



本科毕业设计（论文）

基于计算机视觉及强化学习的非嵌入式游戏 AI 设计

学院（部、中心）：数学与统计学院

专业：应用数学系

班级：强基数学 002

学生姓名：吴天阳

学号：2204210460

指导教师：康艳梅，兰旭光

2024 年 5 月

摘要

本文提出了一种基于非嵌入式离线强化学习的训练策略，通过图像信息输入在即时策略游戏皇室战争（Clash Royale）中实现自主博弈对局。本文结合当前目标识别与光学文本识别的前沿算法对图像信息进行特征提取，并使用离线强化学习算法进行训练，实现了移动设备上的实时图像获取、感知融合、智能体决策及设备控制，与对手进行实时对局，并能够战胜游戏中的内置 AI。

首先，本文设计了一种高效制作切片数据集的方法，在此基础上制作了包含 4654 张切片的数据集，涵盖了游戏中的全部部队、法术、防御塔等总计 150 个类别。本文提出了一种生成目标识别数据集的算法，能够模拟真实对局场景生成带标签图像，用于目标识别模型训练。本文从视频流中截取 6939 张图像，制作了包含 116878 个目标框的目标识别数据集作为验证集^①。通过生成式数据集训练出的目标识别模型在验证集中表现出良好的泛化性，在 50% 交并比阈值下达到了 80.9% 的召回率。

其次，基于目标识别、光学文本识别及图像分类模型的输出特征，本文设计了感知融合算法，能够将感知特征预处理为模型的输入特征。该算法基于视频数据中的上下帧信息及单位的关联信息恢复漏识别目标，排除干扰性特征，优化模型输入特征。

最后，在决策模型方面，本文制作了包含 105 个回合、总共 113981 帧的专家数据集^②，在智能体不与环境交互的条件下进行训练。本文从架构及预测目标两个方面对离线强化学习模型进行了改进，将难以学习的离散动作序列转化为连续动作序列，相比传统模型性能分别提高了 24% 和 37%。

本文的全部代码均已开源^③。

关键词：目标识别；光学文本识别；强化学习

^①图像数据集：<https://github.com/wty-yy/Clash-Royale-Detection-Dataset>

^②专家数据集：<https://github.com/wty-yy/Clash-Royale-Replay-Dataset>

^③全部代码：<https://github.com/wty-yy/katacr>

Abstract

This paper proposes a training strategy based on non-embedded offline reinforcement learning to achieve autonomous gameplay in the real-time strategy game Clash Royale through image information input. The paper integrates cutting-edge algorithms for object recognition and optical character recognition to extract features from image information and uses offline reinforcement learning algorithms for training. This enables real-time image acquisition, perception fusion, agent decision-making, and device control on mobile devices, allowing real-time matches against opponents and defeating the built-in AI in the game.

Firstly, the paper designs an efficient method for creating a sliced dataset and produces a dataset containing 4654 slices, covering all troops, spells, and towers in the game, totaling 150 categories. The paper proposes an algorithm to generate an object recognition dataset that can simulate real-game scenarios to generate labeled images for training object recognition models. The paper extracts 6939 images from video streams, creating an object recognition validation dataset containing 116878 bounding boxes^④. The object recognition model trained with the generative dataset shows good generalization in the validation set, achieving an 80.9% recall rate at a 50% Intersection over Union (IoU) threshold.

Secondly, based on the output features of object recognition, optical character recognition, and image classification models, the paper designs a perception fusion algorithm that preprocesses perception features into model input features. The algorithm recovers missed targets based on inter-frame information and unit association information in the video data, eliminating interfering features and optimizing the model input features.

Finally, in terms of the decision-making model, the paper creates an expert dataset containing 105 rounds and a total of 113981 frames^⑤, training the agent without interacting with the environment. The paper improves traditional offline reinforcement learning models from both architecture and prediction objectives, transforming hard-to-learn discrete action sequences into continuous action sequences, improving performance by 24% and 37% compared to traditional models, respectively.

All code in this paper has been open-sourced^⑥.

Keywords: Object recognition; Optical character recognition; Reinforcement learning

^④Image dataset: <https://github.com/wty-yy/Clash-Royale-Detection-Dataset>

^⑤Expert dataset: <https://github.com/wty-yy/Clash-Royale-Replay-Dataset>

^⑥All code: <https://github.com/wty-yy/katacr>

目 录

摘 要.....	I
Abstract.....	II
主要符号表.....	V
1 绪论.....	1
1.1 嵌入式智能体研究现状.....	1
1.2 计算机视觉研究现状	2
1.3 本毕设的任务	3
1.4 本毕设的贡献	4
2 目标识别模型设计及数据集搭建.....	6
2.1 切片数据集制作流程	7
2.2 目标识别模型设计	7
2.3 生成式目标识别数据集.....	9
2.4 本章小结.....	12
3 感知融合与模型决策.....	13
3.1 感知模型.....	13
3.1.1 目标识别模型	13
3.1.2 光学字符识别模型.....	13
3.1.3 图像分类模型	14
3.2 感知特征提取	14
3.2.1 状态特征提取	14
3.2.2 动作特征提取	16
3.2.3 奖励特征提取	16
3.3 决策模型.....	17
3.3.1 离线强化学习	17
3.3.2 状态空间与动作空间	20
3.3.3 预测目标设计与重采样	20
3.3.4 模型架构设计	21
3.4 本章小结.....	22

4 数据分析及实验结果	23
4.1 生成式数据集分析	23
4.2 目标识别模型	23
4.3 决策模型	24
5 总结与展望	27
致 谢	29
参考文献	30
附录 A 外文文献原文	33
附录 B 外文文献译文	43
附录 C 相关内容补充	52
C.1 硬件交互	52
C.2 动态采样概率分布	52
C.3 数据增强	52
C.4 模型测试卡组	53
附录 D 论文相关代码	54
D.1 生成式数据集	54
D.2 特征融合	55
D.2.1 状态特征提取部分代码	56
D.2.2 动作特征提取部分代码	56
D.2.3 奖励特征提取部分代码	57

主要符号表

\mathbb{R}	实数集
\mathbb{N}	自然数集
\mathbb{Z}	整数集
\mathbb{Z}^+	正整数集
\mathbb{R}^N	N 维空间, 由 N 个实数集 \mathbb{R} 所构成的笛卡尔积
$\mathbb{I}_{\text{条件}}$	当“条件”成立时为 1, 否则为 0
A_{ij}	矩阵 A 的第 i 行第 j 列元素
\mathbf{x}_i	向量 \mathbf{x} 中第 i 个元素
$\mathbf{x}_{i:j}$	向量 \mathbf{x} 中第 i 到 j 个元素所构成的向量
$\text{softmax}(\mathbf{z})$	$\left\{ \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}} \right\}_{i=1}^C$ 其中向量 $\mathbf{z} \in \mathbb{R}^C$, C 为总类别数
$A \odot B$	$(A \odot B)_{ij} = (A)_{ij}(B)_{ij}$ 其中矩阵 A 与矩阵 B 具有相同形状

1 绪论

1.1 嵌入式智能体研究现状

当前强化学习（Reinforcement Learning, RL）已经成为人工领域备受关注的重点之一，它通过智能体与环境交互利用得到的奖励大小来指导策略的改进，通过不断地试错来获得最优策略。

强化学习在游戏领域中有着广泛的应用：围棋作为经典博弈游戏，因其具有巨大的状态空间以及复杂的策略而无法使用传统搜索算法解决，DeepMind 团队通过 AlphaGo^[1]给出了基于价值函数估计和蒙特卡洛树搜索的解决方法，首次打败人类顶级选手，标志着强化学习在处理复杂游戏中的巨大成功；AlphaZero^[2]也使用类似模型架构，但它完全基于自我博弈学习，在摒弃任何人类先验知识的前提下超越了 AlphaGo，它展现了自博弈学习的强大能力。

多人在线战术竞技游戏中强化学习也有着非凡的成就，这类游戏往往有着巨大的动作、状态空间，并带有高动态的前后文信息，需要智能体之间的团队配合能力。OpenAI 公司通过 OpenAI Five^[3]给出了基于强化学习的解决方法，该深度网络模型为长短期记忆循环神经网络（Long Short-Term Memory, LSTM）^[4] 和卷积神经网络（Convolutional Neural Network, CNN）组合，提出并利用了强化学习中经典在线同策略近端策略优化算法（Proximal Policy Optimization, PPO）算法^[5]，使用分布式训练方法使其首次在 Dota2 游戏中战胜顶级职业选手。类似的还有 DeepMind 团队的 AlphaStar^[6] 在即时战略游戏《星际争霸》中解决了复杂的多智能体决策问题。

值得注意的是上述模型使用的算法均为在线强化学习算法即经典强化学习研究内容，经典算法例如 PPO^[5]、深度 Q 函数学习（Deep Q Network, DQN）^[7]、软演员评论家算法（Soft Actor-Critic, SAC）等，这是因为嵌入式智能体与游戏环境之间本身具有低成本高交互次数的特性，使得智能体有无限的试错空间，可以不断对新状态空间进行探索，更好地对动作状态价值函数 $Q(s, a)$ 进行估计，并进一步利用 $Q(s, a)$ 对自身策略 $\pi(a|s)$ 进行优化，从而通过迭代方法接近最优策略 $\pi^*(a|s)$ 。

由于嵌入式智能体对环境信息的读取和人类完全不同，人类理解视频游戏的主要方式是通过图像信息，而嵌入式智能体可以直接获取精确的游戏内部信息，虽然这些人类也可以通过图像信息间接获得，但可视化的图像信息中往往伴有噪声，在信息输入方面就会产生误差，因此就导致了游戏竞技上的不公平性产生（其余的不公平因素，例如感知频率与操作频率，对智能加入相应限制可以解决）。

由于当前计算机视觉技术（Computer Vision, CV）发展迅速，本论文提出一种非嵌入智能体的特征提取方法，这种非嵌入式智能体可以仅通过获取和人类完全相同的图像信息进行决策，由于非嵌入式智能体只能通过模拟器和环境进行交互，无法高效获取数据，所以本文只能收集专家数据集，采用 3.3.1 中所介绍的离线强化学习算法 Decision

Transformer (DT)^[8] 和 StARformer^[9] 来完成决策部分。

1.2 计算机视觉研究现状

非嵌入式智能体主要利用到计算机视觉中图像分类 (Image Classification)、目标识别 (Object Detection) 和光学字符识别 (Optical Character Recognition, OCR) 三个方面，下面分别对其进行简单介绍：

图像分类 在图像分类任务上 LeNet-5^[10] 给出了早期卷积神经网络 (Convolutional Neural Network, CNN) 模型，它作为现在 CNN 架构的先驱之一，在 MNIST 手写数据集上首次展现出深度神经网络的卓越性能。近年来由于自动微分计算速度的飞速提升以及 ImageNet2012 数据集^[11] 的支撑，从而使得基于数据驱动的模型重新被重视，自从 AlexNet^[12] 利用在 2012 年 ImageNet 挑战赛中取得突破性成果后，各类新型网络架构例如 VGG^[13]、GoogleNet^[14]、ResNet^[15]、DenseNet^[16]、ViT^[17]、RepVGG^[18]、MetaFormer^[19] 等相继被提出，这些模型不断刷新图像分类的准确率记录，同时提升了模型的计算能力和泛化能力。

研究者针对网络的宽深度 (VGG)、宽度 (GoogleNet)、残差连接 (ResNet)、分组卷积 (AlexNet)、注意力机制 (ViT, MetaFormer)、模型重新参数化 (RepVGG) 等方面进行了深入探索，其中残差网络的引入解决了深度网络的训练难题，可以将网络深度上升到数百层之上，乃至 Transformer^[20] 类架构中也使用了残差机制；而注意力机制的引入可以帮助模型关注图像的关键特征。

目标识别 目标识别任务 (Object Detection) 旨在从图像或视频数据中识别出特征类别的目标或对象，根据处理流程的不同检测头 (Detection Head) 分为单阶段检测器 (Single-Stage Detectors) 和多阶段检测器 (Two-Stage Detectors)，其中单阶段检测器，通常为端到端问题，直接通过输入的图像给出识别框的位置 (回归问题) 与类别 (分类问题)，当前经典模型包括 YOLO (You Only Look Once) v1 ~ 9 系列^[21]、SSD (Single Shot Multibox Detector)^[22]、DETR (DEtection TRansformer)^[23] 等。这类算法优点在于推理速度较快，适用于实时监测任务，该系列在初次提出时检测率较低，但通过不断的优化改进，已经可以成功的解决小目标检测和重叠目标问题。而多阶段检测器例如 R-CNN^[24] 系列，这类算法虽精度较高，但推理速度较长，不适用于实时监测任务。

光学字符识别 光学字符识别 (OCR, Optical Character Recognition) 目前最常用的模型是卷积循环神经网络 (Convolutional Recurrent Neural Networks, CRNN)^[25]，其将经典的连接时序分类损失函数 (CTC, Connectionist Temporal Classification)^[26] 与深度神经网络架构 LSTM 和 CNN 相结合，首先利用 CNN 将图像信息编码为一维序列信息，再用 (Bi-)LSTM^[27] 进行文本预测，最后利用 CTC 损失函数使用对预测的文本顺序进行纠正。但是仅有文字识别模型并不足够，还需要对图像中的文本位置进行定位再使用

CRNN 算法进行识别，于是发展出了很多 OCR 系统，如百度公司的 PP-OCRv1 ~ 4 系列^[28]。

为了理解每个模型的搭建与训练细节，本文对上述介绍的部分模型进行了复现^①。

1.3 本毕设的任务

《皇室战争》(Clash Royale, CR) 是一款卡牌类即时策略游戏 (RTS)，本文仅考虑一对一对战模式，双方玩家需要通过实时决策使用手牌来战胜对手，首先对游戏基本规则及非嵌入式感知任务目标进行介绍：

感知场景 在对局过程中，游戏场景如右图 1-1 所示，本文所需感知内容主要分为如下四个主要部分和三个子部分：

主要部分：

- 竞技场 (Arena): 包括防御塔及双方所部署的部队和建筑物。
- 手牌：位于竞技场下方，包括当前卡片图像及其所需的圣水 (Elixir)。
- 游戏时间：当前阶段的剩余时间。
- 总圣水：用于卡牌的使用，随时间自动恢复。

子部分：

- 主塔：处于三个防御塔中间，当其被摧毁时，对局直接结束。
- 副塔：处于左右两侧的防御塔。
- 部队：玩家通过卡牌部署的战斗单位，包括可移动地面和空中单位，以及建筑单位。

游戏目标 每位玩家的目标是摧毁尽可能多的敌方防御塔，优先摧毁敌方主塔的玩家立刻获胜。游戏中在两种阶段下存在不同的胜利条件：

^①<https://github.com/wty-yy/katacv>，复现内容包括：LeNet-5、AlexNet、VGG16、GoogleNet、ResNet50、YOLOv1、YOLOv3、YOLOv4、YOLOv5、CTCLoss & CRNN、miniGPT



图 1-1 右上角为当前阶段的剩余时间；中间为竞技场部分，双方可在其上部署部队及建筑；下方显示了可用手牌以及部署所需的圣水，最下方显示了当前的可用圣水总量。

1. 常规时间：持续三分钟，目标为摧毁更多的敌方防御塔，若时间结束时防御塔数量一致，则进入加时赛，否则防御塔多的一方获胜。
2. 加时赛：持续两分钟，在该阶段中，首个摧毁敌方剩余防御塔的玩家获胜，若加时赛结束时仍未分出胜负，则比较双方防御塔的最低生命值，最低生命值较高者获胜，若仍未分出胜负，则为平局。

游戏过程 一次对局中每个玩家牌库大小为固定的 8 张，对局开始时，随机在队列中初始化卡牌的出现顺序，每次从队首取出 4 张手牌，当玩家使用卡牌后，使用完的卡牌将被重新加入队尾，所以当一方使用完 8 张不同卡牌时，可以通过逻辑计算出未出现卡牌类别。玩家需要基于当前竞技场上的实时状态、手牌类别及总圣水信息，实时决策使用卡牌的类别和位置，采取进攻或防守策略，最终按照上述给出的规则自动判断胜负。

1.4 本毕设的贡献

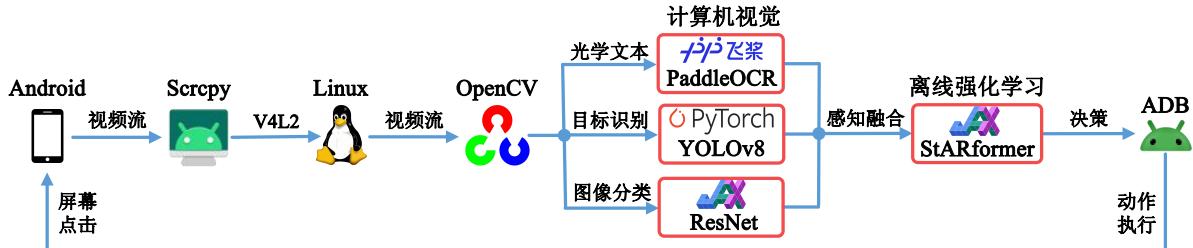


图 1-2 信息流传输流程图

在本文中，首次提出一种基于非嵌入式的离线强化学习设计方案，应用于游戏《皇室战争》，并最终在移动端设备上进行实时决策验证其可行性。图 1-2 中展示了信息流的传输流程，其中感知与决策模型的设计方案如下：

1. 生成式数据集：本文设计了一个与本任务相关的生成式数据集，使其可以模仿真实场景生成含有任意数量部队及种类的带有识别标签的图像，能够非常高效地生成用于目标识别的训练数据集。
2. 感知融合：本文利用 YOLOv8^[29] 并结合目标追踪技术 ByteTrack^[30] 实现了对视频流中部队位置及类别的实时识别，通过 PaddleOCR^[28] 完成了对数字及时间信息的识别，利用 ResNet^[15] 完成对卡牌和圣水图标的分类，结合上述几种感知模型，本文还提出了一系列针对本任务的上下帧信息过滤方法用于优化感知特征。
3. 模型决策：本文使用离线强化学习算法 StARformer 作为决策模型，设计相应的特征输入结构以及损失函数，并将难以学习的离散动作序列转化为连续动作序列进行训练，此方法大幅提高模型性能，最后部署在手机上完成实时对局决策。

离线强化学习模型使用手工录制的 11 万帧专家数据集进行训练，能够战胜游戏内置 AI，虽然无法达到 100% 胜率，但本研究发现该离线强化学习模型能够展现出一定的模仿及泛化能力，从一定程度上说明了这种非嵌入式方法的可行性。

图 1-3 中展示了电脑对手机的实时控制效果。图 1-3 (b) 中具体内容如下：左一为

手机视频流；左二为感知识别结果，左上角为当前实时奖励与总帧数，右上角为当前回合经过的时间，左下角为当前总圣水与手牌，竞技场中目标识别的边界框名称中第一项为类别名称与从属派别，第二项为目标跟踪编号；右二为感知融合特征，其中网格为 32×18 与竞技场中的格子相对应，每个带颜色的方形表示一个部队（数字表示从属派别），圆形表示观测到的动作执行（左上数字为卡牌编号，右下角为动作滞后的帧数）；右一为StARformer-3L决策模型的实时预测的动作分布，其中上方的网格分布表示动作的二维坐标预测，左下角数字表示当前预测动作延迟帧数，右下角四个方格对应四张手牌被选的概率大小。



(a) 现实场景：左侧为被控制的手机，右侧为电脑



(b) 电脑中感知融合与决策：左一为手机视频流，左二为感知识别结果，右二为感知融合特征，右一为模型决策预测动作分布。

图 1-3 实时控制效果

2 目标识别模型设计及数据集搭建

本毕设的首要任务是完成图 1-1 中竞技场部分的目标识别问题，因为决策所需的状态信息主要来自于该部分，对于竞技场中第 i 个单位 u_i ，其所包含的信息可以表示为 $u = (\mathbf{x}, \text{cls}, \text{bel}, \text{bar}_1, \text{bar}_2)$ ，其中 \mathbf{x} 表示 u 所在的网格坐标， cls 表示 u 所属的部队类别， bel 表示 u 所属的派别， $\text{bar}_1, \text{bar}_2$ 表示 u 当前的生命值图像及其余条状图像信息。

在本章节将介绍如何使用目标识别模型完成对每个单位中 $\mathbf{x}, \text{cls}, \text{bel}$ 信息的识别，由于训练目标识别模型需要基于大量的有标记图像，而该任务没有任何相关的开源数据集，若逐帧人工标记效率极低且成本高昂，因此本毕设基于该任务提出一种高效的带标记图像的生成方案，并重构目标识别代码使其可以有效训练该生成式数据集以满足上述识别要求，相关实验结果在 4.1 中展示。

识别数据集制作以及模型更新流程如图 2-1 所示，其中左上部分的“原视频流”为一个回合的视频数据，若存在之前训练的目标识别模型，则使用该模型进行对视频流进行辅助标记，获得“辅助标记视频流”再进行手工标记，否则直接对“原视频流”进行手工标记；在手工标记中，以 0.5 秒作为标记间隔，进行人工目标框标记；然后将目标框和原图像同时传入到 SAM 模型当中获取前景分割，将分割后的结果进行人工筛选获得“切片数据集”；基于已完成的切片数据集，利用生成式数据集算法，对目标识别模型进行迭代更新，从而用于下一次的辅助标记过程。



图 2-1 目标识别数据集制作流程

2.1 切片数据集制作流程

切片数据集中制作本毕设使用了一个强大的通用计算机视觉模型：Segment Anything Model (SAM)^[31]，该模型为 Meta AI 公司于 2023 年开源，可以根据输入的提示信息生成对象的分割掩码，也可对图像中的所有对象生成掩码，其是在一个包含 1100 万张图像和 11 亿个掩码的数据集上进行训练得到的，能够在很多零样本任务上表现出强大的性能，SAM 模型的核型框架包含如下三个部分：

1. 视觉编码器 (Vision Encoder)：基于 ViT 的图像特征提取架构，用于捕获图像的上下帧信息，将输入图像转化为特征编码。
2. 提示编码器 (Prompt Encoder)：用于处理输入中的提示信息，提示信息分为离散（点、框选区域、文本信息）和连续（掩码）两种，分别使用位置嵌入 (Positional Encoding) 和 CNN 对提示信息进行编码。
3. 解码器 (Decoder)：将视觉编码器和提示编码器生成的特征进行融合，并输出提示信息所对应的分割掩码。

由于生成式数据集需要获取每一个单位不同角度的切片图像，人工创建分割掩码过于繁琐，但单位的目标框相对容易标注，所以考虑使用 SAM 中框选区域的提示方式生成对象掩码，图 2-1 的中上以及右上部分展示了使用 SAM 制作切片数据集的过程。

2.2 目标识别模型设计

由于需要追求高效的识别速度，所以本文使用了一阶段识别器 YOLOv8^[29]，下面对 YOLO 系列模型架构进行分析，并给出本文对其进行修改的部分。

设 x, y, w, h 为正实数，四元组 $B = (x, y, w, h)$ 称为边界框，其中 (x, y) 表示当前边界框中心， (w, h) 表示边界框的宽度和高度。

对于任意两个边界框 B_1, B_2 , S_1, S_2 分别表示其所围住的点集，则称 B_1, B_2 的交并比 (Intersection over Union, IOU) 为 $\text{IOU}_{B_2}^{B_1} := \frac{S_1 \cap S_2}{S_1 \cup S_2}$

设 $S, B, C \in \mathbb{Z}^+$ 分别表示图像网格化大小、每个网格中边界框预测框数目、总分类类别数，通过 CNN 可以将图像空间进行压缩，从而得到对应的不同网格化大小。例如，在如图 2-2 中原图像宽高为 416×416 ，CNN 中每次降采样会将图像的宽高均减小一半，于是经过步长 (Stride) 为 $s = 2^3$ 的降采样就可以得到左图中 $(H/s) \times (W/s) =: S \times S$ 即 52×52 的网格大小，每个网格中的感受野大小 (即步长) 即为对应网格中 $2^3 \times 2^3$ 个像素，目标识别模型需要在每个网格处做出 B 个边界框框预测。

图 2-2 中，对于每个目标框 B ，其中心点用黄色点标出，假设该中心点位于网格大小为 52×52 中的第 (i, j) 网格内，则在模型预测的输出中，也应该由 (i, j) 网格对应的边界框进行预测。在 YOLOv3^[32] 中，将 Fast-RCNN^[24] 中锚框的概念引入 YOLO 系列，对于不同的网格划分，对应分配不同的尺度估计框，称为锚框 (Anchor Bounding Box)，这些锚框是基于数据集中的全体锚框做 K 近邻得到 (距离衡量标准为负的相对

交并比），该方法可以将数据集中识别框大小的先验信息引入到模型中。按照网格划分的从大到小，分别分配 B 个从小到大的锚框，因为更大的网格对应更高的分辨率，其具有更多的细节信息，从而可以识别小目标框，所以给其分配更小的锚框，反之亦然。

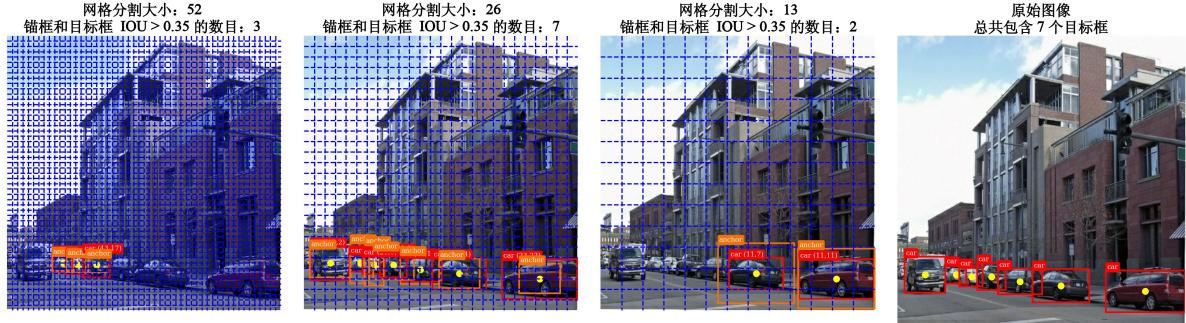


图 2-2 3 个不同网格大小下的目标框与锚框，红色为目标框，橙色为基于数据集分配的锚框，在 3 个不同的网格大小下各分配了 3 种锚框，在图中只显示了相对 $\text{IOU} > 0.35$ 的锚框。演示的标记图片来自于 PASCAL VOC 数据集^[33]，图像大小为 416×416 。

假设当前网格步长为 s ，对应的一个锚框为 (A_w, A_h) ，则网格 (i, j) 处的相对预测框定义为六元组 $(x, y, w, h, c, \{p_c\}_{c=1}^C, \text{bel})$ ，其中：

- (x, y) 表示当前预测框中心点相对于步长 s 的比例，偏移量为 $(i \cdot s, j \cdot s)$ 。
- (w, h) 表示当前预测框的长宽相对于锚框 (A_w, A_h) 的比例。
- c 表示当前预测的置信度，定义为 $\text{Pr}(\text{Object}) \cdot \text{IOU}_{\text{pred}}^{\text{true}}$ 。
- p_c 表示当前预测框预测为类别 c 的概率。
- bel 表示当前预测框预测从属派别为敌方的概率。

不难看出，当前网格步长 s 下网格 (i, j) 处的相对预测框所对应的全局边界框为 $((j + x)s, (i + y)s, wA_w, hA_h)$ 。传统目标识别通常为单类别识别，本毕设任务由于包含单位的从属派别类别，故为多类别识别问题，需要对预测结果以及对应损失函数进行修改。

设模型的输出为 $\hat{z} \in \mathbb{R}^N$ ，则模型预测分布为（Softmax 变换）

$$\hat{y}_i = \text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}} \quad (2-1)$$

对于 N 维离散目标分布为 $y \in \mathbb{R}^N$ ，多元交叉熵损失（Cross-Entropy Loss, CE Loss）为

$$\mathcal{L}_{\text{CE}}(\hat{y}, y) = - \sum_{i=1}^N y_i \log \hat{y}_i \quad (2-2)$$

特殊地，当 $N = 2$ 时，不妨令目标分布为 $\{y, 1 - y\}$ ，则式 (2-2) 被称为二元交叉熵损失（Binary Cross-Entropy Loss, BCE Loss）

$$\mathcal{L}_{\text{BCE}}(\hat{y}, y) = y \cdot \log \hat{y}_1 + (1 - y) \cdot \log \hat{y}_2 \quad (2-3)$$

在目标识别任务中目标分布 y 通常为独热分布（Onehot），其中正确预测的概率为 1，其余类别概率为 0，可以表示 $\text{onehot}(c) = \{[i = c]\}_{i=1}^C$ ，[条件] 表示当内部条件成立

时为 1，否则为 0。然而上述 Softmax 函数只有当 $z_i \gg z_j (i \neq j)$ 时接近该分布，这会导致模型对其预测过于自信，在目标识别任务中容易出现过拟合现象^[34]。

设当前网格步长为 s ，输入图像的大小为 $W \times H$ ，每个网格处所需的预测框数目为 B ，第 i 行 j 列网格预测出的第 k 个预测框记为 $(\widehat{\text{box}}_{ijk}, \widehat{c}_{ijk}, \widehat{p}_{c_{ijk}}, \widehat{\text{bel}}_{ijk})$ ，对应的目标框 $(\text{box}_{ijk}, \text{cls}_{ijk}, \text{bel}_{ijk})$ ，下面给出该步长下的损失函数

$$\begin{aligned} \mathcal{L}_s(\hat{y}, y) = & \sum_{i=1}^{H/s} \sum_{j=1}^{W/s} \sum_{k=1}^B \mathbb{1}_{ijk}^{noobj} \lambda_{noobj} \mathcal{L}_{\text{BCE}}(\widehat{c}_{ijk}, 0) \\ & + \mathbb{1}_{ijk}^{obj} \left[\lambda_{\text{box}} \mathcal{L}_{\text{CIOU}}(\widehat{\text{box}}_{ijk}, \text{box}_{ijk}) + \lambda_{\text{obj}} \mathcal{L}_{\text{BCE}}(\widehat{c}_{ijk}, \text{IOU}_{pred}^{true}) \right. \\ & + \lambda_{\text{class}} \sum_{c=1}^C \mathcal{L}_{\text{BCE}}\left(\{\widehat{p}_{ijk}\}_c, \text{onehot}(\text{cls}_{ijk})_c\right) \\ & \left. + \lambda_{\text{class}} \mathcal{L}_{\text{BCE}}(\widehat{\text{bel}}_{ijk}, \text{bel}_{ijk}) \right] \end{aligned} \quad (2-4)$$

其中 $\mathbb{1}_{ijk}^{noobj}$ 当网格 (i, j) 下的第 k 个预测框没有对应的目标框时为 1，反之为 0，相反的 $\mathbb{1}_{ijk}^{obj} = 1 - \mathbb{1}_{ijk}^{noobj}$ ， $\mathcal{L}_{\text{CIOU}}$ 为 Complete-IOU 损失^[35] 是基于 IOU 损失的一种改进，边界框回归预测问题中相比 MSE 损失能够更快地收敛。

由于本任务的类别数目超过 150 种，而当前通用目标识别数据集 COCO^[36] 类别仅为 80 种，所以本文使用了双模型识别器，如图 2-1 中左下角部分，多个检测器进行组合的模型预测效果见第五章中表 4-1。

2.3 生成式目标识别数据集

假设将每个切片作为绘制单位，定义 $u = (img, box, level)$ ，其中 img 为绘制单位的切片图像； $box = (x, y, w, h, \text{cls}, \text{bel})$ 为绘制单位的边界框参数， (x, y) 为切片中心点位于图像的二维坐标， (w, h) 为切片的宽高大小， cls 为当前切片的所属类别， bel 为当前切片的所属派别； $level$ 为当前切片的所属图层等级，图层等价划分如表 2-1 所示。

表 2-1 图层等级与切片类别关系表

图层等级	切片类别
0	地面法术，地面背景部件
1	地面部队，防御塔
2	空中部队，空中法术
3	其余待识别部件，空中背景部件

绘制单位的插入流程如图 2-1 中右下角部分所示，具体细节如下：

1. 背景选取：从数据集中随机选取一个去除防御塔、部队及文本信息的空竞技场作

为背景图片。

2. 背景增强：加入背景板中的非目标识别部件，用于数据增强，例如：部队阵亡时的圣水，场景中随机出现的蝴蝶、花朵等。
3. 防御塔加入：在双方的三个防御塔固定点位上随机生成完好或被摧毁的防御塔，并随机选取生成与之相关联的生命值信息。
4. 部队加入：按照类别出现次数的反比例 $\left\{ \frac{1}{n_{c_i} - n_{\min} + 1} \right\}_{i=1}^{|C|}$ 所对应的分布进行类别随机选取，其中 $n_{c_i}, (c_i \in C)$ 表示类别 c_i 的切片之前生成的总次数， $n_{\min} = \min\{n_{c_i}\}_{i=1}^{|C|}$ ；在竞技场中按照动态概率分布（具体见附录 C.2）随机选择生成点位，并随机选取生成与之相关的等级、生命值、圣水、时钟等信息。

输入：绘制单位序列 $U = \{u_i\}$, 覆盖率阈值 α , 待识别类别集合 C

输出：image, box

```

1 image ← 空图像, box ← {} // 初始化参数
2  $U \leftarrow \{u_i \in U : u_i^{level} > u_j^{level}, \forall i, j \in \{1, \dots, |U|\} \text{ 且 } i < j\}$ 
3 while True do
4   mask ← 空掩码,  $U_{avail} \leftarrow U$ 
5   for  $i = 1, 2, \dots, |U|$  do
6     if  $\frac{u_i^{img} \cap \text{mask}}{u_i^{img}} > \alpha$  then
7        $| U_{avail} \leftarrow U_{avail} - R(u_i)$  // 删除与  $u_i$  相关联的单位
8     end
9     mask ← mask  $\cup u_i^{img}$ 
10    end
11    if  $|U_{avail}| = |U|$  then
12      break // 覆盖单位筛选完成
13    end
14     $U \leftarrow U_{avail}$ 
15  end
16  $U \leftarrow \{u_i \in U : u_i^{level} < u_j^{level}, \forall i, j \in \{1, \dots, |U|\} \text{ 且 } i < j\}$ 
17 for  $i = 1, 2, \dots, |U|$  do
18   img ← img  $\cup u_i^{img}$  // 图像绘制
19   if  $u_i^{cls} \in C$  then
20     box ← box  $\cup u_i^{box}$  // 边界框保存
21   end
22 end

```

算法 2-1 生成算法伪代码

完成绘制单位加入后，可以按照插入顺序得到待绘制单位序列 U ，但生成的切片可能存在覆盖关系，因此需要引入最大覆盖率阈值 α ，当被覆盖单位面积超过该单位切片面积的 α 倍时，对被覆盖单位进行去除，对单位完成筛选之后，再按照图层等级的从高到低进行绘制，并将识别类别 C 中的边界框信息进行记录，用于后续识别模型训练，具体绘制流程见算法 2-1。通过调整不同的单位生成数量、切片生成类型，最大覆盖率阈值 α ，可以得到如图 2-3 所示的生成结果，生成式代码框架见附录 D.1。



图 2-3 生成式数据集实例

2.4 本章小结

本章介绍了使用目标识别模型完成对每个单位识别的方法，设计了相应的损失函数，并给出了一种制作生成式数据集切片的方法，基于该方法制作了用于目标识别模型训练的生成式数据集（数据集分析与统计信息见 4.1），还通过对视频流数据集逐帧进行人工标记，制作了用于测试模型性能的验证集。

本章设计了双模型以及三模型的模型组合方式，使用生成式数据集训练模型，再在验证集中进行验证，全部模型均表现出良好的泛化能力，其中最好的模型在验证集中达到了 68.8% mAP 指标并在 50% 交并比阈值下达到了 80.9% 召回率，同时保持每张图片 43 毫秒的实时推理性能（模型验证结果见表 4-1）。

3 感知融合与模型决策

3.1 感知模型

3.1.1 目标识别模型

本文使用 YOLO 系列模型作为本次目标识别模型，分别尝试了 YOLOv5^[37] 和 YOLOv8^[29] 作为目标识别器，下面分别对其模型架构改进进行介绍：

YOLOv5 主要是对 YOLOv4^[38] 模型进行了少许改进，YOLOv4 模型在 Backbone 部分（初步特征提取）的主要改进是使用了基于跨阶段部分网络（Cross Stage Partial Networks, CSPNet）^[39] 结构的 DarkNet^[32] 并在输出特征时使用空间金字塔池化（Spatial Pyramid Pooling, SPP）^[40] 对特征进行不同尺度上的提取，在 Neck 部分（进一步特征提取）中使用了路径聚合网络（Path Aggregation Network, PAnet）^[41]，结合特征金字塔与路径压缩，缩短了较低层与最顶层特征之间的信息路径，能够有效进行特征融合。

而 YOLOv8 模型结构上仍然使用 CSP 和 SPP 进行特征提取，相对 YOLOv5 使用了更多的工程优化方法对模型预测速度进行大幅优化，Head 部分（预测框输出）则重新回到 YOLOv1^[21] 无锚框预测的方法，使预测框不再受到锚框约束。

ByteTrack^[30] 是一种基于 Kalman 滤波^[42] 的多目标跟踪（Multiple Object Tracking, MOT）算法，首先需要分别对不同边界框作为跟踪对象，建立独立的线性运动方程，通过 Kalman 滤波结合之前帧的追踪信息，以递归的方式给出目标边界框在当前时刻下的位置预测，再通过计算 IOU 定位当前帧下跟踪对象的位置。与传统 MOT 算法不同的是，ByteTrack 尝试利用滤波预测的方法，从更低的置信度对应的边界框中找到可能被忽略的跟踪边界框，从而恢复可能因遮挡等原因被低估的边界框。

3.1.2 光学字符识别模型

PaddleOCR^[28] 是百度公司研发并开源的一个基于 PaddlePaddle 框架的光学字符识别（Optical Character Recognition, OCR）系统，PaddleOCR 提供了一整套从图像预处理到文字识别的方案，其模型架构主要分为如下两个模块：

1. 文本检测模块：使用 ResNet^[15]、MobileNet^[43] 等经典的卷积神经网络作为特征提取网络，使用特征金字塔网络（Feature Pyramid Networks, FPN）^[44] 等方法来融合多尺度特征，使用可微分二值化（Differentiable Binarization, DB）^[45] 等方法来预测文本框。
2. 文本识别模块：使用卷积循环神经网络（Convolutional Recurrent Neural Networks, CRNN）^[25] 提取文字区域的特征，首先使用卷积网络对图像进行特征提取，再使用 Bi-LSTM^[27] 等序列模型来捕捉字符之间的依赖关系，最后使用 CTC（Connectionist Temporal Classification）^[26] 作为文字序列的损失函数。

3.1.3 图像分类模型

本毕设的图像分类模型使用 ResNet 结构，其核心思想是通过引入短连接（Shortcut Connections），使得网络可以直接学习残差（Residual），从而更容易学习到恒等映射，缓解深度神经网络中梯度消失的问题。具体来说，ResNet 使用残差块（Residual Block）来实现上述操作，残差块由两部分构成，卷积层：卷积变换 $F_W(\cdot)$ 、批归一化（Batch Normalization, BN）^[46] 以及 ReLU 激活函数；残差连接：将输入直接加在卷积层的输出结果上。残差块结构可以表示为

$$y = \sigma(\text{BN}(F_W(x))) + x \quad (3-1)$$

其中 x 为输入图像特征， y 为输出图像特征， F_W 是以卷积核 W 的卷积变换，BN 为批归一化操作， $\sigma(x) = \frac{x+|x|}{2}$ 为 ReLU 函数。

3.2 感知特征提取

在感知特征提取中，本文将同时用到上述三种模型作为一阶段图像特征提取，分别可以得到当前时刻下三个预处理信息：剩余时间（OCR）、竞技场中的预测框（YOLO）、当前手牌及总圣水（图像分类器）。下面将介绍特征提取器的设计方法，可以将预处理信息进一步转化为决策模型的输入信息，它们分别为环境状态信息提取（State）、执行动作信息提取（Action）和环境奖励信息提取（Reward）。

3.2.1 状态特征提取

状态特征包括四种信息：

1. 经过的总时长：直接通过 OCR 对图 1-1 右上角的阶段剩余时间信息识别后，简单处理即可。
2. 竞技场中部队信息：每个部队由五个参数构成二维位置坐标、类别、派别、生命值、其余条状信息构成 $u := (\mathbf{x}, \text{cls}, \text{bel}, \text{bar}_1, \text{bar}_2)$ 。
3. 手牌信息：由于手牌可以被拖出但不释放，因此仅识别手牌图像无法确定其真实状态，需要通过是否做出动作来判断当前手牌是否被使用，故要基于动作特征 3.2.2 判断当前手牌信息。
4. 圣水信息：通过 OCR 对图 1-1 下方可用圣水中的数字进行识别。

在竞技场中部队信息特征提取时，本文引入一种基于上下帧关联性推理的方式来解决部队识别错误或漏识别的问题，在本任务中由于等级和生命值信息容易识别，每



图 3-1 关联性推理

个部队都存在一个与之唯一对应的等级和生命值信息，且部队与等级或生命值信息的相对位置基本不变，所以当部队识别框消失、类别识别错误、派别识别错误时，利用之前与之关联的等级和生命值信息中进行修正，实现效果如图 3-1 所示。当模型在第 1 帧关联了部队 1 与等级、生命值信息的对应关系，通过目标追踪及上下帧信息记忆，即使在第 2, 3 帧未能目标识别模型未能检测到部队 1，通过等级或生命值的关联性推理，模型同样可以推理得到当前单位的真实位置。



图 3-2 动作直接导致的错误状态先验信息：第 1 帧中目标识别错误识别到未部署的部队状态；第 2 帧中 OCR 识别器在目标识别识别到动作之前，产生了错误的总圣水识别信息，因此需要将动作帧提前到该帧之前；第 3 帧中目标识别成功识别到圣水动画，可以判断玩家在该时刻之前部署了部队。

需要注意状态处理的细节问题，在执行动作之前，状态特征中应不包含任何直接与该动作相关的先验信息，否则模仿学习可以通过该先验信息直接给出动作预测，而真实环境中由于不再出现这类动作的先验信息，所以导致模型无法做出决策，错误状态信息如右图 3-2 所示。

假设目标识别每个预测框的识别结果包括七个参数 $b := (x, y, w, h, \text{id}, \text{cls}, \text{bel})$ ，分别为预测框的中心点 (x, y) ，宽高 (w, h) ，目标追踪编号 id ，预测所属类别 cls 以及所属派别 bel 。

定义 3.1 (计数记忆缓存): 设 $I \subset \mathbb{N}$ 为跟踪指标集， S 为可数记忆元素集合， $V \subset \mathbb{R}^+$ 为可用时间集合，定义 $M_T : I \times S \rightarrow \mathbb{Z}^+ \times V$ 为缓存大小为 T 的计数记忆缓存，其满足 $\forall \text{id} \in I, s \in S, (n, t) := M_T(\text{id}, s)$ ，在时间间隔 $[t - T, t] \cap V$ 下跟踪指标为 id 的预测框中，存在 n 个预测框包含元素 s 。 ◇

初始化派别记忆缓存 M_T^{bel} 和类别记忆缓存 M_T^{cls} ，其记忆元素集合分别为派别集合 $\{0, 1\}$ 及全体类别集合，假设当前时间为 t ，于是状态特征提取可以分为下述 6 步^①：

1. 更新派别记忆缓存：对当前每个预测框 b ，更新 $M_T^{\text{bel}}(b^{\text{id}}, b^{\text{bel}})$ ，并对 b^{bel} 进行修正，使得 $M_T^{\text{bel}}(b^{\text{id}}, b^{\text{bel}})_1 = \max \{M_T^{\text{bel}}(b^{\text{id}}, x)_1 : x \in \{0, 1\}\}$ 。
2. 全局文本信息查找：使用 OCR 对当前竞技场中全部文本信息进行识别，用于解决图 3-2 中未放置单位的错误识别问题。
3. 等级、生命值及防御塔信息关联：将等级与部队生命值、防御塔与防御塔生命值进行关联。

^①状态特征提取代码框架见附录 D.2.1。

4. 部队信息关联：基于贪心的关联策略，从下至上，将部队与在一定范围内最近的等级和生命值进行关联，并基于步骤 2 结果，去除具有对应文本的部队单位。
5. 缓存清理：将 $M_T^{\text{bel}}, M_T^{\text{cls}}$ 中超出时间间隔 $[t - T, t]$ 的信息置零。
6. 更新类别记忆缓存：结合关联性信息，对部队及其关联的等级和生命值信息所对应的 M_T^{cls} 进行更新，同时对部队类别进行修正，更新及修正方法与步骤 1 类似。

3.2.2 动作特征提取

动作特征包含两种信息：

- 当前执行动作的二维坐标 x 。
- 当前执行动作所用的卡牌编号 $\text{card} \in \{1, 2, 3, 4\}$ 。

在进行动作特征提取时，需通过目标识别模型识别到的圣水预测框来判断执行动作的时刻以及坐标位置，对圣水预测框的目标检测如图 3-2 中第 3 帧所示，但是在第 2 帧中已经出现了圣水对象，动作执行本应该发生在第 2 帧，但是模型无法对其进行识别导致动作判断出现延迟，所以需通过第 2 帧中总圣水识别发生突变来进行判断动作的执行，因此需要将第 3 帧识别到的动作前移至第 2 帧上。

还需结合当前手牌及圣水上方向的文字信息来判断使用的卡牌编号，通过维护一个手牌记忆缓存记录当前可用手牌，当手牌被玩家或智能体拖出牌库，则将其加入到候选手牌中，每次动作执行的卡牌编号将文本与候选手牌进行比对得到，当两个文本串的 Levenshtein 编辑距离^[47]不超过 2 时，则认为两个文本串相同。

初始化圣水记忆缓存 M_T^{elixir} ，其记忆元素集合为可部署单位的二维坐标空间，假设当前时间为 t ，于是动作特征提取可以分为下述 4 步^②：

1. 缓存清理：将 M_T^{elixir} 中超出时间间隔 $[t - T, t]$ 的信息置零。
2. 更新可用手牌及候选手牌：通过手牌分类器的识别结果以及缓存中记录的手牌，可以完成候选手牌、可用手牌的信息更新。
3. 记录总圣水的突变时刻：通过 OCR 识别当前总圣水以及上一帧总圣水，可用判断当前的总圣水是否发生减少或者无法识别的情况，当此类情况发生则认为动作执行可能发生在当前帧。
4. 动作查找：通过竞技场中对圣水单位的目标识别，判断动作执行的位置，动作执行的真实时刻为一段时间内最早发生的总圣水突变时刻，动作卡牌编号需要结合 OCR 文本识别与候选手牌集合判断，最后再对当前可用手牌进行更新。

3.2.3 奖励特征提取

奖励特征仅包含奖励 $r \in \mathbb{R}$ 一种信息，通过 OCR 识别可以得到敌我防御塔具体生命值，敌我主、副塔如图 1-1 中所示，设 $h_i^{\text{bel}}, (i \in \{0, 1, 2\}, \text{bel} \in \{0, 1\})$ 为防御塔生命

^②动作特征提取代码框架见附录 D.2.2。

值, 当 $i = 1, 2$ 时表示左右两个副塔生命值, $i = 0$ 表示主塔生命值, $\text{bel} = 0, 1$ 分别表示我方和敌方建筑, Δh_i^{bel} 表示前一帧与当前帧生命值的差值, H_i^{bel} 表示对应防御塔的总生命值, 分别定义如下四种奖励函数:

1. 防御塔生命值奖励

$$r_{tower} = \sum_{\text{bel}=0}^1 \sum_{i=0}^2 (-1)^{\text{bel}+1} \frac{\Delta h_i^{\text{bel}}}{H_i^{\text{bel}}} \quad (3-2)$$

2. 防御塔摧毁奖励 $r_{destory}$: 当敌我副塔被摧毁时给予 $(-1)^{\text{bel}+1}$ 奖励, 敌我主塔被摧毁时给予前者的 3 倍奖励。

3. 主塔激活奖励 $r_{activate}$: 当副塔均存活的条件下, 主塔第一次失去生命值时, 给予 $(-1)^{\text{bel}}$ 0.1 奖励。

4. 圣水溢出惩罚 r_{elixir} : 当总圣水持续保持溢出状态时, 每间隔 1 秒产生一次 0.05 的惩罚。

综合上述奖励, 得到总奖励^③:

$$r = r_{tower} + r_{destory} + r_{activate} + r_{elixir} \quad (3-3)$$

需要注意的是, 当图像中的数字信息出现噪声干扰或产生错误目标位置识别时, 需要终止奖励的错误更新, 例如: 部署单位时其他部件对其产生的遮挡 (等级、生命值、圣水、时钟、文本)。

3.3 决策模型

3.3.1 离线强化学习

离线强化学习 (Offline Reinforcement Learning) 是指在没有与环境交互的情况下, 使用预先收集的随机或专家数据来训练强化学习算法, 这类算法适用于无法频繁与环境交互或交互代价高昂的场景, 例如自动驾驶、机器人控制等。由于本任务中的非嵌入环境, 与环境交互速度效率很低, 所以考虑使用离线强化学习作为决策模型。

首先对强化学习中的概念进行介绍, 考虑无限长带折扣 Markov 决策过程 (Markov Decision Process, MDP), 定义为 $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, 其中 \mathcal{S} 为状态空间, \mathcal{A} 为动作空间, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ 为状态转移方程, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ 为奖励函数, $\gamma \in (0, 1)$ 为折扣系数。令 π 表示决策函数 $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, 令 $R(\pi)$ 表示其期望所获得的总奖励 (回报):

$$R(\pi) = \mathbb{E}_{S_1, A_1, S_2, A_2 \dots} \left[\sum_{t=0}^{\infty} r(S_t, A_t) \right], \quad \text{其中 } A_t \sim \pi(\cdot | S_t), S_{t+1} \sim p(\cdot | s_t, a_t) \quad (3-4)$$

强化学习的目标通常是找到最优策略 $\pi^* := \arg \max_{\pi} R(\pi)$, 在线强化学习算法往往通过策略迭代和价值函数估计方法实现策略的更新, 而下文中所使用的离线强化学习算

^③奖励特征提取代码框架见附录 D.2.3。

法不再基于值估计方法，而是更加类似于模仿学习的方法。

Decision Transformer (DT) [8] 是一种将强化学习问题是为序列建模问题的方法，使用了深度学习中的 Transformer 架构，对于离线数据集中的一段长度为 T 的交互轨迹 (Trajectory)

$$\rho = (s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T, s_{T+1}) \quad (3-5)$$

其中 s_{T+1} 为终止状态，则 ρ 可以视为建模为序列

$$R_0, s_1, a_1, R_1, s_2, \dots, a_{T-1}, R_{T-1}, s_T, a_T \quad (3-6)$$

其中 $R_i = \sum_{t=i}^T r_{t+1}, (i = 0, \dots, T-1)$ 为目标回报 (Return-to-Go)。

DT 模型中序列编码模型使用的是 GPT 模型^[48]，即仅含有编码器的因果注意力机制。具体来讲，假设当前处理的序列 $X \in \mathbb{R}^{d \times N}$ 长度为 N ，每个特征编码维度为 d ，则注意力机制^[49] 包含三个可学习矩阵 $W_Q, W_K \in \mathbb{R}^{d_k \times d}, W_V \in \mathbb{R}^{d_v \times d}$ ，分别对应生成询问键 (Query)，查询键 (Key) 和价值键 (Value)

$$\begin{cases} Q = W_Q X \\ K = W_K X \\ V = W_V X \end{cases} \quad (3-7)$$

则对于序列中第 $i \in \{1, \dots, N\}$ 个特征对应的交叉注意力 (Cross-Attention) 为

$$z_i^{cross} = \sum_{j=1}^N \text{softmax} \left(\left\{ \frac{1}{\sqrt{d_k}} \langle \mathbf{q}_i, \mathbf{k}_l \rangle \right\}_{l=1}^N \right)_j \cdot \mathbf{v}_j \iff Z^{cross} = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (3-8)$$

其中包含系数 $1/\sqrt{d_k}$ 的原因：不妨假设 $\mathbf{q}_{ij} \mathbf{k}_{lj} \sim \mathcal{N}(0, \sigma^2), (j \in \{1, \dots, d_k\})$ ，则 $\sum_{j=1}^{d_k} \mathbf{q}_{ij} \mathbf{k}_{lj} \sim \mathcal{N}(0, d_k \sigma^2)$ ，由于初始化的神经网络输出可以保证 $\sigma \approx 1$ ，因此使用系数 $1/\sqrt{d_k}$ 可以保持输出的方差在 1 左右，避免发散。

因果注意力 (Causal-Attention) 为 (每个特征 i 只能看到 $j \leq i$ 的特征)

$$\begin{aligned} z_i^{causal} &= \sum_{j=1}^i \text{softmax} \left(\left\{ \frac{1}{\sqrt{d_k}} \langle \mathbf{q}_i, \mathbf{k}_l \rangle \right\}_{l=1}^N \right)_j \cdot \mathbf{v}_j \\ &\iff Z^{causal} = \text{softmax} \left(\frac{(QK^T) \odot M}{\sqrt{d_k}} \right) V \end{aligned} \quad (3-9)$$

其中 M 为 N 阶下三角阵，引入因果注意力机制后，每个特征由于无法观察到后续特征，所以可以对相邻的下一个特征进行预测，从而无需再对编码器和解码器进行区分，降低了代码复杂性。

DT 训练方法：首先从离线数据集中随机采样得到一段长度为 N 的轨迹

$$\tau_{t-N+1:t} =: (R_0, s_1, a_1, R_1, s_2, a_2, \dots, R_{N-1}, s_N, a_N) = \{R_{n-1}, s_n, a_n\}_{n=1}^N \quad (3-10)$$

将每个特征编码到相同维度 \mathbb{R}^d 下, 按式 (3-6) 建模为长度 $3N$ 的序列 $\tau \in \mathbb{R}^{3N \times d}$, 用 GPT 模型对序列进行特征编码可以得到和 τ 维度相同的编码序列 τ' , 再取出状态序列所对应的编码结果, 通过线性变换映射到动作空间维度, 从而得到对相邻动作的预测

$$(\tau'_2, \tau'_5, \dots, \tau'_{3N-1}) \xrightarrow{\text{线性变换}} (\hat{a}_1, \hat{a}_2, \dots, \hat{a}_N) \quad (3-11)$$

对于离散动作空间使用多元交叉熵损失, 连续动作空间则使用 ℓ^2 范数 (Mean Square Error, MSE) 作为损失函数。

DT 验证方法: 需给出模型期望达到的总奖励 \hat{R}_0 , 通过自迭代的方式使模型完成动作预测, 具体来说, 假设初始状态为 s_1 , 则初始轨迹为 $\tau_1 = (\hat{R}_0, s_1)$, 模型对 τ_1 进行编码, 取出最后一个状态 s_1 所对应的预测动作 \hat{a}_1 与环境交互, 得到新的状态 s_2 和奖励 r_1 , 令 $\hat{R}_1 = \hat{R}_0 - r_1$, 从而得到新的序列 $\tau_2 = (\hat{R}_1, s_1, \hat{a}_1, \hat{R}_1, s_2)$, 模型再对 τ_2 进行编码, 取出 s_2 所对应的预测动作 \hat{a}_2 与环境交互, 以此类推, 直到环境达到终止状态为止。这种方式和自然语言模型中文本生成的做法基本一致。

StARformer^[9] 是一种基于 ViT^[17] 专门对状态中的图像特征编码进行的改进, 通过将轨迹 τ 中的 $\{(a_{n-1}, R_{n-1}, s_n)\}_{n=1}^{3N}$ (a_0 使用特征填充补全) 按照空间维度进行展开 (s_n 使用 ViT 中图像分块的方法将图像分块序列化), 并压缩到空间特征维度 d' , 进而在空间维度上使用交叉注意力机制进行编码, 得到 $\{a'_{n-1}, R'_{n-1}, s'_n\}_{n=1}^{3N}$, 再将第 n 时刻对应的编码 $(a'_{n-1}, R'_{n-1}, s'_n)$ 全部展平, 使用线性变换到时间维度 \mathbb{R}^d 中, 将其记为 $\{l_n\}_{n=1}^N$, 则时间维度的序列建模为

$$\tau := s_1, l_1, s_2, l_2, \dots, s_N, l_N \quad (3-12)$$

通过因果注意力机制 (相邻的 (s_n, l_n) , $n = 1, \dots, N$ 之间仍然具有注意力机制) 可以完成对时序序列信息的编码, 将编码结果记为 τ' , 类似 DT 的预测方法, 只考虑所有状态对应的编码结果, 通过线性变换映射到动作空间维度, 从而得到相邻动作的预测

$$(\tau'_1, \tau'_3, \dots, \tau'_{2N-1}) \xrightarrow{\text{线性变换}} (\hat{a}_1, \hat{a}_2, \dots, \hat{a}_N) \quad (3-13)$$

模型的训练及推理方式与 DT 完全一致, 相比 DT 模型, StARformer 能够有效的提高模型对图像信息的理解能力, 本毕设对上述离线强化学习算法进行了复现^④, 本文的相关复现实验显示相比 DT 更不依赖于初始时目标总奖励 \hat{R}_0 的设定, 这说明其更倾向于对专家数据的模仿学习, 而非与目标总奖励进行对齐。

^④<https://github.com/wty-yy/Decision-Transformer-JAX>, 复现内容包括: Decision Transformer (DT), Return-Aligned Decision Transformer (RADT) 和 StARformer, 在 Atari 环境中进行了对比试验, 并对结果进行可视化。在 5 个不同 Atari 环境下的实验表明, StARformer 的平均得分超过 DT 算法的 30%。

3.3.2 状态空间与动作空间

模型的状态输入由 2 部分构成，分别为 S^{img}, s^{card} ，其中 $S^{img} \in \mathbb{R}^{18 \times 32 \times 15}$ 为单位的网格状特征输入，对于第 i 行 j 列的特征 $z_{ij} := (S^{img})_{ij} \in \mathbb{R}^{15}$ 表示处于该位置的单位具有如下 4 种特征： $(z_{ij})_{1:8}$ 为类别编码， $(z_{ij})_9$ 为从属派别编码， $(z_{ij})_{10:12}$ 为生命值图像编码， $(z_{ij})_{13:15}$ 为其余条状图像编码； $s^{card} \in \mathbb{R}^6$ 表示当前状态下的两个全局特征： $(s^{card})_{1:5}$ 为当前手牌信息， $(s^{card})_6$ 为当前总圣水量。

模型的动作输入由 2 个部分构成： a^{pos}, a^{select} ，其中 $a^{pos} \in \mathbb{R}^2$ 表示动作执行的部署坐标， a^{select} 表示动作执行的手牌编号。

3.3.3 预测目标设计与重采样

由于本任务中动作执行极为离散，总帧数中仅有 4% 为执行动作帧，其余帧均不执行动作，如果直接逐帧预测动作会产生非常严重的长尾问题，导致模型最终基本不执行动作（表 4-2 中离散预测的动作数远低于连续动作预测数），因此需要将预测目标从离散转化为连续，解决方法是引入延迟动作预测：对于第 i 帧，需找到其后（包含自身）最近的动作帧 j ，令最大间隔帧数阈值为 T_{delay} ，则每个非动作帧的预测的延迟动作为 $a_i^{delay} = \min\{j - i, T_{delay}\}$ 。

对离线数据集进行采样时，为避免长尾问题导致模型偏移，本文还设置了重采样频次，设数据集总帧数为 N ，动作帧数为 N_{action} ，则动作帧占比为 $r_a := N_{action}/N$ ，对于第 i 个动作帧位于数据集中的第 t_i 帧，则 $j \in \{t_i, \dots, t_{i+1} - 1\}$ 帧作对应的重采样频次为

$$s_j = \max \left\{ \frac{1}{1 - r_a}, \frac{1}{r_a(j - t_i + 1)} \right\}, \quad (t_i \leq j \leq t_{i+1}) \quad (3-14)$$

则训练轨迹中结束帧的采样分布为 $\left\{ \frac{s_j}{\sum_{j=1}^N s_j} \right\}_{j=1}^N$ ，图 3-3 中展示了离线数据集一段轨迹所对应的重采样频次与动作预测值。

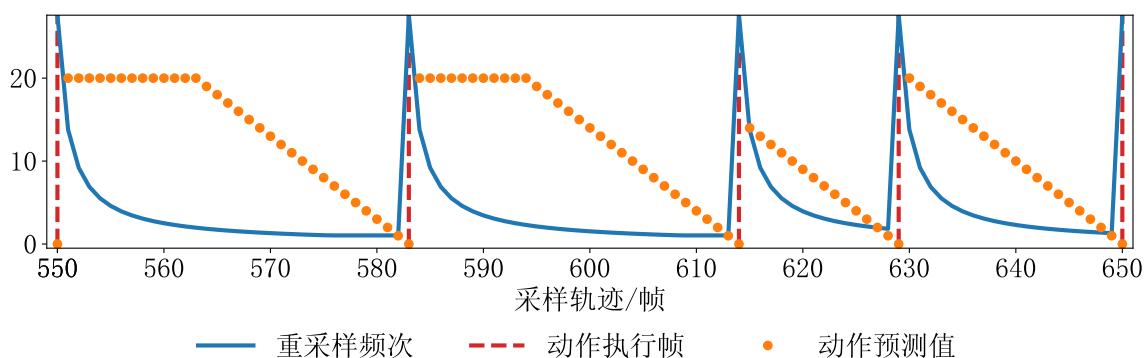


图 3-3 从离线数据集中截取的一段数据，总共包含 5 个动作帧，最大间隔帧数阈值 $T_{delay} = 20$ ，

3.3.4 模型架构设计

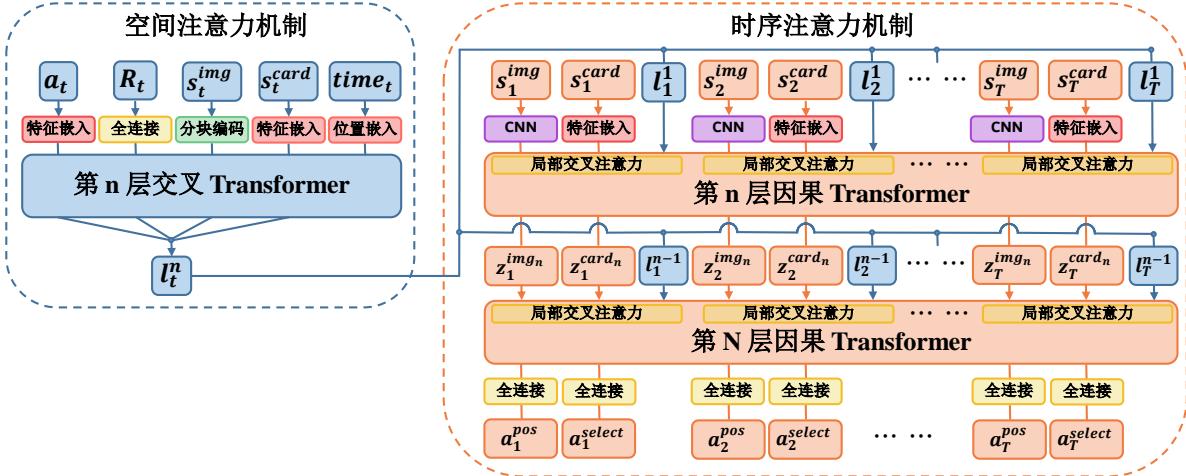


图 3-4 决策模型架构

本文使用的决策模型架构如图 3-4 所示，基于 StARformer 的 ViT+DT 架构，模型输入为轨迹序列 $(a_t, R_t, s_t)_{t=1}^T$ ，输出为动作预测序列 $(a_t^{pos}, a_t^{select})_{t=1}^T$ 。左侧交叉注意力机制对局部信息 (a_t, R_t, s_t) 按空间维度进行编码，并使用 ViT 中分块思路将图像 s_t^{img} 转化为序列；右侧为因果注意力机制对全局信息 (s_t^{img}, s_t^{card}) 按时序维度进行编码，并在每层序列输入中引入对应的局部编码信息 l_t^n 。

本文总共设计了 3 种不同的模型架构，分别基于 StARformer 和 DT 模型，假设输入的轨迹长度为 L ，时序注意力机制中的输入序列长度 T ，使用 N 层 Transformer 堆叠，则每种模型架构设计细节如下：

StARformer-3L: 基于 StARformer 架构，本文设计的 StARformer-3L 决策模型架构如图 3-4 所示，其中输入序列长度 $T = 3L$ ，第 n 层 Transformer 输出的时序序列记为 $\{z_t^{img_n}, z_t^{card_n}, l_t^{n-1}\}_{t=1}^L$ （序列长度 $3L$ ）。右侧时序注意力机制中的因果 Transformer，由于需要使同一时刻下的信息可以相互产生注意力关系，所以需要引入局部交叉注意力，具体实现方法是将式 (3-9) 中的掩码矩阵 M_3 ，其中 M_{L_0} 定义为

$$(M_{L_0})_{ij} = \begin{cases} 1, & i = kL_0 - l, j \leq kL_0 \\ 0, & \text{否则} \end{cases}, \quad k \in \{1, \dots, L\}, l \in \{0, \dots, L_0 - 1\} \quad (3-15)$$

其本质上是在下三角阵的对角线上顺次放置不交的大小为 $L_0 \times L_0$ 的全 1 矩阵。

StARformer-2L: $T = 2L$ 的模型架构与 StARformer^[9] 论文中架构基本一致，其将图像信息 s_t^{img} 与 s_t^{card} 编码到同一特征 z_t 中，得到第 n 层的 Transformer 输出的时序序列 $\{z_t^n, l_t^{n-1}\}_{t=1}^L$ （序列长度 $2L$ ），同理需要使用局部交叉注意力，并将掩码矩阵置为 M_2 ，预测中 a_t^{pos} 和 a_t^{select} 均使用 z_t 进行解码得到。

DT-4L: 基于 DT 架构，可以看作仅包含图 3-4 中时序注意力部分，将时序注意力机制中 l_t^n 进行替换，得到时序序列为 $\{a_{t-1}, R_{t-1}, s_t^{img_n}, s_t^{card_n}\}_{t=1}^L$ （序列长度 $4L$ ），预测中 a_t^{pos} 和 a_t^{select} 分别由 $s_t^{img_n}$ 和 $s_t^{card_n}$ 对应解码得到。

3.4 本章小结

感知融合：首先分别对 YOLO 系列目标识别模型、PaddleOCR 光学字符识别模型、ResNet 图像分类器进行介绍，将其作为非嵌入图像特征信息的初步提取器，再通过状态、动作、奖励三种不同感知特征提取器对特征信息进行融合，通过结合上下帧信息及单位的关联信息恢复漏识别目标，排除对动作决策具有干扰的特征，最后将融合完成的特征作为 3.3 中决策模型的输入信息。

决策模型：先对离线强化学习中的基本定义及 DT 和 StARformer 模型进行介绍，详细介绍了训练及推理的实现方法。由于本任务的离线数据集中存在严重的长尾问题，本文通过对预测目标的重新设计，并在模型训练中加入重采样策略，从而一定程度上缓解了长尾问题。最后本文还对 StARformer 模型架构做出了改进，设计了三种不同的决策模型框架，进行对比实验测试（见表 4-2）。从表中可知，预测目标修改提高了 37% 的模型性能，将 StARformer 模型架构从 2L 修改为 3L 提高了 24% 的模型性能。

4 数据分析及实验结果

本章对前 3 章内容的实验结果进行总结，分别包含第 2 章的生成式数据集分析统计，3.1 的目标识别模型对比实验以及 3.3 的决策模型对比实验。

4.1 生成式数据集分析

数据集总共分为两部分^①：

1. 生成式数据集切片：总计 154 个类别，待识别类别 150 个，总共包含 4654 个切片，在全部待识别类别的切片图像中，切片大小分布如图 4-1 所示。
2. 目标识别验证集：总计 6939 张人工标记的目标识别图像，包含 116878 个目标框，平均每张图片包含 17 个目标框，该数据集均为真实对局视频流逐帧标记得到，而模型训练所使用的完全是生成式数据集，所以该数据集可以做验证集使用。

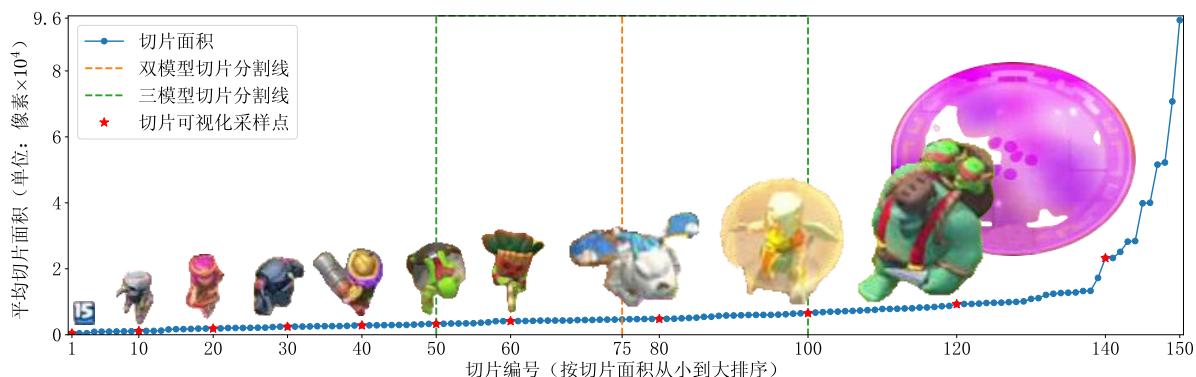


图 4-1 将切片数据集全部切片按平均面积从小到大排序并编号，从中随机采样出部分切片进行可视化。分别以全体切片面积的二等分和三等分点，作为双模型和三模型的识别类别分割线。

4.2 目标识别模型

目标识别模型使用了自己实现的 YOLOv5^② 和重构后的 YOLOv8 的模型^③，每个训练集大小设置为 20000，至多训练 80 个 epoch 收敛。数据增强使用了：HSV 增强，图像旋转，横纵向随机平移，图像缩放，图像左右反转，具体参数见附录表 C-1。

实验结果^④如表 4-1 所示，表中具体内容解释如下：

1. 模型名称：编号后的字母表示模型大小，l,x 分别对应大与特大型模型，YOLOv8-l×n 表示使用 n 个 YOLOv8-l 模型，每个子模型分别识别图 4-1 中分割线所划分区域

^①上述数据集统计信息截止于 2024 年 5 月 6 日，全部图像数据集均已开源：<https://github.com/wty-yy/Clash-Royale-Detection-Dataset>

^②复现 YOLOv5 代码：<https://github.com/wty-yy/KataCV/tree/master/katacv/yolov5>

^③YOLOv8 重构内容：https://github.com/wty-yy/KataCR/blob/master/assets/yolov8_modify.md

^④YOLOv5 全部训练曲线：<https://wandb.ai/wty-yy/ClashRoyale>, YOLOv8 全部训练曲线：<https://wandb.ai/wty-yy/YOLOv8>

中的切片类型，最后将识别的预测框通过非最大值抑制（Non-Maximum Suppression, NMS）进行筛选，NMS 过程中 IOU 阈值设定为 0.6。

2. 评测指标：表中 mAP 评测指标表示 COCO mAP 指标^[36]，即在 10 种不同 IOU 阈值下计算 PR 曲线下面积求平均得到，AP50、P50、R50 和 mAP(S) 分别表示在判断正例的 IOU 阈值为 50% 下的 mAP、平均精度、平均召回率和小目标的 mAP。

3. 验证速度：模型预测时 Batch 大小设置为 1，FPS 为模型在 GeForce RTX 4090 下测试的验证速度，验证测试时置信度设置为 0.001。当对视频流数据进行预测时，将置信度改为 0.1，并使用 ByteTrack^[30] 算法在目标追踪计算过程中对边界框进行筛选，FPS(T) 是在 GeForce RTX 4060 Laptop 下带有目标追踪的识别速度。

从实验结果可以看出，YOLOv8-l 的双识别器对小目标的识别能力与三识别器效果基本一致，并远高出非组合式的识别器，其原因可能在于 150 个预测类别大小远超模型的识别能力范围，最大目标与最小目标的边界框大小差距甚远，又由于 YOLOv8 是无锚框识别头，由于大目标易于识别，可能导致预测的目标框均偏大，所以多个识别器降低平均类别数能够有效对小目标进行识别。

表 4-1 YOLO 模型对比测试结果

模型名称	AP50	P50	R50	mAP	mAP(S)	FPS	FPS(T)	检测器类别数	数据增强
YOLOv5-l	66.2	84.4	63.8	53.2	NA	59	NA	151	
YOLOv8-x	83.1	93.9	68.3	67.7	39.8	68	31	160	
YOLOv8-x	85.3	90.7	80.4	66.8	35.9	68	31	160	✓
YOLOv8-l × 2	84.3	89.5	79.8	67.4	43.9	34	18	85	✓
YOLOv8-l × 3	85.2	89.7	80.9	68.8	48.3	23	10	65	✓

4.3 决策模型

本文基于第 3 章中介绍的感知融合技术，手动构建了玩家与与游戏内置的 8000 分 AI 对局 105 回合的专家数据^⑤，固定双方使用的卡组（具体卡组见附录 C.4），数据集总计 113981 帧，动作帧占比 4.01%，重采样频次比率为 $\frac{\text{动作帧}}{\text{非动作帧}} = 24.92 : 1.04$ ，平均动作延迟大小为 21.26，最大间隔帧数阈值为 $T_{delay} = 20$ （重采样细节见 3.3.3）。模型损失函数分为三个部分，由于均为离散动作，所以损失函数均使用目标动作均使用交叉熵损失，总损失函数如下

$$\mathcal{L} = \sum_{i=1}^N \mathbb{I}_{a_i^{delay} < T_{delay}} \left[\mathcal{L}_{CE}(\hat{a}_i^{pos}, a_i^{pos}) + \mathcal{L}_{CE}(\hat{a}_i^{select}, a_i^{select}) + \mathcal{L}_{CE}(\hat{a}_i^{delay}, a_i^{delay}) \right] \quad (4-1)$$

其中 T_{delay} , a_i^{pos} , a_i^{select} , a_i^{delay} 分别为最大间隔帧数阈值、目标动作的部署坐标、手牌编号以及部署延迟，注意每条轨迹下只考虑 $a_i^{delay} < T_{delay}$ 对应的梯度。

本文分别测试了下述模型参数：

^⑤全部专家数据集均已开源：<https://github.com/wty-yy/Clash-Royale-Replay-Dataset>

- 不同的模型架构（架构设计见 3.3），分别测试了 StARformer 和 DT 架构。
- 模型输入的轨迹步长记为 L ，测试了 $L = 30, 50, 100$ 的情况。
- 不同的预测目标（离散与连续预测见 3.3.3）。
- 不同的手牌预测范围，默认预测当前手牌编号，也尝试了对当前牌库中全部手牌进行预测。

本文使用了如下数据增强方式对模型进行训练：

- 重采样：对稀疏的动作帧按比例进行重采样，加快模型收敛，缓解离线数据集的长尾问题。
- 随机手牌重组：对当前输入轨迹中的全部手牌按照随机排列进行打乱，当预测全部手牌编号时，将动作对应的手牌也进行相应变换。

全部模型训练曲线均进行了上传^⑥，模型实时对局的验证结果总结于表4-2中，在实时对局的实现中包含以下细节：

- 动作执行：对于连续动作预测模型中，由于感知识别中存在延迟，当预测动作延迟在 8 帧以内就会立刻执行动作，并且为了避免总圣水溢出导致的惩罚奖励，每当总圣水达到 10 时就直接下出当前预测的卡牌。
- 无效动作跳过：若模型预测出的卡牌所需圣水超出了当前总圣水量或当前动作执行的卡牌位为空。

表 4-2 决策模型对比

模型框架	步长 L	训练回合	总奖励	对局时长/秒	动作数	胜率
DT-4L	50	8	-5.7 ± 2.5	148.9 ± 33.6	128.7 ± 37.7	5%
StARformer-2L	30	3	-6.0 ± 2.3	135.0 ± 35.1	141.8 ± 57.9	5%
StARformer-2L	50	8	-6.2 ± 2.2	131.9 ± 44.3	195.3 ± 69.8	5%
StARformer-2L	100	1	-4.9 ± 2.8	150.2 ± 35.6	187.6 ± 48.2	0%
StARformer-3L	30	4	-5.1 ± 3.7	147.2 ± 37.4	190.8 ± 52.7	10%
StARformer-3L	50	3	-4.7 ± 3.1	158.9 ± 27.7	207.8 ± 48.2	5%
StARformer-3L	100	5	-6.1 ± 2.2	125.9 ± 37.8	144.6 ± 42.9	5%
StARformer-3L (全卡牌预测)	50	2	-5.6 ± 2.1	150.2 ± 38.6	195.3 ± 69.8	0%
StARformer-2L (离散动作预测)	50	1	-7.5 ± 0.8	123.1 ± 39.2	21.9 ± 9.4	0%

表 4-2 中记录了为每个模型的前 10 次训练结果中，与环境交互 20 个回合得到的最高平均奖励，从中可以看出，将离散预测改为连续预测提高了 37% 的性能、StARformer 架构从 2L 修改为 3L 的改动提高了 24% 的模型性能。表中每列的含义分别为：

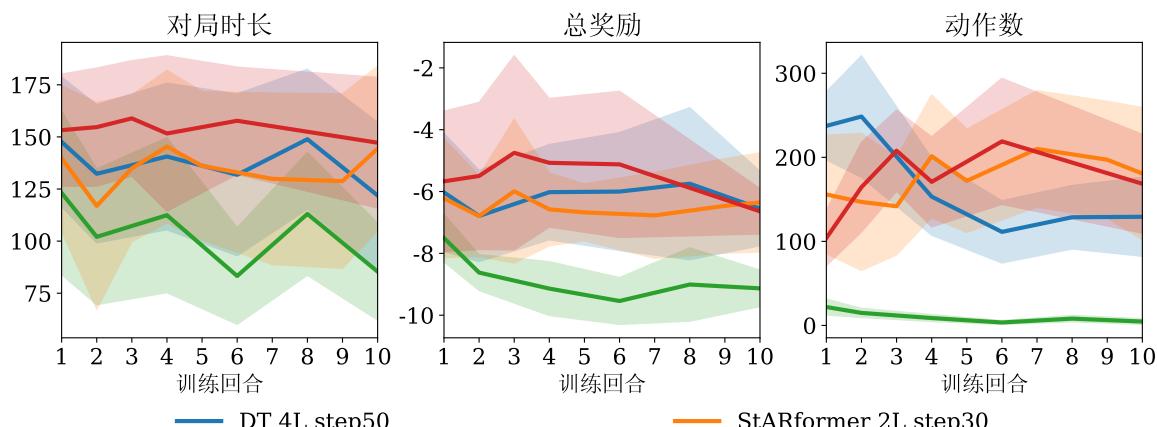
- 步长 L ：为模型架构设计 3.3.4 中的输入轨迹长度。

^⑥决策模型全部训练曲线：<https://wandb.ai/wty-yy/ClashRoyale%20Policy>

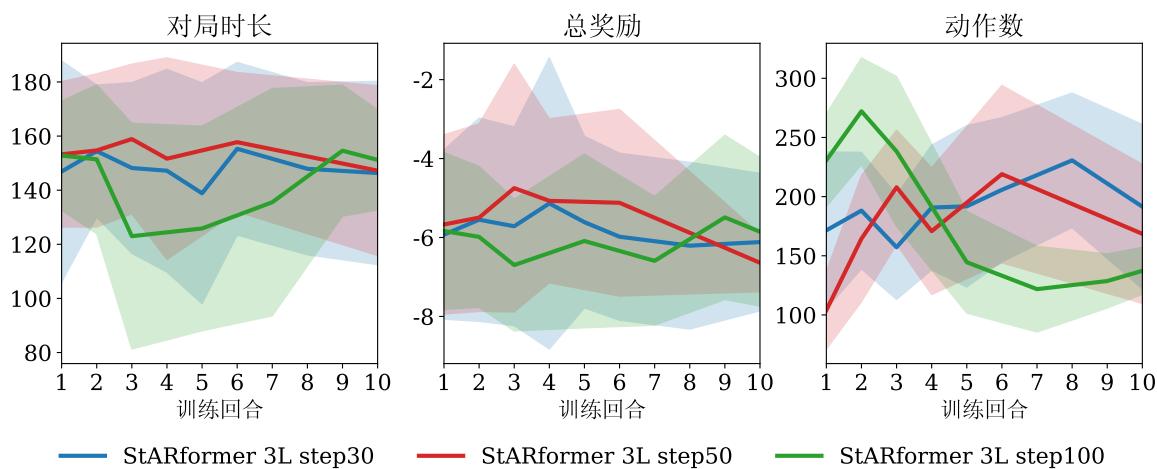
- 训练回合：前 10 个训练结果中，获得最高奖励所对应的回合数。
- 总奖励：按奖励公式 (3-3) 进行累计得到的总奖励。
- 对局长：统计每次对局的时长。
- 动作数：统计每局智能体成功执行的动作数目。
- 胜率：按照 1.3 中介绍的游戏目标，判定智能体的获胜概率。

硬件交互方法见附录 C.1，本文使用的验证环境：手机系统为鸿蒙、电脑系统为 Ubuntu24.04 LTS、CPU: R9 7940H、GPU: RTX GeForce 4060 Laptop，平均决策用时 120ms，感知融合用时 240ms。

图 4-2 中展示了每种模型前 10 个训练结果的验证曲线。



(a) 不同模型结构



(b) StARformer-3L 采取不同轨迹长度

图 4-2 模型验证曲线：展示了前 10 个训练结果，每回合模型在真实对局中的对局时间时长、总奖励和执行动作数，每次进行 20 次对局，实线为均值、虚影为标准差。

5 总结与展望

本文基于游戏皇室战争 (Clash Royale)，首次提出了一种基于非嵌入式的离线强化学习训练策略。结合目标识别和光学文本识别的顶尖算法，成功实现了智能体在移动设备上进行实时对局，并且战胜了游戏中的内置 AI。

主要贡献包含以下三点：

1. 数据集制作：本文设计了一种高效制作切片数据集的方法，制作了包含 4654 张切片、共 150 个类别的切片数据集，以及包含 116878 个目标框、共 6939 张图像的目标识别数据集。提出的生成式目标识别数据集算法可以模拟真实对局场景生成带标签的图像，通过生成式图像训练的模型在真实视频流中表现出良好的泛化性，具有很高的识别准确率。
2. 感知融合算法：基于计算机视觉模型的输出结果，设计了感知融合算法，该算法结合视频数据中的上下帧信息优化特征结果，进一步提升了识别的准确率。
3. 决策模型改进：在决策模型方面，从架构及预测目标两个方面对传统模型进行改进，将难以学习的离散动作序列转化为连续动作序列，大幅提高了模型性能。制作了包含 105 回合、总共 113981 帧的专家数据集，并基于该离线数据集训练出能够战胜游戏内置 AI 的智能体。

本文为非嵌入式强化学习在移动设备上的应用提供了新的思路。未来的工作可以从以下几个方面进行扩展：

1. 数据集的扩展与优化：增加数据集的规模和多样性，进一步提升模型的泛化能力和识别准确率。
2. 算法的改进：当前在固定卡组下进行训练，仍然无法 100% 战胜游戏中的内置 AI，因此远无法达到人类平均水平，而且制作离线强化学习数据集需要花费大量的人力，若要进一步提升智能体能力，应该需要采用在线强化学习算法，与此同时需要使用更加高效的感知融合算法和决策模型架构，才有可能进一步提高智能体的实时决策能力和对局胜率。
3. 实际应用的拓展：将本文的方法应用于更多的游戏和实际场景中，验证其通用性和实用价值。

本文的全部代码均已开源^①，期望能够为相关领域的研究者提供有价值的参考和借鉴。

^①全部代码：<https://github.com/wty-yy/katacr>

致 谢

感谢西安交通大学数学学院能够给我这次自拟毕设题目的机会，如果错过这次机会，之后可能很难再有充足的时间与精力完成本毕设内容；感谢数学学院强基计划，通过本科课程学习，使我有了扎实的数学基础，能够比较轻松地看懂计算机视觉、强化学习等领域中的论文，理解并对其中的公式进行推导加深理解，最后使用代码进行复现。

感谢兰旭光老师对本毕设的支持，即使本毕设内容充满各种未知挑战，但老师仍给我提供了必要的算力支持，如果没有充足算力，本文进展将相当缓慢，完全无法完成任务。感谢课题组中各位师兄师姐为本次毕设提供思路，其中使用 SAM 辅助切片制作和多目标识别模型进行目标检测思路来自唐宇航博士，决策模型中将离散预测转化为连续预测思路来自万里鹏博士，如果没有这些改进，精准的识别模型可能无法实现，决策模型也没有任何性能，战胜游戏中的内置 AI 将成为幻想。感谢我身边朋友的支持，他们的支持使我愈发坚定将不可能变为可能的决心。感谢所有开源社区中贡献者们，他们无私奉献出各种领域的顶尖模型（YOLO 系列，PaddleOCR，GPT，DT 等）也让本次毕设成为可能，从他们的源代码中我总能学到非常多新知识、新架构设计方案，从而能不断提升自身的代码水平，向他们看齐。

感谢我的父母对我一直以来的支持，让我能在高中参加算法竞赛，打下扎实的编程基础，并支持我去追随自己儿时的梦想——“做出一个能够与现实进行直接交互并改善生活的智能体”。如果没有这样坚定的梦想，我可能早在制作数据集的枯燥过程中放弃，但通过大学的各种知识与技术的学习，让我看到了前进的道路，坚定下自己的信念，将全部代码从零实现了出来，从而初步完成了本毕设开题时所设定的目标，成功地走出了第一步。

参考文献

- [1] Silver D, Huang A, Maddison C J, et al. Mastering the game of go with deep neural networks and tree search[J]. *nature*, 2016, 529(7587):484-489.
- [2] Silver D, Hubert T, Schrittwieser J, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm[J]. *arXiv preprint arXiv:1712.01815*, 2017.
- [3] Berner C, Brockman G, Chan B, et al. Dota 2 with large scale deep reinforcement learning[J]. *arXiv preprint arXiv:1912.06680*, 2019.
- [4] Hochreiter S, Schmidhuber J. Long short-term memory[J]. *Neural computation*, 1997, 9(8):1735-1780.
- [5] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. *arXiv preprint arXiv:1707.06347*, 2017.
- [6] Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning[J]. *Nature*, 2019, 575(7782):350-354.
- [7] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. *Nature*, 2015, 518(7540):529-533.
- [8] Chen L, Lu K, Rajeswaran A, et al. Decision transformer: Reinforcement learning via sequence modeling[J]. *Advances in neural information processing systems*, 2021, 34:15084-15097.
- [9] Shang J, Kahatapitiya K, Li X, et al. Starformer: Transformer with state-action-reward representations for visual reinforcement learning[C]//European conference on computer vision. Springer, 2022: 462-479.
- [10] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. *Proceedings of the IEEE*, 1998, 86(11):2278-2324.
- [11] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]//2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009: 248-255.
- [12] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. *Communications of the ACM*, 2017, 60(6):84-90.
- [13] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. *arXiv preprint arXiv:1409.1556*, 2014.
- [14] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
- [15] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [16] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.
- [17] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. *arXiv preprint arXiv:2010.11929*, 2020.
- [18] Ding X, Zhang X, Ma N, et al. Repvgg: Making vgg-style convnets great again[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 13733-13742.
- [19] Yu W, Luo M, Zhou P, et al. Metaformer is actually what you need for vision[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10819-10829.
- [20] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. *Advances in neural information processing systems*, 2017, 30.

- [21] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
- [22] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer, 2016: 21-37.
- [23] Carion N, Massa F, Synnaeve G, et al. End-to-end object detection with transformers[C]//European conference on computer vision. Springer, 2020: 213-229.
- [24] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
- [25] Shi B, Bai X, Yao C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(11):2298-2304.
- [26] Graves A, Fernández S, Gomez F, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks[C]//Proceedings of the 23rd international conference on Machine learning. 2006: 369-376.
- [27] Huang Z, Xu W, Yu K. Bidirectional lstm-crf models for sequence tagging[J]. arXiv preprint arXiv:1508.01991, 2015.
- [28] Du Y, Li C, Guo R, et al. Pp-ocr: A practical ultra lightweight ocr system[J]. arXiv preprint arXiv:2009.09941, 2020.
- [29] Jocher G, Chaurasia A, Qiu J. Ultralytics YOLO[CP/OL]. 2023. <https://github.com/ultralytics/ultralytics>.
- [30] Zhang Y, Sun P, Jiang Y, et al. Bytetrack: Multi-object tracking by associating every detection box [C]//European conference on computer vision. Springer, 2022: 1-21.
- [31] Kirillov A, Mintun E, Ravi N, et al. Segment anything[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 4015-4026.
- [32] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.
- [33] Everingham M, Van Gool L, Williams C K, et al. The pascal visual object classes (voc) challenge[J]. International journal of computer vision, 2010, 88:303-338.
- [34] Zhang Z, He T, Zhang H, et al. Bag of freebies for training object detection neural networks[J]. arXiv preprint arXiv:1902.04103, 2019.
- [35] Zheng Z, Wang P, Liu W, et al. Distance-iou loss: Faster and better learning for bounding box regression [C]//Proceedings of the AAAI conference on artificial intelligence: volume 34. 2020: 12993-13000.
- [36] Lin T Y, Maire M, Belongie S, et al. Microsoft coco: Common objects in context[C]//Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer, 2014: 740-755.
- [37] Jocher G. YOLOv5 by Ultralytics[CP/OL]. 2020. <https://github.com/ultralytics/yolov5>. DOI: 10.5281/zenodo.3908559.
- [38] Bochkovskiy A, Wang C Y, Liao H Y M. Yolov4: Optimal speed and accuracy of object detection[J]. arXiv preprint arXiv:2004.10934, 2020.
- [39] Wang C Y, Liao H Y M, Wu Y H, et al. Cspnet: A new backbone that can enhance learning capability of cnn[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020: 390-391.

- [40] He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(9):1904-1916.
- [41] Liu S, Qi L, Qin H, et al. Path aggregation network for instance segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 8759-8768.
- [42] Kalman R E. A new approach to linear filtering and prediction problems[J]. 1960.
- [43] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
- [44] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
- [45] Liao M, Wan Z, Yao C, et al. Real-time scene text detection with differentiable binarization[C]// Proceedings of the AAAI conference on artificial intelligence: volume 34. 2020: 11474-11481.
- [46] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. pmlr, 2015: 448-456.
- [47] Levenshtein V I, et al. Binary codes capable of correcting deletions, insertions, and reversals[C]// Soviet physics doklady: volume 10. Soviet Union, 1966: 707-710.
- [48] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.
- [49] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

附录 A 外文文献原文

YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss, and combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP₅₀) for the MS COCO dataset at a real-time speed of ~65 FPS on Tesla V100. Source code is at <https://github.com/AlexeyAB/darknet>.

1. Introduction

The majority of CNN-based object detectors are largely applicable only for recommendation systems. For example, searching for free parking spaces via urban video cameras is executed by slow accurate models, whereas car collision warning is related to fast inaccurate models. Improving the real-time object detector accuracy enables using them not only for hint generating recommendation systems, but also for stand-alone process management and human input reduction. Real-time object detector operation on conventional Graphics Processing Units (GPU) allows their mass usage at an affordable price. The most accurate modern neural networks do not operate in real time and require large number of GPUs for training with a large mini-batch-size. We address such problems through creating a CNN that operates in real-time on a conventional GPU, and for which training requires only one conventional GPU.

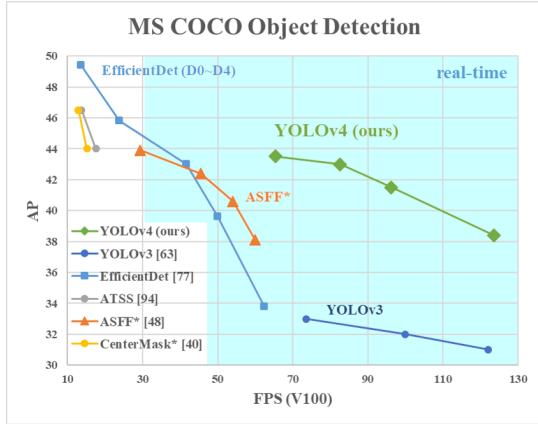


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

The main goal of this work is designing a fast operating speed of an object detector in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP). We hope that the designed object can be easily trained and used. For example, anyone who uses a conventional GPU to train and test can achieve real-time, high quality, and convincing object detection results, as the YOLOv4 results shown in Figure 1. Our contributions are summarized as follows:

1. We develop an efficient and powerful object detection model. It makes everyone can use a 1080 Ti or 2080 Ti GPU to train a super fast and accurate object detector.
2. We verify the influence of state-of-the-art Bag-of-Freebies and Bag-of-Specials methods of object detection during the detector training.
3. We modify state-of-the-art methods and make them more efficient and suitable for single GPU training, including CBN [89], PAN [49], SAM [85], etc.

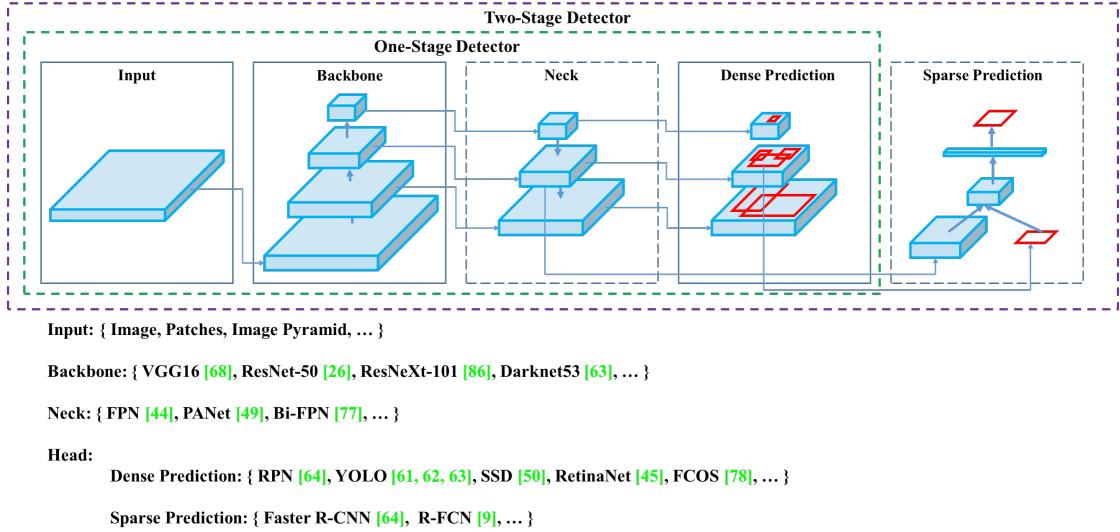


Figure 2: Object detector.

2. Related work

2.1. Object detection models

A modern detector is usually composed of two parts, a backbone which is pre-trained on ImageNet and a head which is used to predict classes and bounding boxes of objects. For those detectors running on GPU platform, their backbone could be VGG [68], ResNet [26], ResNeXt [86], or DenseNet [30]. For those detectors running on CPU platform, their backbone could be SqueezeNet [31], MobileNet [28, 66, 27, 74], or ShuffleNet [97, 53]. As to the head part, it is usually categorized into two kinds, i.e., one-stage object detector and two-stage object detector. The most representative two-stage object detector is the R-CNN [19] series, including fast R-CNN [18], faster R-CNN [64], R-FCN [9], and Libra R-CNN [58]. It is also possible to make a two-stage object detector an anchor-free object detector, such as RepPoints [87]. As for one-stage object detector, the most representative models are YOLO [61, 62, 63], SSD [50], and RetinaNet [45]. In recent years, anchor-free one-stage object detectors are developed. The detectors of this sort are CenterNet [13], CornerNet [37, 38], FCOS [78], etc. Object detectors developed in recent years often insert some layers between backbone and head, and these layers are usually used to collect feature maps from different stages. We can call it the neck of an object detector. Usually, a neck is composed of several bottom-up paths and several top-down paths. Networks equipped with this mechanism include Feature Pyramid Network (FPN) [44], Path Aggregation Network (PAN) [49], BiFPN [77], and NAS-FPN [17].

In addition to the above models, some researchers put their emphasis on directly building a new backbone (DetNet [43], DetNAS [7]) or a new whole model (SpineNet [12], HitDetector [20]) for object detection.

To sum up, an ordinary object detector is composed of several parts:

- **Input:** Image, Patches, Image Pyramid
- **Backbones:** VGG16 [68], ResNet-50 [26], SpineNet [12], EfficientNet-B0/B7 [75], CSPResNeXt50 [81], CSPDarknet53 [81]
- **Neck:**
 - **Additional blocks:** SPP [25], ASPP [5], RFB [47], SAM [85]
 - **Path-aggregation blocks:** FPN [44], PAN [49], NAS-FPN [17], Fully-connected FPN, BiFPN [77], ASFF [48], SFAM [98]
- **Heads:**
 - **Dense Prediction (one-stage):**
 - RPN [64], SSD [50], YOLO [61], RetinaNet [45] (anchor based)
 - CornerNet [37], CenterNet [13], MatrixNet [60], FCOS [78] (anchor free)
 - **Sparse Prediction (two-stage):**
 - Faster R-CNN [64], R-FCN [9], Mask R-CNN [23] (anchor based)
 - RepPoints [87] (anchor free)

2.2. Bag of freebies

Usually, a conventional object detector is trained off-line. Therefore, researchers always like to take this advantage and develop better training methods which can make the object detector receive better accuracy without increasing the inference cost. We call these methods that only change the training strategy or only increase the training cost as “bag of freebies.” What is often adopted by object detection methods and meets the definition of bag of freebies is data augmentation. The purpose of data augmentation is to increase the variability of the input images, so that the designed object detection model has higher robustness to the images obtained from different environments. For examples, photometric distortions and geometric distortions are two commonly used data augmentation method and they definitely benefit the object detection task. In dealing with photometric distortion, we adjust the brightness, contrast, hue, saturation, and noise of an image. For geometric distortion, we add random scaling, cropping, flipping, and rotating.

The data augmentation methods mentioned above are all pixel-wise adjustments, and all original pixel information in the adjusted area is retained. In addition, some researchers engaged in data augmentation put their emphasis on simulating object occlusion issues. They have achieved good results in image classification and object detection. For example, random erase [100] and CutOut [11] can randomly select the rectangle region in an image and fill in a random or complementary value of zero. As for hide-and-seek [69] and grid mask [6], they randomly or evenly select multiple rectangle regions in an image and replace them to all zeros. If similar concepts are applied to feature maps, there are DropOut [71], DropConnect [80], and DropBlock [16] methods. In addition, some researchers have proposed the methods of using multiple images together to perform data augmentation. For example, MixUp [92] uses two images to multiply and superimpose with different coefficient ratios, and then adjusts the label with these superimposed ratios. As for CutMix [91], it is to cover the cropped image to rectangle region of other images, and adjusts the label according to the size of the mix area. In addition to the above mentioned methods, style transfer GAN [15] is also used for data augmentation, and such usage can effectively reduce the texture bias learned by CNN.

Different from the various approaches proposed above, some other bag of freebies methods are dedicated to solving the problem that the semantic distribution in the dataset may have bias. In dealing with the problem of semantic distribution bias, a very important issue is that there is a problem of data imbalance between different classes, and this problem is often solved by hard negative example mining [72] or online hard example mining [67] in two-stage object detector. But the example mining method is not applicable

to one-stage object detector, because this kind of detector belongs to the dense prediction architecture. Therefore Lin *et al.* [45] proposed focal loss to deal with the problem of data imbalance existing between various classes. Another very important issue is that it is difficult to express the relationship of the degree of association between different categories with the one-hot hard representation. This representation scheme is often used when executing labeling. The label smoothing proposed in [73] is to convert hard label into soft label for training, which can make model more robust. In order to obtain a better soft label, Islam *et al.* [33] introduced the concept of knowledge distillation to design the label refinement network.

The last bag of freebies is the objective function of Bounding Box (BBox) regression. The traditional object detector usually uses Mean Square Error (MSE) to directly perform regression on the center point coordinates and height and width of the BBox, i.e., $\{x_{center}, y_{center}, w, h\}$, or the upper left point and the lower right point, i.e., $\{x_{top_left}, y_{top_left}, x_{bottom_right}, y_{bottom_right}\}$. As for anchor-based method, it is to estimate the corresponding offset, for example $\{x_{center_offset}, y_{center_offset}, w_{offset}, h_{offset}\}$ and $\{x_{top_left_offset}, y_{top_left_offset}, x_{bottom_right_offset}, y_{bottom_right_offset}\}$. However, to directly estimate the coordinate values of each point of the BBox is to treat these points as independent variables, but in fact does not consider the integrity of the object itself. In order to make this issue processed better, some researchers recently proposed IoU loss [90], which puts the coverage of predicted BBox area and ground truth BBox area into consideration. The IoU loss computing process will trigger the calculation of the four coordinate points of the BBox by executing IoU with the ground truth, and then connecting the generated results into a whole code. Because IoU is a scale invariant representation, it can solve the problem that when traditional methods calculate the l_1 or l_2 loss of $\{x, y, w, h\}$, the loss will increase with the scale. Recently, some researchers have continued to improve IoU loss. For example, GIoU loss [65] is to include the shape and orientation of object in addition to the coverage area. They proposed to find the smallest area BBox that can simultaneously cover the predicted BBox and ground truth BBox, and use this BBox as the denominator to replace the denominator originally used in IoU loss. As for DIoU loss [99], it additionally considers the distance of the center of an object, and CIoU loss [99], on the other hand simultaneously considers the overlapping area, the distance between center points, and the aspect ratio. CIoU can achieve better convergence speed and accuracy on the BBox regression problem.

2.3. Bag of specials

For those plugin modules and post-processing methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection, we call them “bag of specials”. Generally speaking, these plugin modules are for enhancing certain attributes in a model, such as enlarging receptive field, introducing attention mechanism, or strengthening feature integration capability, etc., and post-processing is a method for screening model prediction results.

Common modules that can be used to enhance receptive field are SPP [25], ASPP [5], and RFB [47]. The SPP module was originated from Spatial Pyramid Matching (SPM) [39], and SPMs original method was to split feature map into several $d \times d$ equal blocks, where d can be $\{1, 2, 3, \dots\}$, thus forming spatial pyramid, and then extracting bag-of-word features. SPP integrates SPM into CNN and use max-pooling operation instead of bag-of-word operation. Since the SPP module proposed by He *et al.* [25] will output one dimensional feature vector, it is infeasible to be applied in Fully Convolutional Network (FCN). Thus in the design of YOLOv3 [63], Redmon and Farhadi improve SPP module to the concatenation of max-pooling outputs with kernel size $k \times k$, where $k = \{1, 5, 9, 13\}$, and stride equals to 1. Under this design, a relatively large $k \times k$ max-pooling effectively increase the receptive field of backbone feature. After adding the improved version of SPP module, YOLOv3-608 upgrades AP₅₀ by 2.7% on the MS COCO object detection task at the cost of 0.5% extra computation. The difference in operation between ASPP [5] module and improved SPP module is mainly from the original $k \times k$ kernel size, max-pooling of stride equals to 1 to several 3×3 kernel size, dilated ratio equals to k , and stride equals to 1 in dilated convolution operation. RFB module is to use several dilated convolutions of $k \times k$ kernel, dilated ratio equals to k , and stride equals to 1 to obtain a more comprehensive spatial coverage than ASPP. RFB [47] only costs 7% extra inference time to increase the AP₅₀ of SSD on MS COCO by 5.7%.

The attention module that is often used in object detection is mainly divided into channel-wise attention and point-wise attention, and the representatives of these two attention models are Squeeze-and-Excitation (SE) [29] and Spatial Attention Module (SAM) [85], respectively. Although SE module can improve the power of ResNet50 in the ImageNet image classification task 1% top-1 accuracy at the cost of only increasing the computational effort by 2%, but on a GPU usually it will increase the inference time by about 10%, so it is more appropriate to be used in mobile devices. But for SAM, it only needs to pay 0.1% extra calculation and it can improve ResNet50-SE 0.5% top-1 accuracy on the ImageNet image classification task. Best of all, it does not affect the speed of inference on the GPU at all.

In terms of feature integration, the early practice is to use skip connection [51] or hyper-column [22] to integrate low-level physical feature to high-level semantic feature. Since multi-scale prediction methods such as FPN have become popular, many lightweight modules that integrate different feature pyramid have been proposed. The modules of this sort include SFAM [98], ASFF [48], and BiFPN [77]. The main idea of SFAM is to use SE module to execute channel-wise level re-weighting on multi-scale concatenated feature maps. As for ASFF, it uses softmax as point-wise level re-weighting and then adds feature maps of different scales. In BiFPN, the multi-input weighted residual connections is proposed to execute scale-wise level re-weighting, and then add feature maps of different scales.

In the research of deep learning, some people put their focus on searching for good activation function. A good activation function can make the gradient more efficiently propagated, and at the same time it will not cause too much extra computational cost. In 2010, Nair and Hinton [56] propose ReLU to substantially solve the gradient vanish problem which is frequently encountered in traditional tanh and sigmoid activation function. Subsequently, LReLU [54], PReLU [24], ReLU6 [28], Scaled Exponential Linear Unit (SELU) [35], Swish [59], hard-Swish [27], and Mish [55], etc., which are also used to solve the gradient vanish problem, have been proposed. The main purpose of LReLU and PReLU is to solve the problem that the gradient of ReLU is zero when the output is less than zero. As for ReLU6 and hard-Swish, they are specially designed for quantization networks. For self-normalizing a neural network, the SELU activation function is proposed to satisfy the goal. One thing to be noted is that both Swish and Mish are continuously differentiable activation function.

The post-processing method commonly used in deep-learning-based object detection is NMS, which can be used to filter those BBoxes that badly predict the same object, and only retain the candidate BBoxes with higher response. The way NMS tries to improve is consistent with the method of optimizing an objective function. The original method proposed by NMS does not consider the context information, so Girshick *et al.* [19] added classification confidence score in R-CNN as a reference, and according to the order of confidence score, greedy NMS was performed in the order of high score to low score. As for soft NMS [1], it considers the problem that the occlusion of an object may cause the degradation of confidence score in greedy NMS with IoU score. The DIoU NMS [99] developers way of thinking is to add the information of the center point distance to the BBox screening process on the basis of soft NMS. It is worth mentioning that, since none of above post-processing methods directly refer to the captured image features, post-processing is no longer required in the subsequent development of an anchor-free method.

Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	1058 K	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	27.6 M	950 K	52 (26.0 FMA)	66
EfficientNet-B3 (ours)	512x512	1311x1311	12.0 M	668 K	11 (5.5 FMA)	26

3. Methodology

The basic aim is fast operating speed of neural network, in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP). We present two options of real-time neural networks:

- For GPU we use a small number of groups (1 - 8) in convolutional layers: CSPResNeXt50 / CSPDarknet53
- For VPU - we use grouped-convolution, but we refrain from using Squeeze-and-excitement (SE) blocks - specifically this includes the following models: EfficientNet-lite / MixNet [76] / GhostNet [21] / MobileNetV3

3.1. Selection of architecture

Our objective is to find the optimal balance among the input network resolution, the convolutional layer number, the parameter number ($\text{filter_size}^2 * \text{filters} * \text{channel} / \text{groups}$), and the number of layer outputs (filters). For instance, our numerous studies demonstrate that the CSPResNext50 is considerably better compared to CSPDarknet53 in terms of object classification on the ILSVRC2012 (ImageNet) dataset [10]. However, conversely, the CSPDarknet53 is better compared to CSPResNext50 in terms of detecting objects on the MS COCO dataset [46].

The next objective is to select additional blocks for increasing the receptive field and the best method of parameter aggregation from different backbone levels for different detector levels: e.g. FPN, PAN, ASFF, BiFPN.

A reference model which is optimal for classification is not always optimal for a detector. In contrast to the classifier, the detector requires the following:

- Higher input network size (resolution) – for detecting multiple small-sized objects
- More layers – for a higher receptive field to cover the increased size of input network
- More parameters – for greater capacity of a model to detect multiple objects of different sizes in a single image

Hypothetically speaking, we can assume that a model with a larger receptive field size (with a larger number of convolutional layers 3×3) and a larger number of parameters should be selected as the backbone. Table 1 shows the information of CSPResNeXt50, CSPDarknet53, and EfficientNet B3. The CSPResNext50 contains only 16 convolutional layers 3×3 , a 425×425 receptive field and 20.6 M parameters, while CSPDarknet53 contains 29 convolutional layers 3×3 , a 725×725 receptive field and 27.6 M parameters. This theoretical justification, together with our numerous experiments, show that CSPDarknet53 neural network is the optimal model of the two as the backbone for a detector.

The influence of the receptive field with different sizes is summarized as follows:

- Up to the object size - allows viewing the entire object
- Up to network size - allows viewing the context around the object
- Exceeding the network size - increases the number of connections between the image point and the final activation

We add the SPP block over the CSPDarknet53, since it significantly increases the receptive field, separates out the most significant context features and causes almost no reduction of the network operation speed. We use PANet as the method of parameter aggregation from different backbone levels for different detector levels, instead of the FPN used in YOLOv3.

Finally, we choose CSPDarknet53 backbone, SPP additional module, PANet path-aggregation neck, and YOLOv3 (anchor based) head as the architecture of YOLOv4.

In the future we plan to expand significantly the content of Bag of Freebies (BoF) for the detector, which theoretically can address some problems and increase the detector accuracy, and sequentially check the influence of each feature in an experimental fashion.

We do not use Cross-GPU Batch Normalization (CGBN or SyncBN) or expensive specialized devices. This allows anyone to reproduce our state-of-the-art outcomes on a conventional graphic processor e.g. GTX 1080Ti or RTX 2080Ti.

3.2. Selection of BoF and BoS

For improving the object detection training, a CNN usually uses the following:

- **Activations:** ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish
- **Bounding box regression loss:** MSE, IoU, GIoU, ClIoU, DIoU
- **Data augmentation:** CutOut, MixUp, CutMix
- **Regularization method:** DropOut, DropPath [36], Spatial DropOut [79], or DropBlock
- **Normalization of the network activations by their mean and variance:** Batch Normalization (BN) [32], Cross-GPU Batch Normalization (CGBN or SyncBN) [93], Filter Response Normalization (FRN) [70], or Cross-Iteration Batch Normalization (CBN) [89]
- **Skip-connections:** Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP)

As for training activation function, since PReLU and SELU are more difficult to train, and ReLU6 is specifically designed for quantization network, we therefore remove the above activation functions from the candidate list. In the method of regularization, the people who published DropBlock have compared their method with other methods in detail, and their regularization method has won a lot. Therefore, we did not hesitate to choose DropBlock as our regularization method. As for the selection of normalization method, since we focus on a training strategy that uses only one GPU, syncBN is not considered.

3.3. Additional improvements

In order to make the designed detector more suitable for training on single GPU, we made additional design and improvement as follows:

- We introduce a new method of data augmentation Mosaic, and Self-Adversarial Training (SAT)
- We select optimal hyper-parameters while applying genetic algorithms
- We modify some existing methods to make our design suitable for efficient training and detection - modified SAM, modified PAN, and Cross mini-Batch Normalization (CmBN)

Mosaic represents a new data augmentation method that mixes 4 training images. Thus 4 different contexts are



Figure 3: Mosaic represents a new method of data augmentation.

mixed, while CutMix mixes only 2 input images. This allows detection of objects outside their normal context. In addition, batch normalization calculates activation statistics from 4 different images on each layer. This significantly reduces the need for a large mini-batch size.

Self-Adversarial Training (SAT) also represents a new data augmentation technique that operates in 2 forward backward stages. In the 1st stage the neural network alters the original image instead of the network weights. In this way the neural network executes an adversarial attack on itself, altering the original image to create the deception that there is no desired object on the image. In the 2nd stage, the neural network is trained to detect an object on this modified image in the normal way.

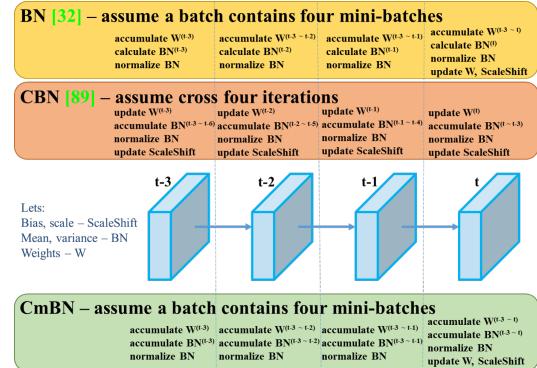


Figure 4: Cross mini-Batch Normalization.

CmBN represents a CBN modified version, as shown in Figure 4, defined as Cross mini-Batch Normalization (CmBN). This collects statistics only between mini-batches within a single batch.

We modify SAM from spatial-wise attention to point-wise attention, and replace shortcut connection of PAN to concatenation, as shown in Figure 5 and Figure 6, respectively.

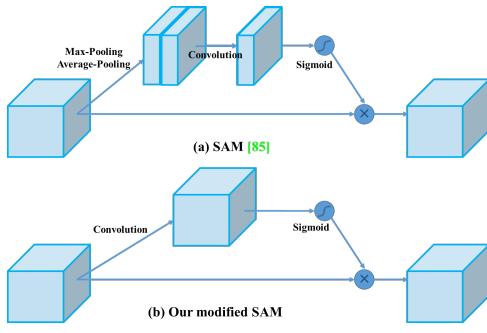


Figure 5: Modified SAM.

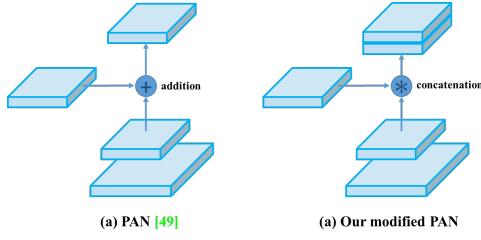


Figure 6: Modified PAN.

3.4. YOLOv4

In this section, we shall elaborate the details of YOLOv4.

YOLOv4 consists of:

- Backbone: CSPDarknet53 [81]
- Neck: SPP [25], PAN [49]
- Head: YOLOv3 [63]

YOLO v4 uses:

- Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing
- Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multi-input weighted residual connections (MiWRC)
- Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler [52], Optimal hyper-parameters, Random training shapes
- Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

4. Experiments

We test the influence of different training improvement techniques on accuracy of the classifier on ImageNet (ILSVRC 2012 val) dataset, and then on the accuracy of the detector on MS COCO (test-dev 2017) dataset.

4.1. Experimental setup

In ImageNet image classification experiments, the default hyper-parameters are as follows: the training steps is 8,000,000; the batch size and the mini-batch size are 128 and 32, respectively; the polynomial decay learning rate scheduling strategy is adopted with initial learning rate 0.1; the warm-up steps is 1000; the momentum and weight decay are respectively set as 0.9 and 0.005. All of our BoS experiments use the same hyper-parameter as the default setting, and in the BoF experiments, we add an additional 50% training steps. In the BoF experiments, we verify MixUp, CutMix, Mosaic, Bluring data augmentation, and label smoothing regularization methods. In the BoS experiments, we compared the effects of LReLU, Swish, and Mish activation function. All experiments are trained with a 1080 Ti or 2080 Ti GPU.

In MS COCO object detection experiments, the default hyper-parameters are as follows: the training steps is 500,500; the step decay learning rate scheduling strategy is adopted with initial learning rate 0.01 and multiply with a factor 0.1 at the 400,000 steps and the 450,000 steps, respectively; The momentum and weight decay are respectively set as 0.9 and 0.0005. All architectures use a single GPU to execute multi-scale training in the batch size of 64 while mini-batch size is 8 or 4 depend on the architectures and GPU memory limitation. Except for using genetic algorithm for hyper-parameter search experiments, all other experiments use default setting. Genetic algorithm used YOLOv3-SPP to train with GIoU loss and search 300 epochs for min-val 5k sets. We adopt searched learning rate 0.00261, momentum 0.949, IoU threshold for assigning ground truth 0.213, and loss normalizer 0.07 for genetic algorithm experiments. We have verified a large number of BoF, including grid sensitivity elimination, mosaic data augmentation, IoU threshold, genetic algorithm, class label smoothing, cross mini-batch normalization, self-adversarial training, cosine annealing scheduler, dynamic mini-batch size, DropBlock, Optimized Anchors, different kind of IoU losses. We also conduct experiments on various BoS, including Mish, SPP, SAM, RFB, BiFPN, and Gaussian YOLO [8]. For all experiments, we only use one GPU for training, so techniques such as syncBN that optimizes multiple GPUs are not used.

4.2. Influence of different features on Classifier training

First, we study the influence of different features on classifier training; specifically, the influence of Class label smoothing, the influence of different data augmentation techniques, bilateral blurring, MixUp, CutMix and Mosaic, as shown in Figure 7, and the influence of different activations, such as Leaky-ReLU (by default), Swish, and Mish.

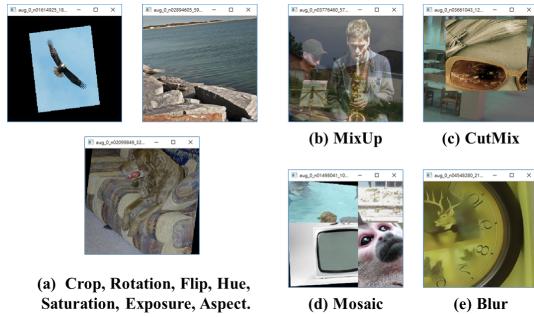


Figure 7: Various method of data augmentation.

In our experiments, as illustrated in Table 2, the classifier's accuracy is improved by introducing the features such as: CutMix and Mosaic data augmentation, Class label smoothing, and Mish activation. As a result, our BoF-backbone (Bag of Freebies) for classifier training includes the following: CutMix and Mosaic data augmentation and Class label smoothing. In addition we use Mish activation as a complementary option, as shown in Table 2 and Table 3.

Table 2: Influence of BoF and Mish on the CSPResNeXt-50 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
✓							77.9%	94.0%
	✓						77.2%	94.0%
		✓					78.0%	94.3%
			✓				78.1%	94.5%
				✓			77.5%	93.8%
					✓		78.1%	94.4%
						✓	64.5%	86.0%
✓	✓		✓				78.9%	94.5%
✓	✓		✓			✓	78.5%	94.8%
						✓	79.8%	95.2%

Table 3: Influence of BoF and Mish on the CSPDarknet-53 classifier accuracy.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
✓	✓						77.2%	93.6%
✓	✓				✓		77.8%	94.4%
						✓	78.7%	94.8%

4.3. Influence of different features on Detector training

Further study concerns the influence of different Bag-of-Freebies (BoF-detector) on the detector training accuracy, as shown in Table 4. We significantly expand the BoF list through studying different features that increase the detector accuracy without affecting FPS:

- S: Eliminate grid sensitivity the equation $b_x = \sigma(t_x) + c_x$, $b_y = \sigma(t_y) + c_y$, where c_x and c_y are always whole numbers, is used in YOLOv3 for evaluating the object coordinates, therefore, extremely high t_x absolute values are required for the b_x value approaching the c_x or $c_x + 1$ values. We solve this problem through multiplying the sigmoid by a factor exceeding 1.0, so eliminating the effect of grid on which the object is undetectable.
- M: Mosaic data augmentation - using the 4-image mosaic during training instead of single image
- IT: IoU threshold - using multiple anchors for a single ground truth IoU (truth, anchor) $>$ IoU_threshold
- GA: Genetic algorithms - using genetic algorithms for selecting the optimal hyperparameters during network training on the first 10% of time periods
- LS: Class label smoothing - using class label smoothing for sigmoid activation
- CBN: CmBN - using Cross mini-Batch Normalization for collecting statistics inside the entire batch, instead of collecting statistics inside a single mini-batch
- CA: Cosine annealing scheduler - altering the learning rate during sinusoid training
- DM: Dynamic mini-batch size - automatic increase of mini-batch size during small resolution training by using Random training shapes
- OA: Optimized Anchors - using the optimized anchors for training with the 512x512 network resolution
- GIoU, CIoU, DIoU, MSE - using different loss algorithms for bounded box regression

Further study concerns the influence of different Bag-of-Specials (BoS-detector) on the detector training accuracy, including PAN, RFB, SAM, Gaussian YOLO (G), and ASFF, as shown in Table 5. In our experiments, the detector gets best performance when using SPP, PAN, and SAM.

Table 4: Ablation Studies of Bag-of-Freebies. (CSPResNeXt50-PANet-SPP, 512x512).

S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP ₅₀	AP ₇₅
✓									MSE	38.0%	60.0%	40.8%
	✓								MSE	37.7%	59.9%	40.5%
		✓							MSE	39.1%	61.8%	42.0%
			✓						MSE	36.9%	59.7%	39.4%
				✓					MSE	38.9%	61.7%	41.9%
					✓				MSE	33.0%	55.4%	35.4%
						✓			MSE	38.4%	60.7%	41.3%
							✓		MSE	38.7%	60.7%	41.9%
								✓	MSE	35.3%	57.2%	38.0%
									GIoU	39.4%	59.4%	42.5%
									DIoU	39.1%	58.8%	42.1%
									CloU	39.6%	59.2%	42.6%
									CloU	41.5%	64.0%	44.8%
									CloU	36.1%	56.5%	38.4%
									MSE	40.3%	64.0%	43.1%
									GIoU	42.4%	64.4%	45.9%
									CloU	42.4%	64.4%	45.9%

Table 5: Ablation Studies of Bag-of-Specials. (Size 512x512).

Model	AP	AP ₅₀	AP ₇₅
CSPResNeXt50-PANet-SPP	42.4%	64.4%	45.9%
CSPResNeXt50-PANet-SPP-RFB	41.8%	62.7%	45.1%
CSPResNeXt50-PANet-SPP-SAM	42.7%	64.6%	46.3%
CSPResNeXt50-PANet-SPP-SAM-G	41.6%	62.7%	45.0%
CSPResNeXt50-PANet-SPP-ASFF-RFB	41.1%	62.6%	44.4%

4.4. Influence of different backbones and pre-trained weightings on Detector training

Further on we study the influence of different backbone models on the detector accuracy, as shown in Table 6. We notice that the model characterized with the best classification accuracy is not always the best in terms of the detector accuracy.

First, although classification accuracy of CSPResNeXt50 models trained with different features is higher compared to CSPDarknet53 models, the CSPDarknet53 model shows higher accuracy in terms of object detection.

Second, using BoF and Mish for the CSPResNeXt50 classifier training increases its classification accuracy, but further application of these pre-trained weightings for detector training reduces the detector accuracy. However, using BoF and Mish for the CSPDarknet53 classifier training increases the accuracy of both the classifier and the detector which uses this classifier pre-trained weightings. The net result is that backbone CSPDarknet53 is more suitable for the detector than for CSPResNeXt50.

We observe that the CSPDarknet53 model demonstrates a greater ability to increase the detector accuracy owing to various improvements.

Table 6: Using different classifier pre-trained weightings for detector training (all other training parameters are similar in all models).

Model (with optimal setting)	Size	AP	AP ₅₀	AP ₇₅
CSPResNeXt50-PANet-SPP	512x512	42.4	64.4	45.9
CSPResNeXt50-PANet-SPP (BoF-backbone)	512x512	42.3	64.3	45.7
CSPResNeXt50-PANet-SPP (BoF-backbone + Mish)	512x512	42.3	64.2	45.8
CSPDarknet53-PANet-SPP (BoF-backbone)	512x512	42.4	64.5	46.0
CSPDarknet53-PANet-SPP (BoF-backbone + Mish)	512x512	43.0	64.9	46.5

4.5. Influence of different mini-batch size on Detector training

Finally, we analyze the results obtained with models trained with different mini-batch sizes, and the results are shown in Table 7. From the results shown in Table 7, we found that after adding BoF and BoS training strategies, the mini-batch size has almost no effect on the detector's performance. This result shows that after the introduction of BoF and BoS, it is no longer necessary to use expensive GPUs for training. In other words, anyone can use only a conventional GPU to train an excellent detector.

Table 7: Using different mini-batch size for detector training.

Model (without OA)	Size	AP	AP ₅₀	AP ₇₅
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 4)	608	37.1	59.2	39.9
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 8)	608	38.4	60.6	41.6
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 4)	512	41.6	64.1	45.0
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 8)	512	41.7	64.2	45.2

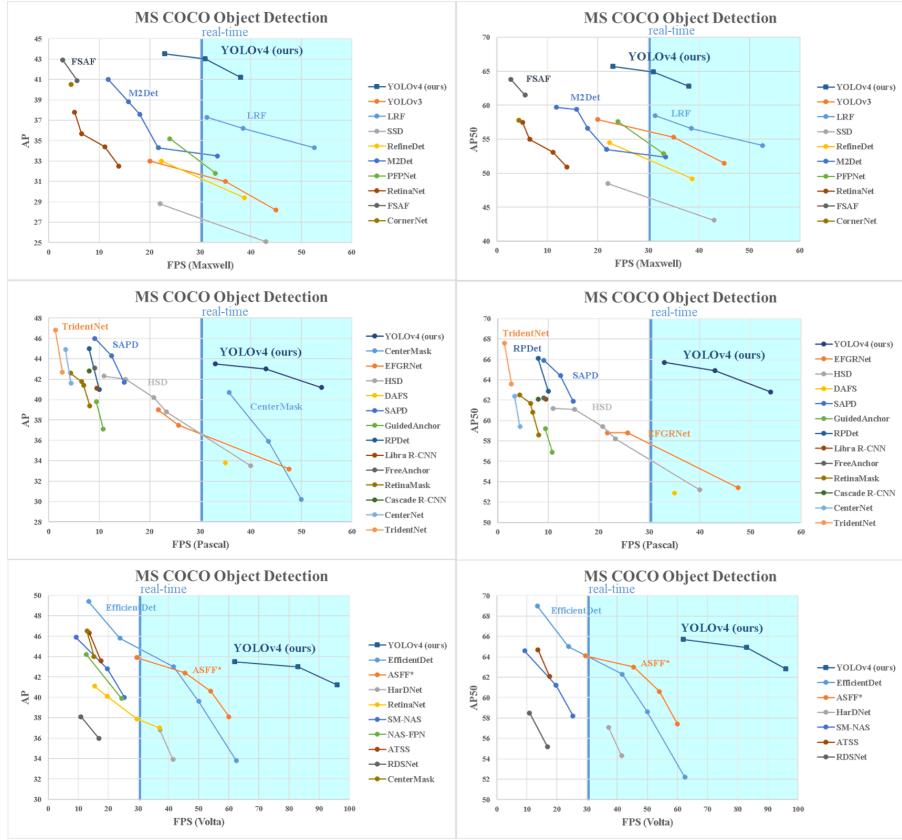


Figure 8: Comparison of the speed and accuracy of different object detectors. (Some articles stated the FPS of their detectors for only one of the GPUs: Maxwell/Pascal/Volta)

5. Results

Comparison of the results obtained with other state-of-the-art object detectors are shown in Figure 8. Our YOLOv4 are located on the Pareto optimality curve and are superior to the fastest and most accurate detectors in terms of both speed and accuracy.

Since different methods use GPUs of different architectures for inference time verification, we operate YOLOv4 on commonly adopted GPUs of Maxwell, Pascal, and Volta architectures, and compare them with other state-of-the-art methods. Table 8 lists the frame rate comparison results of using Maxwell GPU, and it can be GTX Titan X (Maxwell) or Tesla M40 GPU. Table 9 lists the frame rate comparison results of using Pascal GPU, and it can be Titan X (Pascal), Titan Xp, GTX 1080 Ti, or Tesla P100 GPU. As for Table 10, it lists the frame rate comparison results of using Volta GPU, and it can be Titan Volta or Tesla V100 GPU.

6. Conclusions

We offer a state-of-the-art detector which is faster (FPS) and more accurate (MS COCO AP_{50...95} and AP₅₀) than all available alternative detectors. The detector described can be trained and used on a conventional GPU with 8-16 GB-VRAM this makes its broad use possible. The original concept of one-stage anchor-based detectors has proven its viability. We have verified a large number of features, and selected for use such of them for improving the accuracy of both the classifier and the detector. These features can be used as best-practice for future studies and developments.

7. Acknowledgements

The authors wish to thank Glenn Jocher for the ideas of Mosaic data augmentation, the selection of hyper-parameters by using genetic algorithms and solving the grid sensitivity problem <https://github.com/ultralytics/yolov3>.

附录 B 外文文献译文

YOLOv4: 最佳目标检测速度和精度

Alexey Bochkovskiy^{*}
alexeyab84@gmail.com

Chien-Yao Wang^{*}
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

摘要

许多特性据称可以提高卷积神经网络(CNN)的精度。为了验证这些特性是否有效，需要在大型数据集上对各种特性组合进行实际测试，并从理论上证明测试结果的合理性。一些特性仅适用于特定模型、特定问题或小型数据集；而另一些特性（例如批量归一化和残差连接）则适用于大多数模型、任务和数据集。我们假设加权残差连接(WRC)、跨阶段部分连接(CSP)、跨小型批归一化(CmBN)、自我对抗训练(SAT)和Mish激活函数等属于这种通用特性。我们使用了以下新特性：WRC、CSP、CmBN、SAT、Mish激活函数、Mosaic数据增强、CmBN、DropBlock正则化和CIoU损失函数，并通过组合其中的一些特性在MS COCO数据集上实现了最先进的结果：43.5% AP(65.7% AP50)，在Tesla V100上的实时速度约为65 FPS。源代码在<https://github.com/AlexeyAB/darknet>处获得。

1. 引言

目前大多数基于卷积神经网络(CNN)的目标检测器在很大程度上仅适用于推荐系统。例如，通过城市摄像头寻找免费停车位是由速度慢但准确的模型执行的，而防撞预警则依赖于速度快但不太准确的模型。提高实时目标检测器的精度可以使其不仅用于生成提示的推荐系统，还可以用于独立流程管理并减少人工干预。在常规图形处理单元(GPU)上运行实时目标检测器使其能够以实惠的价格大规模使用。然而，最准确的现代神经网络无法实时运行，并且需要大量GPU才能使用大批量数据进行训练。我们通过创建一个可以在常规GPU上实时运行的CNN来解决这些问题，并且该网络的训练只需要一台常规GPU即可完成。

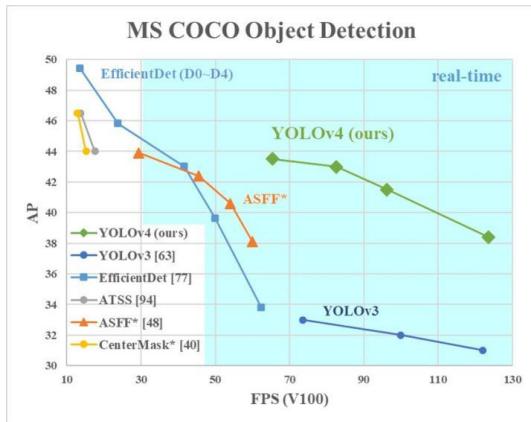


图 1：所建议的YOLOv4与其他最先进的对象检测器之间的比较。在性能相当的情况下，YOLOv4的运行速度比EfficientDet快两倍。分别比YOLOv3的AP和FPS提高了10%和12%。

这项工作的主要目标是设计一个运行速度快的对象检测器，适用于生产系统并针对并行计算进行优化，而非追求低计算量理论指标(BFLOP)。我们希望设计的对象检测器易于训练和使用。例如，任何使用常规GPU训练和测试的人都可以像图1中展示的YOLOv4结果那样，实现实时、高质量且令人信服的对象检测。我们做出的贡献可以总结如下：

1. 开发了一个高效强大的对象检测模型。它使每个人都可以在一块1080 Ti或2080 Ti GPU上训练出一个超快速且准确的对象检测器。
2. 验证了最先进的“免费赠品包”和“特殊赠品包”方法在对象检测器训练过程中的影响。
3. 修改了最先进的方法，使其更有效且适用于单GPU训练，例如CBN[89]、PAN[49]、SAM[85]等。

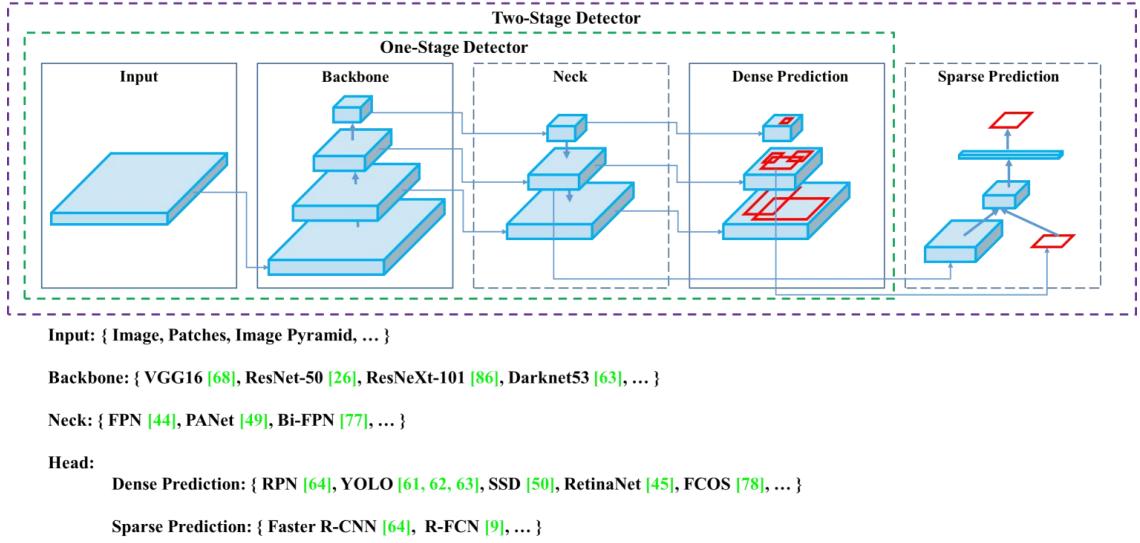


图 2: 目标监测器

2. 相关工作

2.1. 目标监测模型

现代目标检测器通常由两部分组成：主干网络和检测头。主干网络用于提取图像特征，通常预训练于 ImageNet 数据集上。在 GPU 平台上，常用的主干网络包括 VGG [68]、ResNet [26]、ResNeXt [86] 或 DenseNet [30] 等。在 CPU 平台上，则有 SqueezeNet [31]、MobileNet [28, 66, 27, 74] 或 ShuffleNet [97, 53] 等。检测头用于预测目标的类别和边界框。检测头通常分为两类：单阶段和两阶段。最具代表性的两阶段目标检测器是 R-CNN [19] 系列，包括 fast R-CNN [18]、faster R-CNN [64]、R-FCN [9] 和 Libra R-CNN [58]。还有一些特殊的两阶段检测器，例如 RepPoints [87]，它摒弃了锚框机制。最具代表性的单阶段目标检测器是 YOLO [61, 62, 63]、SSD [50] 和 RetinaNet [45]。近年来，一些无锚框的单阶段目标检测器也得到发展，例如 CenterNet [13]、CornerNet [37, 38] 和 FCOS [78] 等。近年来开发的目标检测器经常会在主干网络和检测头之间加入一些额外的层，称为颈部（neck）。这些层通常用于融合来自不同阶段的特征图。常见的颈部结构包括 Feature Pyramid Network (FPN) [44]、Path Aggregation Network (PAN) [49]、BiFPN [77] 和 NAS-FPN [17] 等。除了以上提到的模型，一些研究人员还致力于直接构建新的用于目标检测的主干网络 (DetNet [43]、DetNAS [7]) 或是整个模型 (SpineNet [12]、HitDetector [20])。

综上所述，一个普通的目标检测器由以下几个部分组成：

- **输入：**图像、图像块、图像金字塔
- **主干网络：** VGG16 [68]、ResNet-50 [26]、SpineNet [12]、EfficientNet-B0/B7 [75]、CSPResNeXt50 [81]、CSPDarknet53 [81]
- **颈部：**
 - **附加模块：** SPP [25]、ASPP [5]、RFB [47]、SAM [85]
 - **路径聚合模块：** FPN [44]、PAN [49]、NAS-FPN [17]、全连接 FPN、BiFPN [77]、ASFF [48]、SFAM [98]
- **检测头：**
 - **稠密预测(单阶段):**
 - ◆ RPN [64]、SSD [50]、YOLO [61]、RetinaNet [45] (基于锚框)
 - ◆ CornerNet [37]、CenterNet [13]、MatrixNet [60]、FCOS [78] (无锚框)
 - **稀疏预测(两阶段):**
 - ◆ Faster R-CNN [64]、R-FCN [9]、Mask R-CNN [23] (基于锚框)
 - ◆ RepPoints [87] (无锚框)

2.2. 免费赠品包

通常情况下，物体检测器采用离线训练的方式。因此，研究人员总是倾向于开发更好的训练方法，这些方法可以在不增加推理成本的情况下提高物体检测器的精度。我们将仅改变训练策略或仅增加训练成本的方法称为“免费赠品包”。数据增强是物体检测方法经常采用且符合免费赠品包定义的一种方法。数据增强的目的是增加输入图像的多样性，从而使设计的物体检测模型对来自不同环境的图像具有更高的鲁棒性。例如，光度畸变和几何畸变是两种常用的数据增强方法，它们肯定能对物体检测任务带来益处。在处理光度畸变时，我们会调整图像的亮度、对比度、色调、饱和度和噪声。对于几何畸变，我们会加入随机缩放、裁剪、翻转和旋转操作。

上面提到的数据增强方法都是逐像素的调整，调整区域的所有原始像素信息都会被保留。此外，一些从事数据增强的研究人员将重点放在模拟物体遮挡问题上。他们在图像分类和物体检测方面取得了良好的效果。例如，随机擦除 [100] 和 CutOut [11] 可以随机选择图像中的矩形区域，并用随机值或补零值填充。Hide-and-Seek [69] 和网格掩码 [6] 则会随机或均匀地选择图像中的多个矩形区域，并替换为全零值。如果将类似的概念应用于特征图，则有 Dropout [71]、DropConnect [80] 和 DropBlock [16] 方法。此外，一些研究人员提出了使用多张图像一起进行数据增强的方法。例如，MixUp [92] 使用两张图像以不同的系数比率进行乘法叠加，然后根据这些叠加比率调整标签。CutMix [91] 则是将裁剪的图像覆盖到其他图像的矩形区域，并根据混合区域的大小调整标签。除了上述方法外，风格迁移 GAN [15] 也被用于数据增强，这种用法可以有效减少 CNN 学习到的纹理偏差。

与上面提到的各种方法不同，其他一些免费赠品包方法致力于解决数据集中的语义分布可能存在偏差的问题。在处理语义分布偏差问题时，一个非常重要的问题是不同类别之间存在数据不平衡的问题，这个问题通常可以通过两阶段目标检测器中的困难负样本挖掘 [72] 或在线困难样本挖掘 [67] 来解决。但是，示例挖掘方法并不适用于单阶段目标检测器，因为这种检测器属于密集预测架构。因此，Lin 等人 [45] 提出了焦距损失 (focal loss) 来处理不同类别之间存在的数据不平衡问题。另一个非常重要的问题是，使用 one-hot 硬表示很难表达不同类别之间关联程度的关系。这种表示方案在执行标记时经常使用。论文 [73] 中提出的标签平滑化将硬标签转换为软标签进行训练，可以使模型更加鲁棒。为了获得更好的软标签，Islam 等人 [33] 引入了知识蒸馏的概念来设计标签细化网络。

最后一个免费赠品包是边界框 (BBox) 回归的目标函数。传统的目标检测器通常使用均方误差 (MSE) 直接对 BBox 的中心点坐标和宽高进行回归，即 $\{x_{center}, y_{center}, w, h\}$ 或左上角点和右下角点，即 $\{x_{top_left}, y_{top_left}, x_{bottom_right}, y_{bottom_right}\}$ 。对于基于锚框的方法，则是估计对应的偏移量，例如 $\{x_{center_offset}, y_{center_offset}, w_{offset}, h_{offset}\}$ 和 $\{x_{top_left_offset}, y_{top_left_offset},$

$x_{bottom_right_offset}, y_{bottom_right_offset}\}$ 。然而，直接估计 BBox 的每个点的坐标值，是将这些点作为独立变量对待，并没有考虑物体本身的完整性。

为了更好地处理这个问题，一些研究人员最近提出了 IoU 损失 [90]，它将预测的 BBox 区域和 ground truth BBox 区域的覆盖率纳入考量。IoU 损失的计算过程会通过与 ground truth 执行 IoU 计算来触发 BBox 的四个坐标点的计算，然后将生成的结果连接成一个整体。由于 IoU 是尺度不变的表示，因此它可以解决传统方法在计算 $\{x, y, w, h\}$ 的 L1 或 L2 损失时，损失会随着尺度增加的问题。

最近，一些研究人员又对 IoU 损失进行了改进。例如，GIoU 损失 [65] 除了考虑覆盖面积外，还加入了物体的形状和方向。他们提出寻找一个最小的面积 BBox，可以同时覆盖预测的 BBox 和 ground truth BBox，并使用这个 BBox 作为分母来替换 IoU 损失中原本使用的分母。DIoU 损失 [99] 则额外考虑了物体中心的距离，而 CIoU 损失 [99] 则同时考虑了重叠面积、中心点间距和长宽比。CIoU 可以使 BBox 回归问题取得更好的收敛速度和精度。

2.3. 特殊赠品包

对于那些仅略微增加推理成本但可以显著提高目标检测精度的插件模块和后处理方法，我们称之为“特殊赠品包”。一般来说，这些插件模块用于增强模型的某些属性，例如扩大感受野、引入注意力机制或加强特征融合能力等，而后处理是一种筛选模型预测结果的方法。

可用于扩展感受野的常用模块包括 SPP [25]、ASPP [5] 和 RFB [47]。SPP 模块源自空间金字塔匹配 (SPM) [39]，SPM 的原始方法是将特征图分割成几个大小为 $d \times d$ 的相等块，其中 d 可以取 $\{1, 2, 3, \dots\}$ ，从而形成空间金字塔，然后提取词袋特征。SPP 将 SPM 集成到 CNN 中，并使用最大池化操作代替词袋操作。He 等人 [25] 提出的 SPP 模块会输出一维特征向量，因此无法直接应用于全卷积网络 (FCN)。因此在 YOLOv3 [63] 的设计中，Redmon 和 Farhadi 改进了 SPP 模块，将其变为内核大小为 $k \times k$ 的最大池化输出的连接，其

中 $k = \{1, 5, 9, 13\}$, 步长等于 1。在此设计下, 相对较大的 $k \times k$ 最大池化有效地增加了主干网络特征的感受野。加入改进版的 SPP 模块后, YOLOv3-608 在 MS COCO 目标检测任务上的 AP50 提升了 2.7%, 但额外增加了 0.5% 的计算量。改进后的 SPP 模块与 ASPP [5] 模块之间的操作差异主要在于原始内核大小 $k \times k$, 步长为 1 的最大池化改成了多个 3×3 的内核大小, 空洞率等于 k , 步长等于 1 的膨胀卷积操作。RFB 模块则使用多个内核大小为 $k \times k$ 、空洞率等于 k 、步长等于 1 的膨胀卷积来获得比 ASPP 更全面的空间覆盖范围。RFB [47] 只需额外增加 7% 的推理时间, 即可将 SSD 在 MS COCO 上的 AP50 提高 5.7%。

目标检测中常用的注意力模块主要分为通道注意力和点注意力, 这两种注意力模型的代表分别为 Squeeze-and-Excitation (SE) [29] 和空间注意力模块 (SAM) [85]。虽然 SE 模块可以在 ImageNet 图像分类任务上将 ResNet50 的 top-1 准确率提高 1%, 并且仅增加 2% 的计算量, 但是在 GPU 上它通常会增加大约 10% 的推理时间, 因此更适合用于移动设备。而 SAM 只需增加 0.1% 的额外计算量, 就可以在 ImageNet 图像分类任务上将 ResNet50-SE 的 top-1 准确率提高 0.5%, 而且它完全不影响在 GPU 上的推理速度。

在特征融合方面, 早期做法是使用跳跃连接 [51] 或超列 [22] 将低层次的物理特征与高级语义特征进行融合。随着 FPN 等多尺度预测方法的普及, 许多用于整合不同特征金字塔的轻量级模块被提出。这类模块包括 SFAM [98]、ASFF [48] 和 BiFPN [77]。SFAM 的主要思想是使用 SE 模块对多尺度连接的特征图执行通道级的重新加权。对于 ASFF, 它使用 softmax 作为点级的重新加权, 然后添加不同尺度的特征图。在 BiFPN 中, 引入了多输入加权残差连接来执行尺度级的重新加权, 然后添加不同尺度的特征图。

在深度学习的研究中, 一些人将重点放在寻找良好的激活函数上。良好的激活函数可以使梯度更有效地传播, 同时不会带来太大的额外计算成本。2010 年, Nair 和 Hinton [56] 提出了 ReLU 函数, 有效地解决了传统 tanh 和 sigmoid 激活函数经常遇到的梯度消失问题。随后, 为了解决梯度消失问题, 又陆续提出了 LReLU [54]、PReLU [24]、ReLU6 [28]、缩放指数线

性单元 (SELU) [35]、Swish [59]、hard-Swish [27] 和 Mish [55] 等激活函数。LReLU 和 PReLU 的主要目的是解决 ReLU 在输出小于零时梯度为零的问题。ReLU6 和 hard-Swish 则专门为量化网络而设计。SELU 激活函数的提出是为了实现神经网络的自归一化。值得注意的是, Swish 和 Mish 都是连续可微的激活函数。

深度学习目标检测常用的后处理方法是 NMS, 它可以用于过滤那些错误预测相同物体的边界框, 只保留具有较高响应的候选边界框。NMS 尝试改进的方式与优化目标函数的方法是一致的。NMS 最初提出的方法没有考虑上下文信息, 因此 Girshick 等人 [19] 在 R-CNN 中加入了分类置信度得分作为参考, 并根据置信度得分的高低顺序进行贪婪 NMS。Soft NMS [1] 则考虑了在贪婪 NMS 中使用 IoU 得分可能会导致物体遮挡导致置信度得分降低的问题。DIoU NMS [99] 的开发者在软 NMS 的基础上, 加入了中心点距离信息到边界框筛选过程中。值得一提的是, 由于以上后处理方法都不直接引用捕获的图像特征, 因此后发展的无锚框方法不再需要后处理。

3. 方法

在生产系统中, 我们优化的主要目标是神经网络的快速运行速度和并行计算, 而不是理论计算量指标 (BFLOPs)。下面我们介绍两种实时神经网络的方案:

- 针对 GPU: 我们在卷积层中使用少量的组 (1 - 8), 例如 CSPResNeXt50 / CSPDarknet53
- 针对 VPU: 我们使用分组卷积, 但避免使用 Squeeze-and-Excitement (SE) 模块, 具体包括以下模型: EfficientNet-lite / MixNet [76] / GhostNet [21] / MobileNetV3

3.1. 模型结构选择

我们的目标是在输入网络分辨率、卷积层数量、参数数量 (滤波器大小 * 过滤器 * 通道 / 组数) 和层输出数量 (滤波器) 之间找到最佳平衡。例如, 我们的大量研究表明, 在 ILSVRC2012 (ImageNet) 数据集上进行对象分类时, CSPResNext50 的性能明显优于 CSPDarknet53 [10]。然而, 另一方面, 在 MS COCO 数据集 [46] 上进行对象检测时, CSPDarknet53 的性能优

表1: 用于图像分类的神经网络参数大小

骨干模型	网络输入分辨率	感受野大小	参数量	每层输出的平均大小(WxHxC) (512x512网络输入分辨率)	BFLOPs (512x512网络输入分辨率)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	1058 K	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	27.6 M	950 K	52 (26.0 FMA)	66
EfficientNet-B3 (ours)	512x512	1311x1311	12.0 M	668 K	11 (5.5 FMA)	26

于 CSPResNext50。

下一个目标是选择用于增加感受野的附加模块，以及用于不同检测器层级来自不同主干网络层级参数聚合的最佳方法：例如 FPN、PAN、ASFF、BIFPN。

仅仅在分类任务上表现最佳的模型并不总是适用于检测任务。与分类器相比，检测器需要以下几点：

- 更高的输入网络尺寸（分辨率） - 用于检测多个小型物体
- 更多的层 - 用于更大的感受野以覆盖增加的输入网络尺寸
- 更多的参数 - 用于模型更大的容量，以便在单个图像中检测不同大小的多个物体

从理论上讲，我们可以假设应该选择具有更大感受野大小（具有更多个 3×3 的卷积层）和更多参数的模型作为主干网络。表 1 显示了 CSPResNeXt50、CSPDarknet53 和 EfficientNet B3 的信息。CSPResNeXt50 仅包含 16 个 3×3 的卷积层、 425×425 的感受野和 20.6 M 参数，而 CSPDarknet53 包含 29 个 3×3 的卷积层、 725×725 的感受野和 27.6 M 参数。这种理论依据与我们的大量实验一起表明，CSPDarknet53 神经网络作为检测器的主干网络比两者更优。

不同大小的感受野的影响总结如下：

- 小于等于物体大小 - 允许查看整个物体
- 小于等于网络大小 - 允许查看物体周围的上下文
- 超过网络大小 - 增加图像点和最终激活之间连接的数量

我们在 CSPDarknet53 上添加了 SPP 模块，因为它可以显着增加感受野，分离出最重要的上下文特征，并且几乎不会降低网络运行速度。我们使用 PANet 作为来自不同主干网络层级到不同检测器层级的参数聚合方法，而不是 YOLOv3 中使用的 FPN。

最后，我们选择 CSPDarknet53 主干网络、SPP 附加模块、PANet 路径聚合颈部和 YOLOv3（基于锚框）头作为 YOLOv4 的结构。

未来我们计划大幅扩展免费赠品包 (BoF) 的内容，理论上可以解决一些问题并提高检测器精度，并依次以实验方式检查每个特征的影响。

我们不使用跨 GPU 批归一化 (CGBN 或 SyncBN) 或昂贵的专用设备。这允许任何人在常规图形处理器（例如 GTX 1080Ti 或 RTX 2080Ti）上重现我们最先进的成果。

3.2. BoF 和 BoS 的选择

CNN 通常使用以下组件以改善目标检测训练效果：

- **激活函数**：ReLU、leaky-ReLU、parametric-ReLU、ReLU6、SELU、Swish 或 Mish
- **边框回归损失**：MSE、IoU、GIoU、CIoU、DIOU
- **数据增强**：CutOut、MixUp、CutMix
- **正则化方法**：DropOut、DropPath [36]、Spatial DropOut [79] 或 DropBlock
- **网络激活值的归一化**：Batch Normalization (BN) [32]、Cross-GPU Batch Normalization (CGBN 或 SyncBN) [93]、Filter Response Normalization (FRN) [70] 或 Cross-Iteration Batch Normalization (CBN) [89]
- **跳跃连接**：残差连接、加权残差连接、多输入加权残差连接或 Cross stage partial connections (CSP)

由于 PReLU 和 SELU 训练难度更大，ReLU6 专为量化网络设计，因此我们从候选列表中删除了这些激活函数。对于正则化方法，DropBlock 的发布者详细对比了他们的方法与其他方法，并且他们的正则化方法在对比中表现良好。因此，我们毫不犹豫地选择 DropBlock 作为正则化方法。由于我们专注于仅使用单个 GPU 的训练策略，因此在归一化方法的选择上不考虑 SyncBN。

3.3. 额外的改进

为了使设计的检测器更适合于单个 GPU 训练，我们进行了以下额外的设计和改进：

- 我们引入了一种新的数据增强方法 Mosaic 和对抗生成训练 (SAT)
- 我们在应用遗传算法时选择最佳的超参数
- 我们修改了一些现有方法以使我们的设计适用于高效训练和检测 - 修改后的 SAM、修改后的 PAN 和跨迷你批归一化 (CmBN)

Mosaic 是一种新的数据增强方法，它混合了 4 张训练图像。因此，可以混合 4 种不同的上下文，而 CutMix 只混合 2 张输入图像。这允许检测超出其正常上下文的对象。此外，批量归一化会计算来自每个层上 4 张不同图像的激活统计信息。这大大减少了对大批量大小的需求。

对抗生成训练 (SAT) 也代表了一种新的数据增强技术，它分两个前向后向阶段进行操作。在第一阶段，神经网络不是改变网络权重，而是改变原始图像。这样，神经网络对自己执行对抗攻击，改变原始图像，使其看

起来图像上不存在目标物体。在第二阶段，神经网络以正常方式训练该修改后的图像来检测物体。



图 3: Mosaic 代表了一种新的数据增强方法

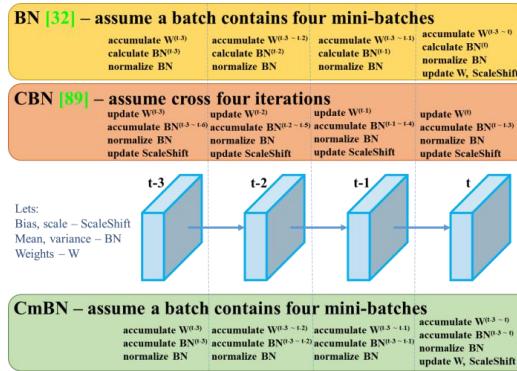


图 4: 跨迷你批归一化

CmBN 表示如图 4 所示的 CBN 修改版本，称为跨迷你批归一化 (CmBN)。它仅收集单个批次内迷你批次之间的统计信息。

我们从空间注意力修改为点注意力修改 SAM，并分别如图 5 和图 6 所示，将 PAN 的捷径连接替换为直接操作。

3.4. YOLOv4

本章节我们将详细介绍 YOLOv4 的组成部分：

- 主干网络 (Backbone): CSPDarknet53 [81]
- 颈部 (Neck): SPP [25]、PAN [49]
- 头部 (Head): YOLOv3 [63]

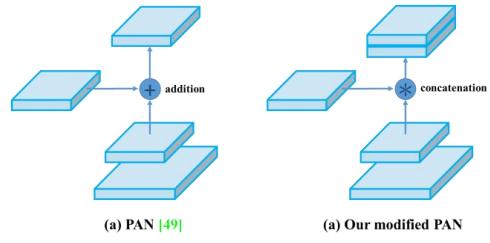


图 5: 修改的 SAM.

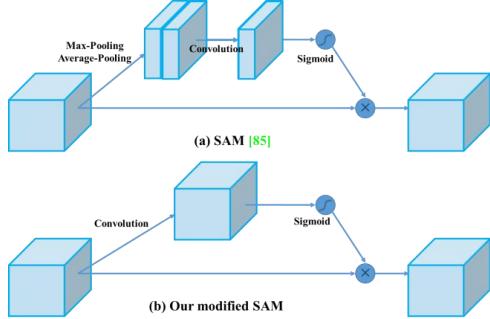


图 6: 修改的 PAN.

YOLOv4 使用了以下方法：

- 主干网络的免费赠品包 (BoF): CutMix 和 Mosaic 数据增强、DropBlock 正则化、类别标签平滑化
- 主干网络的特殊赠品包 (BoS): Mish 激活函数、跨阶段部分连接 (CSP)、多输入加权残差连接 (MiWRC)
- 检测器的免费赠品包 (BoF): CIoU 损失函数、CmBN、DropBlock 正则化、Mosaic 数据增强、对抗生成训练、消除网格敏感性、使用单个真实框匹配多个锚框、Cosine 退火调度器 [52]、最佳超参数、随机训练形状
- 检测器的特殊赠品包 (BoS): Mish 激活函数、SPP 模块、SAM 模块、PAN 路径聚合模块、DIoU-NMS

4. 方法

我们在 ImageNet (ILSVRC 2012 val) 数据集上测试了不同训练改进技术对分类器精度的影响，然后在 MS COCO (test-dev 2017) 数据集上测试了对检测器精度的影响。

4.1. 实验设置

在 ImageNet 图像分类实验中，默认超参数如下：训练步数为 8,000,000；批量大小和迷你批处理大小分别为 128 和 32；采用多项式衰减学习率调度策略，初始学习率为 0.1；热身步数为 1000；动量和权重衰减分

别设置为 0.9 和 0.005。我们所有 BoS 实验都使用与默认设置相同的超参数，在 BoF 实验中，我们额外增加了 50% 的训练步数。在 BoF 实验中，我们验证了 MixUp、CutMix、Mosaic、模糊数据增强和标签平滑正则化方法。在 BoS 实验中，我们比较了 LReLU、Swish 和 Mish 激活函数的效果。所有实验均使用 1080 Ti 或 2080 Ti GPU 训练。

在 MS COCO 目标检测实验中，默认超参数如下：训练步数为 500,500；采用 step decay 学习率调度策略，初始学习率为 0.01，分别在 400,000 步和 450,000 步时乘以因子 0.1；动量和权重衰减分别设置为 0.9 和 0.0005。所有架构都使用单个 GPU 在批量大小为 64 的情况下执行多尺度训练，而迷你批处理大小为 8 或 4，具体取决于架构和 GPU 内存限制。除了使用遗传算法进行超参数搜索实验之外，所有其他实验都使用默认设置。遗传算法使用带 GIoU 损失的 YOLOv3-SPP 进行训练，搜索 300 个 epoch 以获得最小验证集 5k。对于遗传算法实验，我们采用搜索到的学习率 0.00261、动量 0.949、分配 ground truth 的 IoU 阈值 0.213 和损失归一化因子 0.07。我们验证了大量 BoF 技术，包括消除网格敏感性、Mosaic 数据增强、IoU 阈值、遗传算法、类别标签平滑化、跨迷你批归一化、对抗生成训练、cosine 退火调度器、动态迷你批处理大小、DropBlock、优化锚框、不同类型的 IoU 损失函数。我们还对各种 BoS 进行了实验，包括 Mish、SPP、SAM、RFB、BiFPN 和高斯 YOLO [8]。对于所有实验，我们只使用单个 GPU 进行训练，因此不使用诸如 syncBN 之类的优化多个 GPU 的技术。

4.2. 分类器训练中不同特征的影响

首先，我们研究了不同特征对分类器训练的影响；具体而言，研究了类别标签平滑化、不同数据增强技术（双边模糊、MixUp、CutMix 和 Mosaic，如图 7 所示）以及不同激活函数（默认 Leaky-ReLU、Swish 和 Mish）的影响。

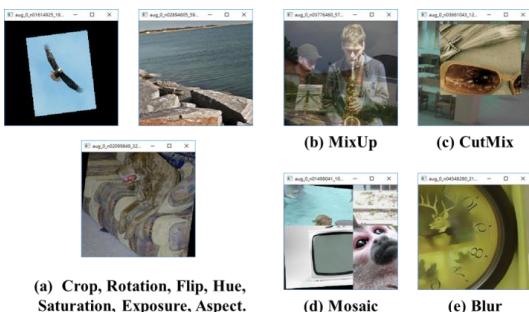


图 7：各种数据增强方法

实验结果如表 2 所示，引入 CutMix 和 Mosaic 数据

增强、类别标签平滑化以及 Mish 激活函数等特征可以提高分类器的精度。因此，我们的分类器训练用 BoF（免费赠品包）主干网络包含以下内容：CutMix 和 Mosaic 数据增强以及类别标签平滑化。此外，我们还可以选择使用 Mish 激活函数，如表 2 和表 3 所示。

表 2：BoF 和 Mish 激活函数对 CSPResNeXt-50 分类器精度的影响。

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
✓							77.9%	94.0%
	✓						77.2%	94.0%
		✓					78.0%	94.3%
			✓				78.1%	94.5%
				✓			77.5%	93.8%
					✓		78.1%	94.4%
						✓	64.5%	86.0%
						✓	78.9%	94.5%
✓	✓			✓			78.5%	94.8%
✓	✓			✓		✓	79.8%	95.2%

表 3：BoF 和 Mish 激活函数对 CSPDarknet-53 分类器精度的影响。

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
✓	✓			✓			77.2%	93.6%
✓	✓			✓		✓	77.8%	94.4%
						✓	78.7%	94.8%

4.3. 检测器训练中不同特征的影响

接下来我们研究了不同免费赠品包 (BoF-detector) 对检测器训练精度的影响，如表 4 所示。我们通过研究可以提高检测精度而不影响 FPS 的不同特征，显著地扩展了 BoF 列表：

- S: 消除网格敏感性 - YOLOv3 中用于评估目标坐标的方程 $b_x = \sigma(t_x) + c_x$ 和 $b_y = \sigma(t_y) + c_y$ 要求 c_x 和 c_y 始终为整数，因此对于 b_x 值接近 c_x 或 $c_x + 1$ 则需要非常高的 t_x 绝对值。我们通过将 sigmoid 乘以大于 1.0 的因子来解决这个问题，从而消除网格对物体检测的影响
- M: Mosaic 数据增强 - 在训练过程中使用 4 张图像的 Mosaic 数据增强而不是单张图像
- IT: IoU 阈值 - 使用多个锚框匹配单个真实框，要求 IoU(真实框, 锚框) > IoU 阈值
- GA: 遗传算法 - 在网络训练开始的 10% 时间内使用遗传算法选择最佳超参数
- LS: 类别标签平滑化 - 使用类别标签平滑化用于

sigmoid 激活函数

- CBN: CmBN - 使用跨迷你批归一化来收集整个批次中的统计信息，而不是收集单个迷你批次中的统计信息
- CA: Cosine 退火调度器 - 在正弦训练过程中改变学习率
- DM: 动态迷你批处理大小 - 通过使用随机训练形状在我们进行小分辨率训练期间自动增加迷你批处理大小
- OA: 优化锚框 - 使用针对 512x512 网络分辨率训练的优化锚框
- GIoU、CIoU、DIoU、MSE - 使用不同的损失算法进行边界框回归

我们进一步研究了不同特殊赠品包 (BoS-detector) 对检测器训练精度的影响，例如 PAN、RFB、SAM、高斯 YOLO (G) 和 ASFF，如表 5 所示。实验结果表明，当使用 SPP、PAN 和 SAM 时，检测器可以获得最佳性能。

4.4. 不同主干网络和预训练权重对检测器训练的影响

接下来我们研究了不同主干网络模型对检测器精度的影响，如表 6 所示。我们注意到，具有最佳分类精度的模型并不总是具有最佳检测精度的模型。

首先，虽然使用不同特征训练的 CSPResNeXt-50 模型的分类精度高于 CSPDarknet53 模型，但是在目标检测方面，CSPDarknet53 模型表现出更高的精度。

其次，使用 BoF 和 Mish 训练 CSPResNeXt50 分类器可以提高其分类精度，但是进一步将这些预训练权重用于检测器训练会降低检测器精度。然而，使用 BoF 和 Mish 训练 CSPDarknet53 分类器可以提高分类器和使用该分类器预训练权重的检测器的精度。最终结果是，主干网络 CSPDarknet53 比 CSPResNeXt50 更适合检测器。

4.5. 不同迷你批处理大小对检测器训练的影响

最后，我们分析了使用不同迷你批处理大小训练的模型的结果，结果如图 7 所示。从表 7 的结果可以看出，加入了 BoF 和 BoS 训练策略后，迷你批处理大小对检测器的性能几乎没有影响。这个结果表明，加入了 BoF 和 BoS 之后，就不再需要使用昂贵的 GPU 进行训练了。换句话说，任何人都可以使用普通的 GPU 来训练出优秀的检测器。

5. 结果

图 8 展示了与其他最先进目标检测器的对比结果。我们的 YOLOv4 位于帕累托最优曲线 (Pareto optimality curve) 上，在速度和精度方面都优于速度和精度最快的检测器。

6. 总结

我们提供了一个最先进的目标检测器，其速度 (FPS) 和精度 (MS COCO AP₅₀...95 和 AP₅₀) 均优于所有现有的替代检测器。所述的检测器可以在具有 8-16 GB 显存的常规 GPU 上训练和使用，这使其能够广泛应用于单阶段基于锚框的检测器这一原始概念已经证明了其可行性。我们验证了大量特征，并从中挑选了一些可以同时提高分类器和检测器精度的特征。这些特征可以作为未来研究和开发的最佳实践。

7. 致谢

作者感谢 Glenn Jocher 提供了 Mosaic 数据增强、使用遗传算法选择超参数以及解决网格敏感性问题的想法 <https://github.com/ultralytics/yolov3>。

表 5：特殊赠品包的消融实验 (Size 512x512)。

模型	AP	AP ₅₀	AP ₇₅
CSPResNeXt50-PANet-SPP	42.4%	64.4%	45.9%
CSPResNeXt50-PANet-SPP-RFB	41.8%	62.7%	45.1%
CSPResNeXt50-PANet-SPP-SAM	42.7%	64.6%	46.3%
CSPResNeXt50-PANet-SPP-SAM-G	41.6%	62.7%	45.0%
CSPResNeXt50-PANet-SPP-ASFF-RFB	41.1%	62.6%	44.4%

表 6：使用不同的分类器预训练权重进行检测器训练（所有其他训练参数在所有模型中都类似）

模型 (最优配置)	Size	AP	AP ₅₀	AP ₇₅
CSPResNeXt50-PANet-SPP	512x512	42.4	64.4	45.9
CSPResNeXt50-PANet-SPP (BoF-backbone)	512x512	42.3	64.3	45.7
CSPResNeXt50-PANet-SPP (BoF-backbone + Mish)	512x512	42.3	64.2	45.8
CSPDarknet53-PANet-SPP (BoF-backbone)	512x512	42.4	64.5	46.0
CSPDarknet53-PANet-SPP (BoF-backbone + Mish)	512x512	43.0	64.9	46.5

表 7：使用不同迷你批处理大小进行检测器训练

模型 (没有 OA)	Size	AP	AP ₅₀	AP ₇₅
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 4)	608	37.1	59.2	39.9
CSPResNeXt50-PANet-SPP (without BoF/BoS, mini-batch 8)	608	38.4	60.6	41.6
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 4)	512	41.6	64.1	45.0
CSPDarknet53-PANet-SPP (with BoF/BoS, mini-batch 8)	512	41.7	64.2	45.2

表 4: 免费赠品包的消融实验 (CSPResNeXt50-PANet-SPP, 512x512).

S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP ₅₀	AP ₇₅
✓									MSE	38.0%	60.0%	40.8%
	✓								MSE	37.7%	59.9%	40.5%
		✓							MSE	39.1%	61.8%	42.0%
			✓						MSE	36.9%	59.7%	39.4%
				✓					MSE	38.9%	61.7%	41.9%
					✓				MSE	33.0%	55.4%	35.4%
						✓			MSE	38.4%	60.7%	41.3%
							✓		MSE	38.7%	60.7%	41.9%
								✓	MSE	35.3%	57.2%	38.0%
									GloU	39.4%	59.4%	42.5%
									DIoU	39.1%	58.8%	42.1%
									Clou	39.6%	59.2%	42.6%
									Clou	41.5%	64.0%	44.8%
									Clou	36.1%	56.5%	38.4%
									MSE	40.3%	64.0%	43.1%
									GloU	42.4%	64.4%	45.9%
									Clou	42.4%	64.4%	45.9%

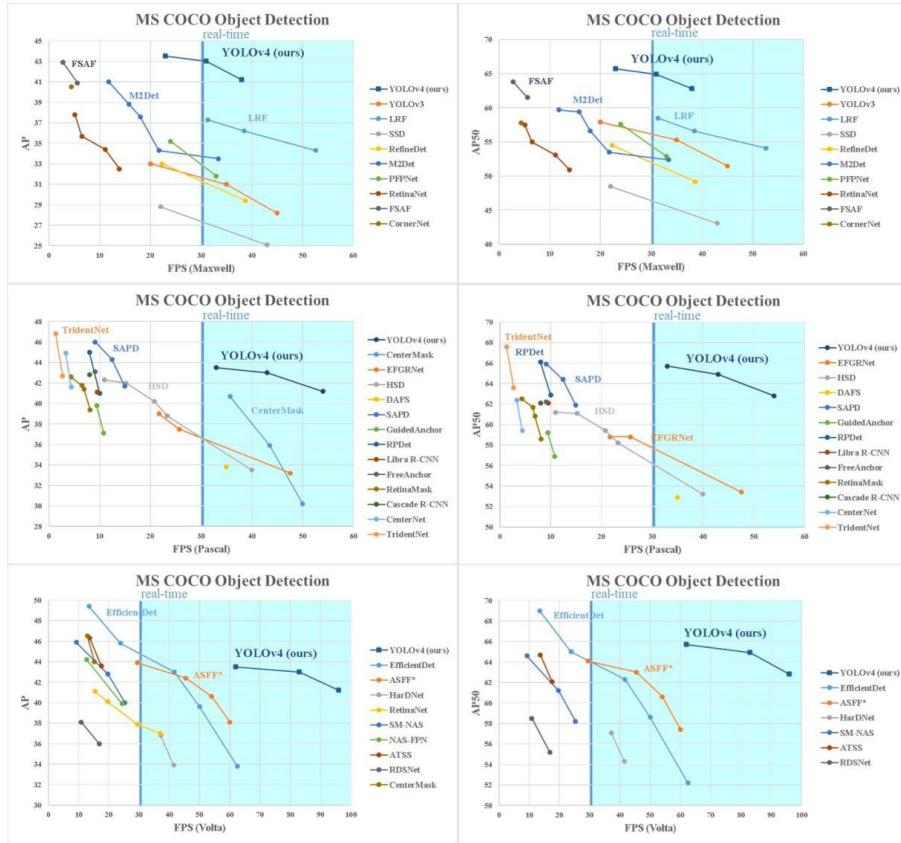


图 8: 不同目标检测器的速度和精度对比 (一些文章只报告了其检测器在 Maxwell/Pascal/Volta 等特定 GPU 上的 FPS)

附录 C 相关内容补充

C.1 硬件交互

手机图像实时传入到模型中的流程如下：手机通过 USB 连接到主机，通过 Scrcpy^①将视频流信息通过 Linux 内核中 V4L2 传入到虚拟设备中，再通过 OpenCV-Python^②从虚拟设备中获取到视频流数据，流程图如图 1-2 所示。

C.2 动态采样概率分布

动态概率分布用于切片的中心点位选取，可以将生成的单位形成团簇，产生更多重叠部分，能有效提高识别数据的多样性，具体实现过程如下，设当前待选取点位为离散的 $W \times H$ 矩阵，切片的待选二维离散分布为 $D : \{1, 2, \dots, W\} \times \{1, 2, \dots, H\} \rightarrow \mathbb{R}$ ，动态修改范围为 $S = 3$ ，令 $(i, j) \sim D$ ，则修改 D 中以 (i, j) 为中心附近大小为 $S \times S$ 的分布：

$$\begin{bmatrix} d_{i-1,j-1} & d_{i-1,j} & d_{i-1,j+1} \\ d_{i,j-1} & d_{ij} & d_{i,j+1} \\ d_{i+1,j-1} & d_{i+1,j} & d_{i+1,j+1} \end{bmatrix} \xrightarrow{\frac{1}{f(d_{ij})}} \begin{bmatrix} d_{i-1,j-1} & d_{i-1,j} & d_{i-1,j+1} \\ d_{i,j-1} & \frac{f(d_{ij})}{2}d_{ij} & d_{i,j+1} \\ d_{i+1,j-1} & d_{i+1,j} & d_{i+1,j+1} \end{bmatrix} \quad (\text{附录 C-1})$$

其中 $f(d_{ij}) = \frac{d_{ij}}{2 \sum_{u,v=1}^S [d_{i+u,j+v} \neq 0]}$ 。做法是使中心概率下降 $d_{ij}/2$ ，并将周围的非零概率点位平均分配 $d_{ij}/2$ 。

C.3 数据增强

表 C-1 数据增强参数范围

数据增强类型	参数变换范围	单位
HSV 增强	$\pm(0.015, 0.7, 0.4)$	比例系数
图像旋转	$(-5, 5)$	度
横纵向随机平移	$(-0.05, 0.05)$	比例系数
图像缩放	$(-0.2, 0.2)$	比例系数
图像左右反转	0.5	概率大小

^①<https://github.com/Genymobile/scrcpy>

^②<https://github.com/opencv/opencv-python>

C.4 模型测试卡组

表 C-2 我方卡组 (平均圣水花费 2.6)

卡牌名称	类型	攻击目标	圣水花费
骷髅兵	部队	地面	1
冰雪精灵	部队	地面和空中	1
滚木	法术	地面	2
戈仑冰人	部队	建筑	2
加农炮	建筑	地面	3
火枪手	部队	地面和空中	4
野猪骑士	部队	建筑	4
火球	法术	地面和空中	4

表 C-3 8000 分内置 AI 敌方卡组 (平均圣水花费 4.6)

卡牌名称	类型	攻击目标	圣水花费
野蛮人滚筒	法术	地面	2
飓风法术	法术	地面和空中	3
狂暴樵夫	部队	地面	4
骷髅飞龙	部队	地面和空中	4
野蛮人	部队	地面	5
雷电飞龙	部队	地面和空中	5
圣水收集器	建筑	无	6
戈仑石人	部队	建筑	8

注：测试时间为 2024 年 5 月，第 59 赛季。

附录 D 论文相关代码

本论文全部代码均已开源 <https://github.com/wty-yy/katacr>，核心代码总计 1.4 万行左右，架构设计如下：

- **build_dataset**
 - 对视频文件进行预处理（划分 episode，逐帧提取，图像不同部分提取）
 - 目标识别数据集搭建工具（辅助标记数据集，数据集版本管理，生成式目标识别，标签转化及识别标签生成，图像切片提取）
- **classification:** 用 ResNet 进行手牌及圣水分类
- **constants:** 常量存储（卡牌名称及对应圣水花费，目标识别类别名称）
- **detection:** 自行用 JAX 复现的 YOLOv5 模型（后弃用）
- **interact:** 测试与手机进行实时交互，包括目标识别，文本识别，GUI
- **ocr_text:** 包括用 JAX 复现的 CRNN（后弃用）和 PaddleOCR 的接口转化
- **policy:**
 - **env:** 两种测试环境：
 - * **VideoEnv:** 将视频数据集作为输入，仅用于调试模型的输入是否与预测相对应
 - * **InteractEnv:** 与手机进行实时交互，使用多进程方式执行感知融合
 - **offline:** 包含了决策模型 StARformer 和 DT 的训练，验证的功能，并包含三种 CNN 测试结构 ResNet, CSPDarkNet, CNNBlocks
 - **perceptron:** 感知融合，包含了 state,action,reward 三种特征生成器，并整合到 SARBuilder 中（感知基于 YOLOv8, PaddleOCR, ResNet Classifier）
 - **replay_data:** 提取专家视频中的感知特征，制作并测试离线数据集
 - **visualization:** 实时监测手机图像，可视化感知融合特征
- **utils:** 用于目标检测相关的工具（绘图、坐标转化、图像数据增强），用于视频处理的 ffmpeg 相关工具
- **yolov8:** 重构 YOLOv8 源码，包括数据读取、模型训练、验证、目标检测、跟踪，模型识别类型设置以及参数配置

下面将展示论文中提到的部分代码架构。

D.1 生成式数据集

```

1  """
2  完整代码: https://github.com/wty-yy/KataCR/blob/master/katacr/build\_dataset/generator.py
3  """
4  ... # import packages, define constants
5
6  class Generator:
7      def __init__(
```

```

8     self, background_index: int | None = None,
9     unit_list: Tuple[Unit, ...] = None,
10    seed: int | None = None,
11    intersect_ratio_thre: float = 0.5,
12    map_update: dict = {'mode': 'naive', 'size': 5},
13    augment: bool = True,
14    dynamic_unit: bool = True,
15    avail_names: Sequence[str] = None,
16    noise_unit_ratio: float = 0.0,
17  ):
18      """
19      Args:
20          background_index: Use image file name in
21          `dataset/images/segment/backgrounds/background{index}.jpg` as current background.
22          unit_list: The list of units will be generated in area.
23          seed: The random seed.
24          intersect_ratio_thre: The threshold to filte overlapping units.
25          map_update_size: The changing size of dynamic generation distribution (SxS).
26          augment: If taggled, the mask augmentation will be used.
27          dynamic_unit: If taggled, the frequency of each unit will tend to average.
28          avail_names: Specify the generation classes.
29          noise_unit_ratio: The ratio of inavailable unit (noise unit) in whole units.
30      Variables:
31          map_cfg (dict):
32              'ground': The 0/1 ground unit map in `katacr/build_dataset/generation_config.py`.
33              'fly': The 0/1 fly unit map in `katacr/build_dataset/generation_config.py`.
34              'update_size': The size of round squara.
35      ...
36
37  def build(self, save_path="", verbose=False, show_box=False, box_format='cxcywh',
38           img_size=None):
39      ... # Build image and bounding box
40
41  def add_tower(self, king=True, queen=True):
42      ... # Add king and queen tower
43
44  def add_unit(self, n=1):
45      ... # Add unit in [ground, flying, others] randomly. Unit list looks at
46      ... # `katacr/constants/label_list.py`
47
48
49 if __name__ == '__main__':
50     generator = Generator(seed=42, background_index=25, intersect_ratio_thre=0.5, augment=True,
51                           map_update={'mode': 'naive', 'size': 5}, avail_names=None)
52     for i in range(10):
53         generator.add_tower()
54         generator.add_unit(n=40)
55         x, box, _ = generator.build(verbose=False, show_box=True)
56         generator.reset()

```

D.2 特征融合

D.2.1 状态特征提取部分代码

```

1  """
2  完整代码:
3      ↳ https://github.com/wty-yy/KataCR/blob/master/katacr/policy/perceptron/state_builder.py
4  """
5  ... # import packages, define constants
6
7  class StateBuilder:
8      def get_state(self, verbose=False):
9          ... # Build state
10
11     def update(self, info: dict, deploy_cards: set):
12         """
13             Args:
14                 info (dict): The return in `VisualFusion.process()`,
15                     which has keys=[time, arena, cards, elixir]
16                 deploy_cards (set): Get deploy_cards from action_builder.
17         """
18
19         self.time: int = info['time'] if not np.isinf(info['time']) else self.time
20         self.arena: CRResults = info['arena']
21         self.cards: List[str] = info['cards']
22         self.elixir: int = info['elixir']
23         self.card2idx: dict = info['card2idx']
24         self.parts_pos: np.ndarray = info['parts_pos'] # shape=(3, 4), part1,2,3, (x,y,w,h)
25         self.box = self.arena.get_data() # xyxy, track_id, conf, cls, bel
26         self.img = self.arena.get_rgb()
27         self.frame_count += 1
28         ### Step 0: Update belong memory ###
29         self._update_bel_memory()
30         ### Step 1: Find text information ###
31         self._find_text_info(deploy_cards)
32         ### Step 2: Build bar items ###
33         self._build_bar_items()
34         ### Step 3: Combine units and bar2 with their BarItem ###
35         self._combine_bar_items()
36         ### Step 4: Update bar history ###
37         self._update_bar_items_history()
38         ### Step 5: Update class memory, if body exists ###
39         self._update_cls_memory()
40

```

D.2.2 动作特征提取部分代码

```

1  """
2  完整代码:
3      ↳ https://github.com/wty-yy/KataCR/blob/master/katacr/policy/perceptron/action_builder.py
4  """
5  ... # import packages, define constants
6
7  class ActionBuilder:
8      ... # define other functions
9
10     def get_action(self, verbose=False):
11         ... # Get from actions queue
12
13     def update(self, info):
14         """
15

```

```

14     Args:
15         info (dict): The return in `VisualFusion.process()`,
16             which has keys=[time, arena, cards, elixir]
17         """
18     self.time: int = info['time'] if not np.isinf(info['time']) else self.time
19     self.arena: CRResults = info['arena']
20     self.cards: dict = info['cards']
21     self.elixir: int = info['elixir']
22     self.card2idx: dict = info['card2idx']
23     self.box = self.arena.get_data() # xyxy, track_id, conf, cls, bel
24     self.img = self.arena.get_rgb()
25     self.frame_count += 1
26     """Step 1: Update elixir history """
27     self._update_elixir()
28     # print("Elixirs:", self.elixirs)
29     """Step 2: Update card memory """
30     self._update_cards()
31     """Step 3: Update last elixir """
32     self._update_mutation_elixir_num()
33     """Step 4: Find new action """
34     self._find_action()

```

D.2.3 奖励特征提取部分代码

```

1     """
2     完整代码:
3         ↪ https://github.com/wty-yy/KataCR/blob/master/katacr/policy/perceptron/reward_builder.py
4     ...
5
6     class RewardBuilder:
7         ... # Define other functions
8
9     def get_reward(self, verbose=False):
10        """Update King Tower """
11        """Update Tower """
12        """Calculate Reward """
13        ...
14
15    def update(self, info):
16        """
17        Args:
18            info (dict): The return in `VisualFusion.process()`,
19                which has keys=[time, arena, cards, elixir]
20        """
21        self.time: int = info['time'] if not np.isinf(info['time']) else self.time
22        self.arena: CRResults = info['arena']
23        self.elixir: int = info['elixir']
24        self.img = self.arena.get_rgb()
25        self.box = self.arena.get_data() # xyxy, track_id, conf, cls, bel
26        self.frame_count += 1

```
