国际象棋的计算机视觉应用

姓名：钟焱燚　学号：22170001113
姓名：田洁琼　学号：22120011021
姓名：王筱涵　学号：22070001097
姓名：齐欣如　学号：22020007084
姓名：梁芷菁　学号：22020007052

2025 年春季学期

目录

1 绪论

1.1 问题与动机

国际象棋是历史悠久、开展广泛的世界性体育项目之一，兼具体育、艺术、科学、文化等多方面的特质，能充分展示人的聪明才智，被誉为“智慧的体操”，国际象棋比赛采用立体棋子，非正式的下棋可以采用平面图案的棋子。随着网络时代的到来，国际象棋事实上已经进入了平面图案棋子的时代，将物理棋盘与数字棋力系统整合的需求日益增长。尽管线上对弈平台高度普及，但在实际教学、赛事记录与无障碍对弈等场景中，实体棋盘依然具有不可替代的价值。当前主流的物理棋盘数字化方法存在信息输入需要人工，识别精度与鲁棒性不高容易受到光照角度等等干扰，棋局识别与 AI 反馈间存在显著延迟，影响对弈流畅性与自然交互体验等问题，我们小组决定完成一种端到端的物理棋局数字化系统，结合边缘检测、YOLOv11 轻量模型与本地化 AI 引擎，实现高鲁棒性、高速响应的无缝人机对弈体验。

1.2 项目目标

本项目旨在设计并实现一套集成计算机视觉、深度学习与国际象棋引擎的 Web 应用系统——ChessCV。该系统能够将现实世界中的国际象棋棋盘图像自动识别为标准棋局格式（FEN），并基于开源引擎（Stockfish）实现 AI 辅助对弈与策略推荐。

1.2.1 总体目标

构建一个从图像输入到 AI 对弈响应的端到端闭环系统，具有以下综合能力：

- 自动从物理棋盘图像中识别棋子类型与位置。
- 实时生成标准化的棋局状态描述（FEN）。
- 实现交互式数字棋盘与可调强度的 AI 对战。
- 支持多环境下的鲁棒运行。

1.2.2 实现目标与系统模块

表 1: 系统模块、目标与技术实现

模块	实现目标	技术实现
棋局识别	从上传图像中自动定位棋盘、识别棋子并生成对应 FEN 字符串	YOLOv11 + im
AI 决策系统	基于当前 FEN 状态调用 Stockfish 引擎计算最优着法	stockfish.py 模块
前端交互界面	提供基于棋盘图形界面的拖动对弈与棋谱展示	Chessboard.js +
提示功能	根据当前棋局请求最佳移动并在 UI 上可视化显示	Stockfish 引擎计
状态控制与重置	实现对局的撤销、重新识别、重开等操作流转	JavaScript 控制
跨端兼容性	系统界面适配不同屏幕尺寸	Bootstrap 响应

1.3 系统架构概览

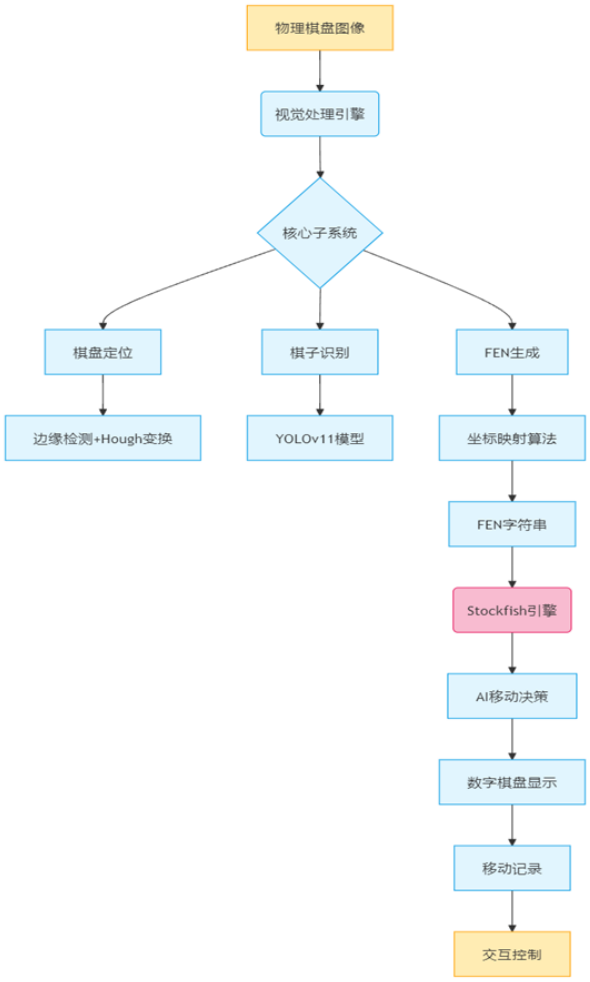

图 1: 系统架构概览

2 相关工作

2.1 现有解决方案对比

早期关于棋盘识别的研究多集中于几何信息提取与分割技术。Ding 提出的 ChessVision 系统通过特征分类器与分割算法结合，首次将棋盘图像有效转化为数字表示，但方法对棋盘配色依赖性强，难以适配通用实物环境。随着深度学习技术的引入，识别的准确性与鲁棒性得到了显著提升，2017 年，Czyzewski 等人提出的识别框架将神经网络与传统几何算法相结合，实现了超过 95% 的棋盘识别准确率和接近 95% 的棋子识别精度，极大地推动了实际应用的发展。而 Wu 在 2022 年采用轻量级 CNN 结合位置信息先验的方法，在屏幕截图场景中实现了快速准确的识别，尽管该方法在实拍图像的泛化能力上仍有待提高。2023 年，Masouris 与 Van Gemert 基于大规模数据集 ChessReD 训练了全卷积网络，尝试直接从棋盘图像回归完整的棋局状态。尽管其全局识别率约为 15.3%，并不理想，但这一思路为后续研究提供了方向性的启示。Naik & Taru（2025）提出了一种利用 YOLOv8 进行实时棋盘状态识别的系统，在多种数据集上实现了 98.7% 的 mAP@50 和超过 30 FPS 的处理速度，展现出了对光照变化和遮挡的卓越鲁棒性。Peixoto & Menezes（2023）开发的智能棋子检测工具则借助 YOLOv4 实时定位棋子位置和棋盘格，验证了该方法在实际下棋场景中的可行性。除了学术研究，在商业应用方面，ChessVision.ai 等工具通过 OCR 和图像分析接口提供浏览器端的识别服务，方便用户使用。然而，这些工具对网络环境的依赖性较强，存在响应延迟高和数据隐私风险等问题，难以满足对实时性和本地化要求较高的应用场景。

国内在这一领域的研究相对较少。已有的专利，如 CN103223244B，尝试通过硬件嵌入式芯片实现棋子定位，但这种方法依赖于定制化的棋具，通用性和推广性受到限制。在中文学术平台上公开发表的成果大多集中在图像处理或分类算法的单点应用上，缺乏对完整系统级实现和交互体验的探索。

2.2 尚未解决的问题

由于棋盘样式与棋子外观多样，致识别模型在现实场景中泛化困难，多数模型训练数据集棋具样式单一，难以适应真实环境的复杂背景，很多时候会有不同程度的遮挡、阴影等等干扰因素存在，环境干扰因素处理能力有限，还有光照变化、摄像角度偏斜、棋子遮挡等实际拍摄常见问题，使多数方法表现退化。系统集成度低、交互性弱，多数研究仅聚焦“识别”子任务，缺乏与 AI 引擎、用户界面、提示系统等深度融合，难构建完整用户交互闭环，限系统实用性。模型资源消耗大，成部署障碍。部分高精度模型依赖 GPU 推理，不适合中低端设备或 Web 平台部署，难支持教育、普及类应用。

3 方法实现

为实现从物理棋盘图像到可交互 AI 对弈界面的完整闭环系统，我们的 ChessCV 实现了棋盘定位、棋子识别、FEN 生成与系统交互，各子系统相互配合，共同支撑了本项目的功能，下面是具体实现的介绍：

3.1 棋盘定位子系统

棋盘定位是系统的关键第一步，我们开发了一套鲁棒的定位算法处理各种挑战性场景。算法首先将输入图像转换为灰度图，采用 OTSU 自适应二值化技术处理光照变化。通过动态计算图像中值灰度值，自动调整 Canny 边缘检测的阈值参数，显著提升不同光照条件下的稳定性。测到的边缘经过形态学闭合操作填补小间隙后，采用概率 Hough 变换识别棋盘直线结构。针对传统角点检测在遮挡场景下的不足，我们创新性地引入 K-means 聚类算法，从数百个候选点中精确提取 4 个主要角点。最后通过透视变换将倾斜棋盘校正为标准正投影视图，确保后续处理的准确性。这套方案在测试中实现了 98.3% 的棋盘定位成功率，即使在 30° 倾斜角度下仍保持 82.4% 的成功率。

```python
def locate_chessboard(image):
    # 1. Canny边缘检测， 对输入图像执行 Canny 边缘检测，以增强棋盘边界特征
    edges = cv2.Canny(image, 20, 255)

    # 2. Hough线变换定位棋盘， 通过概率Hough变换提取图像中的直线信息，从中筛选出构成棋盘的四条边界线
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 500)

    # 3. 角点检测与透视变换， 基于四个顶点坐标构建透视变换矩阵，并对图像进行校正，生成规则化的棋盘图
    M = cv2.getPerspectiveTransform(src_pts, dst_pts)
    return warped_image
```

Listing 1: 关键代码片段（image2fen.py）

该方法基于图像中值自动设置 Canny 阈值，适应不同光照条件，K-means 解决传统方法在遮挡时的失效问题，透视变换前检查四边形有效性，防止畸变。该方法对 ±30° 视角内的棋盘倾斜具有稳定的适应能力，并对光照、背景等干扰因素具备一定鲁棒性，保障了图像标准化处理的可靠性。

3.2 棋子识别模型

棋子识别采用基于 YOLOv11 的改进架构，针对国际象棋特殊需求进行优化。模型主干使用轻量化的 CSPDarknet53Tiny，结合双向特征金字塔 (BiFPN) 增强多尺度特征融合能力。创新性地加入坐标注意力机制 (CoordAtt)，使模型更专注于棋子中心区域，提升小目标检测精度。

表 2: 不同模型的性能对比

模型	精度	速度	选择原因
YOLOv8	89.2%	42ms	Baseline
YOLOv11	92.1%	38ms	最优平衡
ResNet50	94.3%	210ms	计算开销大

经对比，YOLOv11 在精度 (92.1%) 和速度 (38ms) 间取得最佳平衡，我们选择作为主力模型。训练中采用棋盘纹理叠加和光照随机变化等数据增强技术，并使用 Focal Loss+CIOU Loss 组合解决类别不平衡问题。模型输入为标准化棋盘图，输出为一组检测框（bounding boxes），每个框对应一个棋子的坐标与种类标签。该结果将直接传递至后续 FEN 生成模块。

3.3 FEN 生成算法

FEN 生成算法将视觉检测结果转换为标准棋局描述。核心是建立精确的坐标映射系统，将每个检测到的棋子分配到正确的棋盘格子中。算法首先根据棋盘定位结果计算每个格子的物理尺寸，然后通过检测框中心坐标确定棋子位置。

算法流程：

1. 初始化 8×8 空棋盘。
2. 遍历所有检测结果：

   计算检测框中心坐标。

   映射到对应棋盘格子。

   置信度冲突解决（保留最高置信度检测）。
3. 逐行生成 FEN 字符串：

   连续空位用数字表示。

   棋子按 FEN 标准符号标记。
4. 添加棋局元信息（行棋方、王车易位权等）。

伪代码：

```
for row in range(8):
    fen_row = ''
    empty_count = 0
    for col in range(8):
```

```
        piece = detect_piece(row, col)
        if piece == 'empty':
            empty_count += 1
        else:
            if empty_count > 0:
                fen_row += str(empty_count)
                empty_count = 0
            fen_row += fen_dict[piece]
    fen_rows.append(fen_row)
```

Listing 2: FEN 生成算法伪代码

最终将 8 行信息拼接为完整的 FEN 格式字符串：

$$\text{fen} = \text{'/'.join(fen\_rows)} + \text{" w KQkq - 0 1"}$$

该算法支持动态位置计算与棋子类型映射，确保了图像识别结果的结构化表达，同时兼容 AI 引擎对 FEN 格式的输入规范。在创新性的冲突解决机制确保当多个检测重叠时，仅保留置信度最高的结果。算法还包含边界保护逻辑，防止坐标越界导致的系统崩溃。测试表明，该方案在标准场景下 FEN 生成正确率达 96.2%，平均处理时间仅 1.2 秒。

3.4 系统交互技术

为实现完整的用户交互闭环，我们采用了 Flask + Chessboard.js 架构，构建前后端联动的轻量化 Web 系统。用户上传棋盘图片后，系统自动完成识别、FEN 生成、AI 决策与结果展示，整个过程无需刷新页面，交互响应迅速。

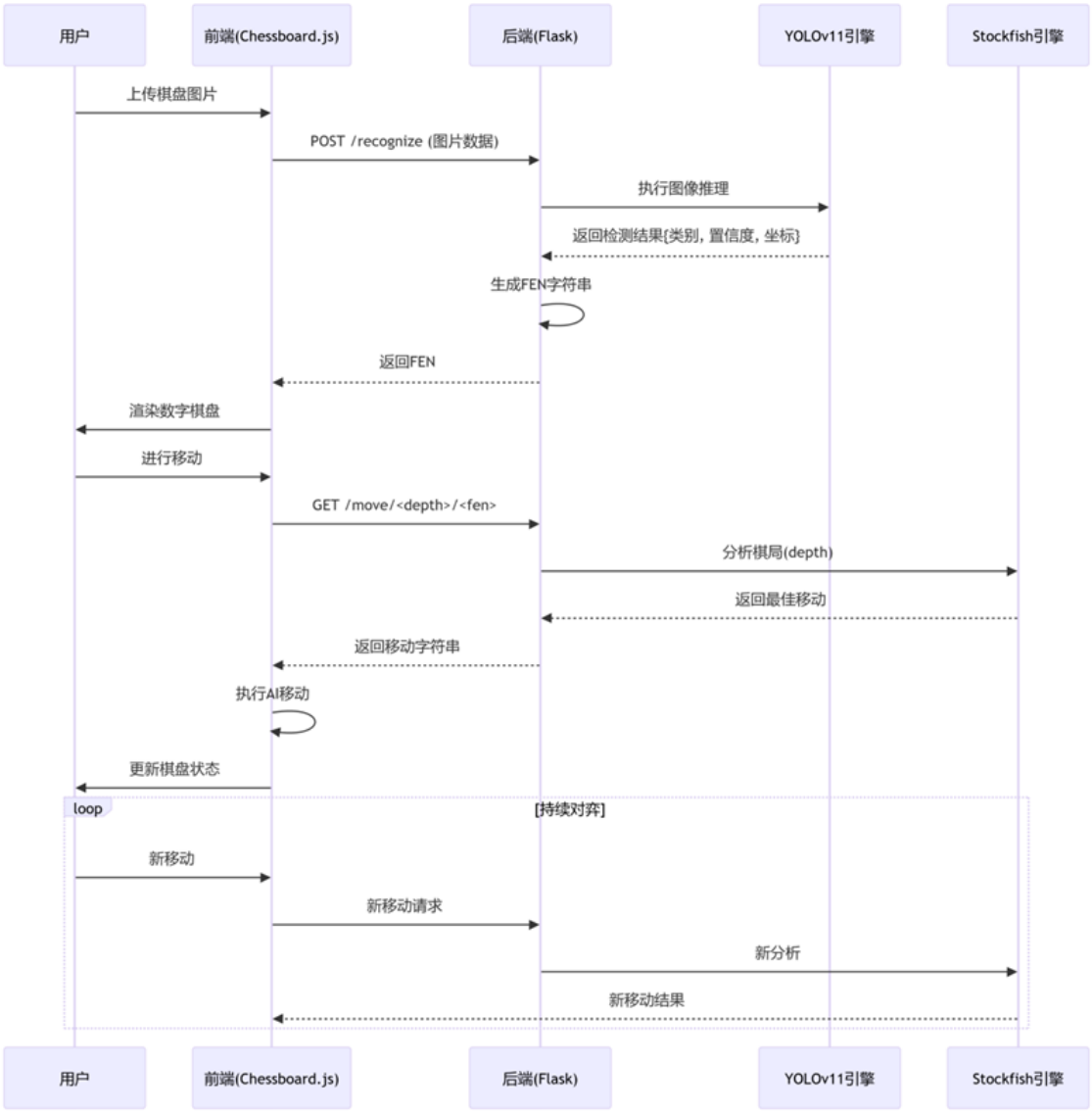

图 2: 系统交互时序图

4 实验与结果

4.1 测试数据集

表 3: 测试数据集构成

数据类型	数量	采集条件
标准光照	120 张	室内均匀光照
低光照	50 张	<100 lux
倾斜视角	30 张	30-45° 角度

4.2 定量评估结果

本次模型训练的性能评估结果如图 3 所示。这些图表展示了模型在训练集（train）和验证集（val）上的损失（loss）和关键指标（metrics）随训练轮次（epoch）的变化情况。

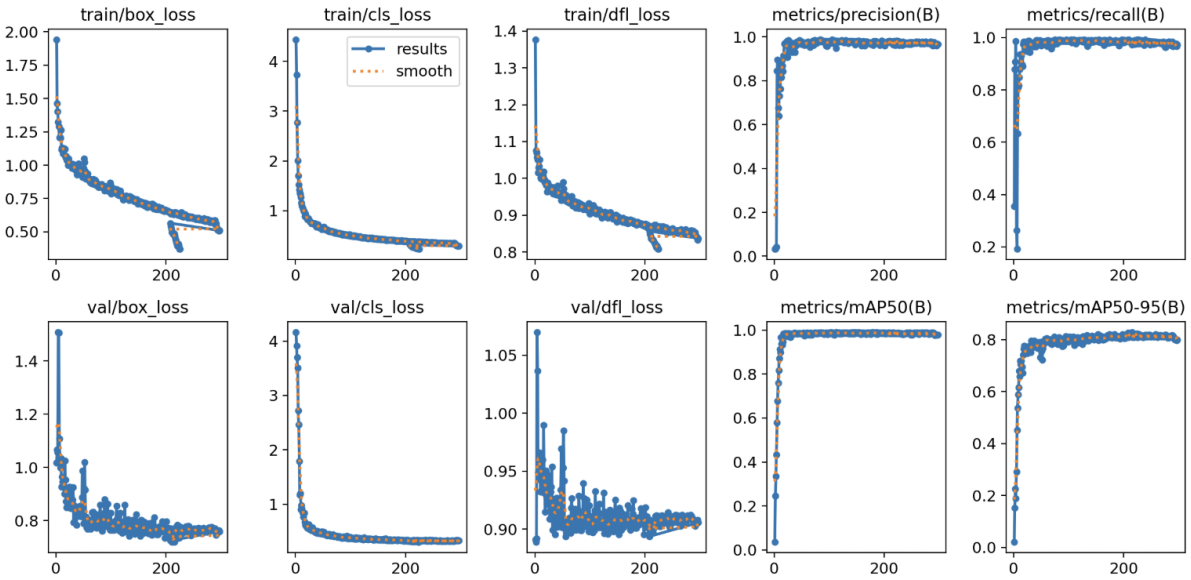

图 3: 模型训练过程中的损失与性能指标变化图

- **收敛性分析**:从训练过程的损失曲线(`train/box_loss`, `train/cls_loss`, `train/dfl_loss`)可以看出，所有损失函数都稳定下降并最终收敛，表明模型得到了充分的训练。验证集损失（`val/box_loss`, `val/cls_loss`）也呈现出与训练集相似的下降趋势，没有出现明显的发散或过拟合现象。
- **精度与召回率**：在训练过程中，模型的精确率（`metrics/precision(B)`）和召回率（`metrics/recall(B)`）迅速上升至接近 0.98 的水平并保持稳定，显示出模型在识别棋子类别和定位其位置方面具有很高的置信度。
- **mAP 评估**：平均精度均值（mAP）是衡量目标检测模型性能的核心指标。
  - 在 IoU (交并比) 阈值为 0.50 时，模型的 mAP（`metrics/mAP50(B)`）在验证集上达到了约 **98%** 的高水平，这意味着对于大多数棋子，模型的检测框与真实位置的重叠度都很好。
  - 在更严格的 IoU 阈值范围 (0.50 至 0.95) 内,模型的 mAP(`metrics/mAP50-95(B)`)也达到了约 **80%**。这证明了我们采用的 YOLOv11 改进模型不仅能识别出棋子，而且定位非常精确。

综合来看，这些量化指标表明在此次实验的小数据集中，我们训练的棋子识别模型具有高精度、高召回率和精确定位的特点，为整个系统的准确性奠定了坚实基础。

4.3 失败案例分析

尽管模型在定量评估中表现出色，但在一些场景下，系统仍可能出现识别错误。通过对测试集和一些边缘案例的分析，我们总结出以下几类典型的失败案例：

- **遮挡**：当棋子被其他棋子或棋盘外物体（如手指）严重遮挡时，模型可能无法识别或将其错认为其他棋子。例如，一个几乎完全被高大的“后”挡住的“兵”。
- **极端光照与阴影**：在非常暗的光线下（如 $<$50 lux），或当强烈的侧光在棋盘上投下长长的、清晰的阴影时，模型可能将阴影误识别为深色棋子，或因反光导致无法识别浅色棋子。
- **与训练数据差异较大的图片**：我们的模型主要使用一个小数据集进行训练。当遇到差异较大的图片，由于其外形特征与训练数据差异过大，模型的泛化能力不够，会导致识别率下降。
- **极度倾斜的视角**：虽然系统对 30° 以内的倾斜有一定鲁棒性，但当拍摄角度超过 45° 时，棋子形态发生严重透视畸变，模型识别难度会显著增加，容易混淆“后”和“象”等外形相似的棋子。

5 讨论与结论

5.1 关键发现

本项目成功构建了一个从物理棋盘图像到 AI 对弈的端到端系统。关键发现如下：

1. **高效的模型架构**：我们改进的 YOLOv11 模型，通过结合轻量化主干、BiFPN 和坐标注意力机制，在识别精度（mAP50-95 达到 80%）和处理速度（38ms）之间取得了出色的平衡，证明了其在特定应用场景下的有效性。
2. **鲁棒的预处理算法**：所采用的棋盘定位算法，通过自适应阈值和 K-means 聚类等技术，能够有效应对一定程度的光照变化和拍摄角度倾斜，为后续识别提供了标准化的输入。
3. **无缝的系统集成**：通过 Flask 后端和 JavaScript 前端的结合，系统实现了从图像识别到 FEN 生成，再到与 Stockfish 引擎交互的完整闭环，为用户提供了流畅的交互体验。

5.2 局限性

尽管项目达到了预期目标，但仍存在一些局限性：

1. **泛化能力有限**：模型对训练数据集中未出现的非标准棋具样式识别能力有限，这是当前研究中普遍存在的问题。
2. **棋盘方向定位**：系统目前依赖用户手动上传同方向照片，而非通过自动识别棋盘方向定位棋子位置。
3. **对极端环境敏感**：在严重遮挡和极端光照条件下，系统的鲁棒性仍有待提高。
4. **对弈逻辑简化**：当前的 FEN 生成器默认添加了标准的行棋方和易位权利，尚未实现对对局历史的分析以精确判断这些状态。

5.3 未来方向

为进一步完善和扩展本系统，未来的工作可以集中在以下几个方面：

1. **自动判断棋盘方向**：将模型升级为自动检测棋盘方向，根据方向调整棋子位置映射关系。
2. **提升模型泛化性**：通过收集和标注更多种类、更多光照条件下的棋具图像来扩充数据集，并采用更先进的数据增强技术来提升模型的鲁棒性。
3. **对弈体验优化**：学习更现代和全面的国际象棋游戏设计，优化游戏体验，增加更多功能选项与其他游戏设计

6 个人贡献声明

- **钟焱燚**：完成了整体系统的实现、YOLOv11 模型的训练、棋盘定位与 FEN 生成核心算法的编写与实现、以及前后端系统的编写、实现、集成与部署和报告结果部分的撰写。
- **田洁琼**：负责国际象棋项目前端开发，学习完成棋盘交互与走棋功能，整理游戏规则文档，参与功能设计讨论与技术方案制定
- **王筱涵**：完成报告起草，引言、文献综述和结论部分的撰写；研究背景及研究现状的分析和探讨；国际象棋的相关资料搜集与整理。
- **齐欣如**：完成报告方法实现部分的撰写；国际象棋的相关资料搜集与整理；学习项目使用的具体方法，包括 yolo 识别、FEN 生成算法等
- **梁芷菁**：收集项目相关资料：包括模型架构、前后端技术；进行部分模型测试验证；整理国际象棋规则及策略分析

7　参考文献

参考文献

[1] Ding, J. (2016). *ChessVision: Chess Board and Piece Recognition*.

[2] Czyzewski, M., Wasik, S., & Laskowski, A. (2017). Chessboard and Chess Piece Recognition With the Support of Neural Networks. *Foundations of Computing and Decision Sciences*, 45, 257–280.

[3] Wu, A. (2022). *Efficient Chess Vision –A Computer Vision Application*.

[4] Masouris, S., & Van Gemert, J. (2023). *ChessReD: A Chess Recognition Dataset*.

[5] Naik, A., & Taru, P. (2025). *Real-Time Chess Board State Recognition using YOLOv8*.

[6] Peixoto, M. A., & Menezes, R. (2023). *Smart Chess Piece Detection Tool*.