

**《软件工程》项目报告**

**题目： 原体育—基于AI的高性能运动检测APP**

**课程名称： 软件工程**

**专业班级： 计卓2001班**

**学 号： U202015282**

**姓 名： 逯润雨**

**同组成员： 苏庆远 U202015270**

**王怡彬 U201914858**

**指导教师： 万胜刚**

**报告日期： 2022.12.8**

**计算机科学与技术学院**

**任 务 书**

**一 总体要求**

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；

2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。

3. 根据需求分析进行总体设计、详细设计、编码与测试等。

**二 基本内容**

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

**1. 问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如NABCD模型，Microsoft Visio，StarUML等工具 (20%)；

**2. 原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；

**3. 编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；

**4．功能创新**：与众不同、特别吸引用户的创新（10%）；

**5. 用户反馈**：包括用户的使用记录，照片，视频等（5%）。

**目 录**

**[1 问题定义 1](#_Toc23015)**

[1.1项目背景与意义 1](#_Toc23689)

[1.2 项目基本目标 3](#_Toc25393)

[1.3 可行性分析 3](#_Toc12504)

[1.4人员管理和项目进度管理 7](#_Toc19255)

**[2 需求分析 10](#_Toc17844)**

[2.1 E-R图、数据流图 10](#_Toc25749)

[2.2 用例图等 11](#_Toc29269)

[2.3 原型系统设计 11](#_Toc30145)

**[3 概要设计和详细设计 14](#_Toc29189)**

[3.1 系统结构 14](#_Toc3034)

[3.2 类图 14](#_Toc24074)

[3.3关键数据结构定义 14](#_Toc4451)

[3.4 关键算法设计 26](#_Toc4207)

[3.5 数据管理说明 32](#_Toc17922)

**[4 实现与测试 35](#_Toc31963)**

[4.1实现环境与代码管理 35](#_Toc26228)

[4.2 关键函数说明 36](#_Toc22097)

[4.3 测试计划和测试用例 41](#_Toc3115)

[4.4 结果分析 46](#_Toc27707)

**[5 总结 47](#_Toc26393)**

[5.1 用户反馈 47](#_Toc2938)

[5.2 全文总结 47](#_Toc19665)

**[6 体会 4](#_Toc5497)8**

**[附录 53](#_Toc16076)**

**1 问题定义**

**1.1项目背景与意义**

**1.1.1 N(Need) 需求**

在疫情时代，面对网课的特殊情况，解决学校体育课对于学生日常锻炼与体育测试上的种种需求，例如对于各种运动项目的合格度检测（一分钟内俯卧撑次数），并防止替考、代考等作弊事项的发生。

目前中学、大学常见的运动方式包括但不限于：运动前热身、仰卧起坐、跳绳、蹲起等等。

针对于现在疫情防控的管理，许许多多大学生、中学生都被困在家中。居家隔离的环境下无法进行正常的体育课的锻炼，所以我们针对以上痛点设计出了此app：可以有效地检测与记录学生的居家运动情况，并使用人脸识别以及手机防抖等多种机制对于作弊进行了有效的防治。

**1.1.2 B(Benefit) 好处**

在疫情时代，满足网课情况下的体育课的正常教学，使学生在家中就可完成各项体育锻炼与测试。项目具体优势有如下几点：

1. **算力要求低，可省去服务器算力负载**

应用直接在手机端即可完成全部流程，设备要求低。不同于当前深度模型大多都在云端进行计算，我们此处采用轻量化以及高性能计算相关知识特定优化卷积等计算，并对产生的计算图进行深度调优，使得模型本身能够在手机端完美运行。

同时速度与精度均满足进行检测要求，目前模型在手机端可以达到30FPS（测试终端处理器为高通骁龙865）的性能，已经达到部分商业公司的优化水平。

1. **模型最终打包大小小，app轻量**

不同于很多模型打包出来就非常大，我们此处最小化所有依赖，剪裁有关动态链接库等，或者手动实现其他的各类依赖，最终将模型部分+可执行程序压缩至20MB以内，完美满足了APP的设计要求。也可开启gpu使用vukan进行加速)，可省去服务器算力负载

不含任何第三方依赖，人体姿态模型手机端最终打包小于<10MB，app轻量，电脑端人脸检测模型最终打包exe 10.3MB。

1. **防止作弊场景**

通过测试前进行人脸识别在一定程度上排除替考、代测等在体育测试中常见的舞弊现象(此处通过人脸识别来进行简单替考检测)。

防止作弊场景，安卓端设计了抖动检测算法，使用人脸检测识别防止中途替考。

1. **检测效果准确**

运动检测逻辑基于神经网络对人体姿态估计实时结果进行判断，可准确鉴别出动作是否合格。同时检测具有防抖机制，可以防止对应异常情况处理。

1. **系统功能完善**

学生端提供简单的本地数据缓存，可以自定义单次运动时长以及模型计算方法等。

同时为教师配备了教师端，方便教师对学生信息进行统一且安全的实时管理。

**1.1.3 C(Competitors) 竞争**

市场上其他体育锻炼检测产品，通常使用重力感应技术，借助配备了陀螺仪的手环等特殊设备完成对于锻炼动作的检测，我们通过使用目标检测等人工智能视觉识别技术，仅需智能手机即可完成整个检测流程，降低了检测设备的要求。

例如任天堂所开发的用于检测运动的健身环，他们根据特殊设备（健身环）内的重力感应器等，基于简单的体感信息对运动进行检测，一方面易欺骗，用户可以通过多种模拟运动的手段对体感检测系统进行欺骗；另一方面健身环作为特殊设备，费用代价过高，不具有普适性。

与腾讯提出的sdk解决方案相比，腾训优图有着一整套关于人体姿态估计的ai套件，准确性方面目前超越我们的方案，但商业费用昂贵，且只是单一的人体姿态估计套件，无法完成对于各项体育运动的检测与判断。

同时，我们提出了完整的端到端的系统服务，从人体姿态估计模型到运动判断逻辑，再到为学生提供服务的app，以及为教师提供服务的后台数据管理系统，整个系统功能完善，可直接落地于实际运用场景，供学校等机构进行体育运动的检测与后台用户的管理。

**1.2 项目基本目标**

课程时间内的基本项目目标为完成设计学生和老师两种用户的使用终端的设计和实现工作，同时实现有效的底层逻辑算法，使得项目成果至少支持若干项运动的检测计数功能并且具备一定的防作弊能力。

**1.2.1 A(Approach) 做法**

1. 基于实时目标检测与姿态估计，完成对于运动项目的合格度检测。
2. 基于人脸识别，完成对于学生真人身份的确认。
3. 所有AI算法均运行在手机端，充分考虑手机芯片算力小、功耗低特点，并可以达到实时效果。
4. 安卓端UI交互，提供对应反馈、交互，并支持连接远程数据库方便教师管理。
5. 基于django框架完成教师端web server实现。
6. 基于MySql数据库完成后台数据统一安全管理

**1.2.2 D(Delivery) 推广**

1. 希望借助社区、借助APP商店等各类开放社区与学校专业机构进行推广。
2. 希望联系到存在疫情的中小学或者大学展开试点，根据对应的反馈进行一定的产品迭代与评估。
3. 希望学院能够首先给与一定的支持，对于封楼地区的大一、大二学生，可以沟通配合好相应的工作

**1.3 可行性分析**

可行性分析，对于我们此次设计的高性能计算与人工智能部分可行性分析如下：

**1.3.1 图像预处理**

在图像识别中，图像质量的好坏直接影响识别算法的设计与效果的精度，因此在对图像分析（特征提取、分割、识别等）前，需要进行预处理。图像预处理的主要目的是消除图像中的无关信息，恢复有用的真实信息，增强有关信息的可检测性、最大限度地简化数据，从而改进特征提取、图像分割、匹配和识别的可靠性。

常见的图像预处理过程一般有数字化、[几何变换](https://baike.baidu.com/item/%E5%87%A0%E4%BD%95%E5%8F%98%E6%8D%A2" \t "https://baike.baidu.com/item/%E5%9B%BE%E5%83%8F%E9%A2%84%E5%A4%84%E7%90%86/_blank)、[归一化](https://baike.baidu.com/item/%E5%BD%92%E4%B8%80%E5%8C%96" \t "https://baike.baidu.com/item/%E5%9B%BE%E5%83%8F%E9%A2%84%E5%A4%84%E7%90%86/_blank)、平滑、复原和增强等步骤。在该项目中为了提高光网络设备识别的准确性，需要采用图像预处理技术对原始设备图像数据进行预处理，提高训练模型的泛化能力和鲁棒性。

**1.3.2 数据标注**

数据为人工智能提供动力，是机器学习模型建立的基础。每个AI应用程序都需要一个合适的数据集，它是创建高效、准确系统的重要组成部分，因此需要大量的优质的数据才能准确地执行复杂的任务。

此次我们使用了网上部分公开模型的数据集以及一些预训练模型进行fine-tuning操作，另外基于我们上述图像预处理技术，最终取得了非常好的训练效果。

**1.3.3 姿态估计技术**

姿态估计，顾名思义是从图像中推断出物体或人的姿势。姿态估计任务本质上分两部分，首先是关键点的识别，然后是从关键点重构人体或物体姿态。

姿态估计根据检测物体的个数可分为单姿态估计和多姿态估计，其中多姿态估计分为自顶向下和自底向上两种。自顶向下的方法从高层抽象开始，首先检测人并在边界框中生成人的位置，然后对每个人进行姿态估计。自底向上的方法首先预测输入图像中每个人的所有身体部位，然后通过人体模型拟合或其他算法对它们进行组合；根据维度可分为2D和3D姿态估计，2D姿态估计预测图像或视频帧中对象关键点的2D平面位置，而3D姿态估计器通过在预测中添加深度将图像中的对象转换为3D对象，从而估计在三维空间中对象关键点的位置。

我们此处采用的是2D姿态检测技术进行检测，采用了lite-hrnet， lite-mspn等多项展开实验，最终选定lite-mspn网络结构。

**1.3.4 人脸识别技术**

人脸识别技术是基于人的脸部特征，对输入的人脸图像或者[视频流](https://baike.baidu.com/item/%E8%A7%86%E9%A2%91%E6%B5%81?fromModule=lemma_inlink" \t "_blank)，首先判断其是否存在人脸，如果存在人脸，则进一步的给出每个脸的位置、大小和各个主要面部器官的位置信息。并依据这些信息，进一步提取每个人脸中所蕴涵的身份特征，并将其与已知的人脸进行对比，从而识别每个人脸的身份。

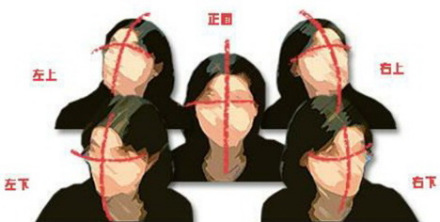
[](https://baike.baidu.com/pic/äººè¸è¯å«ææ¯/1130783/0/a8773912b31bb05149fdd72d3e7adab44bede0de?fr=lemma%26fromModule=lemma_content-image%26ct=single)

图1-1 人脸识别技术

广义的人脸识别实际包括构建[人脸识别系统](https://baike.baidu.com/item/%E4%BA%BA%E8%84%B8%E8%AF%86%E5%88%AB%E7%B3%BB%E7%BB%9F?fromModule=lemma_inlink" \t "_blank)的一系列相关技术，包括人脸图像采集、人脸定位、人脸识别预处理、身份确认以及身份查找等；而狭义的人脸识别特指通过人脸进行身份确认或者身份查找的技术或系统。

此处我们使用谷歌曾提出的FaceNet网络架构作为人脸识别系统的基础。

**1.3.5 Neon指令集**

SIMD，即 single instruction multiple data，单指令流多数据流，也就是说一次运算指令可以执行多个数据流，从而提高程序的运算速度，实质是通过 数据并行 来提高执行效率

ARM NEON 是 ARM 平台下的 SIMD 指令集，利用好这些指令可以使程序获得很大的速度提升。不过对很多人来说，直接利用汇编指令优化代码难度较大，这时就可以利用 ARM NEON intrinsic 指令，它是底层汇编指令的封装，不需要用户考虑底层寄存器的分配，但同时又可以达到原始汇编指令的性能。

NEON 是一种 128 位的 SIMD 扩展指令集，由 ARMv7 引入，在 ARMv8 对其功能进行了扩展（支持向量化运算），支持包括加法、乘法、比较、移位、绝对值 、极大极小极值运算、保存和加载指令等运算

ARM 架构下的下一代 SIMD 指令集为 SVE（Scalable Vector Extension，可扩展矢量指令），支持可变矢量长度编程，SVE 指令集的矢量寄存器的长度最小支持 128 位，最大可以支持 2048 位，以 128 位为增量

ARM NEON 技术的核心是 NEON 单元，主要由四个模块组成：NEON 寄存器文件、整型执行流水线、单精度浮点执行流水线和数据加载存储和重排流水线

**1.3.6 多线程OpenMP技术**

OpenMP是由OpenMP Architecture Review Board牵头提出的，并已被广泛接受，用于[共享内存](https://baike.baidu.com/item/%E5%85%B1%E4%BA%AB%E5%86%85%E5%AD%98/2182364?fromModule=lemma_inlink" \t "_blank)并行系统的多处理器程序设计的一套指导性编译处理方案(Compiler Directive)  。OpenMP支持的编程语言包括C、[C++](https://baike.baidu.com/item/C%2B%2B?fromModule=lemma_inlink" \t "_blank)和[Fortran](https://baike.baidu.com/item/Fortran/674319?fromModule=lemma_inlink" \t "_blank)；而支持OpenMp的[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067?fromModule=lemma_inlink" \t "_blank)包括Sun Compiler，GNU Compiler和Intel Compiler等。OpenMp提供了对[并行算法](https://baike.baidu.com/item/%E5%B9%B6%E8%A1%8C%E7%AE%97%E6%B3%95/967188?fromModule=lemma_inlink" \t "_blank)的高层的抽象描述，程序员通过在[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81?fromModule=lemma_inlink" \t "_blank)中加入专用的pragma来指明自己的意图，由此编译器可以自动将程序进行[并行化](https://baike.baidu.com/item/%E5%B9%B6%E8%A1%8C%E5%8C%96/1585208?fromModule=lemma_inlink" \t "_blank)，并在必要之处加入同步互斥以及通信。当选择忽略这些pragma，或者编译器不支持OpenMp时，程序又可退化为通常的程序(一般为[串行](https://baike.baidu.com/item/%E4%B8%B2%E8%A1%8C/3555818?fromModule=lemma_inlink" \t "_blank))，代码仍然可以正常运作，只是不能利用多线程来加速程序执行。

**1.3.7 内存池技术**

内存池则是在真正使用内存之前，先申请分配一定数量的、大小相等(一般情况下)的内存块留作备用。当有新的内存需求时，就从内存池中分出一部分内存块，若内存块不够再继续申请新的内存。这样做的一个显著优点是，使得内存分配效率得到提升。

在[内核](https://baike.baidu.com/item/%E5%86%85%E6%A0%B8?fromModule=lemma_inlink" \t "_blank)中有不少地方[内存分配](https://baike.baidu.com/item/%E5%86%85%E5%AD%98%E5%88%86%E9%85%8D?fromModule=lemma_inlink" \t "_blank)不允许失败. 作为一个在这些情况下确保分配的方式, 内核开发者创建了一个已知为内存池(或者是 "mempool" )的抽象. 一个内存池真实地只是一类后备缓存, 它尽力一直保持一个空闲内存列表给紧急时使用.

**1.4人员管理和项目进度管理**

**1.4.1 分工总览**

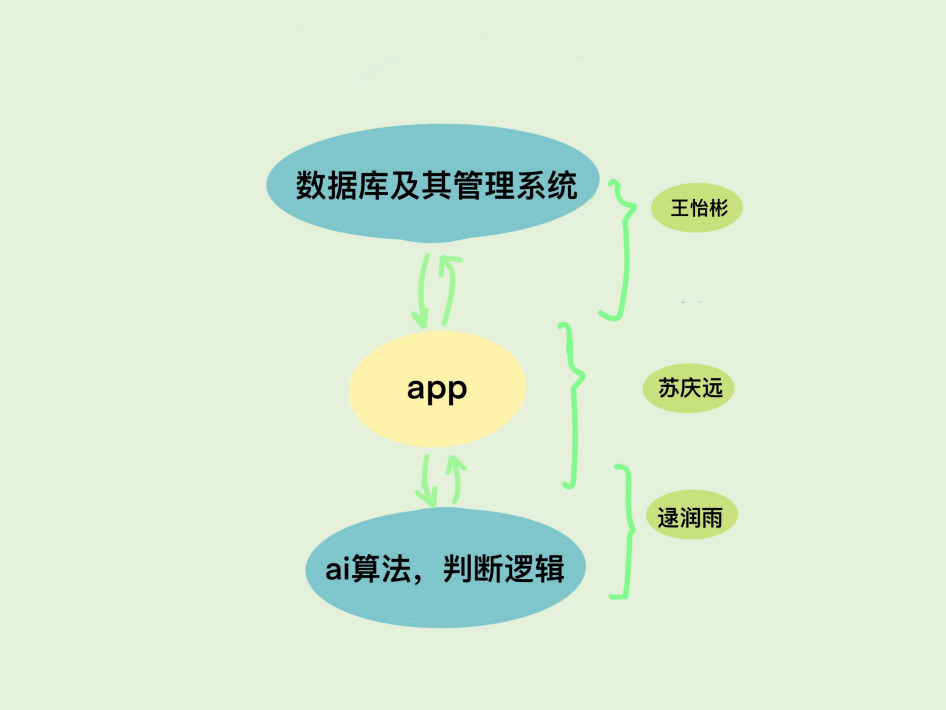
****

图1-2 项目分工总览

1. **个人分工**

王怡彬负责数据库及其管理系统（教师端）编写。

苏庆远负责实现全部Android软件设计，设计软件逻辑、UI交互设计。

逯润雨负责实现人脸、姿态、检测网络模型转化部署，Arm端cpu，gpu性能优化，运动动作识别逻辑的编写。

1. **合作内容**

逯润雨与苏庆远合作完成app与AI判别逻辑的交互接口设计与编写，王怡彬与苏庆远合作完成数据库与app的交互接口，王怡彬与逯润雨合作完成人脸识别模型的选定。

**1.4.2 任务分解**

1. **学生端(安卓)界面设计开发：苏庆远**
2. 主页界面设计
3. 账户界面设计
4. 设置界面设计
5. 登陆界面设计
6. 运动记录界面设计
7. 运动结算界面设计
8. **学生端(安卓)后台逻辑设计：苏庆远**
9. 学生记录本地持久性存储(SQLite)
10. 应用逻辑设计(Java)
11. 远程连接教师端(Java)
12. **教师端前端设计与实现：王怡彬**
13. 远程连接学生端接口设计与实现(Python+Django)
14. 网页UI设计(Html+BootStrap)
15. 网页UI实现(Html+BootStrap)
16. **教师端后端数据库、人脸检测教师端部署：王怡彬**
17. 学生信息存储数据库设计与实现（MySQL)
18. 后端web server框架设计
19. web server实现(Python+Django)
20. 数据库与web server接口实现(Python+Django)
21. 人脸检测模型实现(Python+Pytorch)
22. 后端逻辑设计与实现(Python)
23. 数据库与APP接口实现(Python+Django)
24. **人脸检测、姿态检测算法设计：逯润雨**
25. 人脸检测模型C++重写
26. 姿态检测算法C++重写
27. 姿态运动(仰卧起坐、跳绳、升起蹲)逻辑编写
28. **安卓端模型部署、优化：逯润雨**
29. 人脸模型C++重写、安卓端部署
30. JNI接口提供支持与编写
31. 安卓端人脸性能优化
32. **集成测试：逯润雨，王怡彬，苏庆远**
33. 运动逻辑与APP交互接口设计与实现
34. 教师端与APP交互接口设计与实现
35. 整体系统服务流程调试

**1.4.3 进度安排**

甘特图链接：<https://gantt.mindsup.cn/share/YtXDaUWQ7>

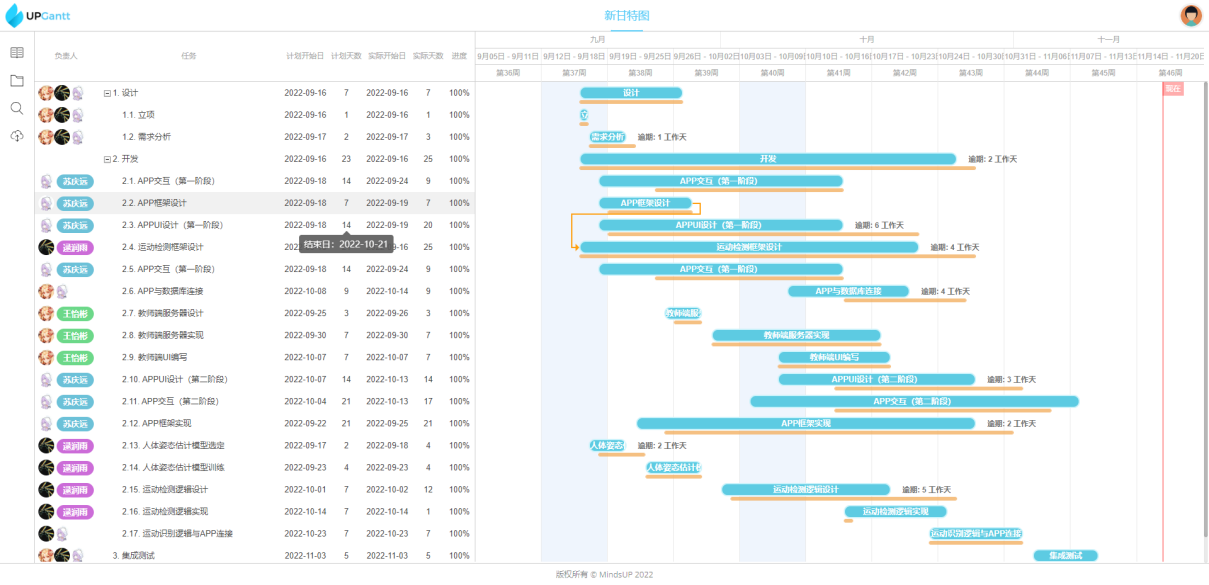


图1-3 项目进度安排甘特图(部分)

**2 需求分析**

**2.1 E-R图、数据流图**

**2.1.1 E-R图**

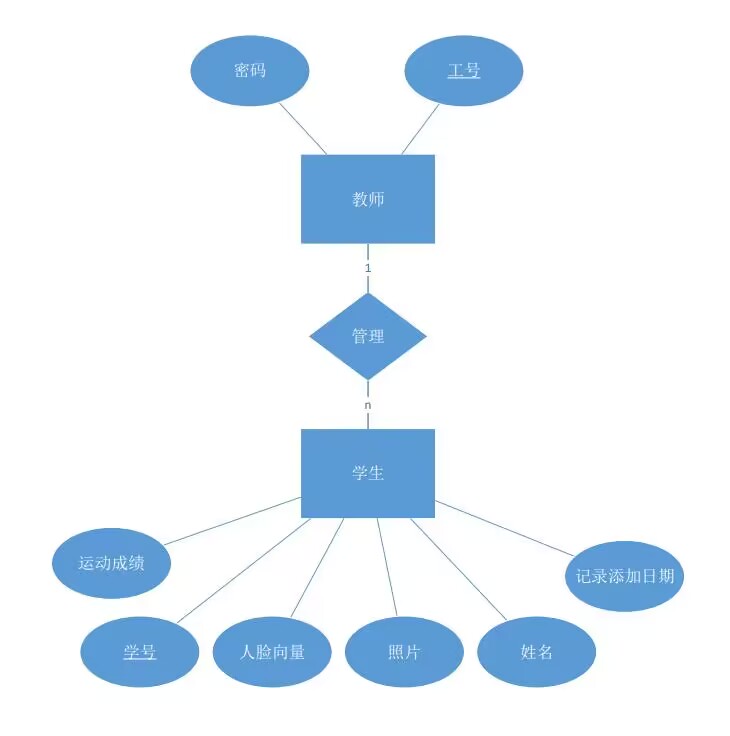


图2-1 E-R图(教师和学生之间的关系)

**2.1.2数据流图**

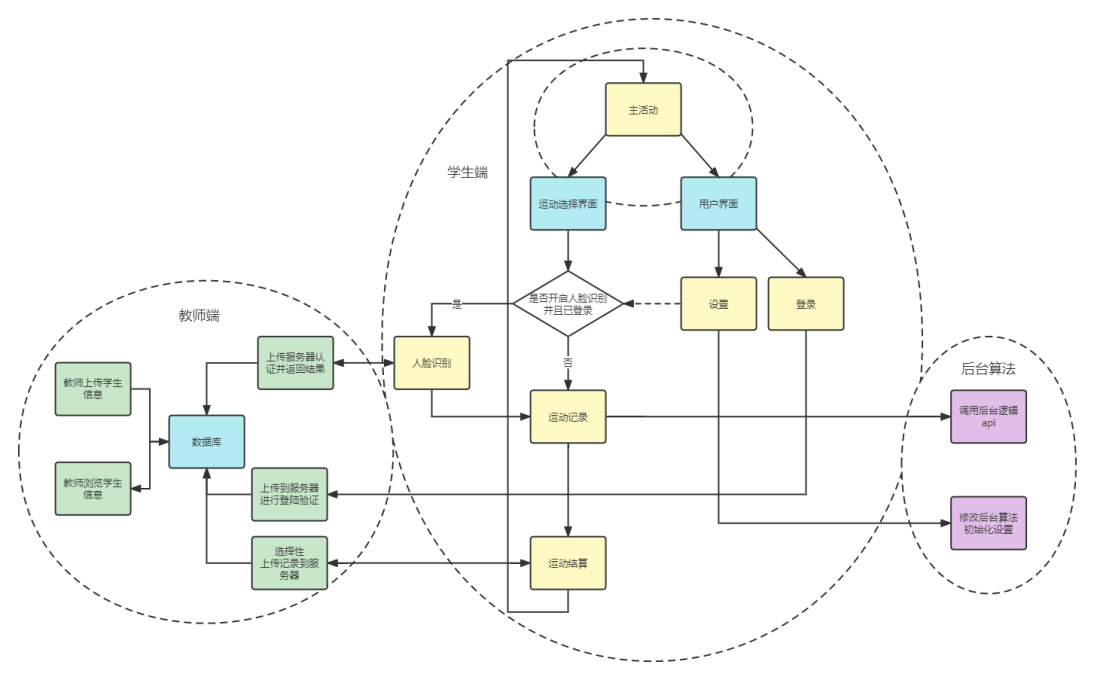


图2-2 项目数据流图

**2.2 用例图等**

图2-3中展示了学生和教师两种不同类型的用户群体与应用之间的关系，用例图中也描述了底层算法逻辑与学生运动信息之间的关系。

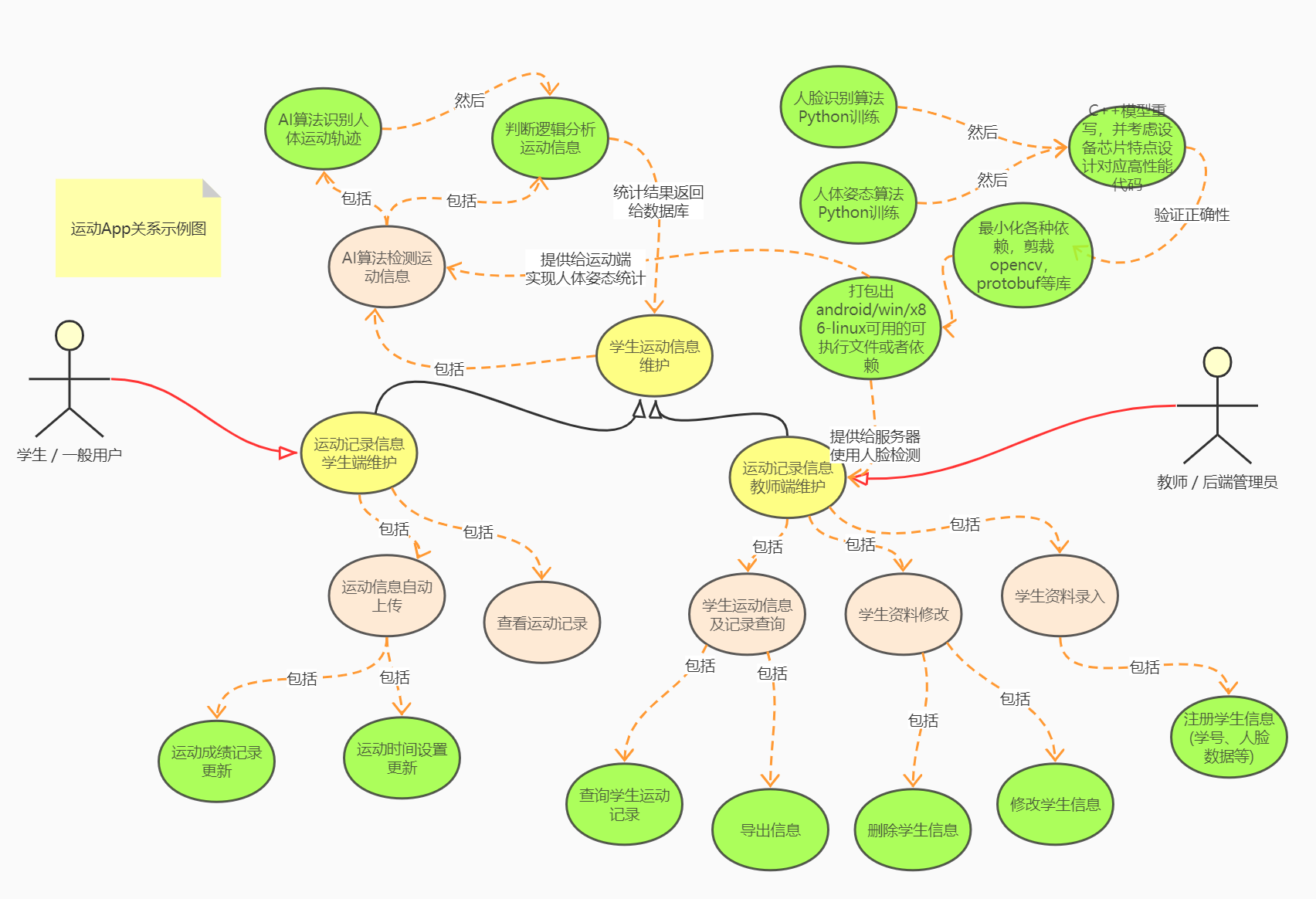


图2-3 用户用例图(包括教师和学生)

**2.3 原型系统设计**

原型系统设计图使用墨刀进行设计。项目初期对于应用界面规划尚有部分不合理的地方，所以项目成果的应用界面和结构可能与原型系统中的设计有较大出入，原型系统设计图仅供参考。

**2.3.1 学生端**

学生端的原型系统界面共设计了三个界面，包括运动选择界面、账户界面和设置界面，分别负责选择要进行的运动、对账户进行登录/退出等操作以及设置运动的限制时间等设置内容。



图2-4 学生端原型系统(从左到右依次为运动选择、账户和设置界面)

**2.3.2 教师端**

教师端的原型系统设计界面包括用户登录界面、学生信息展示界面、学生信息录入界面以及学生信息编辑界面。

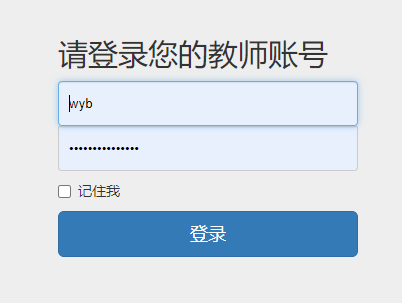


图2-5 用户登录界面



图2-6 学生信息展示界面



图2-7 学生信息录入界面



图2-8 学生信息编辑界面

**3 概要设计和详细设计**

**3.1 系统结构**

项目系统主要分为三大模块，分别为：学生端模块、教师端模块以及底层算法模块。

1. 学生端模块是学生能接触到的主要功能模块，负责提供运动选项、提供运动检测(通过调用底层算法模块的检测程序)和人脸检测功能记录运动数据并及时上传到远程教师端数据库，同时提供基本的防作弊手段；
2. 教师端模块为教师提供管理学生信息的工具，可以从学生端模块获取学生的最新运动记录信息，同时提供基本的学生信息管理系统(数据库)；
3. 底层算法模块为学生端应用提供有力的检测算法支撑，同时为教师端提供了人脸识别算法的接口，是本项目的核心模块。

**3.2 类图**

图略，详情请见3.3关键数据结构定义一节的介绍(包括系统中关键数据结构的类图)

**3.3关键数据结构定义**

**3.1.1 神经网络C++算法框架**

首先介绍C++端算法框架，数据结构类基本结构如下：

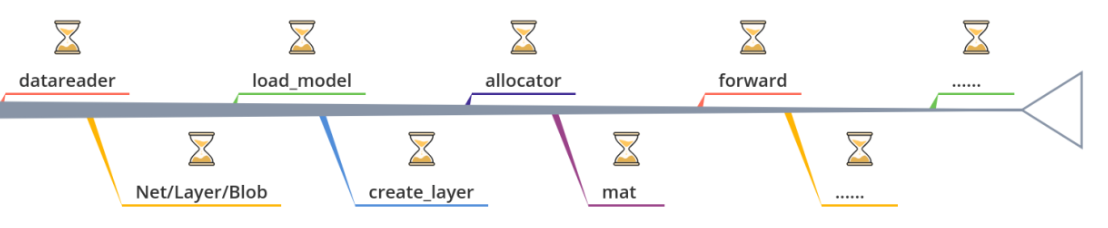


图3-1 C++端算法框架

此为定义的各个前向传播类。Datareader负责读取模型和load\_model负责读取模型、加载权重，allocator负责读取内存，mat负责作为权重存储的基类，creater\_layer、Net/Layer/Blob为模型类、模型每一层的抽象类、每层推理的blob。

首先我们介绍Mat类，算是最基础最核心的实现类：

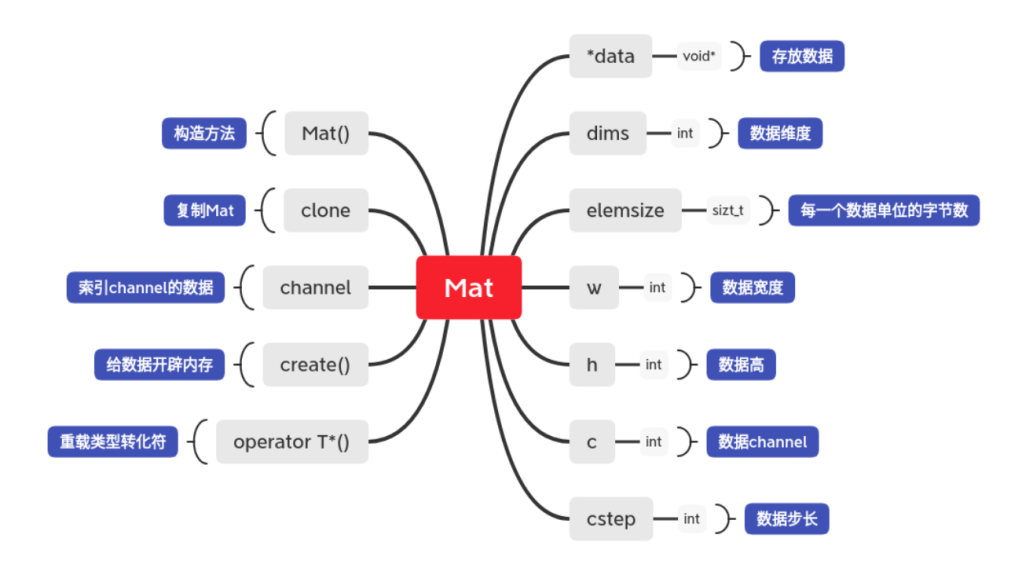


图3-2 Mat类数据结构示意图

这个类的构造风格和opencv很像，算是opencv的精简版。这个类主要功能为存放神经网络需要处理的数据，这些数据一般包含一下几种(不限于)：

* 输入图片
* 模型的weight和bias
* 模型处理的top\_blob和bottom\_blob

Mat好处在于如下(不限于)：

* 方便的构建数据，如根据长宽高构建不同维度的数据．
* 方便的索引数据，如按照channel索引数据．
* 可以很方便的获取数据信息，如维度，长，宽等等．
* 可以很方便的拷贝数据，如clone方法

图3-2右半部分是Mat类所包含的必要的成员变量，左半部分是Mat所包含的必要的成员函数．成员函数很简单，名字和类型都在上图中有写到，这里就不再做过多的说明．下面我们着重说一下Mat类的成员函数．由于不允许贴源代码，我们简单看下函数定义：

inline void Mat::create(int \_w, int \_h, int \_c, size\_t \_elemsize, int \_elempack, Allocator\* \_allocator)

上述函数针对一个三维的数组开辟空间，三个维度分别为长h，宽w和通道数c(channel)．其中cstep使用了AlignSize()函数，它的含义是计算出一个大于步长数字且该数可以刚好被对齐数(16)整除，这个数字其实就是每一个channel所包含的字节空间数．计算这个数字的目的是为了进行另外一个维度的对齐，也就是空间尺寸上的对齐．我们在上一节中讲到，alllocator中的alignPtr函数做到了首地址上的对齐．由于每一个元素的字节数是elemsize，所以一共需要申请的空间为totalsize = alignSize(c\* cstep \* elemsize, 4);

最后还要特别介绍一下，为了编程方便，Mat类中定义了一个重载类型转化符，当Mat类被某个指针强制转化的时候，其实是返回了它的成员函数data的指针被强制转化的结果：

template <typename T>

inline Mat::operator T\*() {

return (T\*)data;

}

其次我们介绍关于数据读取类，其基本的读取文件结构如下：

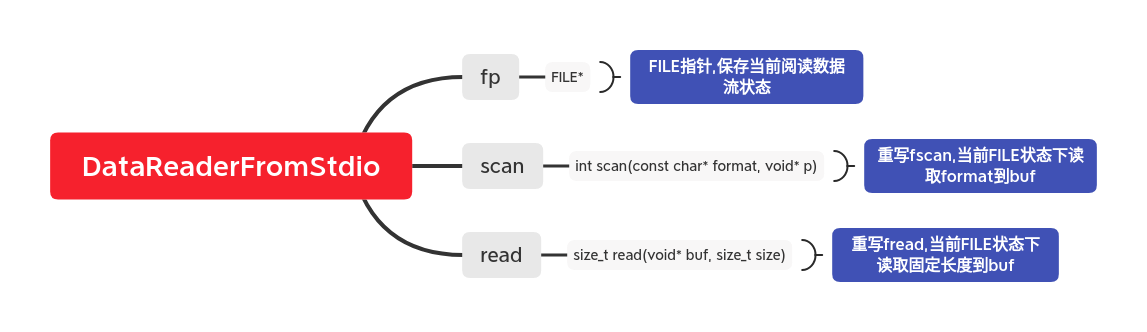


图3-3 数据读取类

DataReaderFromStdio是DataReader（抽象类）的子类，我们主要用它来实现DataReader类的主要功能．

它包含一个成员变量： fp，一个FILE的指针，也就是保存了当前读取的数据流的状态，另外包含两个成员函数： scan和read

这两个成员函数分别是对fscan和fread的一个包装．函数scan是按照format格式读取信息的，所以它非常适合读取模型的proto信息．函数read是安装buffer长度读取信息的，所以它非常适合读取读取模型的权重信息．

然后是关于Net/Layer/Blob类的介绍。Net/Layer/Blob这三个类是组成网络的基础， 首先我们先介绍Net类：

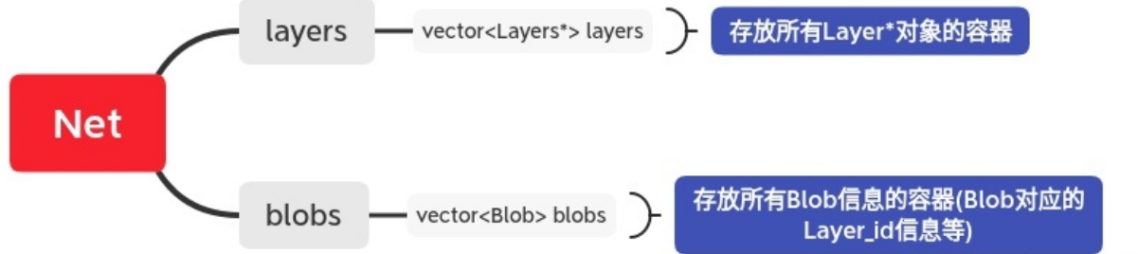


图3-4 Net类

它包含的两个成员变量，layers，blobs。

layers是一个包含了所有layer\*的vector数组，它存放了所有layer的信．blobs是一个包含了所有blob的数组，它存放了所有blob的信息。

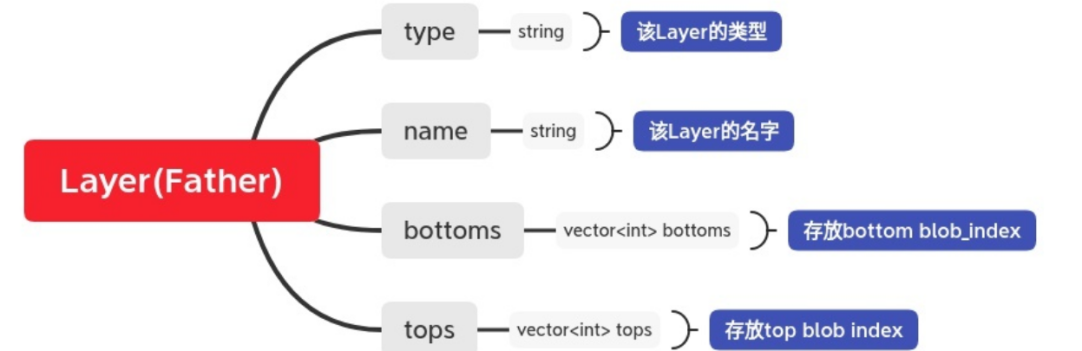


图3-5 Layer类

这里先不提其实现的成员方法，首先还是先介绍他的layer类主要信息：

它包含了4个重要的变量分别是layer的类型(type)，名字(name)，输入blob的id(bottoms)， 输出blob的id(tops)．

之所以要在Layer后面加上Father， 是因为它是具体子Layer实现类的父类，而这些具体的Layer实现类便是我们每层卷积神经网络要运行的层数。

而后为我们的Blob类的实现：

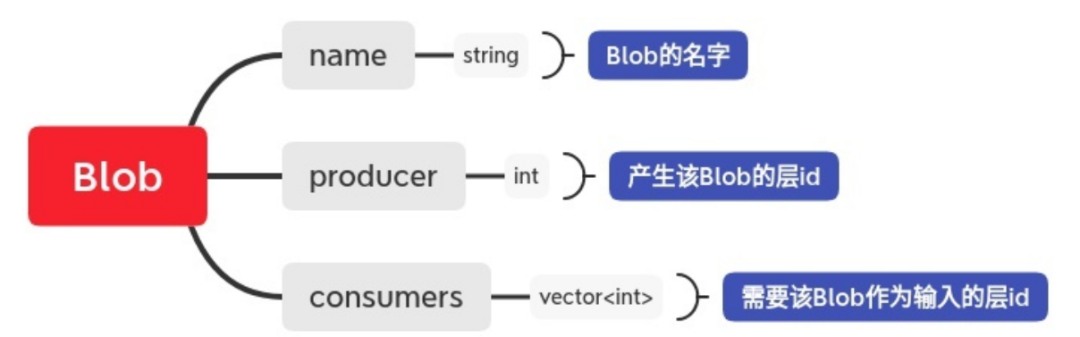


图3-6 Blob类

它包含了3个成员变量，name为该blob的名字，producer是一个int类型的变量，它存放着产生该blob的layer\_id．consumers 是一个vector容器，里面存放着需要该blob作为输入的层的id，也就是字面意思的需要“消费”该blob的layer\_id，因为可能不止一个层需要用到该blob作为输入，所以这个layer\_id可能有多个，它需要一个容器来保存。（当然此处命名借鉴了操作系统的概念，“生产者”，“消费者”）

前向推理Extractor类：

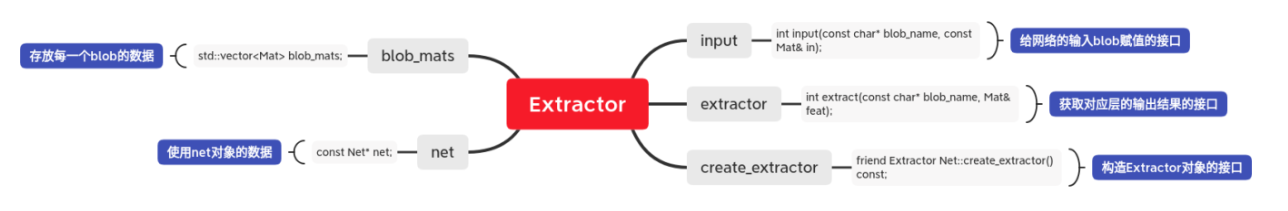


图3-7 Extractor类

这个类的主要作用是提供输入数据的接口和输出数据的接口，同时它还保存着每一个网络层计算的结果，首先把输入图片（已经转化为Mat类实例）通过input函数，给放到对应的blob里面，其实是先通过输入blob\_name去找对应的blob\_index，然后把blob\_index作为blob\_mats的id给blob\_mats赋值。然后调用extractor函数来获取我们对应的输出：通过blob\_name去得到我们想要的输出的blob的id（blob\_index）．

然后检查这个blob\_index对应的blob\_mat[blob\_index]是否是有值的，如果有值则直接返回对应的值．如果这个blob\_mats[blob\_index]没有值，则再基于这个blob\_index去通过net->blobs这个变量去找它对应的输入层的id（layer\_index）．然后再调用net的forward函数去计算该层的输出top\_blob．forward函数是Net类的成员函数，它的作用是检查某一个层id所对应的bottom\_blobs是否有值（之前已经被运算出来）．如果这个bottom\_blob有值（已经被运算出来），那么它就用该层的bottom\_blobs去**调用对应Layer子类的forward函数**来运算得到这个top\_blobs．如果这个bottom\_blob没有值，则递归上述过程，直到某一层的bottom\_blobs是有值的（不可能无限递归，至少input\_blobs肯定有值）。

这样一来，我们神经网络整体的C++数据结构类便介绍完毕。

**3.1.2 学生端数据结构**

**3.1.2.1学生账号**

**Account类**

public class Account {  
    // 学生姓名  
    private String studentName;  
    // 学生学号  
    private String studentId;  
    // 学生账户密码  
    private String password;  
    // 学生各门运动的最好成绩(暂时只记了最好的)  
    private Map<SportsType, Integer> sportsRecords;  
    // 记录一下设置  
    private Map<SportsType, Integer> limitBackup;  
}

Account类表示保存在本地的用户数据，由于应用主要面向学生，这里就以学生信息为准。除了普通的学生能够姓名、学号和账户密码之外，还定义了两个映射负责记录该学生有关的运动的成绩 sportsRecords 以及该学生登录后应加载的每个运动的时间限制 limit。

该类有两个构造方法，默认的无参构造方法代表生成一个游客模式下的账户，带参的则为登陆后才会创建的用户登录帐户。

public Account() {  
        studentName = "请登录";  
        studentId = "U1919810";  
        password = "114514";  
        sportsRecords = new HashMap<>();  
        limitBackup = new HashMap<>();  
        for(SportsType item: SportsType.values()) {  
            sportsRecords.put(item, 0);  
            limitBackup.put(item, 2);   // 默认锻炼时间为 2 分钟  
       }  
   }  
​  
public Account(String studentName, String studentId, String password, int... scores) {  
    this.studentName = studentName;  
    this.studentId = studentId;  
    this.password = password;  
    sportsRecords = new HashMap<>();  
    limitBackup = new HashMap<>();  
    int i = 0;  
    for(SportsType item: SportsType.values()) {  
        sportsRecords.put(item, scores[i++]);  
        limitBackup.put(item, 2);   // 默认锻炼时间为 2 分钟  
   }  
}

类的方法除了基础的 Getter/Setter 方法之外，还有对两个映射的批量处理方法以及重置账户的 rewind() 方法， rewind() 方法的作用就是将重置账户的信息。

public void rewind() {  
        studentName = "请登录";  
        studentId = "1919810";  
        password = "114514";  
        for(SportsType item: SportsType.values()) {  
            sportsRecords.put(item, 0);  
       }  
   }

**AccountHolder类**

该类负责存放一个全局的静态Account对象，表示应用中的账户，在应用的所有活动中，都将通过该类的公有方法获取和修改账户中的信息，包括登录状态更新、更新用户数据、获取用户数据。

public class AccountHolder {  
​  
    private static Account studentAccount = new Account();  
​  
    private static boolean registered = false;  
}

字段 registered 标识当前账号是否处于已登录状态。

**3.1.2.2 设置**

**Setting类**

Setting类是应用中所有可见的设置类选项的功能抽象，负责定义这个按钮的类型、功能和附加信息。Setting类对象具体可以分为三种，DEFAULT 类型只负责打开一个新活动(安卓系统中一般可以把一个界面或者一组碎片界面称为一个活动)，SEEKBAR 类型有一个可调节的滑动条(seekbar)，SWITCH 类型有一个切换开关。为了适应更多的设置类型，后期将考虑将具体类型改为继承关系而不是写在一起。

text 字段是显示在该设置区域上的提示文字，tag 字段在 DEFAULT 类型选项中负责隐式传递打开的新活动的类型，在 SWITCH 类型中将会关联到某个系统设置项，在 SWITCH 类型中则没有作用。

public class Setting {  
​  
    private String text;  
​  
    private String tag;  
​  
    // 使能：指该选项是否可以打开一个对应的新活动  
    private boolean enable;  
​  
    // 状态，指该选项最后是放置于哪个特定列表，根据列表不同显示状态不同和功能不同（想复用一下这个组件）  
    private int status;  
​  
    // 如果设置项属于 switch 类型的话可以设置 switch 对应的两个文本 (分别对应 textOn 和 textOff)。  
    private String[] switchText = {};  
​  
    // default：点击可以打开一个活动（被 enable 控制使能）  
    public static final int DEFAULT = 1;  
​  
    // seekbar：有一个滑动控制条，不能打开新活动  
    public static final int SEEKBAR = 2;  
​  
    // switch：一个开关，不能打开新活动  
    public static final int SWITCH = 3;  
}

##### settingAdapter类

该类是负责将 Setting 中不同类型对象绑定到应用界面中用户可见的选项上的适配器。

**SettingHolder类**

该类负责管理应用中的本地设置，目前支持的设置有使用 CPU/GPU 计算、是否开启限时、是否在运动前进行人脸验证。设置的值将保存在一个 HashMap 中，对应的键值为 Setting 类中 SWITCH类型 的 tag 字段(所以可以看到这里的设置都是只有两个选项的，刚好对应一个开关控件的开和关状态)，getPropertyValue() 和 setPropertyValue()方法分别负责获取和修改某项本地设置的值。

public class settingHolder {  
​  
    public static Map<String, Integer> localProperty = new HashMap<>();  
​  
    public static final int CPU = 0;  
    public static final int GPU = 1;  
    public static final int OFF = 0;  
    public static final int ON = 1;  
​  
    static {  
        localProperty.put("CPUGPU", CPU);     // 0表示使用CPU，1表示使用GPU，默认是CPU  
        localProperty.put("LIMITMODE", ON);  // 0表示不限时，1表示打开限时，默认限时  
        localProperty.put("FACE\_IDENTIFICATION", ON);  // 1表示打开人脸检测，0表示关闭，默认打开  
   }  
​  
    public static int getPropertyValue(String propertyName) {  
        return localProperty.getOrDefault(propertyName, -1);  
   }  
​  
    public static void setPropertyValue(String propertyName, int value) {  
        if(localProperty.containsKey(propertyName)) {  
            localProperty.put(propertyName, value);  
       }  
   }  
​  
}

**3.1.2.3 运动类型**

**Sports类**

该类表示某项运动，具体支持的运动类型在枚举类 SportsType 中可以定义对应的枚举量。

public enum SportsType {  
    SQUAT("蹲起", 2, R.drawable.sports\_bg1),  
    SKIPPING\_ROPE("跳绳", 3, R.drawable.sports\_bg2),  
    SIT\_UP("仰卧起坐", 1, R.drawable.sports\_bg3),  
    HOTBODY("热身", 0, R.drawable.sports\_bg4);  
​  
    private String sportsName;  
​  
    private int index;  
​  
    private int imageID;  
}

sportsType 字段代表 SportsType 类中的某个枚举量，exerciseTimes 记录了该账户(可以是学生也可以是游客)下该运动的运动次数(不过时间有限最后并没有实现这个功能，作为保留字段可能会后续会添加该功能)，bestRecord 和 timeLimit 其实已经在 Account 类中见过了，描述了该项运动项目的最佳成绩和时间限制。

public class Sports {  
​  
    private SportsType sportsType = null;  
​  
    private long exerciseTimes = 0;  
​  
    // 默认采用 xx次 / 分钟 表示  
    private int bestRecord = 0;  
​  
    // 运动的时间，默认为 1 分钟  
    private int timeLimit = 1;  
}

**SportsAdapter类**

和 Setting 类一样，Sports 类也有自己的适配器类，负责将所有的运动项目相关信息绑定到应用的界面上。

**3.1.3 教师端数据结构**

1. **前端数据结构**
2. 登入页面主要数据

username、password分别绑定前端页面的用户名输入框和密码输入框中的内容.

1. 学生信息展示页面主要数据

后端提供学生对象列表：studentlist，前端页面遍历该列表，将各个对象的对应信息进行显示.

1. 学生信息上传页面、学生信息修改页面主要数据

两页面数据结构设计大体相同，student\_name、student\_no、student\_password、student\_img分别绑定前端页面的学生姓名输入框、学生学号输入框、学生密码输入框和学生照片文件上传框中的内容

1. **后端数据库教师数据结构**

教师数据项的表项代码设计如下：

class User(models.Model):

   id=models.AutoField(primary\_key=True)

   name = models.CharField(max\_length=128, unique=True) #教师工号

   password = models.CharField(max\_length=256) #密码

   c\_time = models.DateTimeField(auto\_now\_add=True) #表项添加时间

上表为教师用户数据项表，成员具体说明如下：

**id**：为数据库存取时的唯一索引，具有自增属性，数值为数据库自动为其分配。

**name**：教师的工号，以字符串形式存储，具有唯一性，由系统管理员为教师分配账号时在后台注册。

**password**：教师的密码，与教师的工号一同绑定在一个教师账号下，以字符串形式存储，由系统管理员为教师分配账号时在后台注册。

**c\_time**：教师账号注册的时间，用作特殊记录用。

1. **后端数据库学生数据结构**

学生数据项的表项代码设计如下

class Student(models.Model):

   id=models.AutoField(primary\_key=True)

   student\_no = models.CharField(max\_length=32, unique=True) #学生学号  
    student\_name = models.CharField(max\_length=32) #学生姓名

   student\_img = models.ImageField(upload\_to='./static/pics/', default="") #学生照片

   addtime=models.DateTimeField(default=datetime.now) #学生账号注册时间

   jump\_score=models.IntegerField(default=0) #学生跳绳成绩

   sqd\_score=models.IntegerField(default=0) #学生升起蹲成绩

   situp\_score=models.IntegerField(default=0) #学生仰卧起坐成

绩  
    face\_vector = models.CharField(max\_length=9999) #学生人脸向量（人脸识别用）

   student\_password = models.CharField(max\_length=32, default="123456") #学生账号密码

   temp = models.IntegerField(default=2)

上表为学生用户数据项表，成员具体说明如下：

**id**：为数据库存取时的唯一索引，具有自增属性，数值为数据库自动为其分配。

**student\_no**：学生的学号，以字符串形式存储，具有唯一性。

**student\_password**：学生的密码，以字符串形式存储，为确保系统安全性，在外层调用接口时，只能调用接口的验证函数，无法获取密码的表项。

**addtime**：学生账号注册的时间，用作特殊记录用。

**student\_img**：学生的照片，实际上以字符串形式存储为对应照片文件的路径。

**jump\_score**：学生的跳绳成绩（单位：1分钟）。

**sqd\_score**：学生的升起蹲成绩（单位：1分钟）。

**situp\_score**：学生仰卧起坐成绩（单位：1分钟）。

1. **数据关联结构设计**

前后端数据关联，分页面介绍如下。

教师账号页面登入的前后端数据关联：前端传入的username和password和数据库的教师对应数据：name和password进行匹配验证。

学生信息浏览页面的前后端数据关联：后端提供一个全体学生对象的列表：studentlist，前端页面遍历该列表，将每个对象的学号、姓名、密码、注册时间、照片和成绩进行逐一显示。

学生信息上传、修改页面的前后端数据关联：后端获取到前端从studentlist中选择的学生对象，打开数据库中的对应数据项，根据前端传入的其他信息（姓名、密码、照片）对对应的学生数据项进行修改并保存，前端中的各个数据变量命名均与数据项的对应成员名相同，以便于代码的编写、调试与阅读。

**3.4 关键算法设计**

**3.4.1 调用摄像头算法设计**

传统的调用摄像头算法的方式：

1. java 初始化和设置相机，创建 ImageReader 做预览回调
2. java OnImageAvailable 转换为 rgba / Bitmap
3. java 通过 JNI 调用 C++ 实现的 nanodet 检测函数
4. c++ nanodet 推理
5. java 通过 JNI 返回检测框信息，用 Canvas 在 Bitmap 画框画字，更新
6. 重复 3-5

我们的设计方式：

1. c++ 初始化和设置相机，创建 AImageReader 做预览回调
2. java 通过 JNI 设置显示的 SurfaceView ANativeWindow
3. c++ onImageAvailable 获得 yuv420sp，旋转，转 RGB
4. c++ nanodet 推理
5. c++ opencv-mobile 画框画字，更新 ANativeWindow buffer
6. 重复 3-5

相比之下我们算法优势如下：

* 每帧的处理循环完全用 c++ 实现，避免 JNI 与 java api 的数据传输
* 使用优化的 yuv420sp 旋转，转 RGB 函数实现，更高效
* 使用 opencv画框画字，更高效
* ANativeWindow buffer 到屏幕的缩放由硬件完成

其次是人体运动逻辑算法：

我们选取人体关键点坐标进行测定分析。

对于跳绳，仰卧起坐，蹲起，热身动作均复用一个同一套检测逻辑进行统计示，只不过不同逻辑的检测方式不同。此处设定两个状态，分别为A,B态，同时存在一个“锁”来保证只有发生状态改变的时候才会+1计数。

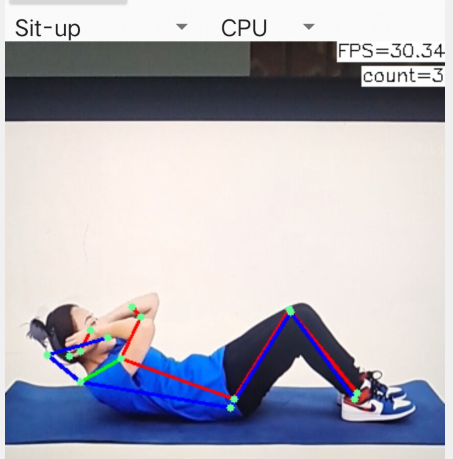


图3-8 仰卧起坐的计数

状态改变判断的标准只有一个，那就是角度，依赖于选择的强大的AI模型，我们对于点的估计是较为准确的，以蹲起为例，只有当膝盖对应大小腿的夹角大于（或小于）一定角度的时候我们才会进行状态改变的判定。

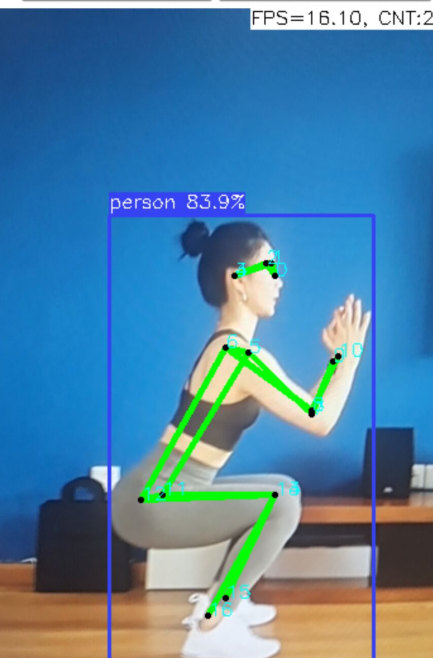


图3-9 蹲起的计数

对于程序健壮性：仍然以蹲起为例，如果我们仅仅摆动弯曲大小腿来追求角度改变是不会计数的。

**3.4.2 Arm端优化算法**

由于卷积神经网络90%以上的算子以及运算开销都是卷积算子。

所以优化卷积的收益是非常大的 ，此处针对不同大小的卷积核使用了不同的优化策略，即著名的**Winograd算法以及Im2col+gemm算法，**选取方法为大块卷积核前者反之后者。

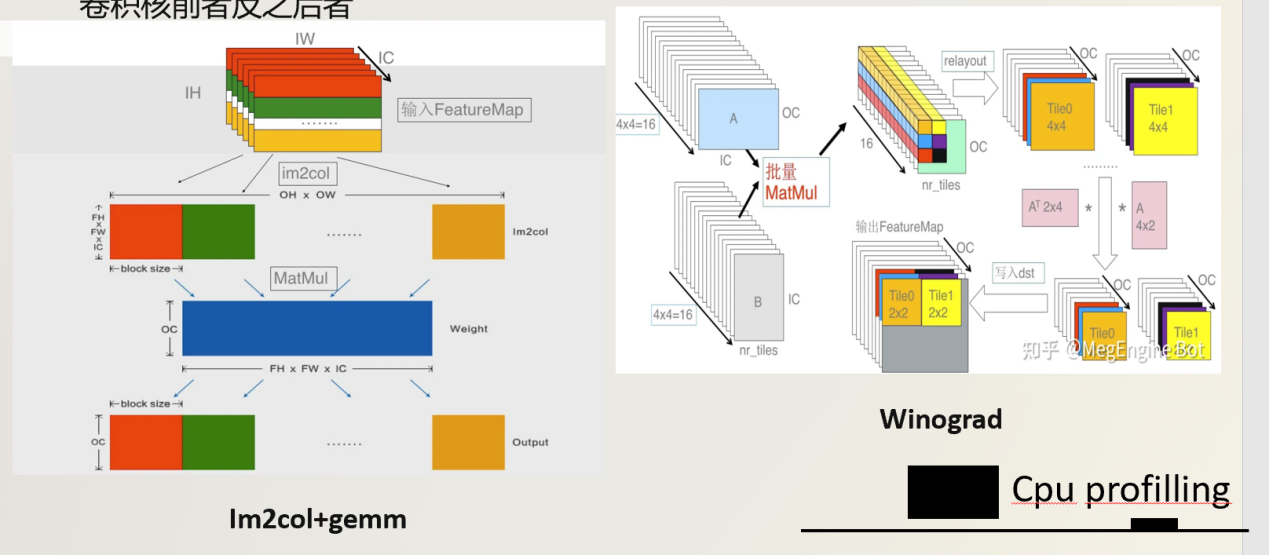


图3-10 Winograd算法以及Im2col+gemm算法

卷积神经网络Arm端采用的Cache、指令集优化：

1. 优化一：进行Pack操作

如果直接使用矩阵乘法计算，在卷积核尺寸比较大并且输出特征图通道数也比较大的时候，我们会发现这个时候Im2Col获得矩阵是一个行非常多列非常少的矩阵，在做矩阵乘法的时候访存会变得比较差，从而计算效率变低。这是因为当代CPU架构的某些原因，导致程序在行方向上的处理始终比列方向慢一个档次，所以我们如果能想个办法使得行方向的元素变少，列方向的元素变多这样就可以有效改善缓存达到加速的目的，另外列方向上元素增多更容易让我们发挥simd指令集的作用。所以，这就是本文将要提到的第一个优化技巧，数据打包(Pack)。

具体来说，对于卷积核我们进行4x4的Pack（所谓4的Pack就是在Im2Col获得的二维矩阵的高维度进行压缩，在宽维度进行膨胀，不知道我这种说法是否合适，参考一下下面的图应该很好理解），如下图所示，一个3x3x4的卷积核进行Im2Col之后再Pack变成了一行。

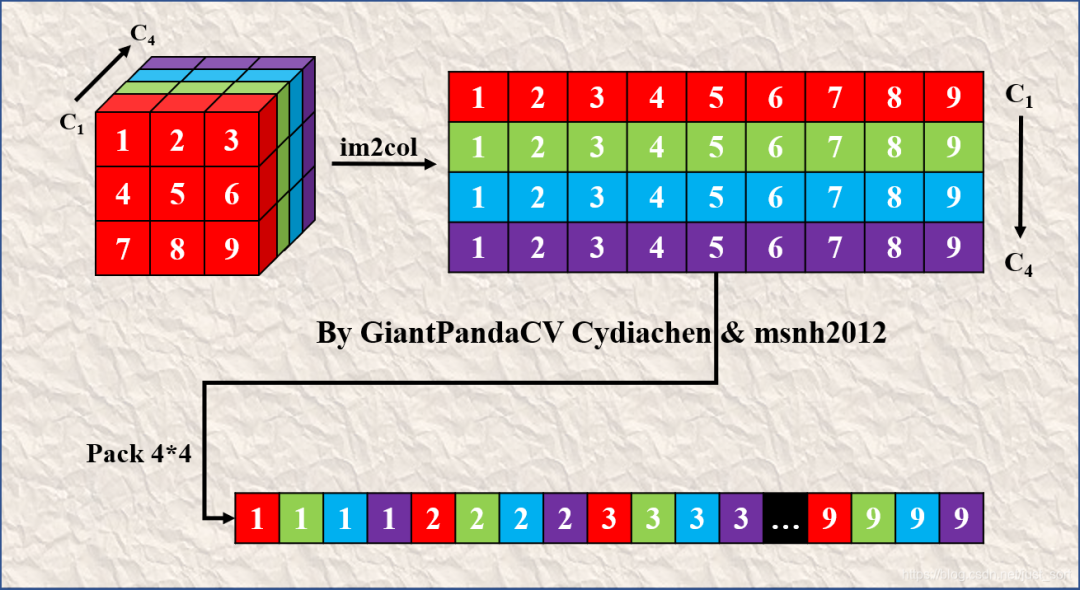


图3-11 Pack操作

这是一个的卷积核并且输出通道数为，它经过Im2Col之后首先变成上图的上半部分，然后经过Pack 之后变成了上图的下半部分，即变成了每个卷积核的元素按列方向交织排列。

这部分的代码也比较容易实现，另外一个技巧**是，**每次在执行卷积计算时，对于Image的Im2col和Pack每次都会执行，但对于卷积核，Im2col和Pack在任意次只用做一次，所以我们可以在模型初始化的时候提前把卷积核给Pack好，这样就可以节省卷积核Im2Col和Pack耗费的时间。

1. 优化二：对 Im2col+Matmul 过程进行分块

上面提到在 Im2col 之后，消耗的内存会超过 CPU 的 Cache 的容量，为了使这部分数据能够保存在 Cache 中，需要对 Im2col+Matmul 的整个过程进行分块，每次 Im2col+Matmul 都只对一个分块进行操作，这样就可以解决内存占用过大，超过 CPU Cache 后造成 Cache Miss 的问题。

分块优化如上图所示：Im2col 每次只对 block\_size 大小的数据进行计算，得到的 Fh\*Fw\*IC\*block\_size 的数据可以保存在 Cache 中。Im2col 得到数据后，对其直接进行 Matmul 计算，将计算得到的结果写入到输出 Tensor 对应的 block\_size 处就可以得到该分块处卷积的计算结果。计算完该分块之后，依次进行下一个 block\_size 的计算，直到整个输入计算完成。

结合矩阵乘的知识，在进行 Matmul A\*B=C 计算时将分块 Im2col 得到的数据视作 B 矩阵，A 矩阵为卷积的权重矩阵，根据 sgemm 的分块规则，以及 cache 的性质，A 矩阵会被调度并保存在 L2 上，B 矩阵基于最内层分块的一列和 A 矩阵基于最内存分块的一行以及 C 矩阵基于最内层的部分分块会被调度保存在 L1 上，因此可以通过 L1，L2 的大小以及 A 矩阵的大小，计算出所有的分块大小。分块优化能有效的减少存储使用，而且还可以提升算子的计算性能。

1. 优化三：融合 Im2col 和 Matmul PACK 数据操作部分

Im2col 过程中将多个窗的展开同时进行时，实际上是对内存的 copy 以及数据的 relayout 的过程，后续 Matmul 的 Pack 操作业是对数据的 copy 的 relayout，因此可以将上面两次数据的 copy 和 relayout 进行合并优化，减少该过程中对内存的读写次数。Im2col+Matmul 的 algo 中实现了将 Im2col 和 Matmul 的 Pack 融合的优化，这样能够减少一次数据的读写操作。由于该 fuse 过程和卷积的参数直接相关，不同的卷积参数将对应不同的融合 kernel，所以不具备通用性。通用情况下我们会使用之前的 Im2col+Matmul 的做法，另外针对一些通用的卷积如：kernel=3x3，stride=2 等，因为参数固定，因此可以直接进行上述融合优化，利用这样的组合优化，既可以保证 im2col 算法的通用性，也可以确保大部分常见的卷积的性能。

1. 优化四：对输入数据进行变换

对卷积核进行变换之后，接下来就轮到对输入矩阵进行变换了，即对V矩阵进行计算，$$$$V\_{c,b}=B^Td\_{c,b}B$$$$。上面我们已经提到过，对于卷积核获得的每一个$$$$U\_{k,c}$$$$,我们都需要一个对应的的图像块(tile)和它做卷积运算（eltwise\_multiply）。所以这里我们首先需要确定输入数据可以被拆成多少个图像块，并且我们需要为变换矩阵V申请空间，从第三节可知：输入变换矩阵，尺寸为(m+r-1)x(m+r-1)，即每个小块的变换矩阵都为(6x3-1)x(6x3-1)=8x8，但是输入特征图长宽不一定会被8整除，这个时候就需要对输入特征图进行扩展(padding)，这部分预处理的代码这里不贴了，大致思路如下：

1. 优化五：Winograd分块优化

上述的 Winograd 算法，还会有以下缺点：

1. 输入转换需要跨 channel 读写整个 feature map，数据读写对 Cache 不友好。
2. feature map 转换之后，矩阵乘时需要再 PACK，数据访存增加。

针对这些问题，可以对 Winograd 算法的整个计算流程做进一步的优化，这些优化主要包括：

* 输入转换时，分块 feature map 的 tiles 进行分块，每次只进行一定数量的tiles 计算；
* 调整分块大小适配 CPU L1 Cache，使得矩阵乘不需要 PACK；对整个输入 feature map 进行分块后，每次只计算一个分块的 nr 个 tiles，这样就可以保证每个批量矩阵的输入数据（除了转换之后的 weight 数据）保存于 L1 Cache，不会出现 Cache miss， 而且矩阵乘时不需要 PACK。

**3.4.3 学生端(Android)优化算法**

学生端作为本项目的用户终端之一，主要任务为运行底层姿态检测算法并做出响应，以及连接远程教师端服务器，故涉及复杂算法的代码部分很少。这里简单介绍运动记录界面中的两个算法内容。

1. **方向检测**

方向检测主要是为了给界面内的控件提供随手机方向旋转的动画而设计的，其主要原理是通过手机的加速度传感器获取手机当前的加速度并进行计算来得到手机当前的角度。

通过注册加速度传感器的服务，我们可以实时获取当前手机的加速度信息。值得注意的是，当加速度发生改变时，系统会传递一个数组的数据给当前的活动，数组长度为3，内容依次为当前手机的x、y、z轴上的加速度分量(x轴方向与手机的短边水平，y轴方向与手机的场边水平，z轴方向垂直于手机屏幕)。得到三个方向的分量，就可以通过勾股定理计算手机的竖直方向上的加速度(通常接近g)，就可以求得分量与竖直加速度之间的夹角，用来控制应用界面中的控件转动角度。

为了减少控件频繁的旋转动画导致的“抖动”，具体的实现方法中设置了角度的阈值，使得手机的微小晃动或者竖屏状态下正常使用时小角度倾斜不会发生过多的动画效果。

1. **抖动检测**

抖动检测是学生端应用针对底层检测和计数算法的一个补丁，主要是针对学生一些可能的作弊情况。最常见的为跳绳运动时，人不动而手机上下晃动，对于底层算法来说一样是有效的跳绳动作(这与具体判断逻辑有关，详见下文关于底层算法的介绍)。同时，由于本应用的使用方法是调用摄像头进行目标检测，所以经常性地晃动也会增加算法运行的不稳定性，不利于构建良好的用户体验。从这方面来说，应用层面的防抖算法是必要的。

抖动检测中同样用到了手机的加速度传感器。其实原本有更好的数据来源——陀螺仪，但考虑到目前仍由部分手机没有陀螺仪这个传感器，最后还是决定使用加速度传感器来代替。

抖动算法原理实际上就是计算一段时间内竖直方向上总加速度的变化率，一旦超过一定阈值，函数便判断这是一次有效地抖动，然后弹出提示性信息告知用户并暂停计数。运动记录活动的类中有一个私有变量 lastTime 记录上一次调用检测函数的时间，还有一组变量记录上一次检测抖动时获取的xyz加速度，每次重新检测是否有抖动时，获得当前时间与 lastTime 取差值 diffTime，当 diffTime 大于预设的时间间隔时，检测一次抖动值(即对加速度做差值再除以 diffTime)，若抖动值大于预设的阈值，则为有效的抖动。无论是否发现有效抖动，都将更新 lastTime 等记录的值。

本抖动检测算法可以通过调整时间间隔阈值和抖动值阈值两个预设常量来改变检测灵敏度。同时，本抖动算法也会对手机的旋转摇晃做出反应，当缓慢抖动时，由于时间间隔较大，所计算的抖动值达不到阈值，系统就不会认为这个是有效抖动，而是作为一次普通的手机旋转。这确保了抖动检测和方向检测不会大概率发生冲突。

**3.5 数据管理说明**

本项目中的教师端为基于Python中的Django框架搭建的Web Server与基于bootstrap框架编写的前端Web页面；在后台使用了Mysql作为数据库进行数据的统一管理，同时使用Django自带的model类对数据库中的数据项进行上层封装。

本项目中的学生端为基于 Java 编写的 Android 应用，内部管理本地数据时使用了嵌入式数据库 SQLite，可以通过 Android 提供的封装类 SQLiteOpenHelper 对数据库进行简单的数据管理。

**3.5.1 学生端数据库**

SQLite，是一款轻型的数据库，是遵守 ACID 的关系型数据库管理系统，它占用资源非常的低，同时能够跟很多程序语言相结合，比如 C#、PHP、Java 等，还有 ODBC 接口，同样比起 MySQL、PostgreSQL 这两款开源的世界著名数据库管理系统来讲，它的处理速度比他们都快。

学生端数据库负责维护三张表, User 表保存账号的基本信息，Score 表保存成绩信息和每项运动对应的时间限制，Properties 表保存应用的设置内容。应用内通过 AccountDatabaseHelper 类的公共方法来进行增删查改操作。

create table User (  
 id text primary key not null,  
   name text not null,  
   password text,  
   registered integer  
);  
​  
create table Score (  
   sportName text primary key not null,  
   bestRecord integer,  
   timeLimit integer  
);  
​  
create table Properties (  
   propertyName text,  
   value integer  
);

**3.5.2 教师端数据库**

MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。MySQL 作为关联数据库，将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，增加了速度并提高了灵活性。项目使用Django自带的model类对Mysql数据库中的数据项进一步进行了上层封装，便于在Web Server中对数据项进行增删改查，提升了数据增删改查的速度与灵活性、并提高了代码的可读性。

**4 实现与测试**

**4.1实现环境与代码管理**

* + 1. **实现环境**

1. 学生端

计算机环境：Windows 11 操作系统

实现语言/工具：Java(jdk-17)，SQLite

IDE: Visual Studio Code，Android Studio

安卓端测试平台信息：高通骁龙™865(soc)，Android 11(系统版本)

1. 教师端

开发工具：VSCode

开发语言：Python+Html5+Css+JavaScript+MySQL

Web框架：Django

数据库：MySQL

操作系统：Windows 11

python环境需求：django 和 mysqlclient

* + 1. **代码版本管理的签入记录**



图4-1 码云平台代码签入

**4.2 关键函数说明**

**4.2.1 学生端**

1. **Account.java**

表4-1 Account类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| public Account() | 无参构造方法，初始化游客模式下的账户信息。 |
| public Account(String studentName, String studentId, String password, int... scores) | 初始化登录状态下的账户信息，参数依次为用户名/学号/密码和教师端保存的最佳成绩(scores数组长度限制为支持的运动类型数量)。 |
| public boolean updateRecord (SportsType sportsType, int score, boolean force) | 按照运动类型 sportsType 更新对应成绩，force表示是否强制更新最佳成绩(force 为 true 时，无需比较成绩高低强制刷新)。 |
| public boolean updateAllRecord (boolean force, int... scores) | 批量更新所有成绩。 |
| public void setSingleLimitBackup(SportsType sportsType, int limit) | 按照运动类型 sportsType 更新对应时间限制。 |
| public void setLimitBackup(int... limits) | 批量更新时间限制。 |
| public void rewind() | 将账户信息恢复到游客模式。 |

1. **AccountHolder.java**

表4-2 AccountHolder类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| public static boolean register(String name, String id, String password, boolean registered) | 登录函数，接收的参数都是要写到本类的私有静态的 Account 对象中的信息。 |
| public static boolean uploadLimit(int index, int limit)，  public static boolean uploadLimit(SportsType sportsType, int limit)，  public static boolean uploadLimits(int... limits)，  public static boolean uploadRecord(SportsType sportsType, int record, boolean force)，  public static boolean uploadRecords(boolean force, int... records)， | 这五个方法分别封装了 Account 类中的 setSingleLimitBackup、setSingleLimitBackup、setLimitBackup、updateRecord 和 updateAllRecord。 |
| public static boolean uploadData(boolean force, int[] records, int[] limits) | 批量更新所有的最佳成绩和时间限制。 |
| public static boolean logout() | 退出当前帐号，内部封装了 Account 类中的 rewind() 方法。 |

1. **Setting.java**

表4-3 Setting类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| public boolean setSwitchText(String[] switchText) | 为 SWITCH 类型设置控件添加和开关关联的文本。 |

1. **HomeActivity.java**

应用的主页(主活动)。onCreate、onStop、onResume、onPause、onDestroy等方法是 Android 应用活动在其生命周期中的不同状态下会执行的方法，所以本应用中的大多数活动类都会重写这些方法，所以只在 HomeActivity.java 中介绍一遍。

表4-4 HomeActivity类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| protected void onCreate(Bundle savedInstanceState) | 活动在创建时完成必要的初始化工作，包括但不限于初始化界面中显示的布局和控件，给控件添加响应事件、初始化本应用的本地数据库并加载其中的数据。 |
| protected void onStop()  protected void onPause()  protected void onDestroy() | 执行父类中的对应方法，同时增加了保存信  息的 save() 方法。 |
| protected void save() | 保存当前应用中的数据到本地数据库。 |
| protected void load() | 从本地数据库获取之前保存的应用数据。 |

1. **collectionFragment.java**

应用主活动(HomeActivity.java)的子界面之一。

表4-5 collectionFragment类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle saveInstanceState) | 初始化界面内容，绑定界面中的控件并为其添加响应事件。 |
| public void onResume() | 作用与 活动类 的 onResume() 相同，当这个界面会到前台显示时，会通知控件显示内容根据数据改变做出更新。 |

1. **accountFragment.java**

应用主活动(HomeActivity.java)的子界面之一。

表4-6 accountFragment类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle saveInstanceState) | 初始化界面内容，绑定界面中的控件并为其添加响应事件。 |
| public void onResume() | 作用与 活动类 的 onResume() 相同，当这个界面会到前台显示时，会通知控件显示内容根据数据改变做出更新。 |

1. **TestSecondActivity.java**

运动记录活动。

表4-7 TestSecondActivity类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| private void reload() | 重新加载底层检测模型。 |
| public void onSensorChanged(SensorEvent sensorEvent) | 获得传感器数据。在本活动中只负责获取加速度数据，然后调用方法calculateNextAngle()计算控件旋转动画的角度值来启动旋转动画，以及调用 checkShaking() 方法检测手机是否抖动。 |
| public void onResume() | 除了运行父类方法外，还重新注册了传感器服务和打开摄像头画面。 |
| public boolean onKeyDown(int keyCode, KeyEvent event) | 当用户退出本活动时会触发(比如按下返回键)，弹出提示信息并暂停检测。 |
| private long calculateNextAngle(float acceX, float acceY, float acceZ) | 计算旋转角度。 |
| private void checkShaking(float acceX, float acceY, float acceZ) | 检测手机是否抖动。 |

1. **SportsCompleteActivity.java**

运动计算界面(活动)。

表4-8 SportsCompleteActivity类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| private void submitData(int scoreData, SportsType sportsType) | 上传数据到远程教师端服务器，根据返回内容提示用户是否上传数据成功。 |

1. **FaceDetectionActivity.java**

人脸检测界面(活动)。

表4-9 FaceDetectionActivity类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) | 获取返回当前活动之前上一个活动的结果码。这里是指人脸检测活动调用了系统的相机来拍摄人脸照片，相机活动退出后，人脸检测活动根据相机退出后返回的结果码来进行下一步动作(将拍摄的照片转为二进制串)。 |
| private void submitFaceData(String faceByteString) | 将照片的二进制串插入报文发送给教师端进行人脸检测。 |
| private void getAcknowledgment() | 等待教师端返回的确认报文和其他信息。 |

1. **LoginActivity.java**

登录界面(活动)。

表4-10 LoginActivity类主要方法

|  |  |
| --- | --- |
| **方法声明** | **作用/功能** |
| private void submit() | 提示并等待用户输入账号密码(这里的账号指的是学生学号)。 |
| private void InternetSever(String username,String password) | 连接远程教师端数据库进行登录操作，并等待教师端的返回报文和信息。 |

**4.2.2 教师端**

1. **doupload函数**

入口参数：request请求对象

返回值：对POST请求返回渲染后的页面：render(request,"student/info.html",context)或以HttpResponse的形式发送简单的错误提示信息，对GET请求返回渲染后的页面

功能：提供对学生信息上传页面的处理，对POST请求传递的对应学生信息进行处理，新建一个数据库学生数据项，将学生的学号、姓名等文本信息赋值给对应对象，将照片文件保存入按规则生成的特定路径，并将路径写入学生数据项的照片表项，同时将照片文件送入人脸检测+识别模型，将输出的人脸向量写入对应数据表项，并向前端返回正确的提示信息；若照片文件为空，则返回照片文件为空的提示信息，若照片文件中不含人脸，则人脸检测+识别模型会返回空结果，返回照片中不含人脸的提示信息。

1. **login\_auth函数**

入口参数：request请求对象

返回值：对POST请求根据用户名与密码的验证结果返回渲染后的不同页面，对GET请求返回渲染后的页面

功能：提供对教师身份的验证登录，将POST请求传入的教师工号与密码，与数据库中存储的教师数据项进行比对，若登录成功，则直接跳转到学生信息浏览界面；若登录失败，则重新刷新当前页面。

1. **judge\_student函数**

入口参数：request请求对象

返回值：对POST请求，根据其传递的学生信息进行验证并返回验证结果；对GET请求返回渲染后的页面

功能：为学生端（APP）提供学生身份验证的交互接口，函数接收APP发送的POST请求的学生信息：学生学号与照片文件（二进制字节流），并将照片文件输入人脸检测+识别模型，将输出的人脸向量与学生学号对应的人脸向量进行点积，即可计算出人脸相似度confidence，若confidence>0.6（满足阈值判断条件），则学生身份验证成功，否则验证失败，最后以json文件的格式返回验证结果。

**4.3 测试计划和测试用例**

**4.3.1 学生端**

由于运动计数需要视频展示，文档部分就不展示运动计数相关的测试了，感兴趣的可以下载项目安装软件后自己测试。

1. 登录测试
2. 网络异常

网络异常的情况对应未连接到服务器或者手机不能访问网络。



图4-2 网络异常示意图

1. 未输入账号

尝试只输入密码。



图4-3 未输入账号示意图

1. 未输入密码

尝试只输入账号(下页图)。



图4-4 未输入密码示意图

1. 登录成功

网络良好时，输入正确的账户和密码即可登录成功，用户界面会显示当前登录者的学号(因为没有保存用户头像功能，所以这里是固定的头像)。



图4-5 登录成功示意图

1. 人脸识别测试
2. 网络异常

网络异常的情况对应未连接到服务器或者手机不能访问网络。



图4-6 网络异常示意图

1. 识别失败

尝试上传一张非账号持有者本人的照片或者没人的照片。



图4-7 识别失败示意图

1. 识别成功



图4-8 识别成功示意图

1. 成绩上传测试
2. 上传成功

成绩可以上传成功当且仅当应用处于已登录状态并且开启了人脸识别功能(当然还不能网络异常)。



图4-9 上传成功示意图

1. 上传失败

未登录或者登录状态下未开启人脸识别功能时会上传失败。



图4-10 上传失败示意图

**4.3.2 教师端**

1. 教师身份验证测试
2. 验证成功（进入学生信息浏览页面）



图4-11 验证成功示意图

1. 验证失败



图4-12 验证失败示意图

1. 学生信息上传测试
2. 上传成功提示



图4-13 上传成功提示示意图

1. 上传文件为空提示(下页图)

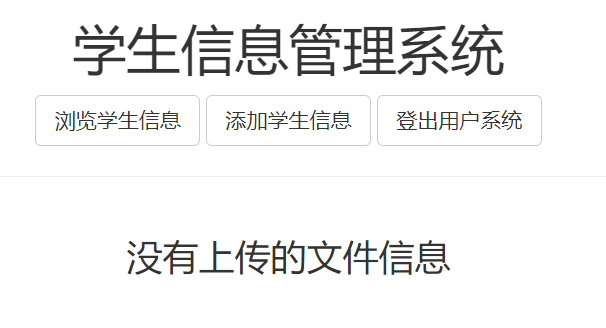


图4-14 上传文件为空示意图

1. 照片文件中未出现人脸提示

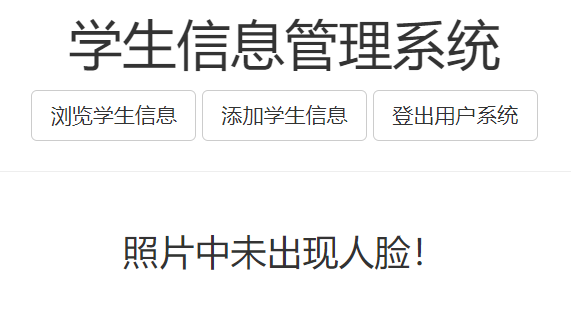


图4-15 照片无人脸示意图

**4.4 结果分析**

**4.4.2 学生端**

4.3节中主要测试了学生端的登录、人脸识别和成绩上传操作时的不同结果。可以看到系统可以有效分辨不同的情况并且对有风险的操作进行规避，并且对用户发出提示性信息。

**4.4.2 教师端**

根据测试分析运行结果、确认软件是否满足需求。由测试结果图11，12可知，教师端能够完成教师账号的验证与前后端信息的通信，并且能够向前端正确返回验证结果和正确重定向到目标页面。由图11可知，教师端能够正确呈现各个学生的信息。由测试结果图13，14，15可知，在上传学生信息的过程中，教师端能够完成学生信息的上传与前后端的通信，并且能够根据前端输入信息的不同情况，向其返回不同情况下对应的提示信息。学生信息的更新、删除等其他功能测试图限于篇幅原因不作展示，但由测试结果可知，教师端能够正确完成学生信息的增删改查，返回对应的提示信息并在学生信息浏览页面上进行及时更新。

**5 总结**

**5.1 用户反馈**

我们邀请了两名组外同学，分别对软件系统进行了整体流程的测试与体验，并收集了他们的反馈如下：

1. 用户1；

**优点**：界面有自己的特色，很有吸引力，采用了人脸识别等AI算法，和其他软件相比具有一定的优势，另外软件手感也很不错，功能完善，响应速度很快。

**建议**：人像识别的界面可以再美化一下，数据的安全性以及用户准入策略可以再完善一下。

1. 用户2；

**优点**：APP界面具有现代特色，符合学生审美，非常美观；网页界面简单整洁，便于老师操作，人脸识别与姿态估计能够正确运行，没有出现判断错误。

**建议**：可以添加更多种类的动作，运动界面也可以多添加一些信息。

**5.2 全文总结**

我们完成了一套端到端的体育运动检测与统计系统，能够直接落地应用于网课情况下学校体育课堂的日常锻炼与检测情景，系统整体分为供教师统计与管理学生信息的教师端（Web Server）和用于体育运动检测的学生端（Android App），体育运动检测算法是在神经网络人体姿态估计算法的基础上，结合不同运动编写对应判断逻辑实现的，同时结合人脸检测识别算法，实现了对学生真实身份的验证，防止舞弊的发生。

在底层算法部分，我们使用了arm汇编优化来优化人脸检测识别模型与人体姿态估计模型的运行速度和内存消耗，使用JNI来让学生端的Java前端部分调用底层C++检测程序；在学生端，实现了简洁的操作界面和较为完整的功能作为连接本次项目三个子项目的桥梁；在教师端，使用django搭建Web Server，基于Bootstrap编写前端页面，使用了MySQL数据库管理后台数据。

**6 体会**

1. 逯润雨

我项目中负责人脸检测、姿态估计AI模型的C++重写前向推理，并使用arm端汇编代码进行性能调优，交叉编译打包到安卓端，另外安卓端实现C++调用摄像头并编写JNI代码实现Java与C++交互，以及运动逻辑的检测实现。以下我会每个部分着重说下遇到的问题以及对应的解决方法。

**内存泄漏**： 首先C++重写模型需要提前设计网络结构各种类以及对应分配的内存。这就导致了“内存泄漏”问题，其实不太算内存泄漏，虽然程序运行完一定会回收，但是由于程序本身如果在安卓端运行会运行非常长时间，所以就会导致我们程序运行的时候反复推理反复申请新的内存，而旧的内存还没有完全来得及释放。不及时的被释放会导致内存申请过大，这样一来就导致我们的内存在程序运行过程中不断增长直到最后超过手机本身运存大小而崩溃。最后尝试了很多奇奇怪怪的内存泄漏工具，比如腾讯开发的matrix，非常经典的valgrind等等，但是由于个人的操作问题感觉都没有太有帮助，后来还是发现实际上还是遵循一个原则，那就是刚开始我先申请一大片内存（提前计算好），然后用到的时候就接着用，用完立马释放，因为神经网络一个blob一个blob往前传，所以很容易就传完这一层就释放这个很容易实现，虽然好像占的内存大一些，但是程序可以非常稳定的运行出来了，这让我想起来程序一开始还是跑的稳一些比较好，不要自以为是的加很多自己以为很省内存的内存管理方案，自己水平到不了就不能做这些事情。

**Gdb调试类析构问题**：另外就是刚开始设计的Mat类比较随便，抄了opencv和ncnn的一些代码，然后自己加了一些运算符重载比方说指针重载等等，同时对于Mat本身使用饮用计数，类似于shared\_ptr。但是当时写release进行析构的时候一些构造函数行为和之前写得不太一致，导致最一开始的时候在我本地linux的环境下进行最普通的运行就会报段错误，这个问题坑了我非常非常长的时间。大概有4、5天全都在调试这部分的bug，一直使用gdb进行单步调试，最后终于将Mat类，以及各个网络层类的拷贝赋值、运算符=重载函数调试完成，至少一个可以正常运行的demo做出来了。

**交叉编译问题：**其实这部分倒是问题不太大，无非是装一个ndk包就行了，主要问题在于我当时使用的时候不太知道Android端adb调试工具，导致之前测试流程非常冗杂，自己先在linux端交叉编译好，然后放到android\_studio上运行出apk，然后再手机上跑测一测fps，这一整套流程下来真的是太慢了！！！我当时就觉得这样开发的效率太低，后来问了问一些Android开发的工作学长发现有一个安卓端调试工具adb，这个很大程度上提高了我的开发效率，手动的移动改成了一个shell脚本从项目编译、到移动到手机端、再到运行全部再linux脚本提前写好，然后直接运行，虽然项目编译本身需要一分钟，但是整体流程自动化了，个人觉得非常nice.

**安卓端实现C++调用摄像头：**刚开始这部分我并不打算负责，我只想让Java端的同学来编写这部分的代码，然后我暴露出一些检测的JNI接口供使用就行。但是这样一来在交叉编译好放到android端进行adb调试的时候，profiling看效果的时候是60~80ms左右，但是一放到android studio上，放到应用端FPS就突然慢了，怎么都上不去，打印一下pringlog查看log分析发现同样的手机，单纯跑编译出模型很快，放到程序中集成突然就时间多了几十毫秒！！！但是之后发现使用Java端的接口光是调用初始化Java相机、传参进入Java函数体内这些操作就要等好久好久（几百ms）。这样一来我们的性能就被Java相机拖累了，没有办法，只能将这部分调用相机代码也是用c++实现，调用的是ndk里面相当于原生的相机类。后来速度问题果然得到了解决。

**运动逻辑的检测实现：**这部分主要难点在于测试，当时b站找了个别人做仰卧起坐、蹲起的视频，然后按照对应的设计写好了运动逻辑，但是后来发现现场做居然有些时候计不了数，后来整理一下思路才发现问题原来在于AI模型本身的阈值不完全一样导致的，后来改了改阈值之类的参数调好了。另外就是反复和组员们在各个场地测试我们的检测逻辑、一起讨论怎么设计能够让他的准确率变高等等。

总体来说，亲自经历过这一整个流程还是感觉收获非常大的，这其中我亲身经历了做一个能够跑出来并且有一定优化的项目是多么的困难，其中开发工具、项目版本管理、打包静态库、C++与Java交互，每一个关卡或多或少都会出现很多问题，最初文件读取函数写得就有问题导致文件迟迟读不进去的时候当时就觉得很绝望，后来gdb调试过去了，感觉开发程序有将近一大半的时间全部都是在使用gdb调试bug，因为不知道什么时候就会突然程序报错，core dump等错误也好，C++类内存拷贝行为错误也好，但是当整个项目能够顺利跑出来，速度和大小控制的都满足了预定的要求的时候，那种自豪感还是非常大的。

1. 王怡彬

在这次的软件工程实验中，我负责实现了教师端整体系统前后端的设计与实现，同时完成了与APP之间交互接口的设计与实现，并完成了人脸识别检测模型的选定。

在立项时，我最初的工作预想只是完成人脸检测模型的实现与训练，但在后续交流讨论中发现，为了完成系统整体服务的端到端实现，我们需要实现一个能够供老师管理学生信息，并与APP实时交互的教师端系统，于是我就接手了这个系统的设计与实现。在项目初期，我由于比较缺乏网页前后端知识，对如何实现这个系统并没有具体的思路，通过咨询老师与搜集资料，结合网络上一些先有的模板，自学相关知识，一步步搭建出了一个简单的Web Server与对应的前后端服务。

在编写后端逻辑时，我对后端框架中的POST/GET请求经常遇到处理错误，在恶补了对应的计算机网络知识后逐步调试解决了问题，同时花费了大量时间学习Django框架的代码，主要是其render渲染器的逻辑与数据库上层封装框架的使用，但Django确实是非常易于上手的一款Web Server框架，对MySQL数据库提供了非常便于使用的上层封装框架，且后端算法逻辑也非常简洁，使得我较为轻松地完成了教师端系统后端的编写，同时使用Bootstrap一些简单的页面设计框架，完成了前端页面的设计实现。

但我们在教师端与APP的交互接口设计实现过程中遇到了很多问题，我们首先在查阅了相关项目后，决定在教师端为APP提供专属的POST请求接口，同时以json文件作为双方互传文件的统一格式。但随后发现，APP在进行人脸验证时，由于需要将学生真人照片上传给教师端服务器，再由服务器运行验证算法后返回验证结果，这一过程中会消耗2-3秒的时间，导致了APP发起的POST请求由于超时而中断连接，故而我们结合TCP的设计思想，在APP给教师端发送验证请求与学生信息后持续监听服务器验证结果，服务器在未验证时始终返回标志0，在验证成功后返回标志1，验证失败后返回标志2，从而解决了这一问题。

在整个工程项目的设计与实现过程中，我独立完成了一个简单的网页服务系统的前后端设计与实现，同时与小组成员合作实现了与APP之间的交互接口，收获了宝贵的工程设计实现经验与项目规划合作经验。

1. 苏庆远

我在本次项目中主要负责设计和实现学生端的Android应用程序。

在这次项目之前，我对Java已有一定的了解，并且也尝试自己做过Android应用的个人开发。但是这次的项目经历还是和之前大有不同，我需要同时调用底层检测模型算法和远程教师端连接的接口，而不是单纯制作一个本地的小应用。项目推进的过程中，我们小组在讨论时不断丰富应用的功能设计，并且在征求老师的意见时，老师也给我们的项目提出了新的合理的要求，这使得这次应用开发对我来说是一个新的挑战。

项目开发阶段刚开始时，由于我有半年多没有接触过Android应用开发，所以大部分时间我都在重新学习。当然这次复盘也让我对应用开发有了新的理解：这次应用的内部逻辑关系和数据结构定义也比我之前做的要合理很多；我也尝试使用了更多之前没有使用过的控件和官方支持库，尝试制作不同的动画效果来丰富软件体验。

在对接其他两位同学负责的部分的时候，我也碰到过不同的问题和困难。在尝试调用底层检测算法时，我只需要将逯润雨提供的模型C++实现、参数文件和JNI文件加入应用的项目文件夹中，再修改项目的本地参数(ndk、cmake路径和版本等)即可通过调用对应的模型类中的实例方法。但是由于之前没有了解过Java本地接口调用C++代码的整个过程，我在这里还是犯了个常识性的错误：由于我和逯润雨一开始并没有在同一个项目文件里面编写我们的代码(其实分开也很有必要，因为他需要单独的模型调试工作而我的应用部分在他那里实际上作用不大，而我也要一直添加应用需要的图片、布局文件等素材，所以我们选择了到项目开发后期再将所有的功能进行整合)，导致他在生成cmake文件和JNI类时的文件路径和我现在使用的有所不同，也就使我不能直接照搬他所有的代码，而是在我的应用项目文件夹里另开一个和他之前一模一样的路径来存放JNI类使得文件识别成功。

在对接教师端服务器的时候我们也遇到了一些问题。其一就是因为我们一开始并没有这方面的知识储备，一开始都是在网上现查别人的博客内容来学习。为了能在演示的时候展示连接功能，我们一开始打算使用WiFi热点建立一个内网来连接(我的手机连接王怡彬同学作为服务器的电脑)，但是后面我们也发现这样做其实并不稳定，而且也不太符合这款应用的实际使用场景，后来就想到使用花生壳客户端来做一个简单的内网穿透，这样应用就可以通过公共网络访问到远程数据库，而不是在每次重新连接时修改代码(因为每次开热点进行连接我这边都需要修改代码里面的发送报文的目标地址，很蠢)。

其二便是做人脸识别模块时候碰到的等待问题。人脸识别是我们在项目后期才打算实装到客户端上的功能，所以短期内的设计有点不太合理：一次人脸识别的过程，要从我这边(学生端)拍照上传图片到教师端(其间需要转换为二进制流)，教师端调用人脸检测程序得到结果后回传到学生端。这期间需要经过两次I/O，本地调用一个独立.exe文件得到结果以及内网穿透等过程，花费时间巨大，大概需要10s左右，而学生端和教师端之间建立的连接没这么久，结果就是连接超时，学生端接收不到回应然后解析报文失败闪退。一昧增加连接超时时间显然也不是一个好办法。于是在没有更多时间优化这个人脸识别功能的效率的情况下，我们的首要目标是让这个功能起码可以跑起来，所以就模仿了计网里学习的TCP的相关内容，在上传一次图片数据之后，由学生端每隔一段时间再发送一次询问报文询问结果，直到得到教师端的对应响应报文为止。这样一来人脸识别就可以正常工作了，可惜的是这也只是一个临时的救急方法。

总体来说，在这次软件工程的项目开发中我还是学到了许多，复习了遗忘的知识，也积累了一些团队开发的经验。我也明白了一个团队想要开发出令人满意的应用，还是非常不容易的。期待这次宝贵的开发经历能对我未来的学习和工作生活有所帮助！

**附录**

代码过多不宜在此展示，更多内容请详见项目src文件夹。

项目地址：[HPC-Android-Human-Detection: A High Performance AI Human Detection System On Embedded System. (gitee.com)](https://gitee.com/moisten-the-rain/HPC-Android-Human-Detection)

**项目小组成员签名**

**王怡彬 逯润雨 苏庆远**