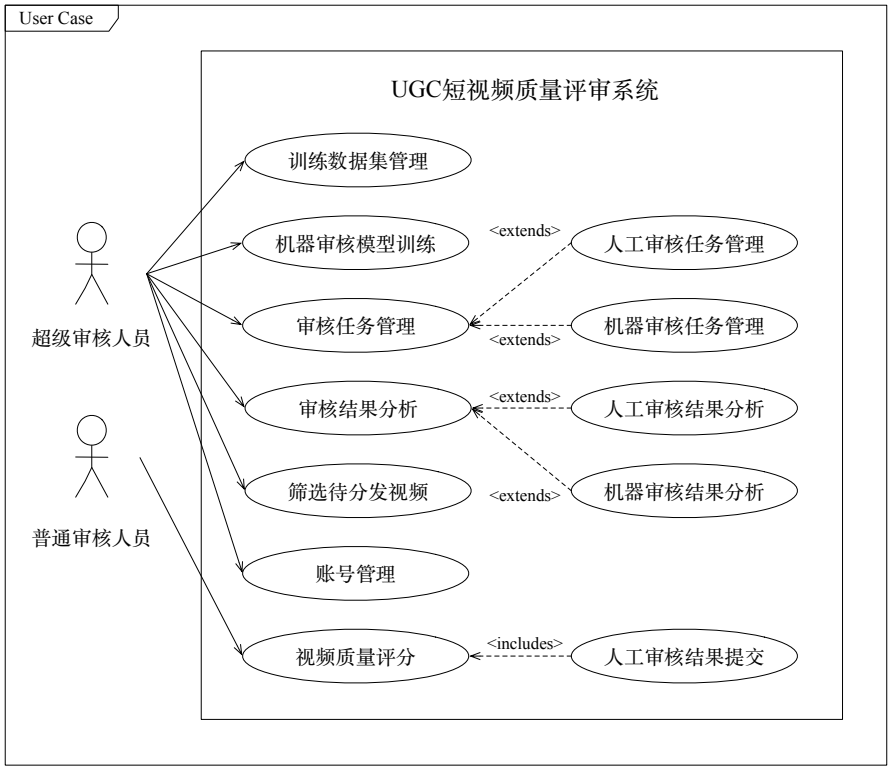


图 4.3: 系统用例图

训练数据集管理用例的描述如表 4.1所示，在这个用例中，超级审核人员可以批量上传训练数据。超级审核人员上传训练数据后，系统会将这些数据批量添加到训练数据集中。

表 4.1: 训练数据集管理用例描述

ID	**UC1**
名称	训练数据集管理
参与者	超级审核人员
前置条件	超级审核人员已使用授权账号登录到系统中
后置条件	系统存储已上传的 UGC 视频及其相关信息
正常流程	1. 超级审核人员批量上传 UGC 视频作为训练数据
	2. 系统将上传的 UGC 视频添加到训练数据集中
	3. 超级审核人员批量上传 UGC 视频的相关信息，包括时长、人工评分等
	4. 系统记录 UGC 视频的相关信息
扩展流程	1-4a. 超级审核人员上删除已上传的 UGC 视频
	1. 系统删除 UGC 视频及其相关信息

机器审核模型训练用例的描述如表 4.2所示，在这个用例中，超级审核人员

可以使用已上传的训练数据，训练机器审核模型。模型训练完成后，系统会将模型保存起来。

表 4.2: 机器审核模型训练用例描述

ID	**UC2**
名称	机器审核模型训练
参与者	超级审核人员
前置条件	超级审核人员已上传 UGC 视频及其相关信息
后置条件	系统存储模型及其相关信息
正常流程	1. 超级审核人员查看已上传的 UGC 视频及其相关信息 2. 超级审核人员从已上传的 UGC 视频中选择部分或全部视频，请求开始训练机器审核模型 3. 系统将超级审核人员选择的 UGC 视频作为训练数据，开始训练模型 4. 机器审核模型训练完成，系统保存模型及其相关信息 5. 用户查看模型信息，并为其添加备注
扩展流程	2a. 超级审核人员选择的 UGC 视频中包含无人工评分的视频 1. 系统拒绝训练模型，并显示无人工评分的 UGC 视频信息 3-4a. 超级审核人员取消训练模型 1. 系统停止训练模型，并提示训练已取消 3-4b. 模型在训练过程中出现了错误 1. 系统提示模型训练失败

表 4.3: 人工审核任务管理用例描述

ID	**UC3**
名称	人工审核任务管理
参与者	超级审核人员
前置条件	系统中有已注册的普通审核人员账户
后置条件	系统存储任务信息和 UGC 短视频的质量分数
正常流程	1. 超级审核人员查看平台用户上传的 UGC 短视频及其相关信息 2. 超级审核人员从平台用户上传的 UGC 短视频中选择部分或全部视频，创建人工审核任务 3. 系统记录任务信息，并将其分发给普通审核人员 4. 普通审核人员审核完成，系统实时更新任务完成进度和 UGC 短视频的质量分数 5. 超级审核人员查看任务信息
扩展流程	2a. 超级审核人员选择的 UGC 短视频中包含已创建人工审核任务的视频 1. 系统拒绝创建任务，并显示已创建人工审核任务的 UGC 短视频信息 3-4a. 超级审核人员取消人工审核任务 1. 系统删除人工审核任务，并提示任务已取消

人工审核任务管理用例的描述如表 4.3所示，在这个用例中，超级审核人员

可以创建人工审核任务。任务创建完成后，系统会将其分发给普通审核人员。在普通审核人员审核的过程中，系统会实时更新任务完成进度和 UGC 短视频的质量分数。

机器审核任务管理用例的描述如表 4.4所示，在这个用例中，超级审核人员可以创建机器审核任务。任务创建完成后，系统会使用机器审核模型预测 UGC 短视频的质量。在模型预测过程中，系统会实时更新任务完成进度和 UGC 短视频的质量分数。

表 4.4: 机器审核任务管理用例描述

ID	**UC4**
名称	机器审核任务管理
参与者	超级审核人员
前置条件	超级审核人员已成功训练机器审核模型
后置条件	系统存储任务信息和 UGC 短视频的质量分数
正常流程	1. 超级审核人员查看平台用户上传的 UGC 短视频及其相关信息 2. 超级审核人员从平台用户上传的 UGC 短视频中选择部分或全部视频，之后再指定一个模型，创建机器审核任务 3. 系统记录任务信息，并使用指定模型预测 UGC 短视频的质量 4. 系统实时更新任务完成进度和 UGC 短视频的质量分数 5. 超级审核人员查看任务信息
扩展流程	3-4a. 超级审核人员取消机器审核任务 1. 系统删除机器审核任务，并提示任务已取消 3-4b. 模型在预测过程中出现了错误 1. 系统提示模型预测失败

表 4.5: 人工审核结果分析用例描述

ID	**UC5**
名称	人工审核结果分析
参与者	超级审核人员
前置条件	系统中有已完成的人工审核任务
后置条件	系统存储对人工审核结果的分析
正常流程	1. 超级审核人员查看已完成的人工审核任务及其相关信息 2. 超级审核人员从已完成的人工审核任务中选择一个任务，查看人工审核结果及系统对结果的分析 3. 系统显示人工审核结果及对结果的分析 4. 超级审核人员选择任务中的部分或全部 UGC 短视频，并将其添加到训练数据集中 5. 系统更新训练数据集，并对选择的 UGC 短视频进行标记

人工审核结果分析用例的描述如表 4.5所示，在这个用例中，超级审核人员可以对人工审核结果进行分析。系统显示出对结果的分析后，超级审核人员可以将这个任务中的 UGC 短视频添加到训练数据集中，以扩充训练数据集。系统在更新训练数据集中数据的同时，也会对这些 UGC 短视频进行标记。

机器审核结果分析用例的描述如表 4.6所示，在这个用例中，超级审核人员可以对机器审核结果进行分析。系统显示出对结果的分析后，超级审核人员可以输入筛选条件，使系统筛选出这个任务中的低质 UGC 短视频。

表 4.6: 机器审核结果分析用例描述

ID	**UC6**
名称	机器审核结果分析
参与者	超级审核人员
前置条件	系统中有已完成的机器审核任务
后置条件	系统存储对机器审核结果的分析及低质 UGC 短视频信息
正常流程	1. 超级审核人员查看已完成的机器审核任务及其相关信息 2. 超级审核人员从已完成的机器审核任务中选择一个任务，查看机器审核结果及系统对结果的分析 3. 系统显示机器审核结果及对结果的分析 4. 超级审核人员输入低质 UGC 短视频的筛选条件 5. 系统筛选低质 UGC 短视频，并对它们进行标记
扩展流程	4b. 超级审核人员输入的筛选条件不符合筛选规则 1. 系统拒绝筛选，并提示筛选条件不符合筛选规则

表 4.7: 账号管理用例描述

ID	**UC7**
名称	账号管理
参与者	超级审核人员
前置条件	系统中有已注册的普通审核人员账号
后置条件	系统存储账号信息
正常流程	1. 超级审核人员查看系统中所有的审核人员账号 2. 超级审核人员从中选择一个普通审核人员账号，并授予超级审核人员权限 3. 系统更新账号信息
扩展流程	2b. 超级审核人员选择一个普通审核人员账号，并将其移除 1. 系统删除账号信息

账号管理用例的部分描述如表 4.7所示，在这个用例中，超级审核人员可以为普通审核人员账号授予超级审核人员权限。权限授予后，系统会相应地更新

账号信息。除此之外，这个用例还包括超级审核人员账号和普通审核人员账号的登录、退出、注销等。

视频质量评分用例的描述如表 4.8所示，在这个用例中，普通审核人员可以对人工审核任务中的 UGC 短视频进行质量评分。评分过程中，系统会记录视频的质量分数。评分完成后，普通审核人员可以提交人工审核结果，系统会更新任务完成进度。

表 4.8: 视频质量评分用例描述

ID	**UC8**
名称	视频质量评分
参与者	普通审核人员
前置条件	普通审核人员已登录到系统中，并有未完成的人工审核任务
后置条件	系统存储任务完成进度及视频质量分数
正常流程	1. 普通审核人员从未完成的人工审核任务中选择一个任务，开始审核 2. 系统显示任务中包含的所有 UGC 短视频 3. 普通审核人员选择一个 UGC 短视频，并输入它的质量分数 4. 系统记录 UGC 短视频的质量分数 普通审核人员重复 2-3 步，直到完成所有视频的质量评分 5. 普通审核人员提交人工审核结果 6. 系统更新任务完成进度
扩展流程	3a. 普通审核人员选择的 UGC 短视频已进行质量评分 1. 系统覆盖 UGC 短视频原有的质量分数 4a. 普通审核人员提交人工审核结果时，还有部分 UGC 短视频未评分 1. 系统不更新任务完成进度，并显示未评分的 UGC 短视频信息 3-4a. 普通审核人员退出评分 1. 系统不保存质量分数，并提示退出后分数将不会被保存

4.2.2　非功能性需求

除上述功能性需求外，我们还分别从性能需求和质量属性两个方面分析了系统应满足的非功能性需求。系统的非功能性需求如表 4.9 所示。

表 4.9: 系统的非功能性需求

类别	**描述**
性能需求	速度：对于所有的用户请求，系统必须在 2 秒之内给出响应 容量：系统能够存储 100 万个 UGC 短视频和 100 个模型 负载：系统可以允许 100 个人同时访问
质量属性	可靠性：在网络出现故障时，系统不出现故障 安全性：普通审核人员账号只能进行视频质量评分，不具有其他功能的操作权限 易用性：超级审核人员可以在一天的时间内审核 5000 个 UGC 短视频

4.3　系统总体设计

4.3.1　系统架构设计

在进行系统架构设计时，我们使用了 4+1 视图 [48]。4+1 视图是一种描述系统架构的方法，最早由 Philippe Kruchten 提出，目前被广泛用于复杂系统的架构设计中。4+1 视图中的 4 分别指的是逻辑视图、开发视图、进程视图和部署视图，1 指的是用例图。4+1 视图中 4 个视图的设计都围绕着用例图展开。其中，逻辑视图是一个满足系统功能需求的抽象技术方案，开发视图主要描述系统在开发环境下的静态组织结构，进程视图主要描述系统运行时的动态行为，部署视图主要反映系统在物理节点上的分布情况。

由 4.2.1 中对系统功能性需求的分析，我们可以知道，系统的主要功能有训练数据集管理、机器审核模型训练、人工审核任务管理、机器审核任务管理、人工审核结果分析、机器审核结果分析、账号管理和视频质量评分。在进行系统设计时，我们把关联较大的功能进行了合并处理：训练数据集管理和机器审核模型训练合并为机器审核模型训练，人工审核任务管理、人工审核结果分析和视频质量评分合并为人工审核，机器审核任务管理和机器审核结果分析合并为机器审核。这样，系统最终需要考虑的概要功能为机器审核模型训练、人工审核、机器审核和账号管理。

因为系统基于 Web 开发，所以我们选择了 MVC（Model−View−Controller，模型–视图–控制器）架构模式。这种模式将系统分成三个基本部分：模型、视图和控制器。模型是对系统状态和数据的封装，用于执行业务逻辑。视图是对用户交互的封装，用于接收用户行为。控制器是模型和视图的桥梁，根据用户行为选择调用模型的业务逻辑。

依据系统概要功能和 MVC 架构模式，我们绘制了如图 4.4 所示的系统逻辑视图。在逻辑视图中，view 层及 controller 层中的 training 包、human_task 包、machine_trak 包和 account 包分别对应概要功能中的机器审核模型训练功能、人工审核功能、机器审核功能、账号管理功能。对于 model 层，我们将涉及机器审核模型训练功能的包拆分为了 training_data 包和 model 包，将涉及人工审核功能的包拆分为了 human_task 包、testing_data 包和 evaluation 包，将涉及机器审核功能的包拆分为了 machine_task 包和 testing_data 包。

系统开发视图在系统逻辑视图的基础上绘制而成，如图 4.5 所示。与系统逻辑设计相比，在进行系统开发设计时，我们考虑了更多的实现细节。开发视图中的 ui 层位于系统前端，api 层、service 层、entity 层位于系统服务端。

根据系统的概要功能，在系统前端中，我们需要为超级审核人员提供训练

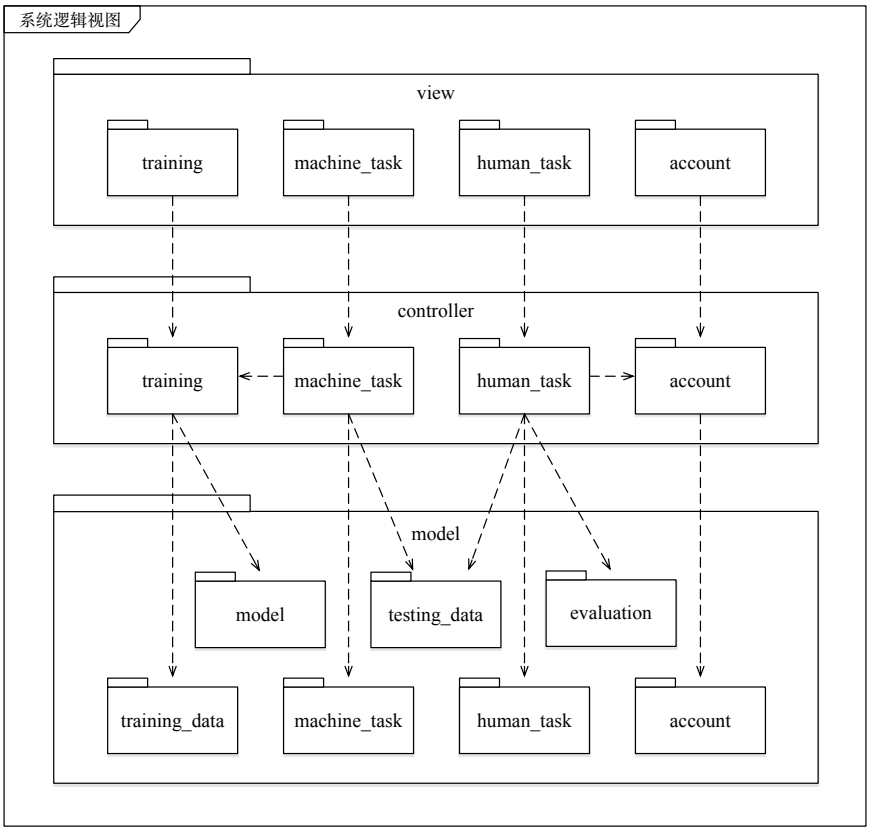


图 4.4: 系统逻辑视图

数据集管理页面、模型列表页面、用户上传视频列表页面、人工审核任务列表页面、机器审核任务列表页面、账号管理页面和视频质量评估页面，需要为普通审核人员提供视频质量评分页面。因此，在 ui 层中，我们创建了 training_data_management 包、model_list 包、video_list 包、human_task_list 包、machine_trak_list 包、account_management 包和 video_evaluation 包，使其分别对应于这七个页面。

因为位于前端的页面需要与服务端进行通信，所以在 api 层中，我们创建了七个与页面同名的包，接收来自页面的请求。

api 层中的包在收到页面请求后，需要调用系统的服务。在 service 层中，我们创建了五个包，这五个包分别对应五个服务。其中，training 包接收 api 层中 training_data_management 包和 model_list 包的调用，video 包接收 api 层中 video_list 包的调用，human_task 包接收 api 层中 human_task_list 包和 video_evaluation 包的调用，machine_task 包接收 api 层中 machine_trak_list 包的调用，account 包接收 api 层中 account_management 包的调用。之所以使 api 层中 human_task_list 包和 video_evaluation 包调用一个服务，是因为人工审核任务的业务逻辑与视频质

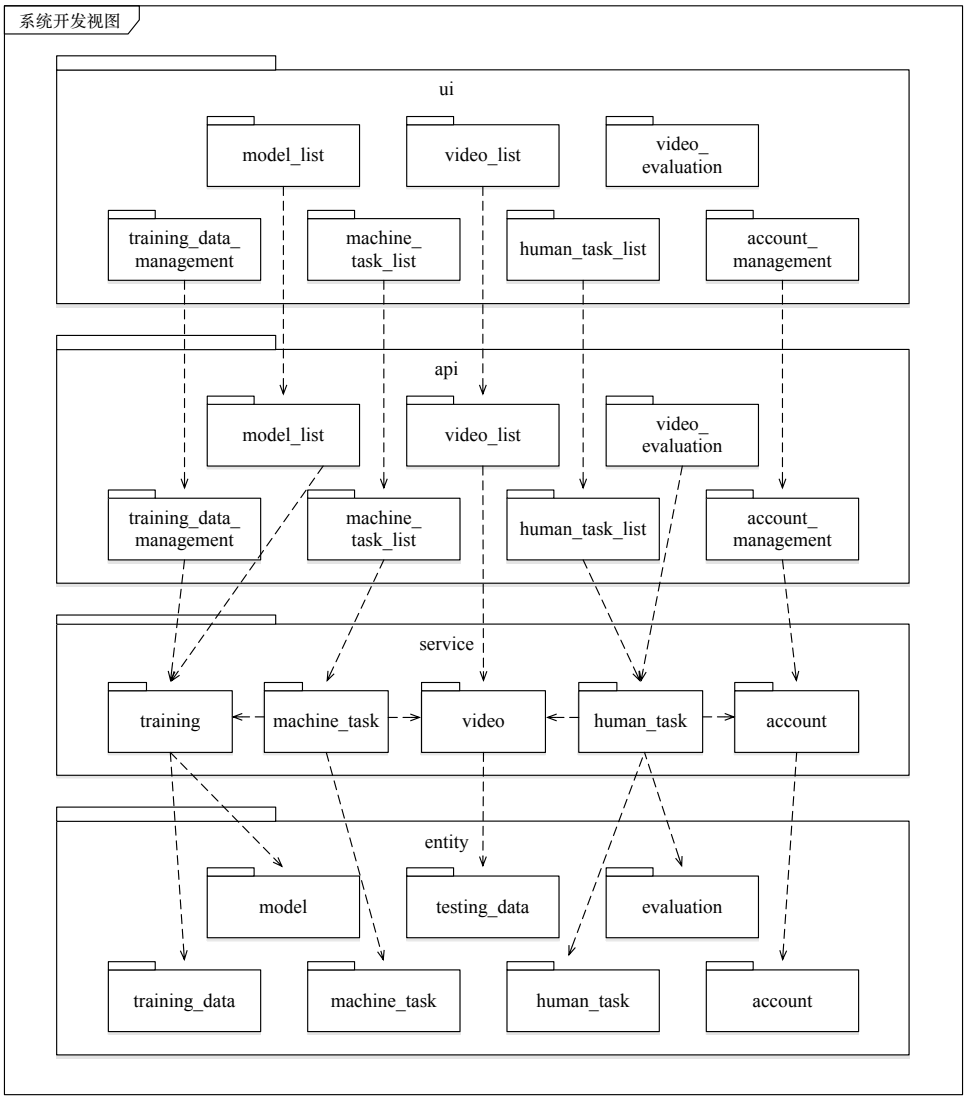


图 4.5: 系统开发视图

量评分的业务逻辑相关性较强，可以在一个服务中实现。

系统服务的操作需要基于数据实体进行。因此，在 entity 层中，我们创建了七个数据实体包。这七个包的设置与逻辑视图中 model 层的包一致。

系统进程视图描述了进行机器审核模型训练和机器审核模型预测时，系统服务端的线程调用情况，如图 4.6 所示。

当进行机器审核模型训练时，系统主线程异步调用模型训练线程，模型训练线程开始训练模型。模型训练完成后，模型训练线程同步调用数据库相关线程，存储模型相关信息。最后，模型训练线程回调主线程，通知主线程训练结果。当进行机器审核模型测试时，系统主线程异步调用模型测试线程，模型测

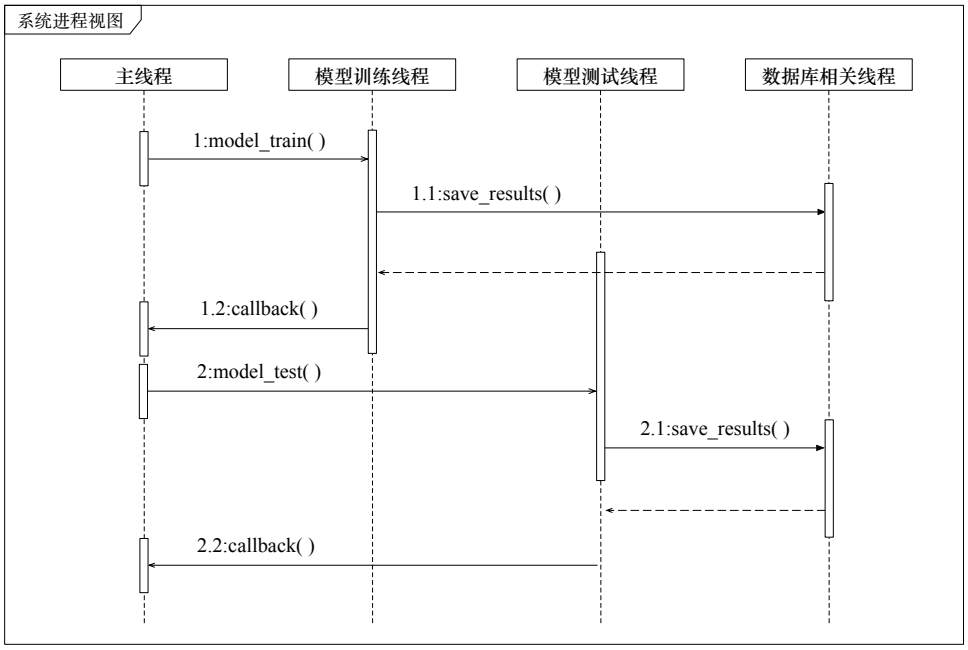


图 4.6: 系统进程视图

试线程开始预测 UGC 短视频的质量分数。预测完成后，模型测试线程同步调用数据库相关线程，存储 UGC 短视频的质量分数。最后，模型测试线程回调主线程，通知主线程测试结果。

系统部署视图如图 4.7 所示。在这个视图中，系统用户通过装有浏览器的个人计算机与系统服务端进行通信，它们之间的通信协议包括 HTTP 协议和 WebSocket 协议。其中，WebSocket 协议被用于实现长连接。当用户向系统服务端发出网络请求时，Nginx 反向代理服务器会接收这个请求，并按照一定的策略转发给 Gunicorn 后台服务器。Gunicorn 后台服务器通过 TCP/IP 协议与 MySQL 数据库服务器进行连接，MySQL 数据库服务器主要被用于存储系统运行过程中产生的数据。

4.3.2　数据库设计

根据系统开发视图中对 entity 层的描述，我们设计了实体类。为了持久化实体类中的数据，我们通过 ORM 将它们映射到关系型数据库中。在我们的关系型数据集库中，共包含了 training_video、model、testing_video、human_task、machine_task、human_task_video、machine_task_video、human_result、machine_result 和 account 十个数据表。下面的表格列出了每个数据表中包含的字段以及它们的类型、属性和含义。字段的 PK 属性表示主键，AI 属性表示自动增长，NN 属性表示不为空，NQ 属性表示不重复，FK 属性表示外键。

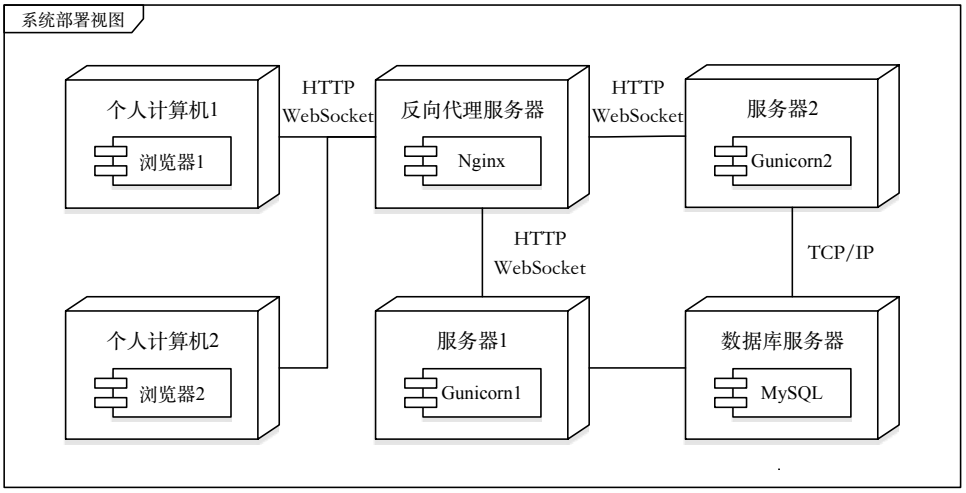


图 4.7: 系统部署视图

training_video 数据表记录了训练数据集中 UGC 视频的相关信息，对应于 Video4Training 实体类。表 4.10 列出了这个数据表中包含的所有字段。除表示名称、时长、分辨率、人工评分的字段外，我们还设置了一个自动增长的 id 字段，使其可以唯一标记训练数据集中的 UGC 视频。另外，训练数据集中 UGC 视频的名称和存储路径是不为空且不重复的。

表 4.10: training_video 表的字段定义

字段	类型	属性	含义
id	INT	PK，AI	训练数据集中 UGC 视频的唯一标识
name	VARCHAR	NN，NQ	视频的名称
length	VARCHAR		视频的时长
size	VARCHAR		视频的分辨率
mos	FLOAT		视频的人工评分
content	VARCHAR	NN，NQ	视频的存储路径

model 数据表记录了机器审核模型的相关信息，对应于 MachineModel 实体类。表 4.11 列出了这个数据表中包含的所有字段。id 字段用于唯一标记机器审核模型，tag 字段用于存储超级审核人员为机器审核模型设置的标签。

testing_video 数据表记录了 UGC 短视频平台用户上传的视频的相关信息，对应于 Video4Testing 实体类。表 4.12 列出了这个数据表中包含的所有字段。与 training_video 数据表相比，这个数据表删除了 mos 字段，增加了两个与视频上传有关的字段，增加了一个标志视频是否已经被创建人工审核任务的字段，增加了一个标志视频是否是低质视频的字段。

表 4.11: model 表的字段定义

字段	类型	属性	含义
id	INT	PK，AI	机器审核模型的唯一标识
name	VARCHAR	NN，NQ	模型的名称
tag	VARCHAR		模型的标签
is_finished	BOOLEAN	NN	模型是否已经训练完成
finished_time	DATATIME		模型的训练完成时间
has_error	BOOLEAN		模型是否在训练过程中出现了错误
content_1	VARCHAR		Siamese 网络的存储路径
content_2	VARCHAR		LSTM 网络的存储路径
frames_num	INT		每个视频中抽取帧的数量

表 4.12: testing_video 表的字段定义

字段	类型	属性	含义
id	INT	PK，AI	用户上传 UGC 短视频的唯一标识
name	VARCHAR		视频的名称
length	VARCHAR		视频的时长
size	VARCHAR		视频的分辨率
content	VARCHAR	NN，NQ	视频的存储路径
user_id	INT	NN	上传视频的平台用户 ID
upload_time	DATATIME	NN	视频的上传时间
is_evaluated	BOOLEAN	NN	视频是否已经被创建人工审核任务
is_low	BOOLEAN		视频是否是低质视频

human_task 数据表记录了人工审核任务的相关信息，对应于 HumanTask 实体类。表 4.13 列出了这个数据表中包含的所有字段。id 字段用于唯一标记人工审核任务，其他字段用于存储人工审核任务的详细信息。create_account_id 是这个数据表的外键，它关联了 account 数据表的主键。

machine_task 数据表记录了机器审核任务的相关信息，对应于 MachineTask 实体类。表 4.14 列出了这个数据表中包含的所有字段。id 字段用于唯一标记机器审核任务，其他字段用于存储机器审核任务的详细信息。create_account_id 是这个数据表的外键，它关联了 account 数据表的主键。

huamn_task_video 数据表记录了人工审核任务中包含的 UGC 短视频，对应于 HumanTaskVideo 实体类。表 4.15 列出了这个数据表中包含的所有字段。task_id 和 video_id 都是这个数据表的外键，它们分别关联了 human_task 数据表和 testing_video 数据表的主键。同时，这两个外键联合起来，共同作为这个数据表的主键。

表 4.13: human_task 表的字段定义

字段	类型	属性	含义
id	INT	PK，AI	人工审核任务的唯一标识
create_account_id	INT	FK，NN	创建任务的账号 ID
name	VARCHAR	NN，NQ	任务的名称
tag	VARCHAR		任务的标签
is_finished	BOOLEAN	NN	任务是否已经被全部完成
finished_count	INT	NN	已完成任务的人数
all_count	INT	NN	需要完成任务的人数

表 4.14: machine_task 表的字段定义

字段	类型	属性	含义
id	INT	PK，AI	机器审核任务的唯一标识
create_account_id	INT	FK，NN	创建任务的账号 ID
name	VARCHAR	NN，NQ	任务的名称
tag	VARCHAR		任务的标签
is_finished	BOOLEAN	NN	任务是否已经被完成
finished_percent	INT	NN	任务完成百分数

表 4.15: huamn_task_video 表的字段定义

字段	类型	属性	含义
task_id	INT	FK，NN	人工审核任务 ID
video_id	INT	FK，NN	任务中包含的 UGC 短视频 ID

machine_task_video 数据表记录了机器审核任务中包含的 UGC 短视频，对应于 MachineTaskVideo 实体类。表 4.16 列出了这个数据表中包含的所有字段。这个数据表中包含字段与 huamn_task_video 数据表的字段类似。

human_result 数据表记录了人工审核结果，对应于 HumanResult 实体类。表 4.17 列出了这个数据表中包含的所有字段。video_id 和 account_id 都是这个数据表的外键，它们分别关联了 testing_video 数据表和 account 数据表的主键。同时，这两个外键联合起来，共同作为这个数据表的主键。

machine_result 数据表记录了机器审核结果，对应于 MachineResult 实体类。表 4.18 列出了这个数据表中包含的所有字段。task_id 和 video_id 都是这个数据表的外键，它们分别关联了 machine_task 数据表和 testing_video 数据表的主键。同时，这两个外键联合起来，共同作为这个数据表的主键。

表 4.16: machine_task_video 表的字段定义

字段	类型	属性	含义
task_id	INT	FK，NN	机器审核任务 ID
video_id	INT	FK，NN	任务中包含的 UGC 短视频 ID

表 4.17: human_result 表的字段定义

字段	类型	属性	含义
video_id	INT	FK，NN	用户上传的 UGC 短视频 ID
account_id	INT	FK，NN	普通审核人员的账号 ID
mos	FLOAT	NN	普通审核人员给出的视频质量评分

表 4.18: machine_result 表的字段定义

字段	类型	属性	含义
task_id	INT	FK，NN	机器审核任务 ID
video_id	INT	FK，NN	任务中包含的 UGC 短视频 ID
mos	FLOAT	NN	机器给出的视频质量评分

account 数据表记录了审核人员账号的相关信息，对应于 Account 实体类。表 4.19 列出了这个数据表中包含的所有字段。id 字段用于唯一标记审核人员账号，其他字段用于存储审核人员账号的详细信息。

4.4　系统详细设计与实现

以系统需求分析和总体设计的结果为出发点，我们对这个系统进行了详细设计与实现。根据系统逻辑视图，这个系统可以被划分为四个模块：机器审核模型训练模块、人工审核模块、机器审核模块和账号管理模块。因此，在详细设计时，我们以类图和顺序图的形式对每个模块中的包进行了细化，以使其满足系统所有的功能性需求和非功能需求。系统详细设计完成后，我们依据每个模块的类图和顺序图对每个模块进行了实现。

4.4.1　机器审核模型训练模块设计与实现

机器审核模型训练模块完成了训练数据集管理和机器审核模型训练两个主要功能。通过这个模块，超级审核人员既可以上传 UGC 视频及其相关信息到训练数据集中，又可以基于上传的数据训练机器审核模型。同时，这个模块也为超级审核人员提供了查看训练数据、删除训练数据和查看机器审核模型的功能。

表 4.19: account 表的字段定义

字段	类型	属性	含义
id	INT	PK，AI	审核人员账号的唯一标识
name	VARCHAR	NN，NQ	账号的用户名
password	VARCHAR	NN	账号的密码
is_manager	BOOLEAN	NN	账号是否是管理审核人员账号

在机器审核模型训练模块中，我们预先在一个大规模的无标签失真图像数据集上完成了 Siamese 网络的训练。而当收到超级审核人员的模型训练请求后，我们又会使用训练数据集中的数据进一步地训练机器审核模型，这其中包括微调 Siamese 网络和训练 LSTM 网络。

（1）详细设计

机器审核模型训练模块的类图如图 4.8 和图 4.9 所示。在这两个图中，我们详细描述了机器审核模型训练模块中涉及到的类以及类与类之间的关系。

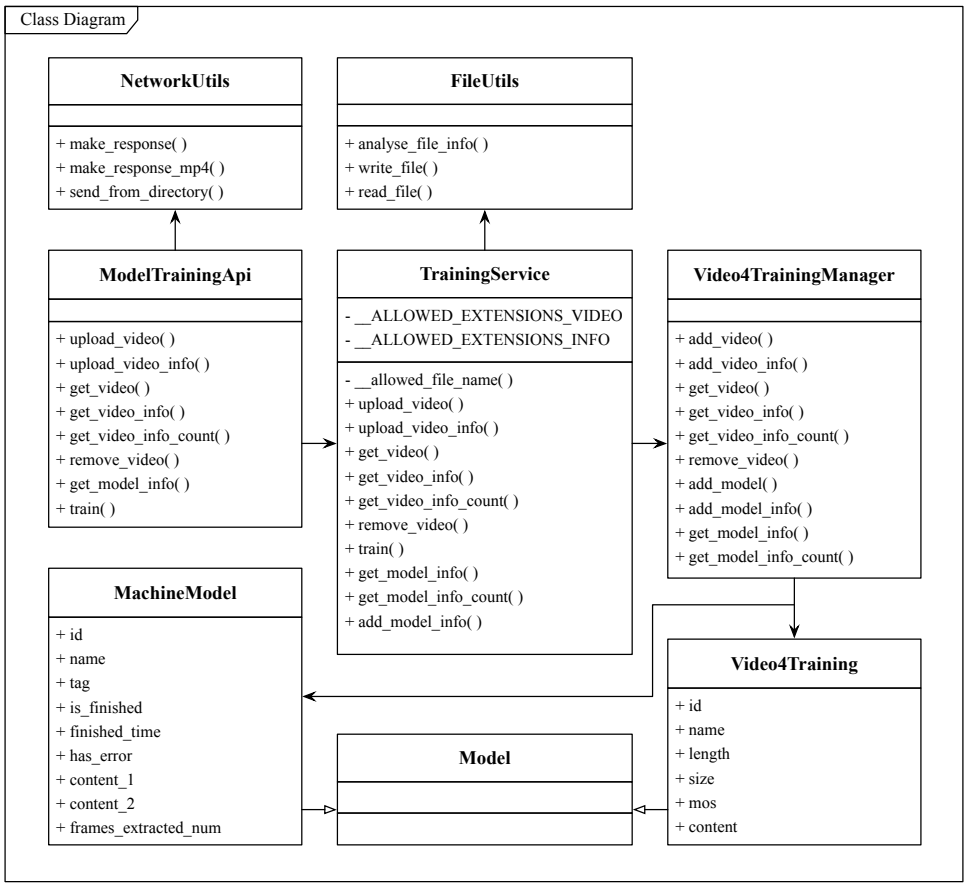


图 4.8: 机器审核模型训练模块类图-1

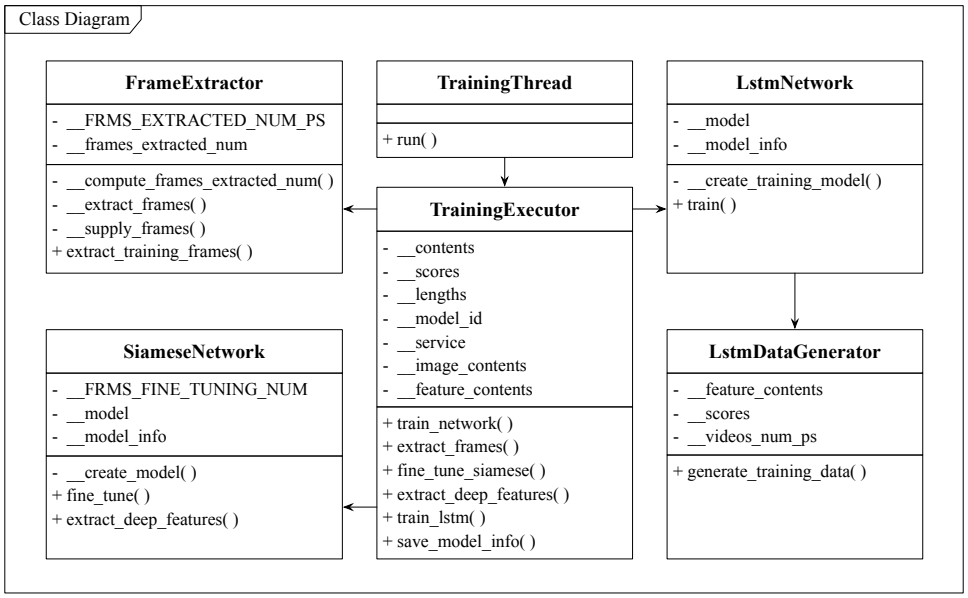


图 4.9: 机器审核模型训练模块类图-2

图 4.8 中各个类的职责如下。ModelTrainingApi 类负责接收训练数据集管理页面和模型列表页面发出的网络请求。TrainingService 类负责对收到的网络请求进行处理，它依赖 TrainingManager 类和 TrainingExecutor 类，并通过这两个类提供与模型训练有关的服务。其中，TrainingManager 类负责管理 Video4Training 和 MachineModel 实体类，进行与实体类相关的操作。Model 类是所有实体类的基类。NetworkUtils 类和 FileUtils 类分别是网络工具类和文件工具类。

图 4.9 中各个类的职责如下。TrainingThread 类是专门用于模型训练工作的线程类。TrainingExecutor 类负责管理 FrameExtractor 类、SiameseNetwork 类和 LstmNetwork 类，进行与机器审核模型训练有关的操作。FrameExtractor 类负责对 UGC 视频进行抽帧，SiameseNetwork 类负责微调 Siamese 网络和使用 Siamese 网络提取 UGC 视频的深度特征，LstmNetwork 类负责训练 LSTM 网络。LstmDataGenerator 类则负责为 LstmNetwork 类提供训练数据。

由于模型训练需要花费较长的时间，当收到超级审核人员的模型训练请求后，机器审核模型训练模块会创建一个子线程，用于异步执行模型训练操作。在图 4.10 所示的顺序图中，我们展示了子线程创建后，机器审核模型训练模块中的相关类是如何协作完成模型训练的。

子线程启动后，TrainingThread 类会依次调用 TrainingExecutor 类的方法，完成 UGC 视频抽帧、微调 Siamese 网络、提取 UGC 视频深度特征、训练 LSTM 网络的操作。TrainingExecutor 类则会将其委托给具体负责完成这些操作的类，即 FrameExtractor 类、SiameseNetwork 类和 LstmNetwork 类。在它们完成这些操作

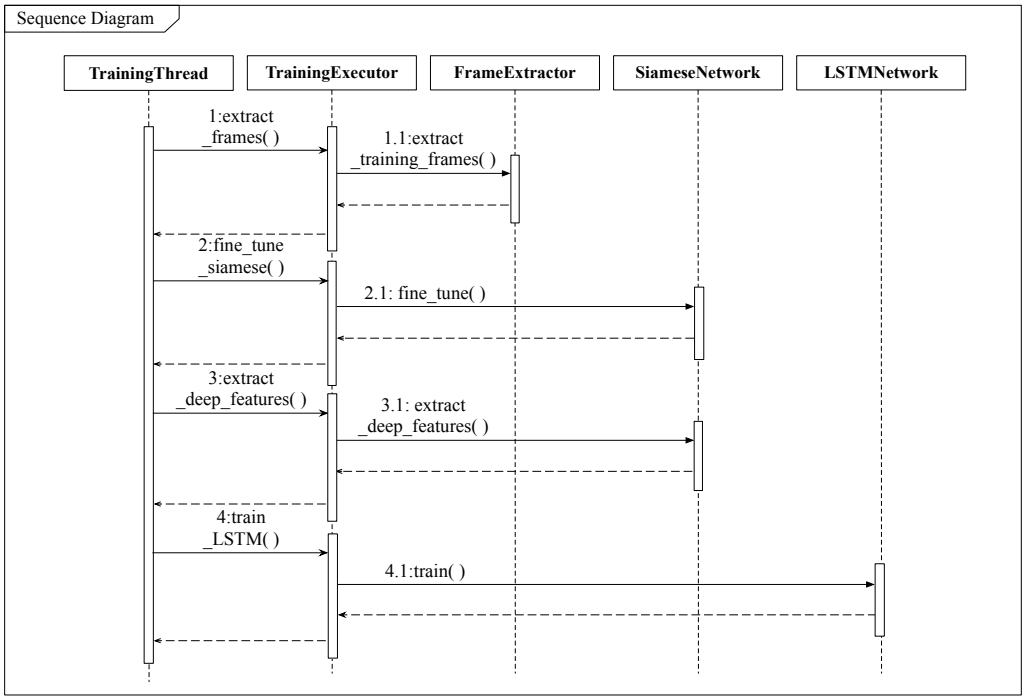


图 4.10: 机器审核模型训练模块局部顺序图

的过程中，TrainingExecutor 类会临时保存它们的中间产物，持久化存储训练产生的 Siamese 网络和 LSTM 网络的信息。

（2）关键代码

TrainingThread 类作为专门用于模型训练工作的线程类，重写了 Thread 类的 run 方法。在 run 方法中，TrainingThread 类通过调用 TrainingExecutor 类的方法，完成了一系列与模型训练有关的操作。同时，TrainingThread 类还对系统运行时可能出现的异常情况进行了处理。如果系统运行时出现了异常，TrainingThread 类会将模型训练失败的信息封装起来，传递给 TrainingExecutor 类。TrainingExecutor 类则会存储模型的失败信息。TrainingThread 类的实现代码如代码段 4.1 所示。

LstmNetwork 类中训练 LSTM 网络的代码如代码段 4.2 所示。在代码中，LstmNetwork 类首先通过调用自身的方法，搭建了一个 LSTM 网络。之后，它为搭建的网络设置了损失函数和优化算法。最后，它配置了训练数据生成器、Epoch 个数等的训练参数，并开始基于这些参数训练模型。模型训练完成后，LstmNetwork 类会先保存模型，再将模型训练成功的信息封装到起来，传递给 TrainingExecutor 类。TrainingExecutor 类则会存储模型的成功信息。

代码段 4.1: TrainingThread 类的实现代码

```
class TrainingThread(threading.Thread):
    def __init__(self, executor):
        super().__init__()
        self.__training_executor=executor
    def run(self):
        try:
            self.__training_executor.extract_frames()
            self.__training_executor.fine_tune_siamese()
            self.__training_executor.extract_deep_features()
            self.__training_executor.train_lstm()
        except:
            current_time=datetime.now()
            model_info={'is_finished':True,'finished_time':current_time,'has_error':
                True}
            self.__training_executor.save_model_info(model_info)
```

代码段 4.2: LstmNetwork 类的实现代码

```
class LstmNetwork:
    # Omitting the definition of class attributes and other methods
    def __create_training_model(self,frames_num):
        self.__model=Sequential()
        self.__model.add(LSTM(units=self.__UNITS_NUM,
                              input_shape=(frames_num,self.__FEATURES_DIM),
                              return_sequences=False))
        self.__model.add(Dense(1))
    def train(self, feature_contents, scores):
        video_num=len(scores)
        frames_num=len(feature_contents[0])
        self.__create(frames_num)
        self.__model.compile(loss='mean_squared_error',optimizer='adam')
        generator=LstmDataGenerator(feature_contents,
                                    scores, self.__VIDEOS_NUM_PS)
        self.__model.fit_generator(generator=generator.generate_data(),
                                   use_multiprocessing=True,
                                   steps_per_epoch=video_num/self.__VIDEOS_NUM_PS,
                                   epochs=self.__EPOCHS_NUM)
        current_time = datetime.now()
        model_content=self.__MODEL_DIR+str(current_time)
        self.__model.save(model_content)
        self.__model_info={'is_finished':True,'finished_time':current_time,'has_error
            ':False,'content_2':model_content,'frames_extracted_num':frames_num}
        return self.__model_info
```

LstmNetwork 类训练模型时，会通过 LstmDataGenerator 类获取当前批次的训练数据。在 LstmDataGenerator 类中，生成训练数据的函数被定义为了生成器函数。LstmNetwork 类每次获取训练数据时，都会调用这个生成器函数返回的生成器。生成器被调用后，会随机选取一定数量的 UGC 视频，读取它们的深度特征，之后，连同它们的质量分数一起返回给 LstmNetwork 类。LstmDataGenerator 类中生成训练数据的代码如代码段 4.3 所示。

代码段 4.3: LstmDataGenerator 类的实现代码

```
class LstmDataGenerator:
    # Omitting the definition of other methods
    def generate_training_data(self):
        video_num=len(self.scores)
        frames_num=len(self.feature_contents[0])
        while True:
            index_list=random.sample(range(video_num),self.videos_num_ps)
            data_x=[]
            data_y=[]
            for i in index_list:
                features=[]
                for j in range(frames_num):
                    feature=np.load(self.feature_contents[i][j])
                    features.append(feature)
                data_x.append(features)
                data_y.append(self.scores[i])
            data_x=np.array(data_x)
            data_y=np.array(data_y)
            yield data_x,data_y
```

4.4.2　人工审核模块设计与实现

人工审核模块完成了人工审核任务管理、人工审核结果分析和视频质量评分三个主要功能。通过这个模块，超级审核人员既可以创建人工审核任务，又可以在任务完成后，查看人工审核结果及系统对结果的分析，使用任务中的 UGC 短视频扩充训练数据集。普通审核人员则既可以对任务中的 UGC 短视频进行质量评分，又可以在评分完成后，提交人工审核结果。同时，这个模块也为超级审核人员和普通审核人员提供了查看人工审核任务的功能。

在人工审核模块中，我们实现了超级审核人员和普通审核人员之间的通信与交互。当收到超级审核人员的人工审核任务创建请求后，我们会将创建的任务分发给所有普通审核人员。而当收到普通审核人员的人工审核结果提交请求

后，我们也会将提交的结果告知给所有超级审核人员。

（1）详细设计

人工审核模块的类图如图 4.11 所示。在这个图中，我们详细描述了人工审核模块中涉及到的类以及类与类之间的关系。

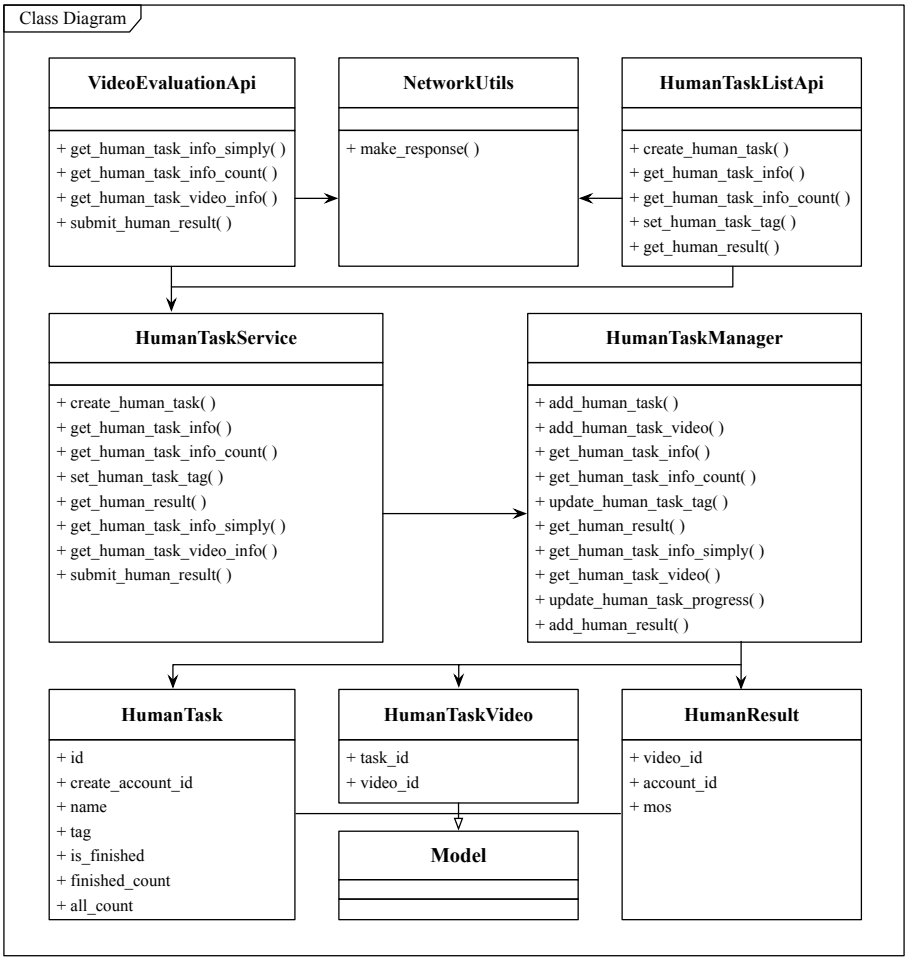


图 4.11: 人工审核模块类图

图 4.11 中各个类的职责如下。HumanTaskListApi 类和 VideoEvaluationApi 类分别负责接收人工审核任务列表页面和视频质量评分页面发出的网络请求。HumanTaskService 类负责对收到的网络请求进行处理，这个类依赖 HumanTaskManager 类，并通过它提供与人工审核有关的服务。HumanTaskManager 类负责管理 HumanTask、HumanTaskVideo 和 HumanResult 实体类，进行与实体类相关的操作。Model 类是所有实体类的基类。NetworkUtils 类是网络工具类。

在图 4.12 所示的顺序图中，我们以人工审核结果的提交和获取为例，展示了人工审核模块中的相关类是如何协作完成超级审核审核和普通审核人员之间

的通信与交互的。

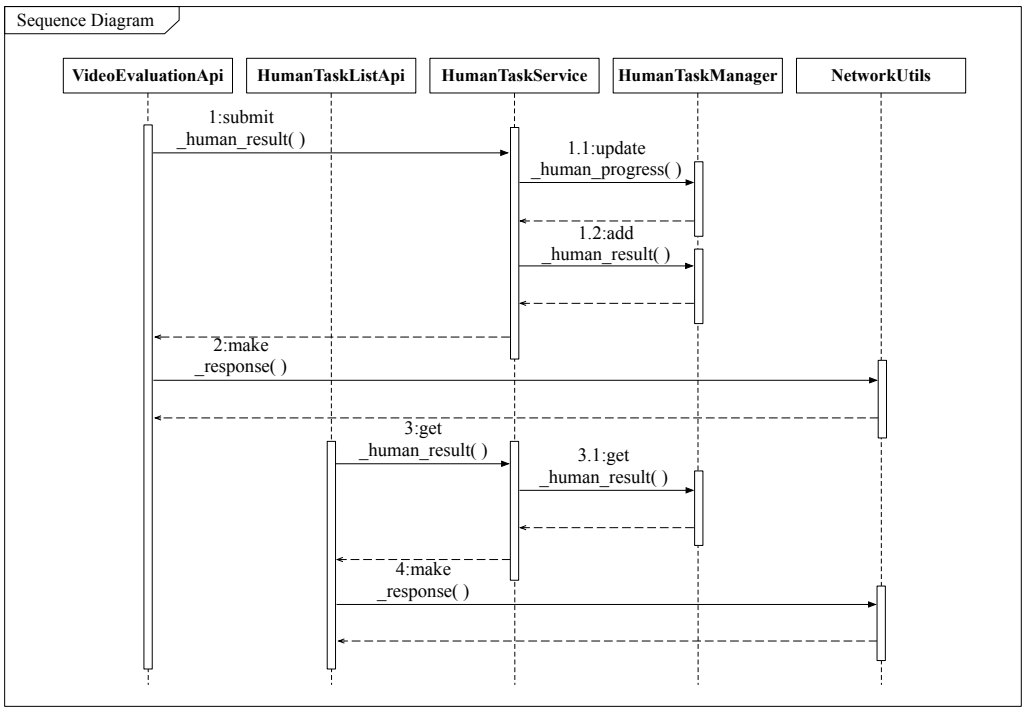


图 4.12: 人工审核模块局部顺序图

在普通审核人员发出人工审核结果提交请求时，VideoEvaluationApi 类会先对其进行解析。之后，这个类会将解析出的结果传递给 HumanTaskService 类。HumanTaskService 类则会通过调用 HumanTaskManager 类的相关方法，更新数据库中的数据。其中，第一个方法用于修改数据库中人工审核任务的进度信息，第二个方法用于向数据库中添加人工审核结果。普通审核人员提交的人工审核结果被存储到数据库中后，超级审核人员即可通过这个模块对其进行获取。在超级审核人员发出人工审核结果查看请求时，HumanTaskService 类会通过调用 HumanTaskManager 类的相关方法，从数据库中读取所有人工审核结果。

（2）关键代码

在这个系统中，我们通过 Flask-SQLAlchemy 扩展实现了 ORM。对于多表查询，这个扩展提供了一种基于外键的关联查询方法。在数据库操作中，外键通常被用于建立两个数据表之间的关联。下面，我们就以读取人工审核结果为例，展示这种查询的具体实现方法。

在 HumanTaskVideo 类中，video_id 作为 human_task_video 数据表的外键，关联了 testing_video 数据表的主键。为对这两个数据表进行关联查询，我们在 HumanTaskVideo 类中创建了一个关联属性 video，并将这个关联属性的关联类设置

为 Video4Testing。这样，通过 video 属性，我们就可以方便地访问到 testing_video 数据表中的数据。HumanTaskVideo 类的实现代码如代码段 4.4 所示。

代码段 4.4: HumanTaskVideo 类的实现代码

```
class HumanTaskVideo(db.Model):
    task_id=db.Column(db.Integer,db.ForeignKey('human_task.id'),
        nullable =False)
    video_id=db.Column(db.Integer,db.ForeignKey('video_testing.id'),
        nullable =False)
    video=db.relationship('Video4Testing',
        backref=db.backref('human_task_video'))
```

同样地，我们在 HumanResult 类中也创建了一个关联属性 account，以便于访问 account 数据表中的数据。HumanResult 类的实现代码如代码段 4.5 所示。

代码段 4.5: HumanResult 类的实现代码

```
class HumanResult(db.Model):
    video_id=db.Column(db.Integer,db.ForeignKey('video_testing.id'),
        nullable =False)
    account_id=db.Column(db.Integer,db.ForeignKey('account.id'),
        nullable =False)
    mos=db.Column(db.Float,nullable=False)
    account=db.relationship('Account',
        backref=db.backref('human_results'))
```

上面的关联属性被创建后，我们便可以在读取人工审核结果时，进行基于外键的关联查询。通过 HumanTaskVideo 类的 video 属性，我们可以方便地获取到人工审核任务的 UGC 短视频信息。通过 HumanResult 类的 account 属性，我们可以方便地获取到对 UGC 短视频进行评分的账号信息。HumanTaskManager 类中读取人工审核结果的代码如代码段 4.6 所示。

4.4.3　机器审核模块设计与实现

机器审核模块完成了机器审核任务管理和机器审核结果分析两个主要功能。通过这个模块，超级审核人员既可以创建机器审核任务，又可以在任务完成后，查看机器审核结果及系统对结果的分析，筛选低质 UGC 短视频。同时，这个模块也为超级审核人员提供了查看机器审核任务的功能。

在机器审核模块中，当收到超级审核人员的机器审核任务创建请求后，我们会使用机器审核模型训练模块中训练好的模型预测任务中 UGC 短视频的分数，这其中包括测试 Siamese 网络和测试 LSTM 网络。

代码段 4.6: HumanTaskManager 类的部分实现代码

```
class HumanTaskManager:
    # Omitting the definition of other methods
    def get_human_result(task_id):
        task_videos=HumanTaskVideo.query.filter(
            HumanTaskVideo.task_id==task_id).all()
        task_video_dicts=[]
        for a_task_video in task_videos:
            task_video_dict={'video_id':a_task_video.video_id,
                             'video_name':a_task_video.video.name}
            human_results=HumanResult.query.filter(
                HumanResult.video_id==a_task_video.video_id).all()
            account_ids=[]
            scores =[]
            account_names=[]
            for a_human_result in human_results:
                account_ids.append(a_human_result.account_id)
                scores .append(a_human_result.mos)
                account_names.append(a_human_result.account.name)
            task_video_dict['account_ids']=account_ids
            task_video_dict[' scores ']=scores
            task_video_dict['account_names']=account_names

            task_video_dicts.append(task_video_dict)

        return task_video_dicts
```

（1）详细设计

机器审核模块的类图如图 4.13 所示。在这个图中，我们详细描述了机器审核模块中涉及到的类以及类与类之间的关系。由于篇幅的限制，我们省略了模块中的实体类。

图 4.13 中各个类的职责如下。MachineTaskListApi 类负责接收机器审核任务列表页面发出的网络请求。MachineTaskService 类负责对收到的网络请求进行处理，它依赖 MachineTaskManager 类和 TestingExecutor 类，并通过这两个类提供与机器审核有关的服务。其中，MachineTaskManager 类负责管理实体类，进行与实体类相关的操作。NetworkUtils 类是网络工具类。

与机器审核模型训练模块较为类似，在这个模块中，TestingThread 类是专门用于模型测试工作的线程类。TestingExecutor 类负责管理 FrameExtractor 类、SiameseNetwork 类和 LstmNetwork 类，进行与机器审核模型测试有关的操作。FrameExtractor 类负责对 UGC 短视频进行抽帧，SiameseNetwork 类负责使用

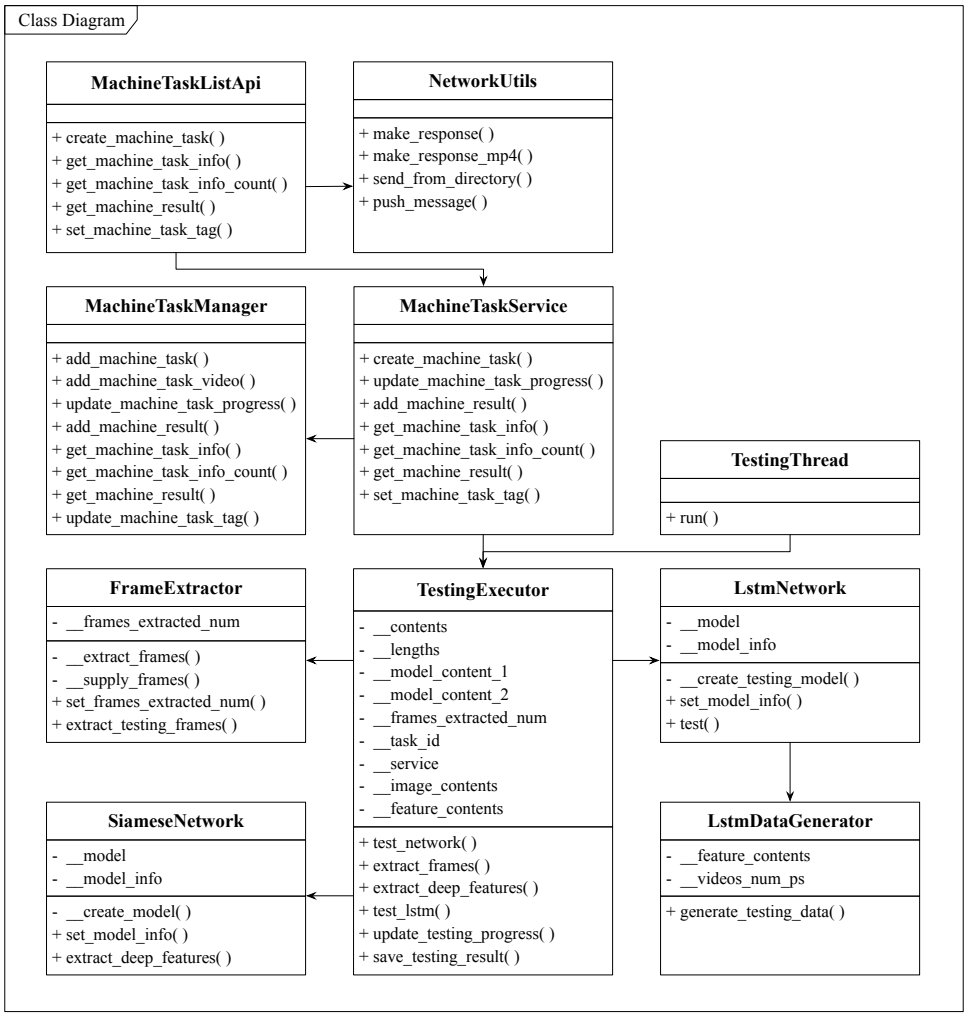


图 4.13: 机器审核模块类图

Siamese 网络提取 UGC 短视频的深度特征，LstmNetwork 类负责使用 LSTM 网络预测 UGC 短视频的最终质量。LstmDataGenerator 类则负责为 LstmNetwork 类提供预测数据。

当收到超级审核人员的机器审核任务创建请求后，机器审核模块会创建一个子线程，用于异步执行模型测试操作，包括 UGC 短视频抽帧、提取 UGC 短视频的深度特征和预测 UGC 短视频的最终质量。在图 4.14 所示的顺序图中，我们展示了预测 UGC 短视频的最终质量时，机器审核模块中的相关类是如何进行协作的。

TestingExecutor 类将预测 UGC 短视频最终质量的操作委托给 LstmNetwork 类后，LstmNetwork 类会先调用自身的方法，加载 LSTM 网络。之后，这个类会分批次预测 UGC 短视频的最终质量。在每一个批次中，LstmNetwork 类会先调

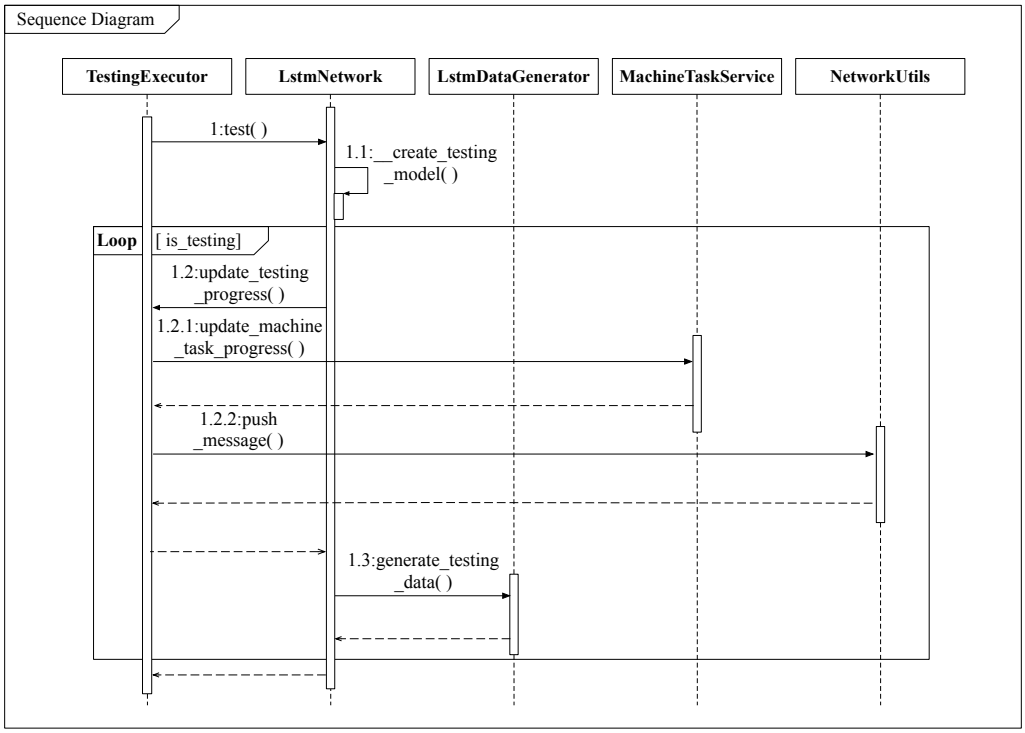


图 4.14: 机器审核模块局部顺序图

用 TestingExecutor 类的相关方法，更新测试进度，再调用 LstmDataGenerator 类的相关方法，获取本批次的预测数据，进而使用加载出的网络预测本批次数据的最终质量。在更新测试进度时，TestingExecutor 类会调用 MachineTaskService 类和 NetworkUtils 类的相关方法，前者用于修改数据库中机器审核任务的进度信息，后者用于将测试进度推送给系统前端。

（2）关键代码

为向超级审核人员实时反馈机器审核任务的完成进度，我们不仅在机器审核任务列表页面发出机器审核任务查看请求时，为其返回包含机器审核任务进度信息的响应，还通过 WebSocket 协议主动向其推送测试进度。WebSocket 协议是一个在单个 TCP 连接上提供全双工通信通道的网络协议。Flask-SocketIO 扩展加入了对 WebSocket 协议的支持。通过这个扩展，我们可以在系统的前端和服务端之间建立长连接，实现服务端向前端的数据推送。

在具体实现时，FrameExtractor 类、SiameseNetwork 类和 LstmNetwork 类每完成一部分模型测试操作，都会向 TestingExecutor 类传递当前的测试进度。对于传递过来的测试进度，TestingExecutor 类一方面会通过 MachineTaskService 类保存到数据库中，另一方面会进行封装，并通过 NetworkUtils 类推送给系统前端。系统前端在收到服务端推送的测试进度后，会进行实时地进度展示。TestingExecutor

类中处理测试进度的代码如代码段 4.7 所示。

代码段 4.7: TestingExecutor 类的实现代码

```
class TestingExecutor:
    # Omitting the definition of other methods
    def update_testing_progress(self, finished_percent):
        self.___service.update_machine_task_progress(finished_percent)
        data={'id': self.___task_id,'finished_percent': finished_percent}
        NetworkUtils.push_message('progress',data,'/getMachineTaskProgress')
```

NetworkUtils 类中向系统前端推送消息的代码如代码段 4.8 所示。在代码中，NetworkUtils 类调用了 SocketIO 类的方法，这个方法的第一个参数是事件名，第二参数是需要推送的数据，第三个参数是命名空间。

代码段 4.8: NetworkUtils 类的实现代码

```
class NetworkUtils:
    # Omitting the definition of class attributes and other methods
    @classmethod
    def push_message(cls,event_name,data,namespace):
        cls.socketio.emit(event_name,data,namespace=namespace)
```

4.4.4　账号管理模块设计与实现

账号管理模块完成了账号管理这个主要功能。通过这个模块，超级审核人员既可以查看系统中的审核人员账号，又可以为普通审核人员账号授予超级审核人员权限。普通审核人员则可以注册普通审核人员账号。同时，这个模块也为超级审核人员和普通审核人员提供了登录、退出和注销账号的功能。

在账号管理模块中，我们预置了一个超级审核人员账号。普通审核人员注册普通审核人员账号后，超级审核人员可以通过为其普通审核人员账号授予超级审核人员权限，将其设置为超级审核人员。

鉴于账号管理模块的逻辑与其他模块相比较为简单，我们不再展示这个模块的类图和顺序图，仅列出这个模块的部分代码。

（1）关键代码

在账号管理模块中，我们通过 Flask-Login 扩展实现了用户认证。为使用这个扩展提供的登录、退出、认证保护等功能，我们首先注册了一个用户加载回调函数。之后，我们让存储用户信息的 Account 类继承了这个扩展中的 UserMixin 类。这样，我们就可以使用 login_required 装饰器拒绝未登录用户的网络请求。

除此之外，我们还使用密码散列值的方式对用户的密码进行了安全保护。Werkzeug.security 模块内置了用于生成和验证密码散列值的函数。在 Account 类中，我们定义了两个方法，第一个方法通过这个模块生成给定密码的散列值并为其赋值，第二个方法通过这个模块验证给定密码与散列值是否一致。Account 类的实现代码如代码段 4.9 所示。

代码段 4.9: Account 类的实现代码

```
class Account(db.Model,UserMixin):
    id=db.Column(db.Integer,primary_key=True)
    name=db.Column(db.String(200),unique=True,nullable=False)
    password=db.Column(db.String(200), nullable =False)
    is_manager=db.Column(db.Boolean,nullable=False)

    def set_password( self ,password):
        self .password=generate_password_hash(password)

    def validate_password( self ,password):
        return check_password_hash(self.password,password)
```

4.5　本章小结

本章我们按照软件开发的基本步骤，详细介绍了 UGC 短视频质量评审系统的搭建流程。首先，我们列出了系统的功能性需求和非功能性需求。其次，我们通过 4+1 视图描述了系统的架构设计。再次，我们给出了数据库的详细设计信息。最后，我们展示了机器审核模型训练模块、人工审核模块、机器审核模块、账号管理模块的类图和顺序图，列出了这四个模块的关键代码。

第五章　实验设计与系统测试

5.1　实验设计

为对本文提出的 UGC 视频质量评估算法进行验证，我们设计了两个实验。实验一为消融实验，主要验证算法中每一个步骤的必要性；实验二为交叉实验，主要验证算法的泛化能力。

5.1.1　实验一：消融实验

在本文提出的算法中，我们首先基于大规模的无标签失真图像数据集，训练了一个能够分辨图像相对质量的 Siamese 网络。之后，我们在 UGC 视频质量数据集上对其进行了微调。最后，我们使用从 Siamese 网络中提取出的深度特征，训练了一个用于预测 UGC 视频最终质量的 LSTM 网络。

为了验证算法中训练 Siamese 网络、微调 Siamese 网络和训练 LSTM 网络的必要性，我们分别对删除这三个步骤后的算法进行了测试。表 5.1 给出了它们在 KoNViD-1k 数据集上的测试结果。

表 5.1: 消融实验结果

算法	PLCC	SROCC
删除训练 Siamese 网络后的算法	0.747	0.746
删除微调 Siamese 网络后的算法	0.736	0.746
删除训练 LSTM 网络后的算法	0.744	0.750
完整算法	**0.788**	**0.789**

从表中可以看出，删除这三个步骤后算法的 PLCC 和 SROCC 指标都明显低于完整算法的指标。这表明删除算法中的任何一个步骤都会影响算法的整体性能。因此，算法中的每一个步骤都是必不可少的。

5.1.2　实验二：交叉实验

在以往的研究中，交叉实验通常被用来评估 UGC 视频质量评估算法的泛化能力。所以，为验证本文所提算法的泛化能力，我们基于 KoNViD-1k 数据集训练了一个 UGC 视频质量评估模型，并使用这个模型评估 LIVE-VQC 数据集

中 UGC 视频的质量。评估完成后，我们计算出了算法的 PLCC 和 SROCC 指标。表 5.2 列出了本文所提算法与其他领先算法的对比结果。

表 5.2: 与其他算法的对比结果

算法	PLCC	SROCC
NIQE	0.629	0.596
ILNIQE	0.544	0.504
VIIDEO	0.215	0.033
我们的算法	**0.653**	**0.627**

通过对比结果可以看出，虽然我们没有使用 LIVE-VQC 数据集训练 UGC 视频质量评估模型，但这个模型仍然在 LIVE-VQC 数据集上表现出了领先的性能。因此，交叉实验的结果证明我们的算法具有良好的泛化能力。

5.2　系统测试

UGC 短视频质量评审系统开发完成后，我们对其进行了功能性测试和非功能性测试。功能性测试的主要目标是验证系统是否可以满足系统需求分析中的功能性需求，非功能性测试的主要目标是验证系统是否可以满足系统需求分析中的非功能性需求。

5.2.1　测试环境

表 5.3 列出了我们测试时系统的软硬件环境。其中，在硬件环境方面，我们根据服务器扮演的不同角色为其选取了不同的配置。因为系统后台涉及神经网络的训练和测试，所以我们将系统后台部署在了两台装有 NVIDIA 显卡的高性能服务器上。我们同样也根据反向代理服务器和数据库服务器的特性对它们进行了配置。反向代理服务器作为所有网络请求的入口和出口，对网络带宽有较高的要求。而对于数据库服务器而言，需要有较大的磁盘存储空间。

5.2.2　功能性测试

在进行功能性测试时，我们针对系统逻辑视图中的四个模块设计了四组测试用例。这四组测试用例均基于系统需求分析中的用例描述设计，覆盖了用例描述中大部分的正常流程和扩展流程。我们的每个测试用例都由测试步骤与预期结果组成，若系统的实际运行结果与预期结果一致，则测试通过，否则测试不

表 5.3: 测试时的软硬件环境

	硬件环境	软件环境
用户计算机	CPU 核数 4，内存大小 8GB，带宽 10Mbps	Windows 10，Chrome 89.0
反向代理服务器	CPU 核数 4，内存大小 8GB，上行带宽 149Mbps，下行带宽 65Mbps	Ubuntu 18.04，Nginx 1.14.0
后台应用服务器	CPU 核数 6，内存大小 64GB，显卡型号 NVIDIA GeForce RTX 2080 Ti	Ubuntu 18.04，Python 3.7，Gunicorn 20.0.4，Flask 1.1.1，Docker 20.10.5
数据库服务器	CPU 核数 4，内存大小 8GB，硬盘大小 200GB	Ubuntu 18.04，MySQL 5.7.33

通过。我们按照测试用例中的测试步骤对系统进行了测试，最终系统通过了所有的测试用例。因此，我们的系统满足 4.2.1 中的功能性需求。

机器审核模型训练模块的测试用例如表 5.4 所示，它们分别用来测试系统查看训练数据、添加训练数据、删除训练数据、训练机器审核模型和查看机器审核模型的功能。

表 5.4: 机器审核模型训练模块测试用例

ID	测试功能	测试步骤	预期结果
TC1-1	查看训练数据	测试人员进入训练数据集管理页面	系统显示训练数据集中包含的 UGC 视频及其相关信息
TC1-2	添加训练数据	测试人员点击 UGC 视频上传按钮；测试人员上传 UGC 视频；测试人员点击 UGC 视频信息上传按钮；测试人员上传 UGC 视频信息	系统弹出文件选择对话框；系统添加这些 UGC 视频；系统弹出文件选择对话框；系统添加这些 UGC 视频信息
TC1-3	删除训练数据	测试人员选择部分或全部 UGC 视频，点击删除按钮	系统删除这些 UGC 视频及其相关信息
TC1-4	训练机器审核模型	测试人员从已上传人工评分的 UGC 视频中选择部分或全部视频，点击训练按钮	系统提示模型已开始训练
TC1-5	查看机器审核模型	测试人员进入模型列表页面	系统显示所有模型的信息

人工审核模块的测试用例如表 5.5 所示。它们分别用来测试系统创建人工审核任务、查看人工审核任务、分析人工审核结果和视频质量评分的功能。

表 5.5: 人工审核模块测试用例

ID	测试功能	测试步骤	预期结果
TC2-1	创建人工审核任务	测试人员进入用户上传视频列表页面，从未创建人工审核任务的 UGC 短视频中选择部分或全部视频，点击创建人工审核任务按钮	系统提示人工审核任务已创建
TC2-2	查看人工审核任务	测试人员进入人工审核任务列表页面	系统显示所有人工审核任务的信息
TC2-3	分析人工审核结果	测试人员从已完成的人工审核任务中选择一个任务，点击查看结果及结果分析按钮；测试人员选择部分或全部 UGC 短视频，点击添加至训练数据集按钮	系统显示这个人工审核任务的结果及对结果的分析；系统将这些 UGC 短视频添加至训练数据集中，同时对它们进行标记
TC2-4	视频质量评分	测试人员进入视频质量评分页面，从未完成的人工审核任务中选择一个任务，点击开始审核按钮；测试人员输入这些 UGC 短视频的质量分数；测试人员点击提交按钮	系统显示这个人工审核任务中包含的 UGC 短视频；系统显示这些 UGC 短视频的质量分数；系统提示质量分数已提交

机器审核模块的测试用例如表 5.6 所示。它们分别用来测试系统创建机器审核任务、查看机器审核任务和分析机器审核结果的功能。

表 5.6: 机器审核模块测试用例

ID	测试功能	测试步骤	预期结果
TC3-1	创建机器审核任务	测试人员进入用户上传视频列表页面，从未添加至训练数据集的 UGC 短视频中选择部分或全部视频，点击创建机器审核任务按钮；测试人员选择一个模型，点击确定按钮	系统弹出模型选择对话框；系统提示机器审核任务已创建
TC3-2	查看机器审核任务	测试人员进入机器审核任务列表页面	系统显示所有机器审核任务的信息
TC3-3	分析机器审核结果	测试人员从已完成的机器审核任务中选择一个任务，点击查看结果及结果分析按钮；测试人员输入低质 UGC 短视频的筛选条件，点击标记为低质视频按钮	系统显示这个机器审核任务的结果及对结果的分析；系统对筛选出的低质 UGC 短视频进行标记

账号模块的测试用例如表 5.7 所示。它们分别用来测试系统登录账号、查看

账号、授予权限、删除账号和退出账号的功能。

表 5.7: 账号管理模块测试用例

ID	测试功能	测试步骤	预期结果
TC4-1	登录账号	测试人员进入登录页面，输入超级审核人员账号的用户名和密码，点击登录按钮	系统提示登录成功，同时跳转到用户上传视频列表页面
TC4-2	查看账号	测试人员进入账号管理页面	系统显示所有账号的信息
TC4-3	授予权限	测试人员从普通审核人员账号中选择一个账号，点击授予超级审核人员权限的按钮	系统将这个账号修改为超级审核人员账号
TC4-4	删除账号	测试人员从普通审核人员账号中选择一个账号，点击删除按钮	系统删除这个账号的信息
TC4-5	退出账号	测试人员点击退出按钮	系统提示退出成功，同时跳转到跳转到登录页面

5.2.3　非功能性测试

为保障 UGC 短视频质量评审系统的性能和质量，我们对其进行了非功能性测试，测试的主要内容包括系统的速度、负载、可靠性、安全性、易用性等。

在对系统的速度和负载进行测试时，我们从系统的所有接口中选取了 6 个关键接口，并使用 JMeter[1] 工具模拟了用户对它们的并发请求。JMeter 是使用 Java 编写的开源测试软件，主要用于分析和衡量 Web 应用的性能。表 5.8 列出了我们测试的 6 个接口，同时也记录了并发量为 100 时每个接口的平均响应时间、中位响应时间、最小响应时间和最大响应时间。从表中可以看出，所有测试接口的平均响应时间都小于 1 秒，所有测试接口的最大响应时间都小于 2 秒。因此，我们的系统可以满足 4.2.2 中对系统速度和负载的要求。

为验证系统的可靠性和安全性，我们有针对性地设计了几个测试用例，包括在断网条件下访问系统页面，使用普通审核人员账号进行除视频质量评分以外的操作等，系统的运行结果均满足预期。

我们对系统易用性的测试主要围绕系统审核视频的速度展开。LIVE-VQC 数据集上的测试结果表明，在机器审核模式下，数据集中所有 UGC 视频的平均审核时间为 11.7 秒。按此审核速度计算，超级审核人员每天可以审核 7000 多个 UGC 短视频，这符合系统的易用性需求。

[1] https://jmeter.apache.org

表 5.8: 系统速度和负载测试的结果

接口	并发量	平均时间	中位时间	最小时间	最大时间
上传 UGC 视频	100	428ms	404ms	43ms	1164ms
获取 UGC 视频信息	100	760ms	668ms	119ms	1383ms
创建人工审核任务	100	512ms	487ms	105ms	991ms
获取人工审核任务信息	100	127ms	133ms	11ms	206ms
创建机器审核任务	100	470ms	432ms	94ms	1056ms
获取机器审核结果	100	948ms	907ms	275ms	1743ms

5.2.4　测试结果展示

为展现系统的真实运行情况，在本节中我们给出了测试时部分系统页面的截图。

图 5.1 为执行测试用例 TC1-1 时训练数据集管理页面的截图。这个页面用于向超级审核人员展示训练数据集中包含的 UGC 视频及其相关信息，包括视频的名称、视频的时长、视频的分辨率、视频的人工评分等。同时，通过点击添加视频和上传视频信息按钮，超级审核人员也可以将更多的训练数据添加到训练数据集中。

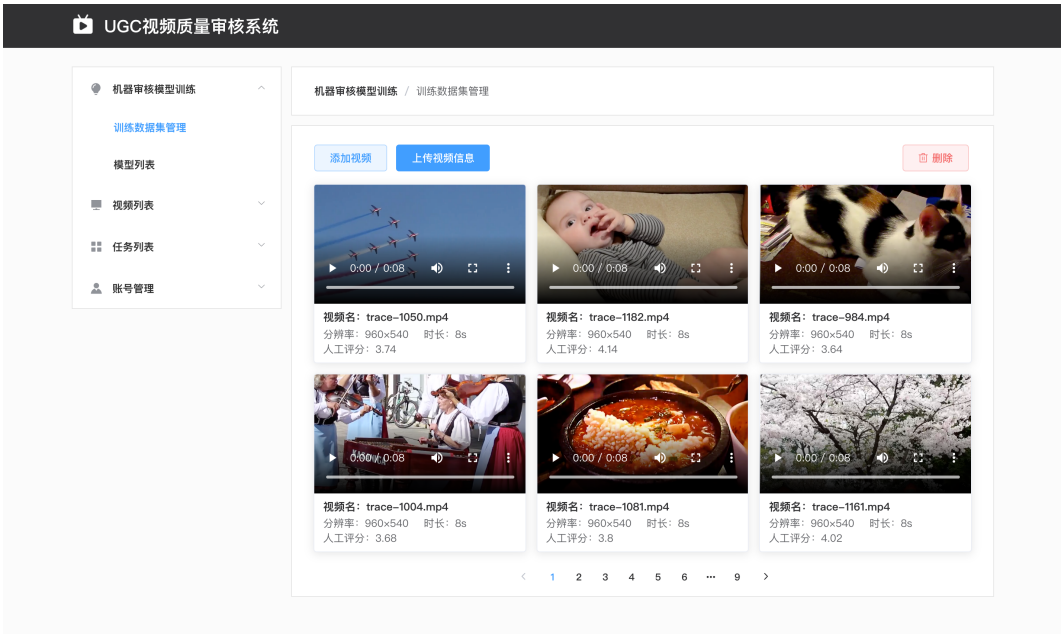


图 5.1: 训练数据集管理页面

图 5.2 为执行测试用例 TC2-2 时人工审核任务列表页面的截图。这个页面用于向超级审核人员展示所有人工审核任务的信息，包括任务的名称、已完成

任务的人数、待完成任务的人数、任务的创建时间、任务的创建者等。通过点击已完成任务的查看结果和查看结果分析按钮，超级审核人员可以查看这个任务的结果及系统对结果的分析。

图 5.2: 人工审核任务列表页面

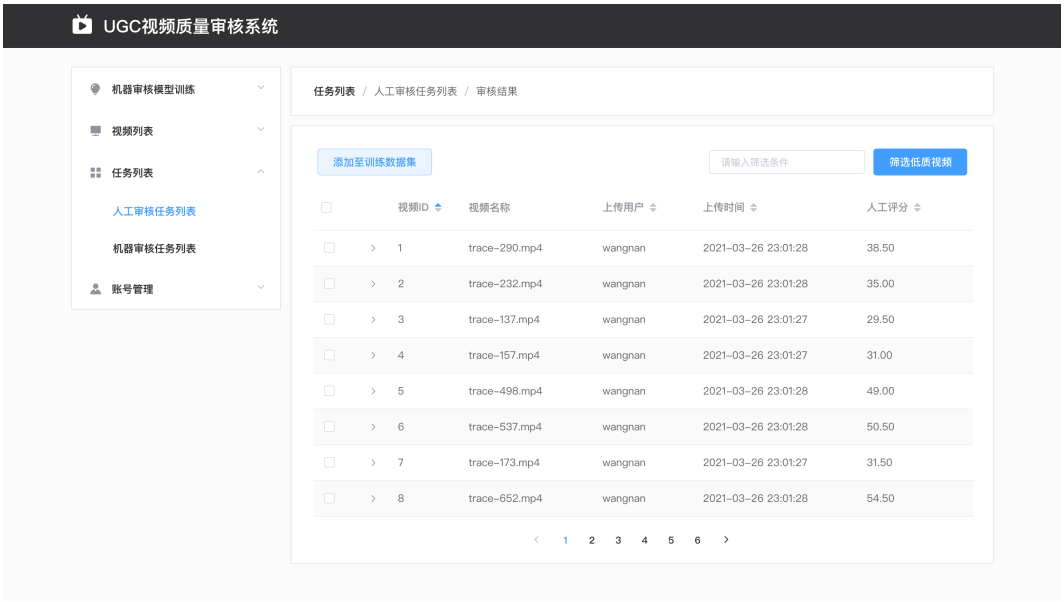


图 5.3: 人工审核任务列表页面

图 5.3 为执行测试用例 TC2-3 时人工审核任务列表页面的截图。点击已完成任务的查看结果按钮后，这个页面会展示任务中所有 UGC 短视频的人工评分

均值。展开页面表格中的行后，超级审核人员可以查看每个普通审核人员对当前行中 UGC 短视频的评分情况。在这个页面中，超级审核人员也可以添加训练数据或筛选低质视频。

图 5.4 为执行测试用例 TC3-3 时机器审核任务列表页面的截图。点击已完成任务的查看结果分析按钮后，这个页面会以扇形图的形式展示任务中所有 UGC 短视频机器评分的分布情况。若任务中所有 UGC 短视频均已完成人工审核，这个页面还会展示机器审核模型的 PLCC 和 SROCC 指标，以及任务中所有 UGC 短视频机器评分和人工评分差值的分布情况。

图 5.4: 机器审核任务列表页面

5.3　本章小结

本章可以划分为实验设计和系统测试两个部分。在实验设计部分，我们分别介绍了消融实验和交叉实验的实验配置，分析了这两个实验的实验结果。在系统测试部分，我们列出了测试时系统的软硬件环境，描述了系统功能性测试和非功能性测试的过程和结果，展示了部分系统测试页面。

第六章　总结与展望

6.1　总结

随着近几年来UGC短视频的大量涌现，人工审核视频质量已难以满足UGC短视频平台用户呈指数增长的优质视频需求，使用机器代替人工审核视频质量、过滤低质视频，已成为UGC短视频质量审核的必然趋势。为帮助平台审核人员高效审核UGC短视频的质量，我们提出了一种基于深度学习UGC视频质量评估算法，并且以此为基础，搭建了一个UGC短视频质量评审系统。

首先，本文阐述了项目的背景与意义，分析了UGC视频质量评估的研究现状，说明了本文的主要工作。

其次，本文详细介绍了UGC视频质量评估算法的设计和实现细节。我们的UGC视频质量评估算法大致可以分为图像相对质量学习、UGC视频深度特征提取、UGC视频质量评估三个部分。在图像相对质量学习部分，我们构造了一个大规模的无标签失真图像数据集，并基于这个数据集训练了一个能够分辨图像相对质量的Siamese网络。在UGC视频深度特征提取部分，我们使用提取的UGC视频帧对Siamese网络的子网络进行微调，并使用微调后的网络提取UGC视频中与空域失真相关的深度特征。在UGC视频质量评估部分，我们训练了一个输出UGC视频最终质量分数的LSTM网络，这个网络被用于捕获UGC视频中的时域失真。

再次，本文详细描述了UGC短视频质量评审系统的搭建流程。我们先分析了UGC短视频质量评审系统的功能性需求和非功能性需求。之后，我们依据系统的主要功能，对系统的总体架构进行了设计，绘制了系统的逻辑视图、开发视图、进程视图和部署视图。最后，我们根据系统逻辑视图，将系统划分为机器审核模型训练、人工审核、机器审核、账号管理四个模块，并分别对这个四个模块进行了详细设计和实现。我们绘制了这四个模块的关键类图和顺序图，列出了这四个模块的核心代码。

最后，为了证明算法中每一个步骤的有效性以及算法的泛化能力，我们分别设计了消融实验和交叉实验。为了验证最终实现的系统是否满足系统需求分析中的功能性需求和非功能性需求，我们对系统进行了功能性测试和非功能性测试。

6.2 展望

UGC 短视频质量评审系统实现了人工审核和机器审核两种审核模式，满足了审核人员的基本需求。然而，在未来，本系统仍然可以从以下方面进行改进。

第一，本系统将训练数据集和机器审核模型直接存储在了服务器的文件系统中。在后续改进时，可以考虑通过 Hadoop 等大数据技术对系统中的文件进行分布式存储，以扩大系统中文件的存储空间。

第二，机器审核模型的训练时间较长，且受限于显卡性能。在后续改进时，可以考虑将系统设计为微服务架构，为机器审核模型训练模块和机器审核模块单独创建一个微服务。

第三，因为目前尚不存在大规模的 UGC 视频质量数据集，所有随着以后更大规模 UGC 视频质量数据集的出现，我们还需要基于这些新的数据集继续优化算法。在后续改进时，可以考虑尝试其他深度学习技术进一步提高 UGC 视频质量的评估效果。

参考文献

[1] V. Hosu, F. Hahn, M. Jenadeleh, H. Lin, H. Men, T. Szirányi, S. Li, D. Saupe, The konstanz natural video database (konvid-1k), in: Ninth International Conference on Quality of Multimedia Experience, QoMEX 2017, Erfurt, Germany, May 31 - June 2, 2017, IEEE, 2017, pp. 1–6.
URL https://doi.org/10.1109/QoMEX.2017.7965673

[2] Z. Sinno, A. C. Bovik, Large-scale study of perceptual video quality, IEEE Trans. Image Process. 28 (2) (2019) 612–627.
URL https://doi.org/10.1109/TIP.2018.2869673

[3] Y. Wang, S. Inguva, B. Adsumilli, Youtube UGC dataset for video compression research, in: 21st IEEE International Workshop on Multimedia Signal Processing, MMSP 2019, Kuala Lumpur, Malaysia, September 27-29, 2019, IEEE, 2019, pp. 1–5.
URL https://doi.org/10.1109/MMSP.2019.8901772

[4] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, L. K. Cormack, Study of subjective and objective quality assessment of video, IEEE Trans. Image Process. 19 (6) (2010) 1427–1441.
URL https://doi.org/10.1109/TIP.2010.2042111

[5] P. V. Vu, D. M. Chandler, Vis3: an algorithm for video quality assessment via analysis of spatial and spatiotemporal slices, J. Electronic Imaging 23 (1) (2014) 013016.
URL https://doi.org/10.1117/1.JEI.23.1.013016

[6] M. Nuutinen, T. Virtanen, M. Vaahteranoksa, T. Vuori, P. Oittinen, J. Häkkinen, CVD2014 - A database for evaluating no-reference video quality assessment algorithms, IEEE Trans. Image Process. 25 (7) (2016) 3073–3086.
URL https://doi.org/10.1109/TIP.2016.2562513

[7] D. Ghadiyaram, J. Pan, A. C. Bovik, A. K. Moorthy, P. Panda, K. Yang, In-capture mobile video distortions: A study of subjective behavior and objective algorithms,

IEEE Trans. Circuits Syst. Video Technol. 28 (9) (2018) 2061–2077.
URL https://doi.org/10.1109/TCSVT.2017.2707479

[8] D. Ghadiyaram, A. C. Bovik, Massive online crowdsourced study of subjective and objective picture quality, IEEE Trans. Image Process. 25 (1) (2016) 372–387.
URL https://doi.org/10.1109/TIP.2015.2500021

[9] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, L. Li, YFCC100M: the new data in multimedia research, Commun. ACM 59 (2) (2016) 64–73.
URL http://doi.acm.org/10.1145/2812802

[10] D. Li, T. Jiang, M. Jiang, Quality assessment of in-the-wild videos, in: L. Amsaleg, B. Huet, M. A. Larson, G. Gravier, H. Hung, C. Ngo, W. T. Ooi (Eds.), Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019, ACM, 2019, pp. 2351–2359.
URL https://doi.org/10.1145/3343031.3351028

[11] J. Chung, Ç. Gülçehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, CoRR abs/1412.3555.
URL http://arxiv.org/abs/1412.3555

[12] Z. Tu, Y. Wang, N. Birkbeck, B. Adsumilli, A. C. Bovik, UGC-VQA: benchmarking blind video quality assessment for user generated content, CoRR abs/2005.14354.
URL https://arxiv.org/abs/2005.14354

[13] Y. Li, S. Meng, X. Zhang, S. Wang, Y. Wang, S. Ma, User-generated video quality assessment: A subjective and objective study, CoRR abs/2005.08527.
URL https://arxiv.org/abs/2005.08527

[14] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
URL https://doi.org/10.1109/TIP.2003.819861

[15] Z. Wang, E. P. Simoncelli, A. C. Bovik, Multiscale structural similarity for image quality assessment, in: The Thrity-Seventh Asilomar Conference on Signals,

Systems & Computers, 2003, Vol. 2, 2003, pp. 1398–1402.
URL https://doi.org/10.1109/ACSSC.2003.1292216

[16] H. R. Sheikh, A. C. Bovik, Image information and visual quality, IEEE Trans. Image Process. 15 (2) (2006) 430–444.
URL https://doi.org/10.1109/TIP.2005.859378

[17] S. Li, F. Zhang, L. Ma, K. N. Ngan, Image quality assessment by separately evaluating detail losses and additive impairments, IEEE Trans. Multim. 13 (5) (2011) 935–949.
URL https://doi.org/10.1109/TMM.2011.2152382

[18] M. A. Saad, A. C. Bovik, C. Charrier, Blind prediction of natural video quality, IEEE Trans. Image Process. 23 (3) (2014) 1352–1365.
URL https://doi.org/10.1109/TIP.2014.2299154

[19] A. Mittal, M. A. Saad, A. C. Bovik, A completely blind video integrity oracle, IEEE Trans. Image Process. 25 (1) (2016) 289–300.
URL https://doi.org/10.1109/TIP.2015.2502725

[20] Y. Zhu, Y. Wang, Y. Shuai, Blind video quality assessment based on spatio-temporal internal generative mechanism, in: 2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017, IEEE, 2017, pp. 305–309.
URL https://doi.org/10.1109/ICIP.2017.8296292

[21] X. Li, Q. Guo, X. Lu, Spatiotemporal statistics for video quality assessment, IEEE Trans. Image Process. 25 (7) (2016) 3329–3342.
URL https://doi.org/10.1109/TIP.2016.2568752

[22] L. Kang, P. Ye, Y. Li, D. S. Doermann, Convolutional neural networks for no-reference image quality assessment, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014, IEEE Computer Society, 2014, pp. 1733–1740.
URL https://doi.org/10.1109/CVPR.2014.224

[23] L. Kang, P. Ye, Y. Li, D. S. Doermann, Simultaneous estimation of image quality and distortion via multi-task convolutional neural networks, in: 2015 IEEE International Conference on Image Processing, ICIP 2015, Quebec City, QC, Canada,

September 27-30, 2015, IEEE, 2015, pp. 2791–2795.
URL https://doi.org/10.1109/ICIP.2015.7351311

[24] S. Bianco, L. Celona, P. Napoletano, R. Schettini, On the use of deep learning for blind image quality assessment, Signal Image Video Process. 12 (2) (2018) 355–362.
URL https://doi.org/10.1007/s11760-017-1166-8

[25] X. Liu, J. van de Weijer, A. D. Bagdanov, Rankiqa: Learning from rankings for no-reference image quality assessment, in: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, IEEE Computer Society, 2017, pp. 1040–1049.
URL https://doi.org/10.1109/ICCV.2017.118

[26] W. Kim, J. Kim, S. Ahn, J. Kim, S. Lee, Deep video quality assessor: From spatio-temporal visual sensitivity to a convolutional neural aggregation network, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I, Vol. 11205 of Lecture Notes in Computer Science, Springer, 2018, pp. 224–241.
URL https://doi.org/10.1007/978-3-030-01246-5_14

[27] W. Liu, Z. Duanmu, Z. Wang, End-to-end blind quality assessment of compressed videos using deep neural networks, in: S. Boll, K. M. Lee, J. Luo, W. Zhu, H. Byun, C. W. Chen, R. Lienhart, T. Mei (Eds.), 2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018, ACM, 2018, pp. 546–554.
URL https://doi.org/10.1145/3240508.3240643

[28] Y. Zhang, X. Gao, L. He, W. Lu, R. He, Blind video quality assessment with weakly supervised learning and resampling strategy, IEEE Trans. Circuits Syst. Video Technol. 29 (8) (2019) 2244–2255.
URL https://doi.org/10.1109/TCSVT.2018.2868063

[29] L. Deng, D. Yu, Deep learning: Methods and applications, Found. Trends Signal Process. 7 (3-4) (2014) 197–387.
URL https://doi.org/10.1561/2000000039

[30] G. E. Hinton, S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
URL https://doi.org/10.1162/neco.2006.18.7.1527

[31] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, Recent advances in convolutional neural networks, CoRR abs/1512.07108.
URL http://arxiv.org/abs/1512.07108

[32] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
URL https://doi.org/10.1109/5.726791

[33] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, 2012, pp. 1106–1114.
URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[34] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1409.1556

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, IEEE Computer Society, 2015, pp. 1–9.
URL https://doi.org/10.1109/CVPR.2015.7298594

[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, 2016, pp.

770–778.
URL https://doi.org/10.1109/CVPR.2016.90

[37] D. Chicco, Siamese neural networks: An overview, in: H. M. Cartwright (Ed.), Artificial Neural Networks - Third Edition, Vol. 2190 of Methods in Molecular Biology, Springer, 2021, pp. 73–94.
URL https://doi.org/10.1007/978-1-0716-0826-5_3

[38] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: Closing the gap to human-level performance in face verification, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014, IEEE Computer Society, 2014, pp. 1701–1708.
URL https://doi.org/10.1109/CVPR.2014.220

[39] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
URL https://doi.org/10.1162/neco.1997.9.8.1735

[40] S. Fernández, A. Graves, J. Schmidhuber, An application of recurrent neural networks to discriminative keyword spotting, in: J. M. de Sá, L. A. Alexandre, W. Duch, D. P. Mandic (Eds.), Artificial Neural Networks - ICANN 2007, 17th International Conference, Porto, Portugal, September 9-13, 2007, Proceedings, Part II, Vol. 4669 of Lecture Notes in Computer Science, Springer, 2007, pp. 220–229.
URL https://doi.org/10.1007/978-3-540-74695-9_23

[41] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, in: V. Kurková, Y. Manolopoulos, B. Hammer, L. S. Iliadis, I. Maglogiannis (Eds.), Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III, Vol. 11141 of Lecture Notes in Computer Science, Springer, 2018, pp. 270–279.
URL https://doi.org/10.1007/978-3-030-01424-7_27

[42] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, L. Zhang, Waterloo exploration database: New challenges for image quality assessment models, IEEE Trans. Image Process. 26 (2) (2017) 1004–1016.
URL https://doi.org/10.1109/TIP.2016.2631888

[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1412.6980

[44] J. C. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
URL http://dl.acm.org/citation.cfm?id=2021068

[45] A. Mittal, A. K. Moorthy, A. C. Bovik, No-reference image quality assessment in the spatial domain, IEEE Trans. Image Process. 21 (12) (2012) 4695–4708.
URL https://doi.org/10.1109/TIP.2012.2214050

[46] A. Mittal, R. Soundararajan, A. C. Bovik, Making a ”completely blind” image quality analyzer, IEEE Signal Process. Lett. 20 (3) (2013) 209–212.
URL https://doi.org/10.1109/LSP.2012.2227726

[47] P. Ye, J. Kumar, L. Kang, D. S. Doermann, Unsupervised feature learning framework for no-reference image quality assessment, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012, IEEE Computer Society, 2012, pp. 1098–1105.
URL https://doi.org/10.1109/CVPR.2012.6247789

[48] P. Kruchten, The 4+1 view model of architecture, IEEE Softw. 12 (6) (1995) 42–50.
URL https://doi.org/10.1109/52.469759

致　　谢

时光如梭，两年的研究生生涯即将落下帷幕。在论文即将完成之际，我想对两年来所有帮助过我的人表达最诚挚的谢意。

首先，我要感谢我的导师陈振宇教授。在两年的研究生生涯里，陈老师给予了我很多指导和帮助，他的严格认真敦促我不断成长和进步。陈老师不仅引领我步入科研的大门，而且在我论文选题、论文写作的过程中提出了很多宝贵建议。陈老师及实验室的各位老师在为我们学术研究和工程实践创造良好学习环境的同时，也为我们树立了很好的榜样。

其次，我要感谢实习期间与我一起工作的同事们。企业的实习经历让我有幸见证了本文涉及技术在工业界的实际应用场景。实习期间，我的主管师兄为我提供了与相关领域专家交流和学习的机会，使我受益匪浅。

再次，我要感谢两年来所有帮助过我的同学和朋友。在平时的学习和生活中，我与他们一起探讨问题，交流经验，分享自己的喜怒哀乐。我将永远铭记与他们在一起的快乐时光。

最后，我要感谢我的父母和家人。在十几年的求学生涯里，他们给予了我充分的理解和信任。无论是顺境还是逆境，他们都始终陪伴我左右。他们的默默支持是我前进的最大动力。