doi:10.6041/j.issn.1000-1298.2023.03.028

基于 Compact - YOLO v4 的茶叶嫩芽移动端识别方法

黄家才[1]　唐　安[1]　陈光明[2]　张　铎[1]　高芳征[3]　陈　田[1]

(1.南京工程学院机械工程学院,南京 211167;2.南京农业大学工学院,南京 210031;
3.南京工程学院自动化学院,南京 211167)

摘要:茶叶嫩芽精准识别是实现嫩芽智能化采摘的前提与基础,采用视觉和深度学习的嫩芽识别方法逐渐成熟,但该方法过度依赖于高性能硬件,不利于采茶机器人移动端的部署,针对这一问题,本文提出一种基于 Compact - YOLO v4 算法的茶叶嫩芽移动端识别方法。首先对 YOLO v4 算法的 Backbone 网络和 Neck 网络进行改进,将 Backbone 网络替换为 GhostNet,将 Neck 网络中传统卷积替换为 Ghost 卷积,改进后的模型内存占用量仅为原来的 1/5。接着运用迁移学习的训练方法提升模型精度,试验表明,Compact - YOLO v4 算法模型的精度、召回率、平均精度均值、F1 值分别为 51.07%、78.67%、72.93% 和 61.45%。最后将本文算法模型移植到 PRO - RK3568 - B 移动端开发板,通过转换模型、量化处理、改进部署环境 3 种方式,降低模型推理计算对硬件性能的需求,最终在保证嫩芽识别准确率的前提下,实现了优化模型推理过程、减轻移动端边缘计算压力的目的,为茶叶嫩芽采摘机器人的实际应用提供了技术支撑。

关键词:茶叶嫩芽识别;轻量化;Compact - YOLO v4;移动端部署

中图分类号:S24;TP391.4　　**文献标识码**:A　　**文章编号**:1000-1298(2023)03-0282-09　　**OSID**:

Mobile Recognition Solution of Tea Buds Based on Compact - YOLO v4 Algorithm

HUANG Jiacai[1]　TANG An[1]　CHEN Guangming[2]　ZHANG Duo[1]　GAO Fangzheng[3]　CHEN Tian[1]

(1. *School of Mechanical Engineering*, *Nanjing Institute of Technology*, *Nanjing* 211167, *China*
2. *College of Engineering*, *Nanjing Agricultural University*, *Nanjing* 210031, *China*
3. *School of Automation*, *Nanjing Institute of Technology*, *Nanjing* 211167, *China*)

Abstract: The precise recognition of tea buds is the prerequisite and foundation of intelligent bud picking. The method of bud identification using vision and deep learning is gradually established, but it relies excessively on high-performance hardware, which is not conducive to the deployment of mobile tea picking robots. To solve this problem, a mobile recognition solution for tea buds based on the Compact - YOLO v4 algorithm was proposed. Firstly, the Backbone and Neck networks of the YOLO v4 algorithm were improved by replacing the Backbone network with GhostNet and the traditional convolution in the Neck network with Ghost convolution, the size of the improved model was only one-fifth of the original one. Secondly, the training method of transfer learning was applied to improve the model accuracy. The experiments showed that the Compact - YOLO v4 algorithm model had P, R, mAP and F1 score values of 51.07%, 78.67%, 72.93% and 61.45%, respectively. Finally, the algorithm model was transplanted to the PRO - RK3568 - B mobile development board to reduce the hardware performance requirements of the model inference calculation by three ways: converting the model, quantization processing, and improving the deployment environment. The aim of optimizing the model inference process and reducing the pressure on edge computing on mobile was eventually achieved, while also ensuring the accuracy of tea bud recognition, providing a theoretical and practical basis for the practical application of tea bud picking robots.

Key words: recognition of tea buds; lightweight; Compact - YOLO v4; mobile deployment

收稿日期:2022 - 05 - 24　修回日期:2022 - 08 - 08

基金项目:江苏省高等学校自然科学研究重大项目(20KJA510007)、国家自然科学基金面上项目(61873120)和江苏省自然科学基金面上项目(BK20201469)

作者简介:黄家才(1977—),男,教授,博士,主要从事机器人控制与系统集成、机器视觉研究,E-mail:huangjc@njit.edu.cn

0　引言

茶叶采摘是一项劳动密集型工作，具有采摘期短、工作量大的特点[1]。为了保证嫩芽质量，提高采摘效率，降低人力成本，智能采茶机器人应运而生。实现智能化嫩芽采摘的前提和基础是准确识别检测茶叶嫩芽。在茶叶嫩芽采摘领域，涌现出许多优秀的茶叶嫩芽识别算法，获得了不错的识别效果[2]。研究初期，研究者们主要是运用图像处理及相关技术手段[3-7]，对茶叶嫩芽图像的色彩空间进行分解并分析，提取其嫩芽图像特征，再对传统图像分割算法进行改进，以此达到分割嫩芽图像、识别嫩芽的目的。但是实际采茶环境复杂多变，基于图像处理的嫩芽识别算法效果并不理想。

随着机器视觉与深度学习技术的发展与应用，目前研究进度已经到了图像处理＋深度学习的阶段。吕军等[8]针对不同光照环境导致的茶丛图像亮度不均问题，提出了一种基于 YOLO v5 的茶叶嫩芽检测模型，实现了区域亮度自适应校正。PARANAVITHANA 等[9]设计了一种卷积神经网络对茶叶嫩芽进行识别，并开发了一款交互式软件用于茶芽分类。朱红春等[10]提出了一种基于 Faster R－CNN 网络的茶叶嫩芽检测方法，当不区分茶叶嫩芽类型时，平均精确率（AP）达到 0.54，均方根误差（RMSE）为 3.32。

然而，深度学习算法无论在模型训练过程还是模型推理过程，都离不开价格高昂的硬件设备（高性能工作站或服务器）和宝贵的时间成本（反复的模型训练耗时），并且受限于移动端处理器性能，大多研究仍停留在试验阶段，难以应用于实际工程。因此在保证嫩芽识别准确率的基础上，对现有深度学习算法进行轻量化改进，并部署在移动端（嵌入式开发板等硬件）的研究工作具有重要的工程实践价值。

ZHANG 等[11]将 YOLO v4 结构中 CSPDarknet53 替换为 MobileNet v2，使得改进后的网络参数量和模型内存占用量仅为原来的 16.76%、19.53%。LI 等[12]在 YOLO v4 中引入 K-means 聚类算法和卷积注意力机制模块（CBAM），进行通道剪枝，与原模型相比，参数量和模型内存占用量分别下降 34% 和 47.6%。

本文提出一种基于 Compact－YOLO v4 算法的茶叶嫩芽识别方法。该算法是在传统 YOLO v4 基础上，引入 GhostNet，GhostNet 是一个专为移动端设计的网络，保证一定模型精度的前提下，压缩网络，精简模型[13]。具体方法是将 Backbone 网络（主干网络，用于特征提取）中原有的 CSPDarknet53 替换为 GhostNet，Neck 网络（用于特征加强）中原有重复的 CBL 模块组件替换为 Ghost 卷积，并将轻量化改进后的算法移植到 PRO－RK3568－B 嵌入式开发板中，以此达到精简模型参数运算量、优化模型推理过程、减轻移动端机器人硬件部署压力的目的。

1　图像采集与数据集制作

为了最大程度还原实际采摘作业环境，本文采用实地拍摄采集的方式自建茶叶嫩芽数据集。图 1 为部分图像样本。



图 1　数据集图像样本

Fig. 1　Image samples of dataset

数据集拍摄于江苏茶博园，涵盖不同拍摄角度、不同曝光程度的嫩芽图像，以提高数据集的丰富程度，使模型学习到更多的特征，从而增强泛化能力。

2　基于 GhostNet 的 YOLO v4 算法轻量化改进

本文算法的应用平台为智能采茶机器人，机器人移动端所搭载的处理器多为嵌入式控制器，运算能力有限[14]。YOLO v4 算法参数量庞大、结构复杂，需要高性能硬件支撑，因此该算法不适合直接在嵌入式平台使用。现对其进行轻量化改进，将 Backbone 网络替换为更轻量化的 GhostNet，将 Neck 网络中传统卷积替换为 Ghost 卷积，称改进后的算法为 Compact－YOLO v4。

2.1　YOLO v4 算法

YOLO v4 算法具有检测速度快、泛化能力强的优点[15]，按结构功能划分为 Backbone、Neck 与 Head 3 部分网络，分别用于图像特征提取、图像特征融合加强和图像预测输出。若输入图像尺寸为 640 像素×640 像素，则整体算法结构如图 2 所示。

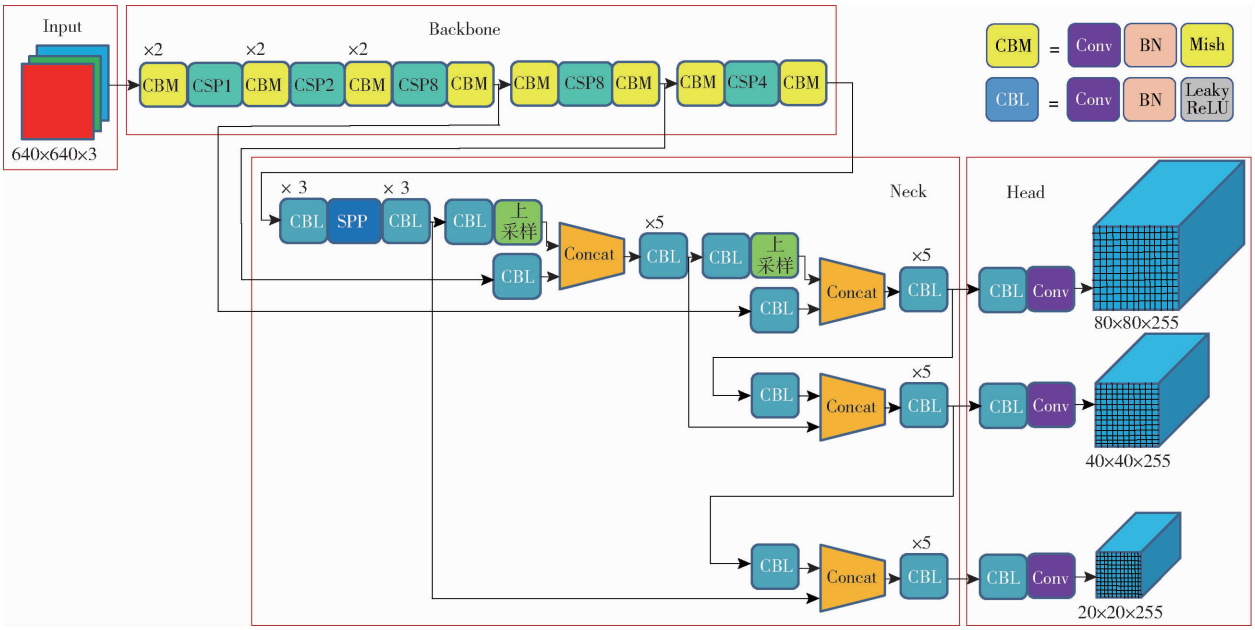

图 2 YOLO v4 结构图

Fig. 2 Structure of YOLO v4

从图 2 中可知,CBM 与 CBL 结构是 YOLO v4 中最基本的结构,两者结构基本相同。

2.2 GhostNet 结构

GhostNet 特征网络的核心思想是以更少的参数来生成更多特征,减少模型参数量和模型运算量[16],是一个适合移动端应用的轻量型神经网络,能够在保持检测精度的同时大幅减少网络参数,提升推理速度。

相较于传统结构,GhostNet 的优势在于分两步生成特征图[17]。首先使用少量传统卷积生成部分特征图,然后对这部分特征图做线性运算处理,生成新的基于原始特征的特征图,新生成的特征图称为 Ghost 特征图,线性运算的过程称为 cheap operation(廉价操作)。最后将线性运算前后的两部分特征图通过拼接操作生成最终需要的特征图,此结构即为 Ghost 卷积,如图 3 所示。

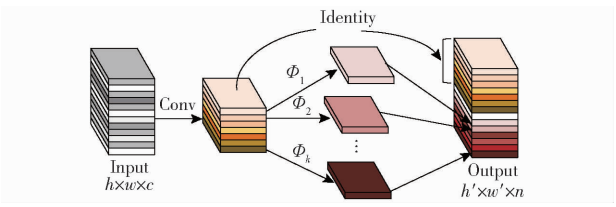

图 3 Ghost 卷积结构

Fig. 3 Structure of Ghost convolutional

若输入大小为 $h\times w\times c$ 的特征图,使用 $k\times k$ 卷积,cheap operation 变换的通道数为 m,变换次数为 s,输出 n 个特征向量通道,宽与高为 w' 和 h',且 $m\ll n$,则可得

$$n = ms \tag{1}$$

由于整个过程存在一次恒等变换(Identity),因此实际有效的变换次数为 $s-1$,式(1)变换为

$$m(s-1) = \frac{n}{s}(s-1) \tag{2}$$

假设线性运算内核均值为 $d\times d$,则能够计算出其相对于传统卷积的理论加速比 r_s 为

$$r_s = \frac{nh'w'ckk}{\frac{n}{s}h'w'ckk + (s-1)\frac{n}{s}h'w'dd} = \frac{ckk}{\frac{1}{s}ckk + \frac{s-1}{s}dd} \approx \frac{sc}{s+c-1} \approx s \tag{3}$$

其中 k 与 d 近似,且 $s\ll c$。

通过上述计算过程不难发现,由于线性运算与传统卷积运算相比,计算消耗更低,将原来全部使用卷积核生成特征图的操作,改为仅保留少部分数量卷积核,其他部分卷积运算由线性运算代替,这样能够大幅减少生成特征图所需要的计算量和时间[18]。

G-bneck(Ghost-bottleneck)是 GhostNet 中的重要结构,包含两个 Ghost 卷积,但这两个卷积作用并不相同,如图 4 所示。第 1 个 Ghost 卷积用于扩展通道数,第 2 个 Ghost 卷积作用与之相反,用于压缩通道数,方便实现 shortcut(捷径)连接。

根据使用步长的不同,G-bneck 的实现方法也不同。当步长为 1 时,G-bneck 由 2 个不同作用的 Ghost 卷积直接堆叠而成,且不对第 2 个 Ghost 卷积输出使用激活函数;而当步长为 2 时,通过在两个 Ghost 卷积间添加一个步长为 2 的深度卷积(Depthwise Conv)构建 G-bneck 结构。

此外,GhostNet 中还使用到了 2017 年提出的 SE(Squeeze-and-Excitation)模块[19]。该模块属于通

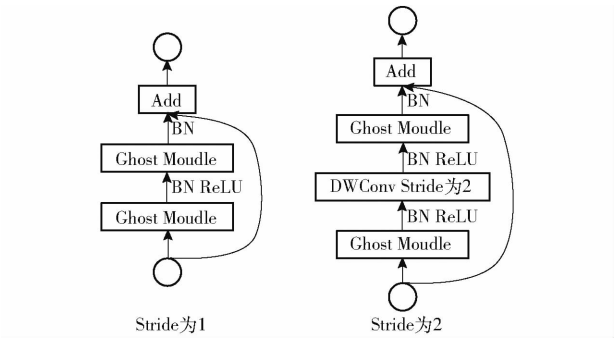

图 4　G－bneck 结构

Fig. 4　Structure of G－bneck

道注意力机制，可以自适应学习不同通道之间的依赖关系[20]，具有结构简单、即插即用的优点。将它加入网络中，不仅不会改变原有结构功能，还可以增强网络提取特征的能力。

从图 5 中可以看出，SE 模块主要包括 3 个操作步骤：压缩（Squeeze）、激励（Excitation）和缩放（Scale）。SE 模块能够给出特征图各通道的权重值，提升重要特征的贡献度并削弱不重要特征的贡献度，有效增强了网络提取重要特征的能力。

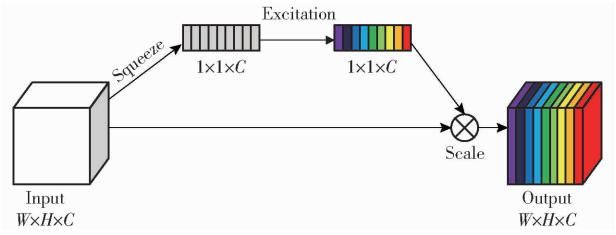

图 5　SE 模块结构

Fig. 5　Structure of SE module

GhostNet 总体结构如图 6 所示，包含 Ghost 卷积、G－bneck 结构、SE 模块等。

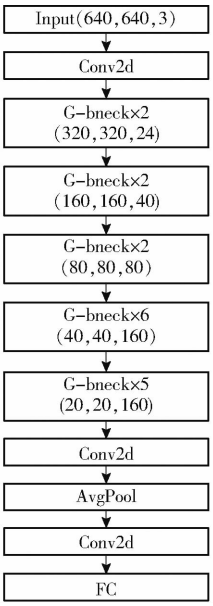

图 6　GhostNet 结构图

Fig. 6　Structure diagram of GhostNet

2.3　改进过程

2.3.1　Backbone 网络改进

YOLO v4 的 Backbone 网络采用 CSPDarknet53，此结构由多个重复的残差结构组成，特征图输出通道数量多，表明它使用的卷积核数量十分庞大，包含大量冗余参数[21]。CSPDarknet53 每层网络结构的详细信息如表 1 所示，其中每层网络的输入均来自于前一层，表中不再展示。

表 1　CSPDarknet53 网络结构参数

Tab. 1　Network structure of CSPDarknet53

重复使用次数	参数量	结构名称	输出特征图通道数
1	928	models. common. Conv	32
1	18 560	models. common. Conv	64
1	20 672	models. common. Bottleneck	64
1	73 984	models. common. Conv	128
2	119 936	models. common. BottleneckCSP	128
1	295 424	models. common. Conv	256
8	1 463 552	models. common. BottleneckCSP	256
1	1 180 672	models. common. Conv	512
8	5 843 456	models. common. BottleneckCSP	512
1	4 720 640	models. common. Conv	1 024
4	12 858 368	models. common. BottleneckCSP	1 024

由表 1 发现，随着卷积核数量增加，每层网络的参数量也随之大幅上升，最终总参数量约为 $2.659\,62\times10^7$，浮点计算量也高达 7.78×10^9。大量的参数带来高昂的计算成本，使得该结构很难部署在移动端的计算平台上。根据实际需求，本文将原始 GhostNet 中最后 4 层网络层直接删除，再替换到原 Backbone 网络中，精简后每层网络结构的详细信息如表 2 所示。

表 2　GhostNet 网络结构参数

Tab. 2　Network structure of GhostNet

重复使用次数	参数量	结构名称	输出特征图通道数
1	464	models. common. Conv	16
1	464	models. common. GhostBottleneck	16
1	2 564	models. common. GhostBottleneck	24
1	2 352	models. common. GhostBottleneck	24
1	9 636	models. common. GhostBottleneck	40
1	13 672	models. common. GhostBottleneck	40
1	22 920	models. common. GhostBottleneck	80
5	533 476	models. common. GhostBottleneck	112
1	362 840	models. common. GhostBottleneck	160
4	1 567 520	models. common. GhostBottleneck	160

通过表 1 和表 2 的对比、计算，总参数量减少至 $2.516\,0\times10^6$，下降了一个数量级，浮点计算量也下降至 2.5×10^8。显然，相较于 CSPDarknet53，GhostNet 更适合在硬件条件和运算能力有限的嵌入式平台使用。

2.3.2　Neck 网络改进

在 YOLO v4 算法中，Neck 网络部分与

Backbone 网络部分类似,使用多个重复的 CBL 模块组件,说明也存在大量冗余[22],有轻量化改进的空间。如果使用 Ghost 卷积对特征图进行处理,在保证效果前提下,能够大幅减少网络参数,提升网络性能。基于上述分析,将 YOLO v4 的 Neck 网络中传统卷积替换为 Ghost 卷积,替换后的 Neck 网络参数量大幅降低,且网络性能几乎不变。忽略其中的拼接操作与 SPP 池化操作(这两部分参数量都为 0),替换前后 Neck 网络每层结构详细信息如表 3、4 所示。

表 3 原 Neck 网络结构参数

Tab. 3 Network structure of original Neck

重复使用次数	参数量	结构名称	输出特征图通道数
1	3 148 800	models. common. Conv3	512
1	3 673 088	models. common. Conv3	512
1	131 584	models. common. UpSample	256
1	131 584	models. common. Conv	256
1	2 756 096	models. common. Conv5	256
1	33 024	models. common. UpSample	128
1	33 024	models. common. Conv	128
1	689 920	models. common. Conv5	128
1	295 424	models. common. DownSample	256
1	2 756 096	models. common. Conv5	256
1	1 180 672	models. common. DownSample	512
1	11 017 216	models. common. Conv5	512

表 4 改进后 Neck 网络结构参数

Tab. 4 Network structure of improved Neck

重复使用次数	参数量	结构名称	输出特征图通道数
1	1 140 224	models. common. GhostConv3	512
1	2 106 880	models. common. GhostConv3	512
1	131 584	models. common. UpSample	256
1	29 184	models. common. GhostConv	256
1	664 576	models. common. GhostConv5	256
1	33 024	models. common. UpSample	128
1	5 376	models. common. GhostConv	128
1	168 448	models. common. GhostConv5	128
1	180 992	models. common. DownSample	256
1	664 576	models. common. GhostConv5	256
1	722 432	models. common. DownSample	512
1	2 639 872	models. common. GhostConv5	512

由表 3 和表 4 可知,虽然替换前后 Neck 网络每层的输出通道数不变,但由于 Ghost 卷积使用的卷积核数量更少,因此参数量也更少。经过上述对 Neck 网络的改进,总参数量下降约 67%,浮点计算量由 4.37×10^9 下降为 1.27×10^9。

2.3.3 改进分析

经计算,改进后 Compact-YOLO v4 与 YOLO v4 相比,整体参数量仅为原来的 1/45,浮点计算量由原来的 1.232×10^{10} 减少为 1.69×10^9,实现了对 YOLO v4 算法的轻量化改进。完整改进过程如图 7 所示,图中虚线部分为精简删除部分,原 Backbone

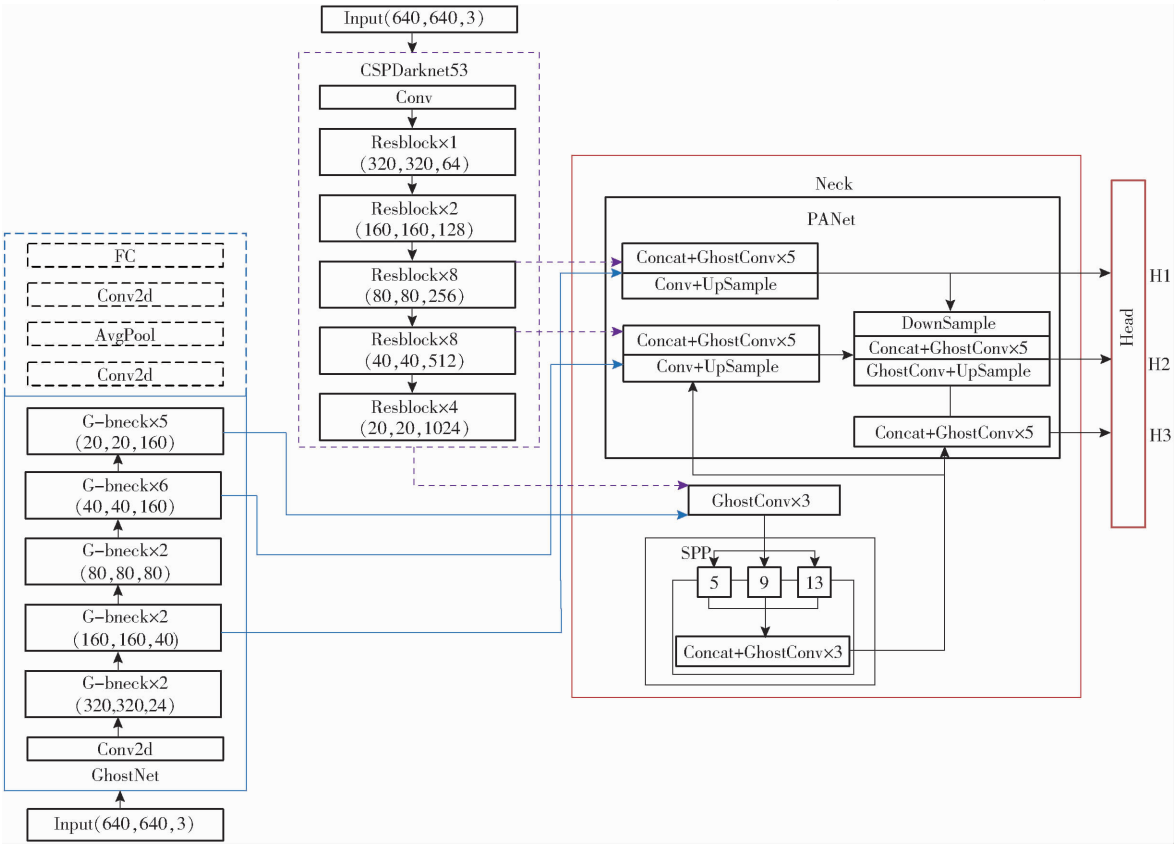

图 7 总体改进流程

Fig. 7 Overall improvement process

网络中输出到 Neck 网络中的 3 种尺度特征图，在新的 Backbone 网络中也将对应相应尺度的特征图输出到 Neck 网络中，不做改动。

3　模型训练与移动端部署

本试验系统平台选取 Ubuntu 18.04，深度学习框架为 PyTorch 1.9.1，Epochs（迭代周期）为 300，batch_size（批量大小）设置为 64，输入训练图像尺寸为 640 像素×640 像素，learning_rate（学习率）设置为 0.01。

本文试验采用深度学习目标检测领域常见的性能评价指标，包括 P（精度）、R（召回率）、mAP（平均精度均值）、F1 值等参数。

3.1　训练结果与分析

损失变化曲线可以直观展示神经网络模型对所给数据的学习和拟合情况。Compact－YOLO v4 算法的损失函数主要包含 GIoU（预测框位置损失）、Obj（预测框置信度损失）、Cls（分类损失）。

从图 8 和图 9 中可看出，在训练或验证一开始，损失函数梯度下降速度较快；随着训练的进行，当训练轮次达到 50 时，损失函数曲线梯度继续下降但趋势明显放缓，且曲线伴随着震荡；当训练轮次达到 300 时，曲线趋于平稳，损失函数收敛到某一固定值，结束训练。选取此过程 mAP 指标最高的一轮作为最终训练结果，对应的各参数指标如表 5 所示。

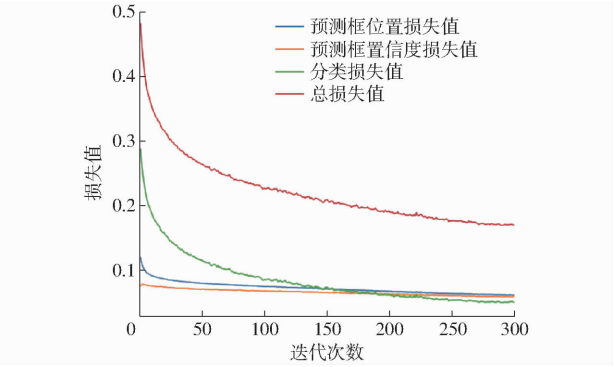

图 8　训练集损失值变化曲线

Fig. 8　Loss variation curves of training set

由表 5 可知，Compact－YOLO v4 算法的模型内存占用量仅为 23.2 MB，仅为 YOLO v4 模型的 1/5。但是，它的各项指标参数有待提高，因此通过加入迁移学习来改善训练效果。

迁移学习的优势是利用预训练好的模型的优异泛化性能，并省去训练初期的冗杂过程，节省训练时间。训练前，将 GhostNet 预训练模型对应层的参数赋值给本文算法模型对应层，两者一一对应。训练时，使用冻结训练的方法，在训练前 50 个轮次，不更

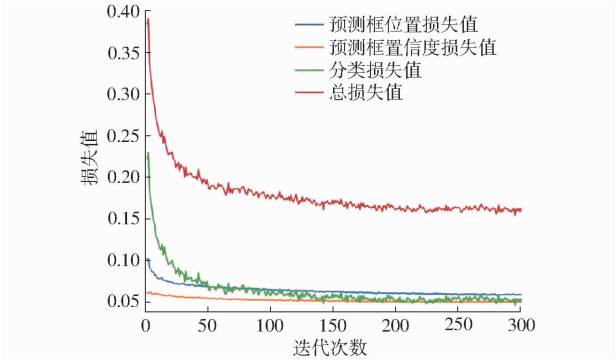

图 9　验证集损失值变化曲线

Fig. 9　Loss variation curves of validation set

表 5　Compact－YOLO v4 算法训练结果

Tab. 5　Training results of Compact－YOLO v4

参数	数值
P/%	46.87
R/%	71.63
mAP/%	67.45
F1 值/%	56.88
模型内存占用量/MB	23.20

新上述赋值参数层的梯度，即 Backbone 网络部分，仅对 Neck 网络以及 Head 网络部分进行参数更新，以此加快训练效率，防止权值被破坏。在 50 个轮次训练完成后，解冻 Backbone 网络，继续训练，当损失函数收敛，停止训练，共 300 个轮次。

使用迁移训练方法后，训练和验证过程损失函数曲线如图 10 和图 11 所示。不难发现使用迁移学习训练方法后，损失值下降迅速，且最终收敛值也较使用迁移训练之前更低。通过观察图 12，能够更直观了解到各参数指标在应用迁移学习后，得到了不同程度的提升。

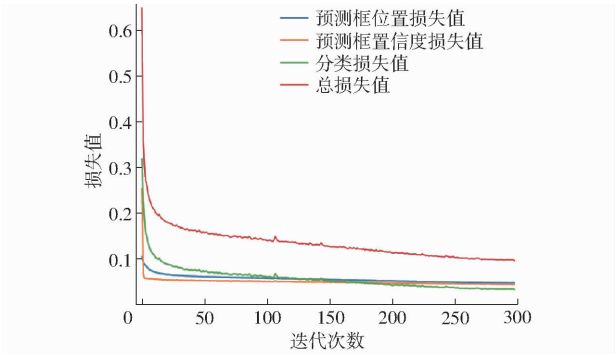

图 10　训练集损失变化曲线（迁移学习）

Fig. 10　Loss variation curves of training set (transfer learning)

为了验证本文算法的检测效果，在相同的测试集下，分别与 AlexNet[23]、Inception v3[24]、Faster R－CNN[25] 和 YOLO v4[26] 进行对比。如表 6 所示，Compact－YOLO v4 相比于 AlexNet 和 Inception v3 而言，各项参数指标均有较大提升；与 Faster R－

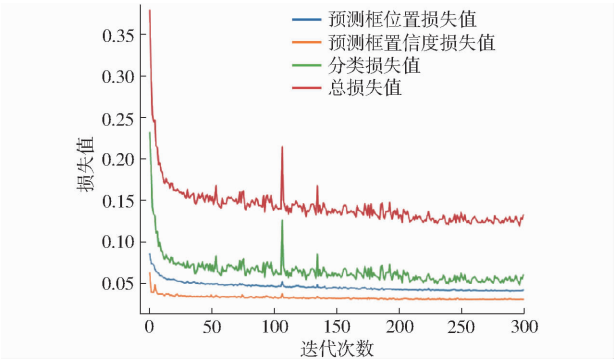

图 11 验证集损失变化曲线(迁移学习)

Fig. 11 Loss variation curves of validation set (transfer learning)

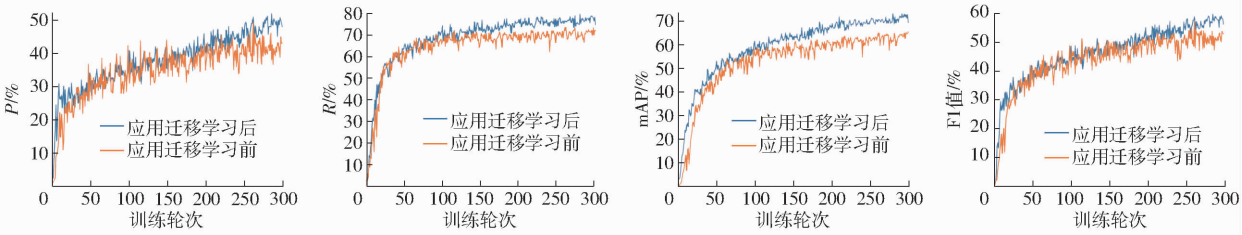

图 12 使用迁移学习前后各指标在验证集上的对比曲线

Fig. 12 Comparison curves for each indicator on validation set before and after using transfer learning

表 6 不同模型检测性能对比

Tab. 6 Comparison of detection performance with different models

模型	*P*/%	*R*/%	mAP/%	F1 值/%	模型内存占用量/MB	检测速度/(f·s^{-1})
AlexNet	53.78	77.43	71.36	59.12	68.6	13.3
Inception v3	50.41	73.96	67.58	62.83	83.4	16.5
Faster R-CNN	58.76	79.29	74.57	74.91	70.8	12.7
YOLO v4	61.32	87.52	84.61	76.32	115.8	32.1
Compact-YOLO v4	51.07	78.67	72.93	61.45	23.2	43.6

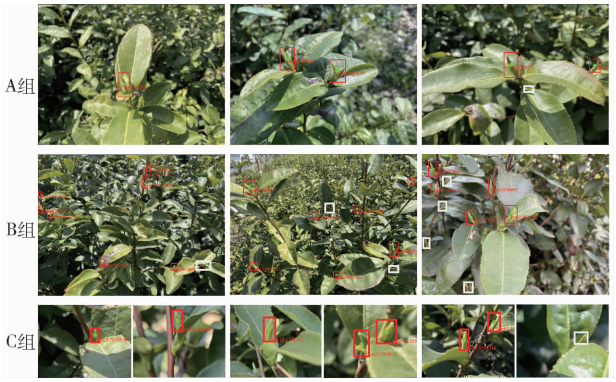

图 13 模型推理结果

Fig. 13 Results of model inference

3.2 移动平台部署

PRO-RK3568-B 是基于瑞星微 RK3568 芯片研发的开发板,性能远不如计算机或工作站,这也是本文对 YOLO v4 算法进行精简轻量化改进的原因之一,即减轻 Compact-YOLO v4 算法对于移动端控制器的性能需求压力。PRO-RK3568-B 开发板实物如图 14 所示。

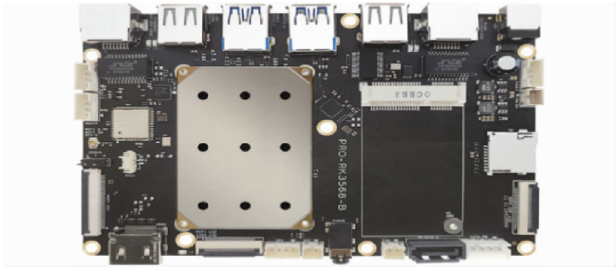

图 14 PRO-RK3568-B 开发板

Fig. 14 PRO-RK3568-B development board

PyTorch 框架所生成的模型无法直接应用于 PRO-RK3568-B 开发板,且硬件性能还不足以满足模型推理计算性能,所以现需要进行如下试验工作,将原模型进一步轻量化,并部署到移动端开发板。

将 PyTorch 模型转换为适用于 PRO-RK3568-B 开发板的 RKNN 模型,转换流程如图 15 所示。需要注意的是,官方提供的工具并没有直接转换 PyTorch 模型的相关接口,需要将原模型转换为 ONNX 模型,然后再转换为 RKNN 模型。

对转换后的模型进行量化处理,进一步轻量化处理。量化的主要目的是降低模型推理所需的计算和内存,常用的手段是将原模型的浮点计算转换为低比特定点计算,但这样会带来一定的精度损失。未量化的 RKNN 模型内存占用量和原 PyTorch 模型内存占用量一样,均为 23.2 MB,但量化处理后的模

CNN 检测效果接近;相较于传统 YOLO v4,参数指标平均下降 11 个百分点,因为轻量化改进后精度损失问题难以避免,但模型大幅精简,并在检测速度上提升 36%。Compact-YOLO v4 兼顾了检测精度和推理速度,更适用于茶叶嫩芽检测。

图 13 中,A 组样本为嫩芽近景特写样本,检测效果良好。B 组为远景样本,画面场景较为复杂,老叶、嫩芽、枝干等元素相互影响,对于一些被老叶遮挡或体积微小的嫩芽,存在漏检现象。C 组为嫩芽遮挡样本,对于部分遮挡情况,仍可以准确识别,若遮挡面积超过 50%,则会造成漏检。其中白色标定框为漏检嫩芽。

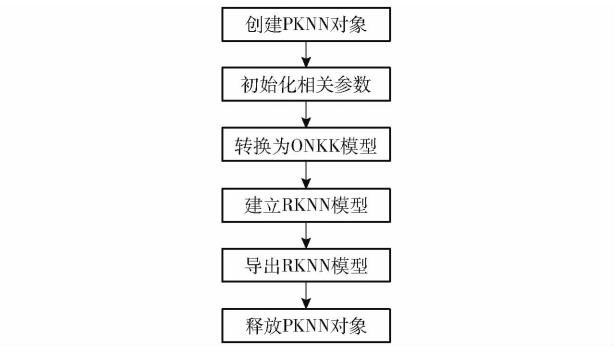

图 15　模型转换过程

Fig. 15　Process of model conversion

型内存占用量仅为 9.8 MB。

使用 C++环境部署，进一步提高性能。为进一步加快模型推理速度，在终端部署时使用 C++环境进行部署，先对 C++代码进行交叉编译，最后移植入 RK3568 使用。

移动端部署完毕后，通过循环推理试验，计算出单幅图像推理耗时约为 67.1 ms，检测速度约 15 f/s，满足实际工程应用条件。

为验证转换后的模型在 RK3568 的实际表现，使用测试集中部分茶叶嫩芽图像再次进行推理测试，推理效果如图 16 所示，对于近景图像样本，模型保持了对嫩芽的良好推理效果，对于远景图像样本中清晰、明显的嫩芽，也有着较为理想的识别准确率。



(a) 近景　　(b) 远景

图 16　模型推理结果(移动端)

Fig. 16　Results of model inference (on mobile)

至此，本文的研究工作结束，最终移植部署到移动端(PRO－RK3568－B 开发板)的 Compact－YOLO v4 算法模型内存占用量仅为 9.8 MB，同时保持了良好的模型精度。表 7 对比了 YOLO v4 算法在计算机推理运行和 Compact－YOLO v4 算法在移动端推理运行这两种方案在推理速度、模型精度、硬件成本方面的性能。

表 7　2 种算法性能对比

Tab. 7　Performance comparison of two algorithms

参数	YOLO v4 (计算机端)	Compact－YOLO v4 (移动端)
mAP/%	84.6	70.2
F1 值/%	76.3	60.1
推理速度/(f·s^{-1})	32.1	15.0
模型内存占用量/MB	115.8	9.8
硬件成本/元	≥8 000	1 200

将 Compact－YOLO v4 算法部署到移动端后，模型精度有所下降，且受限于移动端处理器的性能，推理速度降低，但推理速度 15 f/s 基本满足工程应用。得益于移动端处理器的售价低廉、集成程度高、结构紧凑的优势，使得原本停留在实验室的茶叶嫩芽检测方法可以产业化应用，减轻智能采茶机器人机械设计、电器安装的压力。

4　结论

(1)在 YOLO v4 算法的基础上，进行轻量化改进，改进后的算法浮点计算量由原来的 1.232×10^{10} 减少为 1.69×10^{9}，模型内存占用量为 23.2 MB，仅为原模型的 1/5。

(2)构建茶叶嫩芽图像数据集，训练 Compact－YOLO v4 茶叶嫩芽检测模型。应用迁移学习后的模型参数指标 P、R、mAP、F1 值分别为 51.07%、78.67%、72.93%、61.45%，与 AlexNet、Inception v3、Faster R－CNN 和 YOLO v4 对比后，发现各项参数指标接近，但模型大幅精简，检测速度提升明显，兼顾了检测精度和推理速度。

(3)将 Compact－YOLO v4 模型部署至移动端(PRO－RK3568－B 开发板)，模型进一步精简，内存占用量仅为 9.8 MB，但受限于移动端处理器的性能，推理速度降低，约为 15 f/s，满足实际工程应用要求。

参　考　文　献

[1] HAN Y, XIAO H, QIN G, et al. Developing situations of tea plucking machine[J]. Engineering, 2014, 6(6): 45606.

[2] 王文明，肖宏儒，陈巧敏，等. 基于图像处理的茶叶智能识别与检测技术研究进展分析[J]. 中国农机化学报，2020，41(7)：178－184.
WANG Wenming, XIAO Hongru, CHEN Qiaomin, et al. Research progress analysis of tea intelligent recognition and detection technology based on image processing[J]. Journal of Chinese Agricultural Mechanization, 2020, 41(7): 178－184. (in Chinese)

[3] CHEN Z, HE L, YE Y, et al. Automatic sorting of fresh tea leaves using vision-based recognition method[J]. Journal of Food Process Engineering, 2020, 43(9): e13474.

[4] 杨福增，杨亮亮，田艳娜，等. 基于颜色和形状特征的茶叶嫩芽识别方法[J]. 农业机械学报，2009，40(增刊)：119－123.
YANG Fuzeng, YANG Liangliang, TIAN Yanna, et al. Recognition of the tea sprout based on color and shape features[J].

Transactions of the Chinese Society for Agricultural Machinery, 2009,40(Supp.): 119 - 123. (in Chinese)
[5] SHAO P, WU M, WANG X, et al. Research on the tea bud recognition based on improved K-means algorithm[C]// MATEC Web of Conferences. EDP Sciences, 2018, 232: 03050.
[6] QI F, XIE Z, TANG Z, et al. Related study based on Otsu watershed algorithm and new squeeze-and-excitation networks for segmentation and level classification of tea buds[J]. Neural Processing Letters, 2021, 53(3): 2261 - 2275.
[7] ZHANG L, ZOU L, WU C, et al. Method of famous tea sprout identification and segmentation based on improved watershed algorithm[J]. Computers and Electronics in Agriculture, 2021, 184: 106108.
[8] 吕军,方梦瑞,姚青,等. 基于区域亮度自适应校正的茶叶嫩芽检测模型[J]. 农业工程学报,2021,37(22):278 - 285.
LÜ Jun, FANG Mengrui, YAO Qing, et al. Detection model for tea buds based on region brightness adaptive correction[J]. Transactions of the CSAE, 2021,37(22):278 - 285. (in Chinese)
[9] PARANAVITHANA I R, KALANSURIYA V R. Deep convolutional neural network model for tea bud (s) classification[J]. IAENG International Journal of Computer Science, 2021, 48(3): 1 - 6.
[10] 朱红春,李旭,孟炀,等. 基于 Faster R - CNN 网络的茶叶嫩芽检测[J]. 农业机械学报,2022,53(5):217 - 224.
ZHU Hongchun, LI Xu, MENG Yang, et al. Tea bud detection based on Faster R - CNN network[J]. Transactions of the Chinese Society for Agricultural Machmery,2022,53(5):217 - 224. (in Chinese)
[11] ZHANG M, XU S, SONG W, et al. Lightweight underwater object detection based on YOLO v4 and multi-scale attentional feature fusion[J]. Remote Sensing, 2021, 13(22): 4706.
[12] LI F, JIANG Z, ZHOU S, et al. Spilled load detection based on lightweight YOLO v4 trained with easily accessible synthetic dataset[J]. Computers and Electrical Engineering, 2022, 100: 107944.
[13] YE Z, CUI X, QIU X, et al. GSnet: combine Ghostnet and Shufflenetv2 to get better performance[C]// 2nd International Conference on Computer Vision, Image, and Deep Learning. SPIE, 2021, 11911: 324 - 328.
[14] BATURONE I, GERSNOVIEZ A, BARRIGA Á. Neuro-fuzzy techniques to optimize an FPGA embedded controller for robot navigation[J]. Applied Soft Computing, 2014, 21: 95 - 106.
[15] WU D, LÜ S, JIANG M, et al. Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments[J]. Computers and Electronics in Agriculture, 2020, 178: 105742.
[16] PAOLETTI M E, HAUT J M, PEREIRA N S, et al. Ghostnet for hyperspectral image classification[J]. IEEE Transactions on Geoscience and Remote Sensing, 2021, 59(12): 10378 - 10393.
[17] GHOSH M, MUKHERJEE H, OBAIDULLAH S M, et al. LWSINet: a deep learning-based approach towards video script identification[J]. Multimedia Tools and Applications, 2021, 80(19): 29095 - 29128.
[18] DU K L. Clustering: a neural network approach[J]. Neural Networks, 2010, 23(1): 89 - 107.
[19] HU J, SHEN L, SUN G. Squeeze-and-excitation networks[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018: 7132 - 7141.
[20] ROY A G, NAVAB N, WACHINGER C. Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks[J]. IEEE Transactions on Medical Imaging, 2018, 38(2): 540 - 549.
[21] DING P, QIAN H, CHU S. SlimYOLO v4: lightweight object detector based on YOLO v4[J]. Journal of Real-Time Image Processing, 2022,19(3): 1 - 12.
[22] WANG L, ZHOU K, CHU A, et al. An improved light-weight traffic sign recognition algorithm based on YOLO v4-tiny[J]. IEEE Access, 2021, 9: 124963 - 124971.
[23] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84 - 90.
[24] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the inception architecture for computer vision[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,2016: 2818 - 2826.
[25] REN S, HE K, GIRSHICK R, et al. Faster R - CNN: towards real-time object detection with region proposal networks[J]. Advances in Neural Information Processing Systems, 2015, 28:91 - 99.
[26] BOCHKOVSKIY A, WANG C Y, LIAO H Y M. YOLO v4: optimal speed and accuracy of object detection[J]. arXiv preprint arXiv:2004.10934, 2020.