暨南大学

本科生论文

论文题目：基于卷积神经网络的中国象棋棋子识别

学　　院：信息科学技术学院

专　　业：信息与计算科学

姓　　名：周峻申，项昱又

学　　号：2022100607，2022100608

指导教师：肖亮海

二〇二五年六月十八日

基于卷积神经网络的中国象棋棋子识别

[**摘要**] 本研究旨在解决中国象棋棋子在不同角度、样式和光照条件下的自动识别问题。采用 roboflow 上的一个小型公开数据集。本文采用 CNN 模型。通过迭代实验，探索了与本任务数据特性高度匹配的最优模型架构。实验构建了并行的、I/O 优化数据加载流程，利用 GPU 加速处理，极大地提升了训练效率。相比优化前训练用时缩短 **50%** 以上。此外，本文系统性地开展了超参数敏感性分析，通过控制变量法对比了多个关键因素对模型性能的影响。优化后的模型在训练集上达到了超过 **99%** 的精度，并在独立测试集上达到了 **87.32%** 的分类准确率。实验结果清晰地展示了模型容量、正则化强度和学习策略之间相互作用的重要性，证明了本研究方法的可行性与有效性，为解决类似的细粒度图像分类问题提供了有价值的实践经验。

[**关键词**] 深度学习；卷积神经网络；图像识别；中国象棋；超参数优化

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。国内外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、型号、样式的棋子，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

本研究围绕“基于深度学习的中国象棋棋子识别”这一核心问题展开，旨在构建一个能够应对不同角度、型号和样式的棋子图像，并进行精准分类的卷积神经网络（CNN）模型。为了实现这一目标，本文开展了以下主要工作：

1. **构建并优化了一个自定义的深度卷积神经网络模型**。我们没有直接采用现有的庞大模型，而是参考经典 CNN 思想，结合现代深度学习组件（如 ReLU 激活函数、批量归一化、L2 正则化等），设计了一个与本任务数据特性高度匹配的轻量级 CNN 架构。通过一系列“从复杂到简单，再到精细化约束”的迭代实验，最终确定了最优的模型结构。
2. **实施了系统性的数据处理与增强策略**。针对数据集规模有限的挑战，我

们设计了一套健壮的数据集划分方案以保证评估的可靠性，在模型中集成了在线数据增强层，有效提升了模型的泛化能力和鲁棒性。

3. **开展了超参数敏感性分析与对比实验**。为了找到最优的模型配置，我们采用控制变量法，系统性地探究了多个关键超参数对模型性能的影响，并通过量化对比，为最终模型的构建提供了坚实的实验依据。

1.3 本文的组织结构

为了更好地阐述本研究的工作，本文的组织结构安排如下：

第一章：主要介绍了本研究的背景与意义。

第二章：阐述了本研究所需的核心理论。

第三章：介绍了卷积神经网络的整体架构、各网络层的具体功能。

第四章：描述了实验数据集的来源与概况，数据预处理的完整流程。

第五章：通过一系列对比实验，探究不同超参数对模型性能的影响。

第六章：对整个研究工作进行总结。

2 模型理论基础

本章节将阐述本项目的深度学习核心理论基础。

2.1 人工神经网络 (ANN)

人工神经网络，是一种模拟生物神经系统的计算模型。通过模拟神经元之间的作用来处理信息。

2.1.1 基本概念

神经网络的核心思想是连接主义。其主要特性包括：1. 信息并非存储在单一的单元中，而是分布在多个单元上；2. 学习到的知识体现在神经元间的连接上；3. 通过改变单元之间的连接强度，来实现适应数据，完成任务。

2.1.2 网络基本结构

人工神经网络具有层次化的结构，包括输入层，隐藏层，输出层。信息在网络中的传播方式决定了网络的类型。在前馈神经网络 (FNN) 中，信息单向从输入层传播至输出层。本项目采用的卷积神经网络 (CNN) 属于 FNN 的特殊形式。

2.2 神经元模型

神经元是 ANN 的基本计算单元，模拟生物神经元的信息处理。

2.2.1 人工神经元结构

一个典型的人工神经元接收来自前一层神经元或外部输入的多个信号。其计算过程通常包含以下两个步骤：

1. **加权和**：每个输入信号 x_i 被乘以一个相应的权重 w_i。所有加权输入与一个偏置项 (bias) b 相加，得到神经元的净输入 z：
$$z = \sum_{i=1}^{d}(w_i \cdot x_i) + b = \mathbf{w}^T\mathbf{x} + b \tag{2.1}$$
其中，d 是输入的数量，$\mathbf{w}$ 是权重向量，$\mathbf{x}$ 是输入向量。偏置项 b 允许激活函数在输入为零时仍有非零输出，增加了模型的灵活性。

2. **激活函数**：净输入 z 随后通过一个非线性激活函数 $f(\cdot)$ 进行转换，产生该神经元的激活值 a：
$$a = f(z) = f(\mathbf{w}^T\mathbf{x} + b) \tag{2.2}$$

权重 $\mathbf{w}$ 和偏置 b 是神经元的可学习参数，在网络训练过程中通过优化算法进行调整。

2.3 激活函数

激活函数为神经网络引入了非线性，是学习复杂模式的关键。主要作用包括引入非线性和决定神经元输出。

以下介绍几种与本项目相关的典型激活函数。

2.3.1 Sigmoid (Logistic) 函数

Sigmoid 函数是最早被广泛使用的激活函数之一，其数学表达式为：
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$
特性与缺点：连续可导，便于梯度计算。但可能存在梯度消失问题：当输入值过大或过小时，Sigmoid 函数进入饱和区，其导数趋近于 0，导致在反向传播过程中梯度逐层衰减，使得深层网络难以训练。

2.3.2 Tanh (双曲正切) 函数

Tanh 函数是 Sigmoid 函数的改进版本，其数学表达式为：

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1 \tag{2.4}$$

特性与缺点：输出范围在 (-1, 1) 之间，是零中心化的，通常比 Sigmoid 函数具有更好的收敛性能，且连续可导。但与 Sigmoid 类似，Tanh 函数在输入值较大或较小时也会进入饱和区，存在梯度消失问题。

2.3.3 ReLU 函数

ReLU 函数是目前深度学习中，特别是在卷积神经网络中，应用最为广泛的激活函数之一。其数学表达式为：

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.5}$$

特性与优点：计算高效：只涉及简单的比较和赋值操作。缓解梯度消失：当输入为正时，梯度恒为 1，有利于梯度在网络中更有效地传播，加速训练。

2.3.4 Softmax 函数

Softmax 函数通常用于多分类问题的输出层，它将一个包含 K 个实数值的向量转换为一个 K 维的概率分布，其中每个元素表示对应类别的概率，且所有概率之和为 1。对于输入向量 $\mathbf{z} = (z_1, z_2, \ldots, z_K)$，Softmax 函数的计算方式为：

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \tag{2.6}$$

特性：输出向量的每个元素都在 (0, 1) 范围内。输出向量的所有元素之和为 1。能够突出输入向量中最大的元素所对应的概率。

在中国象棋棋子识别任务中，我们需要将棋子图像分为 14 个不同的类别，因此 Softmax 函数将作为模型输出层的激活函数，输出每个棋子类别的预测概率。

2.4 损失函数

2.4.1 定义与作用

损失函数 $L(y, \hat{y})$ 计算单个样本的预测值 $\hat{y}$（模型输出）与真实值 y 之间的“距离”或“误差”。模型训练的目标是通过调整模型参数 θ，使得在整个训练数据集上的平均损失（即经验风险）最小化。

$$\mathcal{R}_{\text{emp}}(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f(\mathbf{x}^{(i)}; \theta)) \tag{2.7}$$

其中 N 是训练样本的数量，$\mathbf{x}^{(i)}$ 和 $y^{(i)}$ 是第 i 个样本的输入和真实标签，$f(\mathbf{x}^{(i)}; \theta)$ 是模型对第 i 个样本的预测。理论上，我们希望最小化期望风险 $\mathcal{R}(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}[L(y, f(\mathbf{x}; \theta))]$，但由于真实数据分布 p_{data} 未知，实践中通常最小化经验风险。

2.4.2 针对分类问题的常用损失函数

对于中国象棋棋子识别这类多分类任务，交叉熵损失函数是最常用的选择。

2.4.2.1 交叉熵损失

交叉熵用于衡量两个概率分布之间的差异。在分类任务中，这两个分布分别是：

模型的预测概率分布：通常由输出层的 Softmax 函数给出，表示模型预测输入属于各个类别的概率。

真实的标签分布：对于多分类问题，通常采用独热编码表示，即真实类别对应的位置为 1，其他位置为 0。

分类交叉熵 ：当有 K 个类别，真实标签 y_k (第 k 类为 1，其余为 0) 和模型预测概率 p_k (模型预测为第 k 类的概率) 时，单个样本的分类交叉熵损失定义为：

$$L_{\text{CE}} = -\sum_{k=1}^{K} y_k \log(p_k) \tag{2.8}$$

由于 y_k 是 one-hot 编码，上式可以简化为 $L_{\text{CE}} = -\log(p_c)$，其中 c 是真实的类别索引，p_c 是模型预测为真实类别 c 的概率。最小化交叉熵损失等

价于最大化模型预测正确类别的对数似然。在本项目中，由于输出层采用 Softmax 激活函数输出 14 个棋子类别的概率，因此交叉熵损失（具体是分类交叉熵或稀疏分类交叉熵，取决于标签的表示方式）是衡量模型性能和指导优化的理想选择。

2.5 优化算法

本项目中相关的优化算法为自适应学习率算法：这类算法为每个参数独立地调整学习率。

RMSProp：是 AdaGrad 的改进，通过使用指数加权移动平均来累积历史梯度平方，缓解了 AdaGrad 学习率急剧下降的问题。

Adam：是目前最常用的优化算法之一。它结合了动量法和 RMSProp 的思想。Adam 通常能够快速收敛并取得较好的性能。

在本项目中，将优先考虑使用 Adam 优化器进行模型训练。

3 本项目的模型类型

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。

3.1 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。

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）。

3.1.1 模型整体架构图

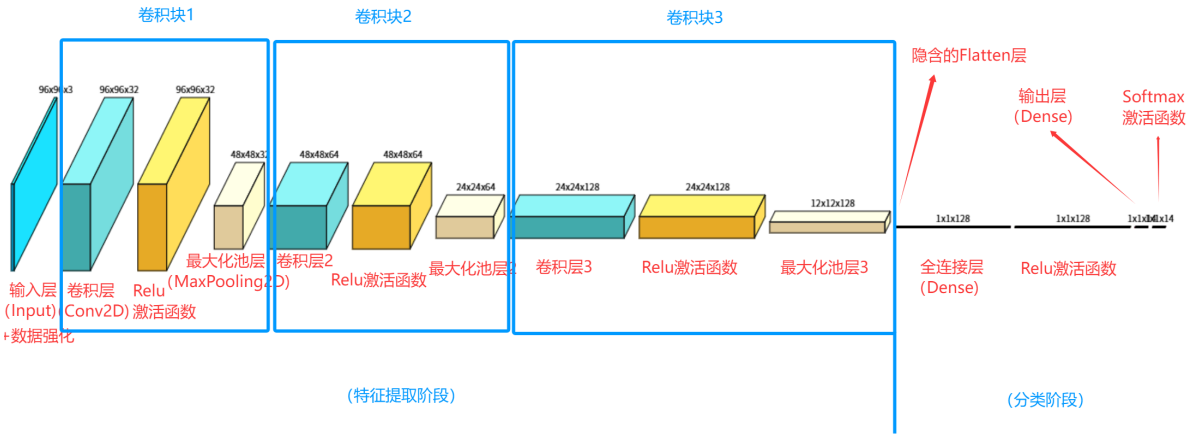


图 3.1: 模型整体架构图

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。

3.1.2 各层具体结构和功能

3.1.2.1 输入与预处理模块

分为输入层，数据增强层，归一化层

表 3.1: 输入与预处理模块结构

层类型	参数	主要功能
`Input Layer`	输入尺寸: $96\times96\times3$	接收 RGB 三通道的棋子图像。
`Data Augmentation`	• 随机水平翻转 • 随机旋转 • 随机缩放	增加数据多样性，提高模型泛化能力。
`Rescaling`	缩放因子: $1/255.0$	将像素值从 $[0,255]$ 归一化到 $[0,1]$。

3.1.2.2 特征提取模块

该模块由三个卷积块堆叠而成，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，深度逐层增加。

卷积块 1：

1. 卷积层 (`Conv2D`):

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。从而学习图像的初级局部特征，如边缘、角点和颜色块。

2. 批归一化层 (`BatchNormalization`): 紧跟在卷积层之后，对卷积层的输出进行归一化，并提供轻微的正则化效果。

3. 最大池化层 (`MaxPooling2D`): 使用 $(2,2)$ 大小的池化窗口，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:

与卷积块 1 结构类似，但卷积核的数量增加到 64 个。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:

与前序模块结构类似，卷积核数量进一步增加到 **128 个**, 学习更高层次的特征。

表 3.2: 特征提取模块结构 (卷积块)

卷积块	层类型	输出形状	参数
卷积块 1	Conv2D	(96, 96, 32)	32 个 (3,3) 卷积核, ReLU
	Batch Norm	(96, 96, 32)	-
	MaxPooling2D	(48, 48, 32)	(2,2) 池化窗口
卷积块 2	Conv2D	(48, 48, 64)	64 个 (3,3) 卷积核, ReLU
	Batch Norm	(48, 48, 64)	-
	MaxPooling2D	(24, 24, 64)	(2,2) 池化窗口
卷积块 3	Conv2D	(24, 24, 128)	128 个 (3,3) 卷积核, ReLU
	Batch Norm	(24, 24, 128)	-
	MaxPooling2D	(12, 12, 128)	(2,2) 池化窗口

3.1.2.3 展平层 (Flatten Layer)

位于特征提取模块之后, 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。

3.1.2.4 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:

1. 全连接层 (Dense): 包含 128 个神经元，并使用 ReLU 激活函数, 整合所有从卷积部分提取的高级特征，并进行非线性映射。

2. 批归一化层 (`BatchNormalization`): 对全连接层的输出进行归一化。

3.Dropout 层 (`Dropout`): 设置丢弃率为 **0.5**。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`) 包含 14 个神经元，每个神经元对应一个棋子类别。

表 3.3: 分类模块结构

模块	层类型	输出形状	参数
展平层	`Flatten`	(18432,)	-
全连接块	`Dense`	(128,)	128 个神经元, ReLU 激活
	`Batch Norm`	(128,)	-
	`Dropout`	(128,)	丢弃率: 0.5
输出层	`Dense (Output)`	(14,)	14 个神经元, Softmax 激活

激活函数 (`Softmax`): 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。

3.2 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[4]。

4 数据处理

4.1 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。

每种标签的数据数量如图 4.1, 不同的棋子样式如图 4.2

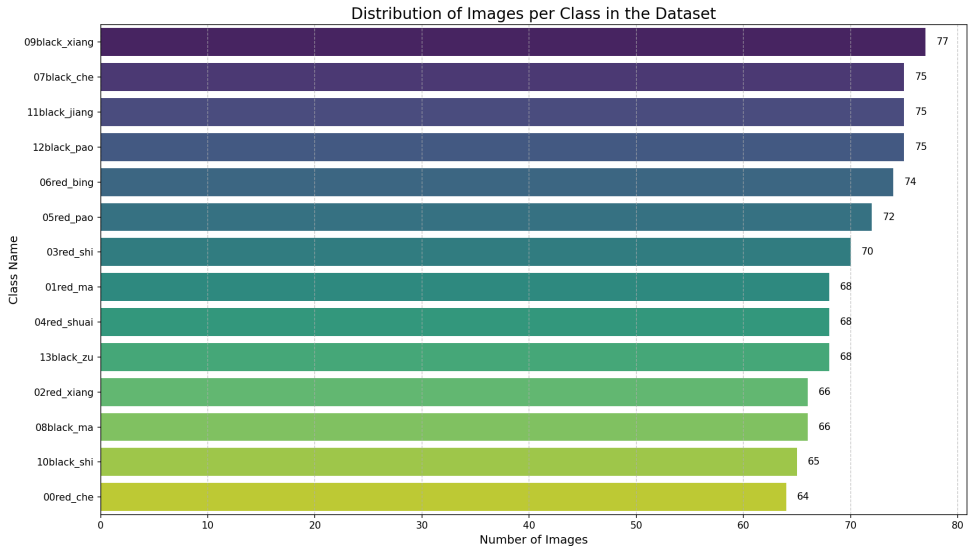


图 4.1: 数据分布

图 4.2: 不同样式的红車棋子

4.2 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。

4.3 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。

4.3.1 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。

4.3.2 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。

数据增强的前后对比图如下：

(a) 原始图片

(b) 数据增强后的图片

图 4.3: 数据增强前后对比

4.3.3 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。

在后续介绍的配置环境下，模型训练的总时间可以控制在 5-6 分钟（300s）左右。

5 模型测试与参数调优

5.1 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5.1所示。

表 5.1: 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]

5.2 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：

5.2.1 准确率曲线和损失曲线

这包含两张子图，见图 5.1：

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。

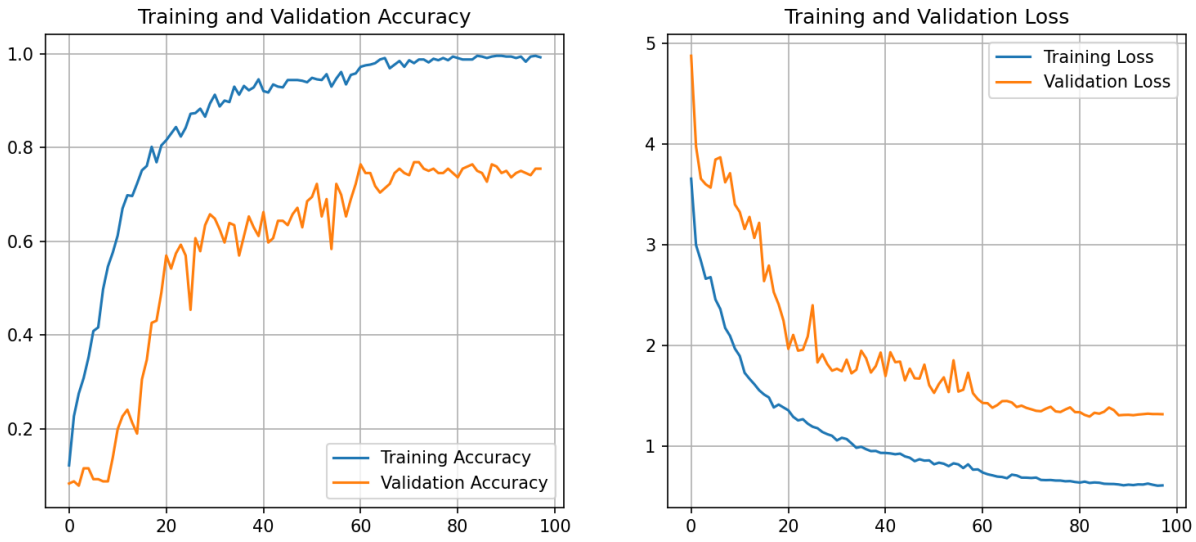


图 5.1: 准确率曲线和损失曲线

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：

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{5.1}$$

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。

5.2.2 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：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2}$$

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：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.3}$$

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：

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

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。如表 5.2

表 5.2: 模型在测试集上的详细分类报告

类别	精确率	召回率	F1 分数	样本数
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
准确率 (Accuracy)			0.79	128
宏平均 (Macro Avg)	0.80	0.80	0.78	128
加权平均 (Weighted Avg)	0.83	0.79	0.79	128

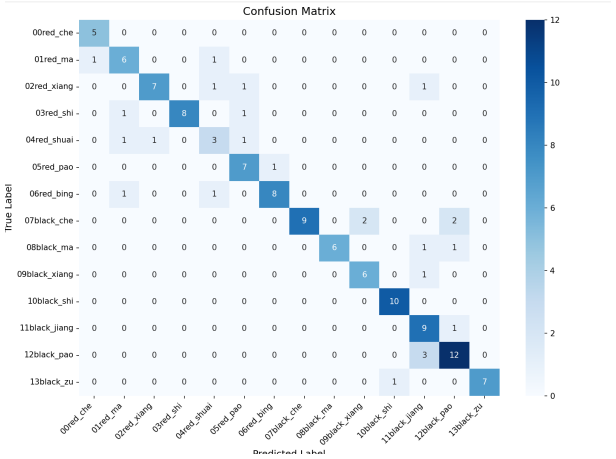
5.2.3 混淆矩阵 (Confusion Matrix)

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”**。

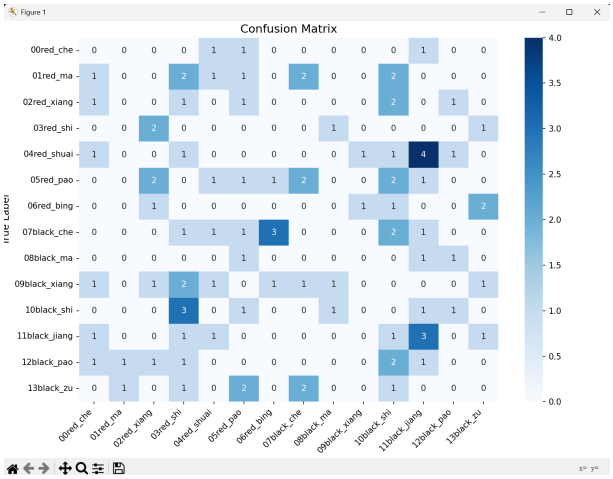
混淆矩阵的行 (Rows) 代表样本的真实类别 (True Label)，列 (Columns) 代表模型给出的预测类别 (Predicted Label)。矩阵对角线上的数值表示正确分类的样本数，非对角线上的数值则表示被错误分类的样本数。

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 5.2)



(a) 训练较好的模型



(b) 训练很差的模型，类似于随机分类

图 5.2: 混淆矩阵如何直观反映模型好坏与问题所在

5.3 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 章）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。

5.3.1 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。其核心配置包含 4 个卷积块（通道数分别为 32, 64, 128, 256）和 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”。在训练后，该模型表现出如表5.3中第一行所示的性能。

5.3.2 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（通道数分别为 32, 64, 128），并减小了全连接层的规模（128 个神经元）

我们称此模型为 “浅层模型”。

5.3.3 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。

5.3.4 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：

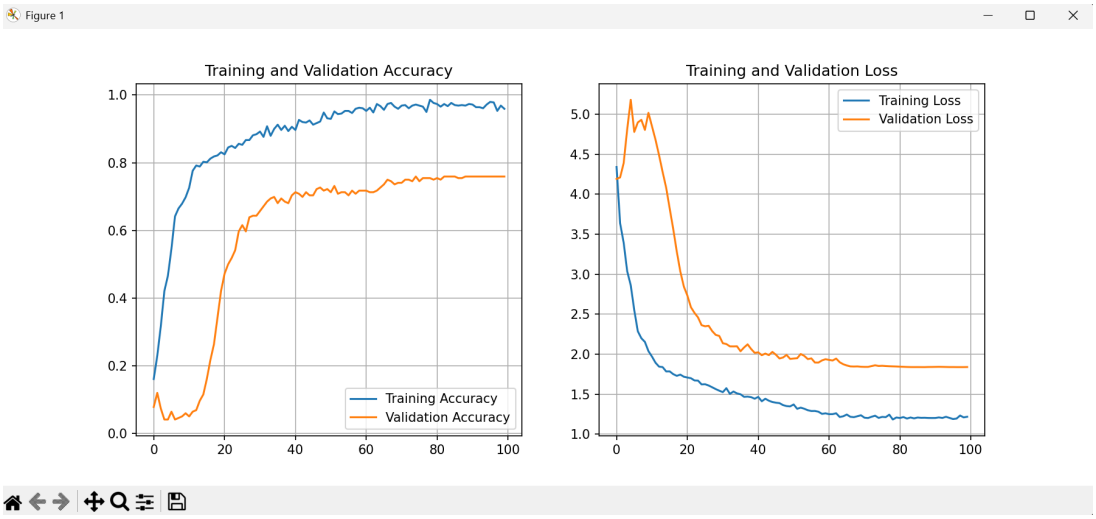


图 5.3: 过拟合

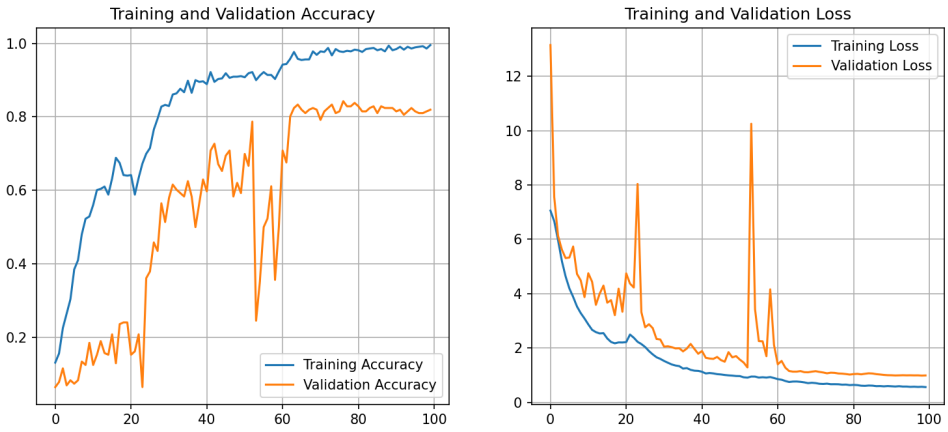


图 5.4: 抑制过拟合

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[6]。

5.3.5 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5.3所示。

表 5.3: 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。

5.4 探究超参数对模型好坏的影响

5.4.1 基线模型实验与分析

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。

5.4.1.1 基线实验配置

基线模型的训练采用了表5.4中所示的超参数。

表 5.4: 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)

5.4.1.2 模型性能评估

训练后，我们在独立的测试集上对测试三次得到的最佳模型进行了评估。其在关键指标上的表现如表5.5所示。

表 5.5: 基线模型在测试集上的性能表现

评估指标	数值
训练集准确率 (Train Accuracy)	99.34%
测试集准确率 (Test Accuracy)	82.58%
宏平均 F1 分数 (Macro Avg F1-score)	0.81
加权平均 F1 分数 (Weighted Avg F1-score)	0.82
测试集损失 (Test Loss)	0.8654

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。

5.4.2 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5.6所示。

表 5.6: 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	**82.33%**	**0.82**
SGD (momentum=0.9)	55.91%	0.54
RMSprop	76.78%	0.76

分析: 实验表明，Adam 优化器在本任务中表现最佳，其测试准确率和 F1 分数远超 SGD 和 RMSprop。SGD 优化器收敛困难，性能最差；RMSprop 表现居中但仍与 Adam 有显著差距。因此，Adam 被选为最终模型的优化器。

5.4.3 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，基准模型采用的是默认的 Adam 优化器（L2 正则化强度为 0.001）。

表 5.7: 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
0.001	84.03%	0.83
0.002	82.47%	0.82
0.005	**87.32%**	**0.87**
0.010	86.16%	0.86

分析: 结果显示，将 L2 正则化强度从 0.001 提升至 0.005 时，模型的性能得到了最显著的提升，准确率提高了约 3.3 个百分点。这表明基准模型的正则化强度不足。然而，当正则化强度继续增加到 0.010 时，性能出现轻微下降，说明正则化过度可能惩罚了部分有效权重。因此，0.005 被确定为最优的正则化强度。

5.4.4 初始学习率对模型性能的影响

测试不同初始学习率对模型的影响。

表 5.8: 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
0.0005	79.34%	0.79
0.001	85.12%	0.85
0.002	85.38%	0.84
0.005	**85.16%**	**0.85**

分析: 在本实验中，学习率为 0.001 时取得了最佳的性能。过低（0.0005）或过高（0.002）的学习率都导致了性能下降。有趣的是，0.005 的学习率表现也相当不错，但考虑到其训练过程可能存在的潜在不稳定性，我们选择 0.001 作为最优学习率。

5.4.5 Dropout 率对模型性能的影响

改变 Dropout 率进行实验。

表 5.9: 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	80.47%	0.81
0.5	**82.72%**	**0.82**
0.6	79.47%	0.79

分析: Dropout 率为 0.5 时达到了最佳平衡。降低 Dropout 率至 0.4 或提高 Dropout 率至 0.6 均导致了性能下降，表明需要恰当的正则化来防止全连接层的过拟合。

5.4.6 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 5.10: 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	72.66%	27
10	79.25%	84
15	**83.72%**	**98**
20	78.12%	100

分析: 过小的 Patience 值（5）导致训练过早停止，模型未能充分收敛，性能最差。随着 Patience 值的增加，模型有更多机会通过学习率衰减等策略找到更优解。Patience=15 在本实验中是一个较好的折中，获得了最佳性能。

5.4.7 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 激活函数和其几个变体的效果。

表 5.11: 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	**82.57%**	**0.83**
Sigmoid	17.62%	0.06
Tanh	42.62%	0.41
ELU	77.25%	0.77

分析: 实验结果戏剧性地证明了 ReLU 的优越性。采用 Sigmoid 或 Tanh 作为隐藏层激活函数，由于梯度消失问题，导致模型几乎无法有效训练，性能极差。ELU 的表现不错，但仍不及标准的 ReLU。这证实了 ReLU 是现代深度卷积网络中的首选。

5.4.8 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 5.12: 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)	**82.39%**	**0.82**
旋转 (0.2)+ 缩放 (0.2)	78.47%	0.78
旋转 (0.1)+ 缩放 (0.1)+ 对比度 + 亮度 (0.1)	75.22%	0.75
旋转 (0.2)+ 缩放 (0.2)+ 对比度 + 亮度 (0.2)	72.09%	0.73

分析: 在本任务中，基础的、幅度较小（0.1）的几何变换（旋转和缩放）取得了最佳效果。增加数据增强的强度（0.2）或引入颜色变换（对比度、亮度）反而导致了性能下降。这可能是因为过于激进的增强引入了过多噪声，或者改变了棋子颜色这一重要特征，从而对模型学习造成了干扰。

5.4.9 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_normal'）的有效性，将其与不指定初始化方法（默认为 Glorot Uniform）进行对比。

表 5.13: 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	宏平均 F1 分数
He Normal	**83.33%**	**0.83**
不指定 (默认)	75.00%	0.74

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，相比默认的 Glorot 初始化，能显著提高模型的最终性能，准确

率提升超过 8 个百分点，是本模型成功的关键因素之一。

6 分析与总结

6.1 局限性

尽管本研究构建的 CNN 模型在象棋棋子识别任务上取得了较为理想的性能，但受限于实验条件和数据集规模，研究工作仍存在以下几点局限性：

- **数据集规模与多样性有限**。本研究采用的数据集虽然经过了增强处理，但其绝对数量和场景多样性仍然有限。模型可能未完全学习到在极端光照、角度、或高度相似背景等复杂情况下的泛化能力。对于字体或图案比较独特的棋子，可能识别不完全准确。
- **模型结构的探索深度**。虽然我们对自定义 CNN 的结构进行了迭代优化，但并未系统性地与更多先进的模型架构进行严格的性能对比，理论上可能存在性能更优的解决方案。
- **超参数优化的局限性**。本研究采用的控制变量法虽然有效分析单个超参数的影响，但对于组合参数之间可能存在的复杂交互作用没有做深入研究。

6.2 未来可优化点

针对上述局限性，未来的研究工作可以从以下几个方面展开，以进一步提升模型的性能和实用性：

- **扩充与构建更丰富的训练数据集**。通过采集更多样化的真实世界图像，或利用生成对抗网络（GAN）等技术生成高度逼真的合成数据，来提升模型在各种复杂环境下的鲁棒性。

- **尝试更多的模型架构**。引入基于迁移学习的预训练模型作为特征提取器，或在当前 CNN 架构中集成注意力机制模块，引导模型聚焦于棋子上的文字等关键区域，可能进一步提升识别精度。
- **探索更系统的超参数优化策略**。采用自动化机器学习（AutoML）工具[7]，进行更全面的超参数搜索，以探索不同参数组合下的最优性能。
- **从图像分类拓展到目标检测**。将当前模型与 YOLO、SSD 等先进的目标检测框架相结合，构建一个能够直接从完整棋盘图像中定位并识别所有棋子的端到端（End-to-End）智能系统[8]，这将极大地提升其实际应用价值。

6.3 总结

本研究旨在解决中国象棋棋子的自动识别问题，通过深度学习技术构建了一个高效的图像分类模型。在整个研究过程中，我们系统性地完成了从数据处理、模型设计、迭代优化到性能评估的全流程工作。

我们首先构建了一个自定义的卷积神经网络（CNN）架构，并针对其在初始实验中暴露出的过拟合问题，通过一系列科学的对比实验，成功地探索并确定了更优的模型结构、正则化策略和关键超参数组合。最终优化后的模型在一个独立的测试集上达到了 **87.32%** 的分类准确率，证明了本研究方法的可行性和有效性。实验结果清晰地展示了模型容量、正则化强度和学习策略之间相互作用的重要性，为解决类似的细粒度图像分类问题提供了有价值的实践经验。

尽管研究存在一定的局限性，但本项目成功地完成了一项有意义的探索工作，为实现更复杂的智能象棋系统奠定了坚实的视觉感知基础，也体现了现代 AI 技术在赋能和传承传统文化方面的巨大潜力。

附录 A：数据可用性

代码文件 `Chinese chess cls` 可从 https://github.com/Rymisz/Maching-Learning-Chinese-chess-cls- 下载

原始数据集可从 https://universe.roboflow.com/pro-zpfy4/chesscls/dataset/1 下载

注意：如果要直接运行代码，下载数据集后需重新整理分类图片并修改文件路径

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] 程廉升, 王立萍, 刘倩, 等.ResNet-CNN 模型在滚动轴承故障诊断中的应用 [J]. 设备管理与维修,2024,(15):54-59.DOI:10.16621/j.cnki.issn1001-0599.2024.08.17.

[5] Li, C. and Chen, G. (2020). Research on Chinese Chess Detection and Recognition Based on Convolutional Neural Network. In: Jain, V., Patnaik, S., Popențiu Vlădicescu, F., & Sethi, I. (eds), *Recent Trends in Intelligent Computing, Communication and Devices*. Advances in Intelligent Systems and Computing, vol 1006. Springer, Singapore.

[6] 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.

[7] 汪文蝶. 基于自动机器学习的 QoE 预测模型研究 [J]. 无线互联科技,2024,21(14):85-91.

[8] 曹国强, 张京龙, 尹航. 基于凸包特征象棋棋子识别系统设计 [J]. 黑龙江大学自然科学学报,2017,34(06):732-736.DOI:10.13482/j.issn1001-7011.2017.09.023.