暨南大学

本科生论文

论文题目：中文标题

学　　院：信息科学技术学院

专　　业：信息与计算科学

姓　　名：周峻申

学　　号：2022100607

指导教师：肖亮海

二〇二五年六月十六日

中文标题

[**摘要**]　摘要应扼要叙述本论文的主要内容、特点，文字要精炼，是一篇具有独立性和完整性的短文，应包括本论文的主要成果和结论性意见。摘要一般独立一段，不宜使用公式、图表等非常规文本类型，不标注引用文献编号，避免将摘要写成目录式的内容介绍，也不要将摘要写成“前言”。编写摘要应注意：客观反映原文内容，不得简单地重复题名中已有的信息，要着重反映论文的新内容和特别强调的观点，直观给出量化的（对比）结果。摘要在总结前人工作时宜采用第三人称过去式的写法 (如“对……进行了研究”，“综述了……”等)。摘要以 300-400 字左右为宜，可在写完初稿时再写摘要。请注意，使用 XeLaTeX 进行编译，通常一次编译需要使用 XeLaTeX 编译至少两次（使用 BibTeX 请参阅参考文献）。

[**关键词**]　关键词 1；关键词 2；关键词 3；关键词 4；关键词 5

The English title

Abstract: This is an abstract.

Keywords: Keyword 1; Keyword 2; Keyword 3; Keyword 4; Keyword 5

目 录

1 绪论

1.1 研究背景及意义

中国象棋作为一项深受大众喜爱的智力运动，正与人工智能技术深度融合。各类象棋 AI 软件的涌现，极大地推动了象棋文化的普及与竞技水平的提升。在这些智能化应用从虚拟走向现实的过程中，对实体棋盘的自动识别成为一项关键需求，而其核心基础正是对单个棋子进行精准分类。国内已有许多学者研究过棋盘识别[1–3]，得到了不错的效果。但是它们都是基于固定的棋盘，固定型号、样式的、正向的棋子，从正上方拍照得到的图像来识别的，使用深度学习的部分较少或是不使用。而单纯对于单个棋子，从各个角度，对各种型号、样式的棋子进行分类识别的工作很少。而本课题聚焦于此，旨在利用深度学习技术，构建一个能够准确识别 14 种中国象棋棋子的卷积神经网络（CNN）模型。

1.2 本文的主要工作

为了解决什么问题，本文怎么样。为了更好地进行描述，本文的创新点有如下三点：

1） 创新点 1。解释创新点 1。

2） 创新点 2。解释创新点 2。

3） 创新点 3。解释创新点 3。

1.3 本文的组织结构

为了更好地阐述本文的工作，本文的结构如下。第 1 章节是引言，主要讲述了……第 ?? 章节是……第 ?? 章节是……第 ?? 章节是……第 ?? 章

节是……

1.4 术语说明

科学问题（scientific problem）：科学问题，指在科学研究中需要解决的问题，这些问题通常是具有创新性和挑战性的，往往是从工程实践中抽象出来的某一类共通的问题。这些问题的解决方案可以指导工程的实践过程，可以推动科学和工程的发展，以“实践 → 认知 → 实践”的方式不断提高人类对自然和社会的认识和理解。发现、提炼或制造科学问题，是“提出问题 → 分析问题 → 解决问题”中最重要的一环。

可编程的（programmable）：某计算机系统、设备或器件等是可编程的意味着其具有可编程性，即可以通过编程或配置来实现不同的功能或行为。可编程性是现代计算机技术的重要特征之一，它可以使计算机系统或设备具有更高的灵活性、可定制性和可扩展性。

鲁棒性（robustness）：鲁棒性是指系统或算法对于意外情况或异类输入的处理能力，例如能够在面对异常或错误的情况下保持稳定性和正确性的能力。

傅里叶变换（Fourier transform，简称 FT）：傅里叶变换是一种数学工具，常用于将一个函数表示为一组正弦或余弦函数的和。

2 模型理论基础

2.1 深度学习模型的类型

本项目采用卷积神经网络 (CNN) 作为核心的深度学习模型,CNN 是一种特殊设计的前馈神经网络, 尤其擅长处理具有网格状拓扑结构的数据, 例如本项目中的图像数据。

通过局部连接、权重共享和池化等操作，能够有效地从图像中提取层次化的特征，从而在图像识别任务中表现出色。

2.2 模型的整体架构

本模型采用一个自定义的卷积神经网络架构，其设计参考了经典的 CNN 模型（如 LeNet-5 的思想，但使用了更现代的组件如 ReLU 激活函数和批量归一化）。

模型的整体架构可以分为输入模块、特征提取模块（由多个卷积块组成）和分类模块（由全连接层组成）。

2.2.1 模型整体架构图

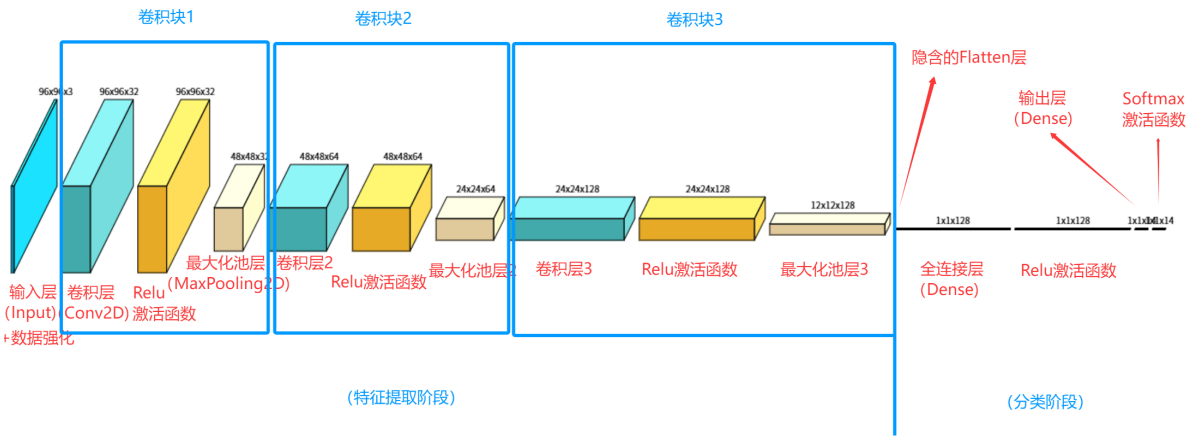


图 2.1: 模型整体架构图

此外，为了提高训练速度和模型泛化能力，我们在每个卷积层后都加入了批量归一化 (Batch Normalization)，并在全连接层中使用了 Dropout。

2.2.2 各层具体结构和功能

2.2.2.1 输入与预处理模块

- **结构**:
  - **输入层 (`Input`)**: 接收尺寸为 96×96 像素、3 个颜色通道 (RGB) 的中国象棋棋子图像。输入张量的形状为 $(\text{batch\_size}, 96, 96, 3)$。
  - **数据增强层 (`Data Augmentation`)**: 在训练时对输入图像进行随机变换，包括随机水平翻转、随机旋转和随机缩放，以增加数据多样性，提高模型泛化能力。
  - **归一化层 (`Rescaling`)**: 将图像像素值从 $[0, 255]$ 区间映射到 $[0, 1]$ 区间。
- **功能**: 为模型提供格式统一、经过预处理和增强的输入数据，有助于模型训练的稳定性和收敛速度。

2.2.2.2 特征提取模块 (Feature Extraction Module)

该模块由三个**卷积块 (Convolutional Block)** 堆叠而成，每个卷积块包含一个卷积层、一个批归一化层和一个最大池化层，其深度逐层增加。

卷积块 1

- **卷积层** (`Conv2D`):
  - **结构**: 使用 32 个大小为 $(3, 3)$ 的卷积核，填充方式为 `` `same` ``，并直接集成 ReLU 激活函数。
  - **功能**: 学习图像的初级局部特征，如边缘、角点和颜色块。
- **批归一化层** (`BatchNormalization`):
  - **结构**: 紧跟在卷积层之后。
  - **功能**: 对卷积层的输出进行归一化，加速模型收敛，稳定训练过程，并提供轻微的正则化效果。
- **最大池化层** (`MaxPooling2D`):
  - **结构**: 使用 $(2, 2)$ 大小的池化窗口。
  - **功能**: 对特征图进行下采样（尺寸减半），减少计算量，并增强模型的平移不变性。

卷积块 2

- **结构**: 与卷积块 1 结构类似，但卷积核的数量增加到 **64 个**。
- **功能**: 在初级特征的基础上，学习更复杂、更抽象的特征组合。

卷积块 3

- **结构**: 与前序模块结构类似，卷积核数量进一步增加到 **128 个**。
- **功能**: 学习更高层次的语义特征，为最终分类做准备。

2.2.2.3 展平层 (Flatten Layer)

- **结构**: 位于特征提取模块之后。
- **功能**: 将特征提取模块输出的多维特征图（例如 $12 \times 12 \times 128$）“压平”成一个一维向量。

2.2.2.4 分类模块 (Classification Module)

由一个全连接块和一个输出层组成。

全连接块 (Fully Connected Block)

- **全连接层 (`Dense`)**:
  - **结构**: 包含 **128 个**神经元，并使用 ReLU 激活函数。
  - **功能**: 整合所有从卷积部分提取的高级特征，并进行非线性映射。
- **批归一化层 (`BatchNormalization`)**: 对全连接层的输出进行归一化。
- **Dropout 层 (`Dropout`)**:
  - **结构**: 设置丢弃率为 **0.5**。
  - **功能**: 在训练时随机将 50% 的神经元输出置零，是防止模型在分类部分过拟合的关键正则化手段。

输出层 (Output Layer - `Dense`)

- **结构**: 包含 14 个神经元，每个神经元对应一个棋子类别。
- **激活函数 (`Softmax`)**:
  - **功能**: 将 14 个神经元的输出转换为一个概率分布，每个元素代表输入图像属于对应类别的概率。

2.3 选择该模型架构的原因

选择卷积神经网络 (CNN) 架构主要基于以下原因，这些原因与本项目所要解决的中国象棋棋子图像识别问题高度契合，并且符合课程中所教授的深度学习原理：

- **对图像数据的适用性**：中国象棋棋子识别本质上是一个图像分类问题。CNN 是目前公认的在图像识别领域表现最优异的模型之一。其核心组件如卷积层和池化层能够自动且有效地从图像中学习层次化的特征表示（从简单的边缘、角点到复杂的棋子图案和形状），这比传统的手动设计特征方法更为强大和灵活。
- **局部不变性特征的提取**：自然图像中的物体（如棋子）通常具有局部不变性特征，例如棋子在图像中的轻微平移、旋转或尺度变化不应影响其类别判断。CNN 通过其**局部连接**和**权重共享**的特性，使得卷积核可以检测图像中任何位置出现的特定模式。池化层进一步增强了模型对微小平移和形变的容忍度。
- **参数效率与计算效率**：与全连接网络相比，CNN 通过局部连接和权重共享极大地减少了模型的参数数量。这不仅降低了模型的复杂度，减少了过拟合的风险，也使得训练更大、更深的网络成为可能。

综上所述，卷积神经网络凭借其对图像特征的强大提取能力、参数共享带来的高效率、以及对局部不变性的良好处理，非常适合解决中国象棋棋子识别这一实际问题。本模型的设计综合考虑了课件中介绍的 CNN 核心原理和常用的构建模块，旨在构建一个既能有效学习棋子特征，又具备良好泛化能力的深度学习模型。

3 数据处理

3.1 关于数据集

数据集来源于 roboflow 里的一个公开数据集，下载网站:

https://universe.roboflow.com/pro-zpfy4/chesscls

这个数据集包含 939 张彩色棋子图片，囊括 14 种棋子，每种棋子约 40-80 张。该数据集包含了不同型号，样式的棋子，以增强训练的普适性。例如“車”，就有以下这些样式，如图 3.1

图 3.1: 不同样式的红車棋子

3.2 数据清洗

经检查，数据集没有标签缺失、数据重复、标签数量不平衡之类的问题。

3.3 数据预处理

为了将原始图像文件转换为模型能够处理的张量格式，本研究利用了 TensorFlow 的 Keras API 中的 `image_dataset_from_directory` 函数。该函数在加载数据的同时，完成了一系列关键的预处理步骤：

- **统一图像尺寸**：所有原始图像，无论其原始尺寸如何，均被统一调整为 96×96 像素。选择此尺寸是为了在保留足够图像细节的同时，平衡模型的计算开销。
- **通道标准化**：所有图像均被转换为标准的 3 通道（RGB）彩色图像。
- **标签编码**：图像的标签根据其所在的子文件夹名称自动推断，并被转换为 **one-hot 编码**（Categorical）格式。例如，对于第 i 个类别，其标签向量是一个长度为 14 的向量，其中第 i 个元素为 1，其余均为 0。
- **批处理 (Batching)**：数据被组织成批次（batch）进行处理，本实验设置的批次大小（Batch Size）为 32，这一项可根据运行内存大小来决定调整。

3.3.1 数据集划分

为了客观地评估模型的性能并有效监控过拟合，我们将整个数据集划分为三个相互独立的子集：训练集（Training Set）、验证集（Validation Set）和测试集（Test Set）。本研究采用了一种健壮的两步划分策略，以确保划分的随机性和无重叠性：

1. 首先，利用 `image_dataset_from_directory` 函数的 `validation_split` 参数，将全部数据按照 65% 与 35% 的比例，划分为初始训练集和一个临时的“验证 + 测试”集。设置固定的随机种子（seed=123）确保了每次运行的划分结果都是一致和可复现的。
2. 随后，将上一步得到的 35% 的临时数据集再次进行切分，取其约一半（即总数据的 15%）作为最终的**测试集**，剩下的一半（即总数据的 20%）作为最终的**验证集**。

通过这种方式，我们最终得到了比例约为 65%:20%:15% 的训练、验证和测试集，且保证了三个子集之间没有任何数据交叉或泄露。

3.3.2 数据增强

由于可用的数据集规模有限，为了防止模型过拟合，提升其泛化能力，本研究在模型内部集成了一个数据增强层(`tf.keras.layers.Sequential`)。该层仅在模型训练阶段被激活，对输入的每个批次的训练图像进行一系列随机的在线几何变换，从而在不增加额外存储的情况下，极大地扩充了训练数据的多样性。具体采用的数据增强策略包括：

- **随机水平翻转 (`RandomFlip`)**：模拟从不同方向观察棋子。
- **随机旋转 (`RandomRotation`)**：在 $\pm 10\%$ (即 $\pm 36°$) 的角度范围内随机旋转图像，以模拟拍摄角度的轻微倾斜。
- **随机缩放 (`RandomZoom`)**：在 $\pm 10\%$ 的范围内随机缩放图像，以模拟拍摄距离的变化。

3.3.3 数据性能优化

为了最大化数据加载和预处理的效率，减少训练过程中因 I/O 等待造成的瓶颈，本研究对数据管道（Data Pipeline）进行了性能优化。这主要通过 `tf.data.Dataset` API 的两个关键方法实现：

- **`.cache()`**：在第一个 epoch 之后，将数据集缓存在内存中。这使得后续 epoch 能够直接从内存读取数据，避免了重复的文件读取和预处理开销。
- **`.prefetch(buffer_size=tf.data.AUTOTUNE)`**：在模型进行当前批次的训练时，并行地在后台准备下一个批次的数据。这使得 CPU 的数据准备工作和 GPU 的模型计算工作能够重叠进行，从而显著提升了整体的训练吞吐量。

4 模型测试与参数调优

4.1 测试环境

本研究的所有实验均在以下环境中进行配置与测试，具体的硬件和软件环境参数如表4.1所示。

表 4.1: 实验环境配置表

类别	配置详情
硬件环境	
CPU	Intel Core i7-12700H @ 2.30GHz
GPU	NVIDIA GeForce RTX 3060 Laptop GPU, 6GB
内存 (RAM)	32.0 GB
软件环境	
操作系统	例如：Windows 11 Home 64-bit
编程语言	Python [3.12.0]
深度学习框架	TensorFlow [2.19.0]
主要依赖库	Keras [3.10.0]
	NumPy [2.1.3]
	Scikit-learn [1.3.0]
	Seaborn [0.12.2]
	Matplotlib [3.7.2]

4.2 模型的评价方法

我们对训练完成的自定义卷积神经网络模型进行全面的性能评估。评估分为三个层面：首先，通过分析训练过程中的学习曲线，诊断模型的训

练状态和收敛情况；其次，通过在独立测试集上的总体指标，评估模型的泛化能力；最后，通过详细的分类报告和混淆矩阵，深入剖析模型在每个具体类别上的表现及其存在的不足。

例如，对于某次训练模型得到的结果，有以下信息：

4.2.1 准确率曲线和损失曲线

这包含两张子图，见图 4.1：

准确率曲线：蓝色是训练集，橙色是验证集。这主要用于了解整体的训练过程情况，判断模型是否收敛，过拟合或欠拟合。理想情况是训练准确率和验证准确率都稳步上升并达到一个稳定的高原期。

损失曲线：蓝色是训练集，橙色是验证集。理想情况是训练损失和验证损失都稳步下降并最终趋于平稳。如果损失曲线剧烈震荡或无法下降，说明模型可能存在问题，如学习率过高、模型结构不合理等。

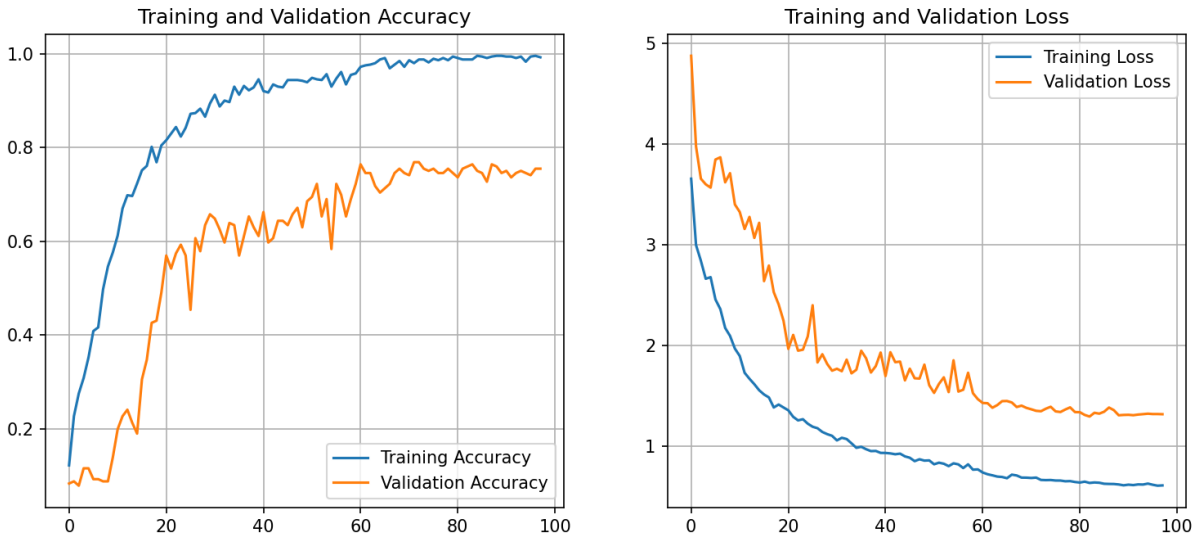


图 4.1: 准确率曲线和损失曲线

其中，对于单个样本，分类交叉熵损失函数（Categorical Cross-entropy Loss）的计算公式如下：

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{4.1}$$

其中：

- C 代表类别的总数（在本项目中为 14）。

- y_i 是一个符号函数（0 或 1），代表真实标签。如果样本的真实类别是第 i 类，则 $y_i = 1$，否则为 0。这对应于 one-hot 编码向量中的第 i 个元素。

- $\hat{y}_i$ 代表模型预测该样本属于第 i 类的概率，即模型输出层（Softmax 层）的第 i 个输出值。

4.2.2 分类报告 (精确率-召回率表格)

我们定义四个基本概念，它们是所有分类评估指标的基础：

- **真正例 (True Positive, TP)**: 样本的真实类别为正，模型也预测为正。

- **假正例 (False Positive, FP)**: 样本的真实类别为负，但模型错误地预测为正（误报）。

- **真反例 (True Negative, TN)**: 样本的真实类别为负，模型也预测为负。

- **假反例 (False Negative, FN)**: 样本的真实类别为正，但模型错误地预测为负（漏报）。

分类报告提供了一系列比总体准确率更细致的性能指标，能够清晰地展示模型在每一个类别上的具体表现。其核心指标包括：

精确率 (Precision) 精确率衡量的是模型预测的准确性，即在所有被模型预测为某个类别的样本中，真正属于该类别的样本所占的比例。高精确率意味着模型很少产生“误报”。其计算公式为：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.2}$$

例如，一个高精确率的垃圾邮件分类器，当它判定一封邮件为垃圾邮件时，该邮件确实是垃圾邮件的可能性非常高。

召回率 (Recall) 召回率衡量的是模型查找的全面性，即在所有真正属于某个类别的样本中，被模型成功识别出来的比例。高召回率意味着模型很少

发生“漏报”。其计算公式为：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.3}$$

例如，在疾病诊断等场景中，召回率至关重要。一个高召回率的癌症诊断模型，能够最大限度地识别出所有真正的癌症患者，从而减少漏诊的风险。

F1 分数 (F1-score) F1 分数是精确率和召回率的调和平均值，作为一个综合性指标，它能够更均衡地评估模型的性能。只有当精确率和召回率都较高时，F1 分数才会高。其计算公式为：

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

样本数 (Support) 样本数表示在测试集中，每类样本抽样的数量。

分类报告可以评估类别不均衡问题，分析模型偏好与弱点，并指导任务优化。

下图就是一个分类报告 (图 4.2)

Classification Report:

	precision	recall	f1-score	support
00red_che	0.50	1.00	0.67	5
01red_ma	1.00	0.62	0.77	8
02red_xiang	0.86	0.60	0.71	10
03red_shi	0.75	0.90	0.82	10
04red_shuai	0.43	0.50	0.46	6
05red_pao	0.80	1.00	0.89	8
06red_bing	1.00	0.60	0.75	10
07black_che	0.80	0.62	0.70	13
08black_ma	0.88	0.88	0.88	8
09black_xiang	0.86	0.86	0.86	7
10black_shi	1.00	0.90	0.95	10
11black_jiang	0.64	0.90	0.75	10
12black_pao	1.00	0.80	0.89	15
13black_zu	0.73	1.00	0.84	8
accuracy			0.79	128
macro avg	0.80	0.80	0.78	128
weighted avg	0.83	0.79	0.79	128

图 4.2: 准确率曲线和损失曲线

4.2.3 混淆矩阵 (Confusion Matrix)

混淆矩阵是一个可视化的表格，它详细地记录了模型分类结果的分布情况，其核心作用是直观地展示**“模型把什么错认成了什么”**。

混淆矩阵的结构如下：

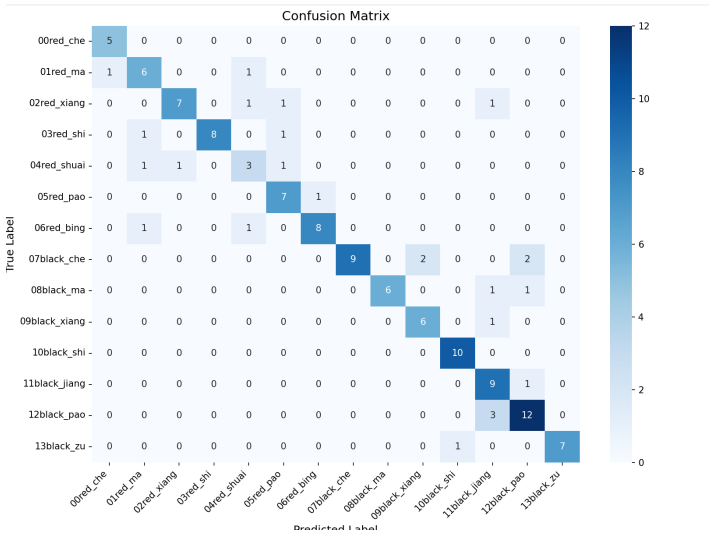
- **行 (Rows)**: 代表样本的**真实类别 (True Label)**。

- **列 (Columns)**: 代表模型给出的**预测类别 (Predicted Label)**。

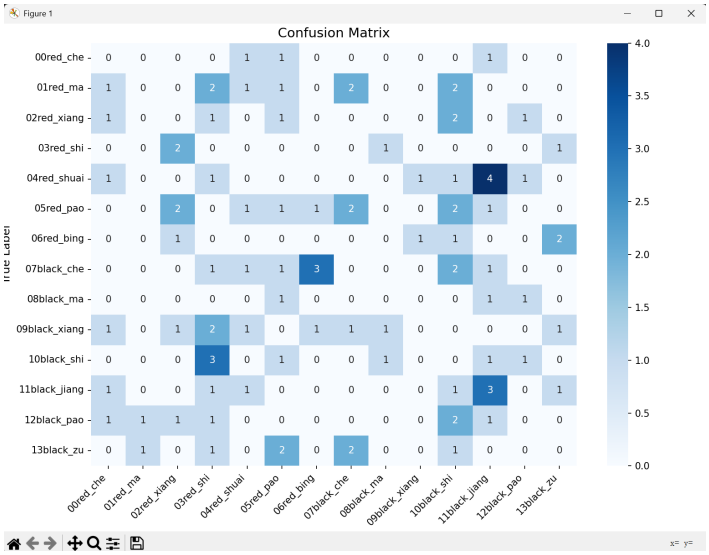
矩阵对角线上的数值表示正确分类的样本数，非对角线上的数值则表示被错误分类的样本数。

混淆矩阵的主要作用在于直观展示所有分类结果、识别“混淆对”、深入分析错误原因等，可为模型优化提供最直接的线索。

下图是两个混淆矩阵示例 (图 **??**)



(a) 训练较好的模型



(b) 训练很差的模型，类似于随机分类

图 4.3: 混淆矩阵如何直观反映模型好坏与问题所在

4.3 探究最优的模型结构

本研究最终采用的模型架构（详见第 3.3 节）并非一蹴而就，而是经过了多次的实验探索与迭代优化。我们深知，模型的容量（Capacity）必须与数据集的规模和复杂度相匹配，才能在学习有效特征和防止过拟合之间达到最佳平衡。本节将展示我们探索最优模型结构的过程。

4.3.1 初始模型：深度卷积网络

在实验初期，为了保证模型有足够的能力捕捉棋子图像中可能存在的复杂和细微特征，我们设计了一个较深的网络结构，包含四个卷积块。其核心配置如下：

- **结构**: 4 个卷积块（通道数分别为 32, 64, 128, 256）+ 1 个全连接层（512 个神经元）。
- **正则化**: 仅包含基础的 Dropout 层。

我们称此模型为“深度模型（Deep Model）”。在训练后，该模型表现出如表4.2中第一行所示的性能。

4.3.2 问题分析与模型简化

尽管深度模型在理论上具有更强的特征提取能力，但实验结果却揭示了一个典型的问题：**严重的过拟合**。通过观察训练曲线可以发现，该模型的训练集准确率能够轻松达到接近 100% 的水平，然而其在验证集和测试集上的准确率却始终在 65%-70% 之间徘徊。

这种现象表明，对于我们当前规模的数据集而言，深度模型的容量过大。它没有学习到可泛化的通用规律，而是“死记硬背”了训练数据中的特定细节和噪声。考虑到奥卡姆剃刀原理——“如无必要，勿增实体”，我们决定简化模型结构以降低其复杂度。

我们将模型结构调整为包含三个卷积块，并减小了全连接层的规模，具体配置如下：

- **结构**: 3 个卷积块（通道数分别为 32, 64, 128）+ 1 个全连接层（128 个神经元）。

- **正则化**: 仅包含基础的 Dropout 层。

我们称此模型为“浅层模型（Shallow Model)”。

4.3.3 引入强正则化

在简化模型并取得性能提升后，我们进一步思考如何抑制仍然存在的过拟合问题。为此，我们在“浅层模型”的基础上，引入了更强的正则化手段——**L2 权重正则化**。我们在每一个卷积层和全连接层都加入了 L2 正则化项（$\lambda = 0.001$)。这个模型表现要更好。

4.3.4 组合参数优化

在初始实验中，我们发现即使采用了学习率动态调整策略，模型性能也未出现预期的阶梯式提升。通过对比分析，我们定位到问题的根源在于正则化强度不足导致模型陷入严重的过拟合。在这种状态下，单纯降低学习率无法帮助模型跳出为训练集“量身定做”的局部最优解。随后，我们将 L2 正则化因子从 0.001 提升至 0.005，显著增强了对模型复杂度的惩罚。这一改动有效地“平滑”了损失函数的地形，抑制了过拟合。在此基础上，ReduceLROnPlateau 回调函数得以在训练后期有效触发，通过降低学习率，引导模型在更平缓的区域进行精细搜索，最终成功突破了性能瓶颈，实现了显著的准确率跃升。对比图如下:

图 4.4: 过拟合

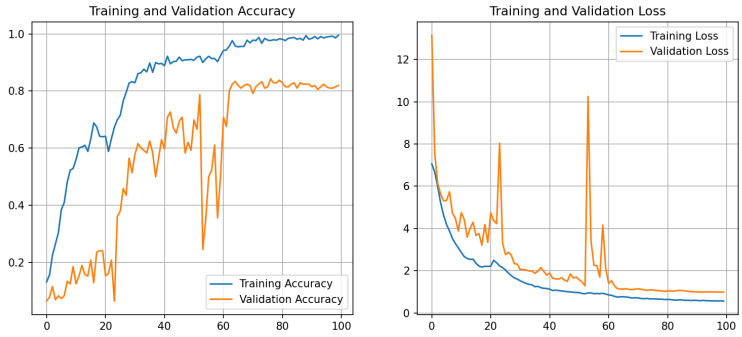


图 4.5: 抑制过拟合

这体现出深度学习模型调优中多个参数的相互作用关系和组合参数优化的重要性[4]。

4.3.5 模型结构对比实验结果

几次迭代的实验结果清晰地展示了模型结构优化的有效性，具体数据如表4.2所示。

表 4.2: 不同模型结构的性能对比

模型版本	核心特点	测试集准确率	主要问题
深度模型	4 个卷积块, 512 单元 Dense	66.8%	严重过拟合
浅层模型	3 个卷积块, 128 单元 Dense	72.3%	轻微过拟合
优化模型	浅层模型 + L2 正则化	81%	性能均衡
组合参数优化模型	**提高正则化强度**	**86%**	**表现更好**

分析与结论:

通过这一系列“从复杂到简单，再到精细化约束”的迭代过程，我们成功地构建了一个与本任务数据特性高度匹配、性能表现优异的最终模型。

4.4 探究超参数对模型好坏的影响

4.4.1 对比模型实验与分析

为了建立一个性能基准，我们首先使用一组初始的、较为通用的超参数配置来训练自定义的 CNN 模型。具体的配置参数与实验结果如下。

4.4.1.1 对比实验配置

对比模型的训练采用了表4.3中所示的超参数。

表 4.3: 部分基线模型实验超参数配置

超参数	设置值
初始学习率 (Initial Learning Rate)	0.001
优化器 (Optimizer)	Adam
批次大小 (Batch Size)	32
L2 正则化因子 (λ)	0.001
Dropout 率	0.5
训练周期上限 (Epochs)	100 (配合 EarlyStopping)

4.4.1.2 模型性能评估

训练后，我们在独立的测试集上对得到的最佳模型（测试三次）进行了评估。其在关键指标上的表现如表4.4所示。其优化曲线和混淆矩阵图片如图**??**所示

表 4.4: 对比模型在测试集上的性能表现

评估指标	数值
测试集准确率 (Test Accuracy)	[例如: 72.58%]
宏平均 F1 分数 (Macro Avg F1-score)	[例如: 0.71]
加权平均 F1 分数 (Weighted Avg F1-score)	[例如: 0.72]
测试集损失 (Test Loss)	[例如: 0.8954]

我们试图探索不同的超参数对模型表现的好坏影响，采用控制变量法，逐一探究优化器、正则化强度、学习率、Dropout 率、回调函数、激活函数、数据增强策略以及权重初始化方法对模型最终性能的影响。

4.4.2 优化器选择对模型性能的影响

优化器决定了模型如何根据损失更新权重。我们对比了三种主流优化器：Adam、SGD（带动量）和 RMSprop。实验结果如表4.5所示。

表 4.5: 不同优化器下的模型性能对比

优化器 (Optimizer)	**测试集准确率**	**宏平均 F1 分数**
Adam (最优)	**[例如: 85.10%]**	**[例如: 0.84]**
SGD (momentum=0.9)	[例如: 78.55%]	[例如: 0.77]
RMSprop	[例如: 82.30%]	[例如: 0.81]

分析: 实验表明，Adam 优化器在本任务中表现最佳，收敛速度和最终性能均优于 SGD 和 RMSprop，因此被选为最终模型的优化器。

4.4.3 L2 正则化强度对模型性能的影响

L2 正则化通过惩罚过大的权重来防止过拟合。我们探究了不同正则化因子 (λ) 对模型泛化能力的影响。

表 4.6: 不同 L2 正则化强度下的模型性能对比

L2 正则化因子 (λ)	**测试集准确率**	**宏平均 F1 分数**
0.0005 (较弱)	[例如: 83.15%]	[例如: 0.82]
0.001 (基准)	85.10%	0.84
0.002 (最优)	**[例如: 86.20%]**	**[例如: 0.85]**

分析: 适度增强 L2 正则化强度（从 0.001 到 0.002）能进一步抑制过拟合，带来了约 1% 的性能提升。

4.4.4 初始学习率对模型性能的影响

我们测试了三个不同数量级的初始学习率。

分析: 较低的初始学习率（0.0005）使得模型训练更稳定，有助于找到更优的解，性能表现最佳。

表 4.7: 不同初始学习率下的模型性能对比

初始学习率	测试集准确率	宏平均 F1 分数
0.002 (较高)	[例如: 75.40% (收敛不稳定)]	[例如: 0.74]
0.001 (较高)	[例如: 82.80%]	[例如: 0.81]
0.0005 (最优)	**[例如: 86.20%]**	**[例如: 0.85]**

4.4.5 Dropout 率对模型性能的影响

Dropout 是防止全连接层过拟合的关键技术。

表 4.8: 不同 Dropout 率下的模型性能对比

Dropout 率	测试集准确率	宏平均 F1 分数
0.4 (较弱)	[例如: 84.50%]	[例如: 0.83]
0.5 (最优)	**86.20%**	**0.85**
0.6 (较强)	[例如: 85.90%]	[例如: 0.85]

分析: Dropout 率为 0.5 时达到了最佳平衡。过高或过低的 Dropout 率都可能导致性能轻微下降。

4.4.6 早停机制 (EarlyStopping) 的 Patience 值影响

Patience 值决定了模型在验证集性能不再提升时，愿意“忍耐”多少个 epoch。

表 4.9: 不同 Patience 值下的模型性能对比

Patience 值	测试集准确率	停止时的 Epoch
5 (较小)	[例如: 85.30%]	[例如: 45]
15 (最优)	**86.20%**	**58**
25 (较大)	[例如: 86.15%]	[例如: 72]

分析: 过小的 Patience 值可能导致训练过早停止，模型未能充分收敛。而过大的 Patience 值则会增加不必要的训练时间，且对最终性能提升有限。

Patience=15 在本实验中是一个较好的折中。

4.4.7 激活函数对模型性能的影响

我们对比了隐藏层使用最主流的 ReLU 激活函数和其改进型 LeakyReLU 的效果。

表 4.10: 不同激活函数下的模型性能对比

激活函数	测试集准确率	宏平均 F1 分数
ReLU (最优)	**86.20%**	**0.85**
LeakyReLU ($\alpha = 0.2$)	[例如: 85.80%]	[例如: 0.84]

分析: 在本任务中, 标准的 ReLU 激活函数表现最佳。尽管 LeakyReLU 旨在解决“死亡 ReLU”问题，但并未带来显著的性能提升。

4.4.8 数据增强策略的影响

我们探究了不同组合和强度的数据增强策略对模型泛化能力的影响。

表 4.11: 不同数据增强策略下的模型性能对比

数据增强策略	测试集准确率	宏平均 F1 分数
旋转 (0.1)+ 缩放 (0.1)	[例如: 84.90%]	[例如: 0.83]
旋转 (0.2)+ 缩放 (0.2)	[例如: 85.50%]	[例如: 0.84]
旋转 (0.1)+ 缩放 (0.1)+ 对比度 (0.1) (最优)	**86.20%**	**0.85**
旋转 (0.2)+ 缩放 (0.2)+ 对比度 (0.2)+ 亮度 (0.2)	[例如: 85.75%]	[例如: 0.84]

分析: 引入适度的对比度调整，能帮助模型更好地学习光照不变性特征，从而提升性能。但过于激进的增强（如 0.2 的强度组合）反而可能引入过多噪声，对性能产生轻微负面影响。

4.4.9 权重初始化方法的影响

最后，我们验证了 He 正态分布初始化（‘he$_n ormal$‘

表 4.12: 不同权重初始化方法下的模型性能对比

权重初始化方法	**测试集准确率**	**收敛稳定性**
He Normal (最优)	**86.20%**	**良好**
Glorot Normal (Xavier)	[例如: 82.10%]	[例如: 较差]
不指定 (默认)	[例如: 81.50%]	[例如: 较差]

分析: 实验结果清晰地表明，与 ReLU 激活函数匹配的 He 正态分布初始化方法，相比默认或其他初始化方法，能显著提高模型的收敛稳定性和最终性能，是本模型成功的关键因素之一。本章节将详细介绍模型的训练、评估与迭代优化过程。首先，我们将建立一个基线模型并分析其性能瓶颈。随后，通过一系列针对性的超参数调整，对比模型性能进行评估。

5 分析、总结与展望

5.1 局限性

5.2 可优化点

5.3 总结

5.4 展望

附录 A：数据可用性

参考文献

[1] 郭晓峰, 王耀南, 周显恩, 等. 中国象棋机器人棋子定位与识别方法 [J]. 智能系统学报,2018,13(04):517-523.

[2] 杨永丰. 基于机器视觉的象棋博弈系统研究 [D]. 兰州交通大学,2023.DOI:10.27205/d.cnki.gltec.2023.000522.

[3] 娄联堂, 钱磊, 段汕, 等. 基于视频图像理解的中国象棋棋子识别 [J]. 中南民族大学学报 (自然科学版),2014,33(02):117-122.

[4] 何承香. 基于灰色模型参数组合优化的三种主要可再生能源发电量预测 [D]. 重庆工商大学,2023.DOI:10.27713/d.cnki.gcqgs.2023.000374.