文章编号：1003-0530(2022)09-1954-11

Ghost-YOLO：轻量化口罩人脸检测算法

陈继平[1] 陈永平[1] 谢 懿[1] 朱建清[1,2] 曾焕强[1]
（1. 华侨大学工学院，福建泉州 362021；2. 厦门亿联网络技术股份有限公司，福建厦门 361015）

摘 要：在嵌入式设备上，由于算力及存储空间的限制，当前的大型高精度目标检测模型的推理速度较低。为此，本文设计了一种轻量化目标检测模型，用于口罩人脸检测。首先，本文设计了一种高激活性鬼影（High Active Ghost，HAG）模块，以轻量的计算代价减少特征图中的冗余。其次，利用HAG实现高激活性鬼影跨段部分（High Active Ghost Cross Stage Partial，HAG-CSP）连接模块，提升了跨段部分连接网络结构的特征学习能力。再次，利用HAG-CSP对你只需看一次（You Only Look Once，YOLO）模型进行轻量化改造来得到完整的Ghost-YOLO网络，并构造出一个口罩人脸检测器。实验结果表明，本文提出方法在NVIDIA Jetson NX嵌入式设备上，在检测精度优于其他目标检测算法的前提下，对于640×640的图片，实现了24.72 ms每帧的检测速度，并且减少了模型的参数量。

关键词：嵌入式设备；目标识别；鬼影模块；YOLOv5；跨阶段部分模块
中图分类号：TP391.4 **文献标识码：**A **DOI：**10.16798/j.issn.1003-0530.2022.09.018

引用格式：陈继平，陈永平，谢懿，等. Ghost-YOLO：轻量化口罩人脸检测算法[J]. 信号处理，2022，38(9)：1954-1964. DOI：10.16798/j.issn.1003-0530.2022.09.018.
Reference format：CHEN Jiping，CHEN Yongping，XIE Yi，et al. Ghost-YOLO：Lightweight masked face detection algorithm[J]. Journal of Signal Processing，2022，38(9)：1954-1964. DOI: 10.16798/j.issn.1003-0530.2022.09.018.

Ghost-YOLO：Lightweight Masked Face Detection Algorithm

CHEN Jiping[1] CHEN Yongping[1] XIE Yi[1] ZHU Jianqing[1,2] ZENG Huanqiang[1]
（1. College of Engineering，Huaqiao University，Quanzhou，Fujian 362021，China；
2. Xiamen Yealink Network Technology Co.，Ltd，Xiamen，Fujian 361015，China）

Abstract: In embedded devices，due to the limitation of low computing power and small storage space，the inference speed of current large-scale high-precision object detection algorithms is low. For that，this paper designed a lightweight masked face detection algorithm. Firstly，a high active ghost（HAG）module was designed to reduce the redundancy in feature learning with a light computational cost. Secondly，based on HAG，a high active ghost cross stage partial（HAG-CSP）connection module was designed to improve the feature learning ability. Thirdly，the HAG-CSP was used to lightweight the you only look once（YOLO）model to raise a Ghost-YOLO model for masked face detection. Experimental results show that the speed of proposed method is 24.72 milliseconds per image for 640×640 sized images on a NVIDIA Jetson NX embedded device and reduces the number of parameters，under the condition that the proposed method had a better accuracy performance than existing object detection models.

Key words: embedded device；object detection；ghost module；YOLOv5；cross stage partial connection

收稿日期：2021-08-09；修回日期：2022-03-14
基金项目：国家重点研发计划项目(2021YFE0205400)；国家自然科学基金面上项目(61976098，61871434)；福建省自然科学基金杰出青年项目(2022J06023)；福厦泉国家自主创新示范区协同创新平台项目(2021FX03)

1 引言

2019年的新型冠状病毒在全世界范围内迅猛传播给人类健康带来了严峻的挑战。佩戴口罩能够有效地防止新型冠状病毒的传播，因此，口罩已是每位出行者必须佩戴的防护设备。在疫情之前，深度学习在人脸检测任务中已有广泛应用，例如毛秀萍[1]将通用目标检测模型应用至人脸检测任务，以及文献[2]提出的基于深度学习的人脸实时检测方法等。但是，这些人脸检测多数处理完整人脸，并不完全适应疫情情况下口罩人脸检测，因为人脸戴上口罩后丢失了几乎一半的特征信息。因此，口罩人脸检测是一个亟待研究的课题。

得益于深度学习的高速发展，学术界涌现出很多基于深度学习的目标检测模型，例如：你只需看一次（You Only Look Once，YOLO）[3]、单帧多框检测器（Single Shot Multi-Box Detector，SSD）[4]和以区域卷积神经网络（Region Convolutional Neural Network，R-CNN）[5]为基础的 Fast R-CNN[6]、Faster R-CNN[7]等目标检测算法。目标检测模型在嵌入式设备中落地部署时，由于嵌入式设备性能低下，使得在嵌入式设备中很难平衡算法的精度与速度。为此，嵌入式设备上的目标检测模型往往采用轻量化网络，如 MobileNet[8-9]、SqueezeNet[10]等，通过压缩再扩展结构或使用深度可分离卷积（Depthwise Separable Convolution）[11]来提高模型推理速度或者减小权重大小，但代价是其精度会降低。NanoDet[12]、MobileNet以及Pelee[13]等模型的参数量大小虽然不到10M甚至不到1M，但其mAP表现分别只有23.5%、22.4%，MobileNet家族的V1、V2、V3以及V3-small的mAP表现均不超过22.2%。而在YOLO家族中的YOLOv4-tiny[14]以及YOLOv3-tiny[15]，虽然是从YOLOv4、YOLOv3这些大参数量且高精度的模型进行轻量化改造而来，但其参数量大小分别为6.06M、8.86M，mAP仅为21.7%、16.6%，均达不到当前轻量化模型的性能。而YOLOv5-s[16]这个参数量只有7.3M的模型，在COCO[17]数据集中获得36.7%的mAP性能指标。因此，在这些目标检测模型中，YOLOv5-s模型能够较好地平衡检测速度和检测精度，在目标检测领域中备受青睐。

除了采用轻量化网络，GhostNet[18]指出了通过卷积所得的特征图中存在着大量冗余的特征，为此他们提出了鬼影模块（Ghost Module）来实现高效的特征去冗余。Ghost Module[18]能用更少的参数来获得与普通卷积输出的相同通道数的特征，相比于MobileNet中使用的大量1×1卷积，Ghost模块可以自定义卷积核尺寸大小，而且其先使用的一组用来减少通道数的普通卷积，使得后一层的分组卷积能够减少高通道数带来的高参数量。

为了解决在嵌入式设备上口罩人脸检测算法的检测速度与精度的平衡问题，本文设计了一个少参数量、检测速度和精度都能处在一个较高水平的目标检测网络。关于检测网络模型，本文使用了YOLOv5-s目标检测算法，并在其架构上引入了基于Ghost Module改进的HAG模块，设计出Ghost-YOLO轻量化目标检测算法，该模型相比于YOLOv5-s减少了大量参数。另外为了解决口罩人脸数据集较少的问题，本文整理了口罩人脸的数据集来扩充训练数据集。结合以上模型与数据集，本文设计了基于Ghost-YOLO的口罩人脸检测器。

2 轻量化Ghost-YOLO网络设计

YOLOv5模型的主体基本由基于跨阶段部分模块（Cross Stage Partial，CSP）[19]结构组成，由于模型中大部分卷积计算和参数都集中于该模块，因此本文首先对CSP模块进行改进，提出一种轻量化的高激活性鬼影跨段部分（High Active Ghost Cross Stage Partial，HAG-CSP）。紧接着，基于轻量化的HAG-CSP结构，本文进一步构建出一个轻量化Ghost-YOLO模型。接下来，本文对HAG-CSP结构和基于HAG-CSP的轻量化Ghost-YOLO模型分别进行介绍，并设计基于Ghost-YOLO模型的口罩人脸检测器。

2.1 轻量化的HAG-CSP结构

本文首先在Ghost Module的基础上提出一种高激活性鬼影（High Active Ghost，HAG）模块；其次，使用所提出HAG结构设计出轻量化的HAG-CSP结构。

2.1.1 高激活性的鬼影结构HAG

HAG结构如图1所示，其中的CBH模块包括一个$k\times k$尺寸卷积层、一个Batch Normalized层和一个Hard-Swish[9,20]激活函数，在卷积层后使用了Batch

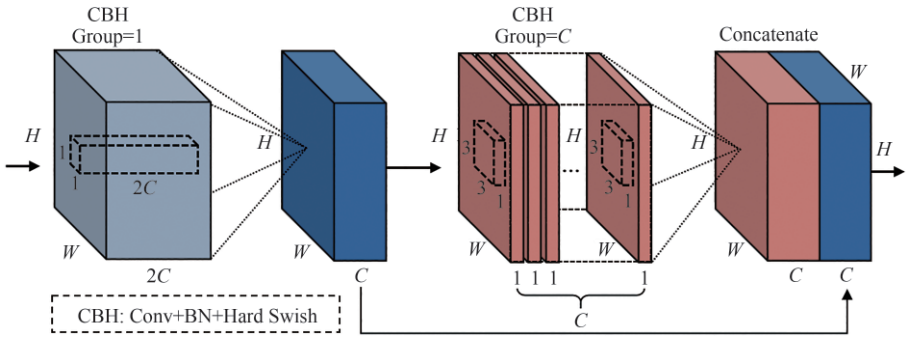

图 1 改进后的 Ghost Module 模块 HAG，图中展示了其工作流程及特征图尺寸维度变化

Fig. 1 HAG is improved based on Ghost Module, and its workflow and dimension change of feature diagram are shown in the figure

Normalized 层来对数据进行归一化，以加快训练速度，再使用激活函数来激活有效特征。HAG 的前一部分由一个 1×1 大小、输出通道数为输入通道数的一半、分组为 1 的 CBH 模块组成，后一部分以前一部分 CBH 模块输出的结果作为输入，经过一个 3×3 大小、输入输出通道数相同、但分组数与输入通道数相同的 CBH 模块，最后将两部分输出的特征图进行拼接操作，将其作为 HAG 的最终输出特征图。在 HAG 结构中对特征图的操作不会改变其高宽。

可从 HAG 的工作流程中发现，其输出的特征图中一半通道数的特征图 A 是输入特征图经过第一个 CBH 后得出来的，而另一半通道数的特征图 B 是特征图 A 经过第二个 CBH 后得出来的，所以特征图 A 会影响特征图 B 生成的内容。

在 Ghost Module 模块中，两次卷积操作之后都使用了修正线性单元（Rectified Linear Unit，ReLU）[21] 作为激活函数，而在本文初期工作中使用原始的 Ghost Module 模块改造的 Ghost-YOLO 模型进行训练时，随着训练世代数的升高，拥有大量卷积层的 Ghost-YOLO 就越可能出现梯度消失的情况。由于 ReLU 激活函数在横坐标负半轴的梯度为 0，导致负的梯度在经过 ReLU 处理之后被置为 0，梯度为负值的部分会被稀疏掉，某个神经元可能不再会有任何数据被激活，使得该神经元“坏死”，导致模型无法学习到有效的特征。

因此，同采用 ReLU 激活函数的原始的 Ghost Module 相比，本文的 HAG 结构采用的 Hard-swish 激活函数在 $x\in(-3,0)$ 时函数值非 0，对于 ReLU 函数负半轴为 0 的特性，Hard Swish 有效地防止负梯度信息被稀释，具有更高的激活性。在计算成本方面，相比于 Swish[20] 激活函数包含的 sigmoid 函数，Hard-Swish 激活函数使用了 ReLU6（x+3）进行线性计算来减少计算量，并且保持了 Swish 函数无上界有下界的特点，因此使用 Hard-Swish 激活函数不仅能够拥有与 Swish 相似的特性，并且能够降低计算开销，对嵌入式设备较低性能的运行条件更友好。

2.1.2 基于 HAG 的轻量化 HAG-CSP 结构

图 2 为基于 HAG 的轻量化 HAG-CSP 结构。在 HAG-CSP 中我们将多个 HAG 模块串联组成 Multi-HAG 作为特征提取的主要模块，通过多层的卷积的堆叠来充分提取特征。

特征图进入 HAG-CSP 后，首先分成两个分支 Branch A 和 Branch B 分别进行运算，两个 Branch 先均使用 1×1 卷积层来进行通道缩减，从而优化了 HAG-CSP 模块的时间复杂度和空间复杂度；在 Branch A 中将通道缩减后的特征图送入 Multi- HAG 进行特征提取，然后经过 1×1 卷积层进行通道特征整合；之后通过 Concatenate 将两个 Branch 输出特征图拼接起来，由于拼接之前一个操作是使用了单个卷积层进行卷积，拼接后得到了线性特征，因此之后需通过 BN 层和 Hard Swish 激活函数转化为非线性特征；最后再经过卷积核尺寸为 1×1 的 CBH 模块实现特征的融合。相比原 YOLOv5 中 CSP 结构使用的普通卷积来进行特征提取，HAG-CSP 由于使用了 HAG 模块，使之能够降低大部分参数量。

接下来对 HAG-CSP 结构的轻量化特性给予证明。对于单一的普通卷积层和 HAG 结构，假设输入与输出特征图的尺寸均为 $F^{in}=F^{out}=R^{H\times W\times 2C}$，$H$、$W$、$C$ 表示特征图的高、宽和通道数，HAG 中的分组卷积尺寸为 $Y=k\times k\times(C/g)$，k、g 分别为卷积核的尺寸和分组数，为了比较普通卷积与 HAG 模块中的

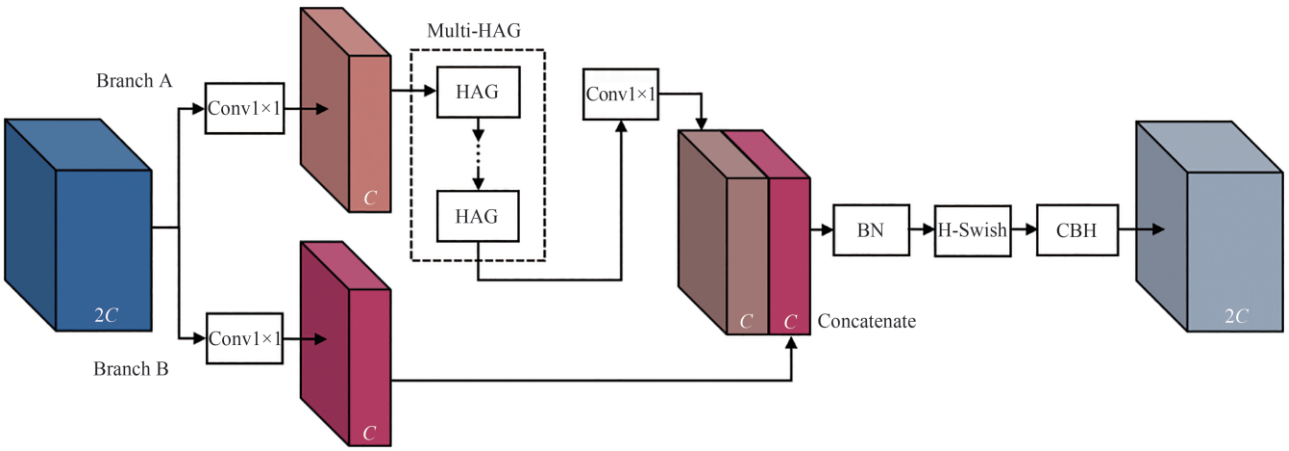

图2　基于HAG模块改进的HAG-CSP结构

Fig. 2　The Improved HAG-CSP structure based on HAG module

两个卷积层的参数量，因此设特征图R经过尺寸为3×3的普通卷积层处理后的参数量为 $r_1 = 3 \times 3 \times 2C \times 2C$，而特征图R经过HAG模块所得卷积层参数量 r_2 为式(1)：

$$r_2 = 1 \times 1 \times 2C \times C + k \times k \times \frac{C}{g} \times \frac{C}{g} \times g \quad (1)$$

其中加法的前部分为第一层卷积的参数量，其使用了尺寸为1×1的卷积核，并且输出通道数为 C，即为输入通道数的一半，后一部分为分组卷积的参数量，将输入通道分为 g 组分别进行卷积操作，在本文中，分组卷积核尺寸 k 取3，g 取特征图R一半通道数，即与第一层卷积输出的通道数相同为 C。式(2)中对 r_1 和 r_2 两个公式进行简化：

$$\begin{cases} r_1 = 36C^2 \\ r_2 = 2C^2 + 9C \end{cases} \quad (2)$$

之后再对其求比值，因此普通卷积与HAG模块的参数量比值如式(3)：

$$Z = \frac{r_1}{r_2} = \frac{36C}{2C + 9} \quad (3)$$

按照式(3)的比值结果，当通道数越大，比值 Z 也就越大，因此在输入相同尺寸的特征图时，本文的HAG所用参数更少，并且随着通道数的升高，将普通卷积模块替换为HAG模块后获得的减少参数量的收益越大。经过多个HAG模块的堆叠，致使HAG-CSP相比于CSP能够减少大量的参数，因此证明了本文HAG-CSP结构的轻量化的特点。

在HAG中，本文之所以将分组卷积的卷积核尺寸设定为3×3，是因为嵌入式设备的计算性能较弱，同时当前的硬件与软件对于3×3卷积核进行了计算优化，虽然大尺寸卷积核能更好地联系像素与周边之间的信息，但会带来更多的参数，因此堆叠多个小卷积核来达成大卷积核的效果。

2.2　基于HAG-CSP的轻量化YOLO模型

经过以上对Ghost Module、CSP结构的轻量化改进，本文将其引入至YOLOv5-s模型进行轻量化改造。本文通过对YOLOv5-s中的CSP结构改进为成本文的HAG-CSP结构，进而得到了如图3的Ghost-YOLO模型的网络结构。与YOLO系列算法相同，Ghost-YOLO中包含了特征提取结构Backbone、特征融合结构Neck以及预测结构Prediction。以下对其分别进行介绍。

Backbone为Ghost-YOLO的第一个结构，输入图片首先经过Focus[16]模块，通过类似于临近采样的方法在图片中每隔一个像素取一个值，总共取4组值来得到4组特征图，由于是隔像素取值，因此这4组特征图基本保持了原图片的特征分布，之后将4组特征图进行拼接操作，最后经过1×1卷积层进行特征整合，从而能以不损失特征的方式实现下采样操作。随后使用多个堆叠的HAG-CSP结构进行特征提取，图3中堆叠的HAG-CSP模块之间使用步长为2的卷积层进行下采样操作，Focus结构后的第二和第三次下采样之后各使用了3个HAG-CSP结构进行串联来加深网络结构。最后经过空间金字塔池化(Spatial Pyramid Pooling，SPP)[22]，使得模型能够适应不同尺寸的输入图片。本文将Backbone的输入图片尺寸规定为640×640由于Backbone结

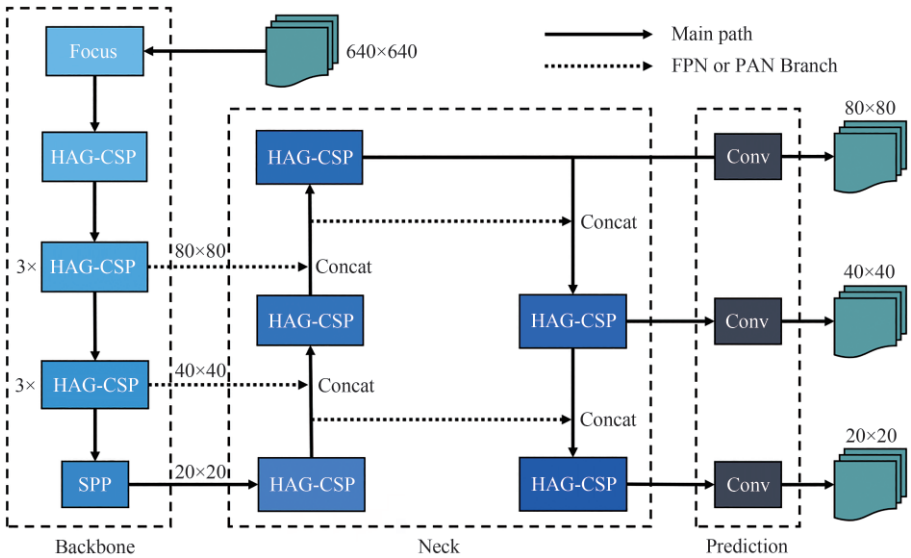

图 3 Ghost-YOLO 模型结构(图中省略了上采样、下采样操作)

Fig. 3 Model structure of Ghost-YOLO (up-sampling and down-sampling operations are omitted in this figure)

构的总步长为 32,因此图片经过该结构的特征提取后,输出了 3 个尺寸和通道数分别为 20×20×512、40×40×256、80×80×128 的特征图给 Neck 进行特征融合。

特征融合是使多个含有不同信息的特征通过拼接、相加等方法进行融合,从而能在一组特征中包含更多的信息,例如阮海涛等人[23]通过特征金字塔网络来得到不同尺度下的信息,而本文在 Neck 中使用了 YOLOv5 中的特征金字塔网络(Feature Pyramid Networks, FPN)[24]结构和路径聚合网络(Path Aggregation Network, PAN)[25]结构,即通过上采样、下采样和 Concatenate 等操作,将两个尺寸与通道数不相同且经过不同卷积操作的特征图进行融合,如图 3 中的虚线分支部分,让两个特征图所提取的特征细节进行融合,从而实现不同尺寸目标的识别。最后将 Neck 中输出的三个尺度的特征融合结果输出给目标预测模块 Prediction,通过卷积核尺寸为 1×1 的卷积层便输出三个尺寸分别为 20×20、40×40、80×80 的最终预测结果,尺寸越大则能够检测出越小的目标,因此 Ghost-YOLO 能检测不同尺度的目标。

2.3 采用轻量化 Ghost-YOLO 模型的口罩人脸检测器

本文基于 Ghost-YOLO 设计了口罩人脸检测器,训练所需数据集类别需有为三类:非人脸、未带口罩的人脸以及戴上口罩的人脸,其中非人脸一类仅作为背景,并不对其进行检测,实际检测目标为未戴口罩人脸和戴上口罩的人脸。

该口罩人脸检测器的工作流程为:将特征图分成 $S\times S$ 个区域 Grid,由存在目标中心点的 Grid 负责检测。首先在每个区域中会生成 a 个预测框,其中每个预测框会预测(5+2)个值,其中前 5 个值代表该预测框的中心坐标距当前 Grid 左上角起点的距离 t_x 和 t_y、预测框的高宽缩放系数 t_w 和 t_h 以及预测框的置信度 c,另外 2 个值代表未戴口罩人脸以及戴口罩人脸的分类概率。之后,对于同一类且在空间上是同一个目标所生成的多个预测框,对其预测框和目标的真实框计算交并比(Intersection of Union, IoU),从所得交并比中选择交并比最大的预测框,即为最终所得预测框。

$$\begin{cases} x = \sigma(t_x) + c_x \\ y = \sigma(t_y) + c_y \\ w = l_w \mathrm{e}^{t_w} \\ h = l_h \mathrm{e}^{t_h} \end{cases} \tag{4}$$

预测框位置的 4 个预测值在进行损失计算时会转换为相对位置,其转换公式如式(4),其中符号 $\sigma(x)$ 为 sigmoid 函数,c_x 和 c_y 为预测框所在 Grid 的左上角点坐标,l_w 和 l_h 为预测框的宽和高。

为了使生成的预测框更贴合实际目标,以及提高对目标分类的准确度,因此在检测器的损失函数

方面，Ghost-YOLO 使用了 YOLOv5 的损失函数，其由三种损失组成：如式(5)的置信度损失、式(6)的分类损失以及式(7)的预测框位置损失，其总损失函数如式(8)所示：

$$L_{\text{obj}} = \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{a} I_{i,j}^{\text{noobj}} (c_i - \hat{c}_i)^2 + \lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{i,j}^{\text{obj}} (c_i - \hat{c}_i)^2 \tag{5}$$

$$L_{\text{class}} = \lambda_{\text{class}} \sum_{i=0}^{S^2} \sum_{j=0}^{a} I_{i,j}^{\text{obj}} \sum_{C \in \text{classes}} p_i(c) \log(\hat{p}_i(c)) \tag{6}$$

$$L_{\text{box}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{a} I_{i,j}^{\text{obj}} (2 - w_i \times h_i) \left[(x_i - \hat{x}_i^j)^2 + (y_i - \hat{y}_i^j)^2 + (w_i - \hat{w}_i^j)^2 + (h_i - \hat{h}_i^j)^2 \right] \tag{7}$$

$$L_{\text{Total}} = L_{\text{obj}} + L_{\text{class}} + L_{\text{box}} \tag{8}$$

上面的式子中，c、w、h、x、y 为模型得出的预测值，S^2 代表图片划分成 $S \times S$ 后的 Grid，a 为某一 Grid 中的第 a 个预测框，$I_{i,j}^{\text{obj}}$ 代表第 i,j 处若有包含目标中心点，则其值为 1，否则为 0，而 $I_{i,j}^{\text{noobj}}$ 与其相反。对于置信度损失 L_{obj}，理想情况下，当框中有物体时，置信度值 c_i 为 1，否则为 0，式(5)中 λ_{noobj} 和 λ_{obj} 分别为无目标和有目标的置信度损失系数。当预测框中的置信度达到阈值后，即认为框中有目标存在，则进行分类损失 L_{class} 计算，式(5)中 λ_{class} 为分类损失系数，$p_i(c)$ 为类别预测。与分类损失相同，预测框位置损失 L_{box} 只在包含目标中心点的预测框中进行损失计算，通过计算预测框与真实框的位置差异来施加惩罚。结合以上损失函数以及检测器的工作方法，设计出了基于 Ghost-YOLO 的口罩人脸检测器。

3　实验与评估

3.1　实验数据及环境

本文收集并整理了人脸与戴口罩人脸的图像数据集，该数据集包含 9817 张口罩人脸图片，其中包括了医用口罩、N95 口罩、防毒面具、带颜色花纹口罩等多种类型的口罩，图像分辨率从 100×100 至 5989×3993，用于训练的有 8836 张图片，其中有 6661 个戴口罩人脸标注框，14884 个人脸标注框；另外有 889 张图片作为测试集，其中有 645 个戴口罩人脸标注框，1734 个人脸标注框，标注效果如图 4 所示。

另外，本文还使用了 AIZOO 数据集（https://github.com/AIZOOTech/FaceMaskDetection）以及 FMDD 数据集（https://www.kaggle.com/wobotintelligence/face-mask-detection-dataset）来验证本文算法。AIZOO 数据集包含了 7959 张带标注信息的图片，其中 6120 张用作训练集，1839 张用作测试集。FMDD 数据集拥有 20 个类，本文参照文献[27]的方法，从中选取标签为“face_no_mask”和“face_with_mask”两类共 3384 张图片作为数据集，其中训练集包含 2707 张图片，测试集包含 677 张图片。

自 YOLO9000[26] 开始引入的 Anchor-base[7] 方法帮助其取得了更高的性能表现，因此需要一组提前设定好尺寸的 Anchor，本文采用了 K-means 聚类方法，对每个真实框未经过值归一化的长宽进行了统计。最后得出 9 组 Anchor 尺寸，分别为[(9，12)，

图 4　人脸与戴口罩人脸数据集的标注示例

Fig. 4　Annotation examples of face and mask wearing face data sets

(25,32),(32,38)]、[(50,61),(56,80),(76,78)]、[(93,103),(141,166),(177,234)],小尺寸 Anchor 被使用在 Neck 输出的 20×20 尺寸的特征图上进行预测,另外两个尺寸则分别使用在 40×40、80×80 的特征图上进行预测。

本文算法的实验评估工作均在 NVIDIA Jetson Xavier NX 嵌入式设备上进行,模型训练过程所使用的显卡为一块 NVIDIA GeForce GTX TITAN X,操作系统为 Ubuntu18.04,深度学习框架为 PyTorch1.6,CUDA 版本为 10.2。

3.2 实验细节及评价指标

Ghost-YOLO 的训练过程中,为了尽可能实现公平对比,以下参数值设定和损失函数系数设置与 YOLOv5[16]所提供的设置一致:输入分辨率为 640×640,选择 Adam 优化器进行参数优化,模型的初始学习率为 0.01,权重衰减系数为 0.0005,训练世代数 Epoch 设定为 150,Warmup 的世代数为 3,在 Warmup 中的学习率动量为 0.937,偏差学习倍数设置为 0.2。经过 Warmup 的 3 个 Epoch 后,进入余弦退火学习周期,该周期的学习率为模型初始学习率乘以当前 Epoch 的余弦退火学习率,余弦退火学习率初始值设定为 0.02,随着 Epoch 的增加而减少。在损失函数系数的设置中,检测框位置损失系数为 0.05,分类损失系数和置信度损失系数都为 0.5。对于数据增强参数,对图片的色调、饱和度和亮度的参数分别设定为 0.015、0.7、0.4,图片左右翻转概率设定为 0.5,Mosaic 数据增强的系数为 1。关于参数设置,与 YOLOv5 的唯一不同之处在于本文将 BatchSize 设置成 16,以尽可能合理利用本地显存资源。

在检测实时性方面,使用平均单张图片平均处理时间作为评价指标。在检测准确性评价指标方面,本文选择在目标检测领域中常用 mAP50 和 mAP 0.5:0.95 指标[17]。mAP 表示平均精度均值(mean average precision),其计算公式如下:

$$\mathrm{mAP}=\frac{\sum_{c=1}^{n}\mathrm{AP}_c}{k} \tag{9}$$

其中,n 表示类别数量;AP_c 表示第 c 类的平均精度(average precision, AP),即为精确率-召回率(Precision-Recall,P-R)曲线的曲线下面积。实践中需要预先设定预测框与真实框交并比(Intersection of Union, IoU)阈值,再将所有预测框的精确率和召回率组合绘制出 P-R 曲线。mAP50 指 IoU 阈值取 50% 时的 mAP 值。进一步的,将阈值从 50% 开始,以 5% 为步长逐渐升高至 95%,获得不同阈值下的 mAP,最后对所有 mAP 求均值即获得 mAP 0.5:0.95。

3.3 消融实验

为了验证本文的 Ghost 轻量化方法的效果,在共同应用 YOLOv5-s 骨架网络的前提下,对采用本文设计的 Ghost 轻量化方法和不用本文设计的 Ghost 轻量化方法进行性能对比,结果如表 1 所示。

表 1 采用 Ghost 轻量化方法改进前后的 YOLOv5-s 在各数据集上的性能对比

Tab. 1 The performance comparison of YOLOv5-s on each dataset before and after Ghost lightweight method improvement

数据集	网络结构	模型大小	mAP50 (%)	mAP 0.5:0.95 (%)
自建数据集	采用 Ghost 轻量化	21.0M	84.9	51.9
	不用 Ghost 轻量化	29.6M	83.8	50
AIZOO	采用 Ghost 轻量化	21.0M	95.3	66.9
	不用 Ghost 轻量化	29.6M	94.9	64.5
FMDD	采用 Ghost 轻量化	21.0M	91.2	66.1
	不用 Ghost 轻量化	29.6M	90.2	62.3

在模型大小上,采用 Ghost 轻量化方法的模型相较于不采用 Ghost 轻量化方法的模型减少了 8.6 M。在本文自建数据集、AIZOO 数据集以及 FMDD 数据集上,采用 Ghost 轻量化方法的模型在 mAP50 上较不采用 Ghost 轻量化方法的模型分别提升了 1.1%、0.4%、1%,而在 mAP 0.5:0.95 指标上则分别提升了 1.9%、2.4%、3.8%。以上数据说明本文的 Ghost 轻量化方法能够在减小模型大小的同时提高检测精度,证明了本文方法的优越性。

3.4 Ghost-YOLO 在数据集上性能分析与对比

为了证明本文算法的性能,在表 2 中给出在本文数据集上本文算法与其他轻量化目标检测算法在模型大小、mAP50 和 mAP 0.5:0.95 指标上的表现。另外在表 3 中给出了在 AIZOO 数据集和 FMDD 数据集上本文算法与其他目标检测算法的精度对比。

由表 2 可见,在检测的准确性能方面,本文的 Ghost-YOLO 在口罩人脸数据集中达到了 84.9% 的 mAP50 和 51.9% 的 mAP 0.5:0.95,相比本文算法

表2　本文算法与其他检测算法在本文数据集上的性能比较

Tab. 2　The performance comparison between our algorithm and other detection algorithms on our dataset

网络结构	模型大小	mAP50 (%)	mAP 0.5:0.95 (%)
Ghost-YOLO	21.0M	84.9	51.9
NanoDet-320[12]	7.6M	63.9	36.9
YOLOv4-tiny[14]	23.5M	80.9	/
YOLOv5-s[16]	29.6M	83.8	50

基础的 YOLOv5-s 分别提升了 1.1% 和 1.9%；相对于 YOLOv4-tiny 则有 3% 的提升。对比 NanoDet-320，本文算法在 mAP50 上有 21% 的领先，在 mAP 0.5:0.95 上领先幅度达到 15%。在模型大小方面，表2中模型权重均以32位精度保存，本文算法模型的大小均小于 YOLOv5-s 和 YOLOv4-tiny，虽然 NanoDet-320 在模型大小上有非常大的优势，但是其检测性能远低于 Ghost-YOLO。因此，本文的 Ghost-YOLO 算法在模型大小和准确性表现方面都优于表内的其他轻量型目标检测算法。

在表3中，本文算法在 AIZOO 数据集上的 mAP50 相较于其他7种方法至少有 0.4% 的领先，其中，本文算法较专用人脸检测算法 RetinaFace 的精度提升了 2.3%，而较于 RetinaFaceMask 则有 0.5% 的领先，mAP 0.5:0.95 指标较于 NanoDet 和 YOLOv5-s 分别有 6.9%、2.4% 的领先；在 FMDD 数据集上，本文算法相较于其他通用目标检测算法的 mAP50 至少有 1% 的提升，mAP 0.5:0.95 指标较于 NanoDet 和 YOLOv5-s 分别有 9.3%、3.8% 的领先。以上对比进一步验证了本文算法的优越性。

在模型推理速度方面，图5给出采用不同 Batch Size 时不同算法在 NVIDIA Jetson Xavier NX 嵌入式设备上的推理速度比较。本文算法在 Batch size 为 8 时，其推理速度达到最高为 24.72 ms，相较于该实验最快的 YOLOv5-s 的 24.42 ms 仅多出了 0.3 ms 的推理时间，而相对于 YOLOv4-tiny 和 NanoDet 分别减少了 23.62 ms 和 3.8 ms。因此，结合表2的检测精度比较结果和图5的检测速度比较结果，证明本文算法具备了优秀的推理速度和更好的识别性能表现。

3.5　特征图可视化

为了分析神经元坏死的情况，以及证明本文基于 Ghost Module 改进的 HAG 的有效性，本文以 ResNet18[31] 为基础，分别使用本文的 HAG 和 Ghost Module 来替换 ResNet18 中的卷积模块，从而构建出两种模型。在同一数据集下经过相同训练之后，分别对其第二个经改造后的卷积模块输出的特征图进行观察，其可视化结果分别如图6(a)和图6(b)所示，使用原 Ghost Module 改造的 ResNet18 模型输出的特征图中，有几个通道的特征图为空，而使用了本文 HAG 改造的 ResNet18 模型输出的特征图中未

表3　本文算法与其他检测算法在 AIZOO 数据集和 FMDD 数据集下的性能比较

Tab. 3　The performance comparison between our algorithm and other detection algorithms in AIZOO dataset and FMDD dataset

数据集	网络结构	mAP50 (%)	mAP 0.5:0.95 (%)
AIZOO	Ghost-YOLO	95.3	66.9
	NanoDet-320[12]	91.3	60
	YOLOv4-tiny[14]	92.5	/
	YOLOv5-s[16]	94.9	64.5
	MSAF R-CNN[27]	90.4	/
	RetinaFaceMask[28]	94.8	/
	RetinaFace[29-30]	93	/
	SL-FMDet[30]	93.8	/
FMDD	Ghost-YOLO	91.2	66.1
	NanoDet-320[12]	83.5	56.8
	YOLOv4-tiny[14]	88.7	/
	YOLOv5-s[16]	90.2	62.3
	MSAF R-CNN[27]	88.7	/

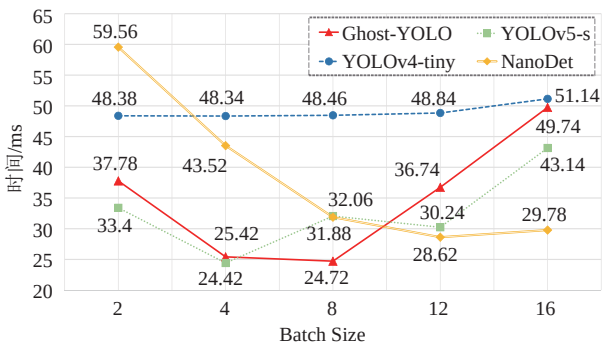

图5　采用不同 Batch Size 时，不同模型的推理速度比较

Fig. 5　The comparison of inference speed of different models via different batch sizes

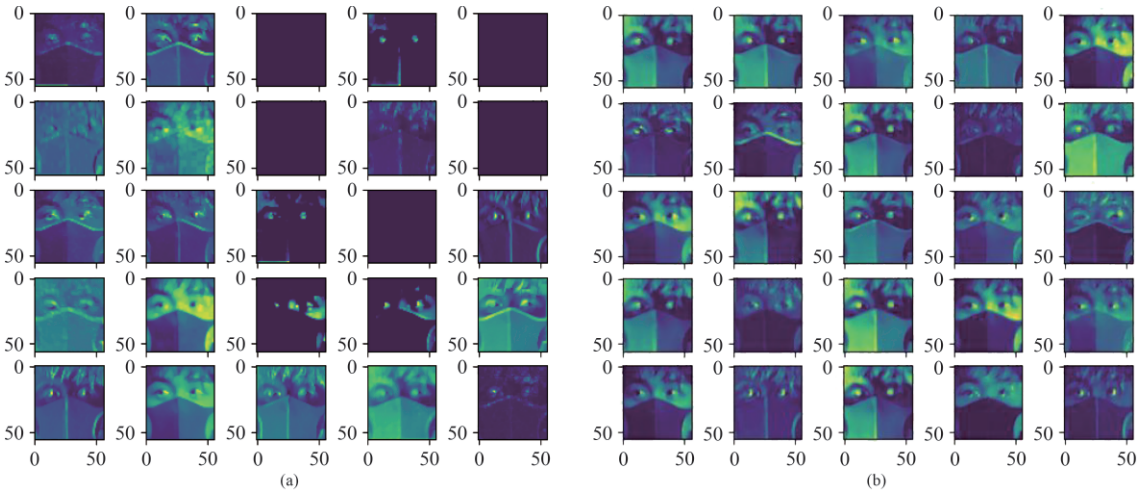

图6 第二层卷积层特征图可视化，(a)为使用原Ghost Module时的结果，(b)为采用HAG时的结果

Fig. 6 Visualization of feature map. The left figure (a) shows the result when using the original Ghost Module, and the right figure (b) shows the result when using HAG

出现特征图为空的问题。

对其原因进行分析，本文认为对于一些离散的数据，由于原始Ghost Module中ReLU激活函数的特性，致使部分神经元“坏死”导致的图6(a)中部分特征图消失，影响了后续的特征提取操作。而图6(b)中，由于HAG中的Hard-Swish激活函数能够把一些负值信息进行激活，使得不丢失负值信息，因此相比左图能够保留更多信息。

3.6 TensorRT加速与参数量化对比

TensorRT[32]为NVIDIA开发的一个专用于NVIDIA系列计算核心的高性能深度学习推理SDK。它包括一个深度学习推理运行优化器，为深度学习推理应用程序提供低延迟和高吞吐量。由于本文实验评估所用的NVIDIA Jetson Xavier NX设备支持该SDK。在深度学习发展前期，例如文献[33]等基于非深度学习模型的人脸检测算法能够实现实时人脸检测，但卷积等深度学习基本模块的运算量远大于之前的方法。因此为了进一步提高模型在嵌入式设备的推理速度以及减少模型的占用空间，在该实验中对比了本文算法使用TensorRT进行模型加速前后以及参数量化的效果，同时也对YOLOv5-s进行了对比，相关结果置于表4。

由表4可见，Ghost-YOLO经过TensorRT加速过后，其单图平均处理时间从24.72 ms减少至18.41 ms，其每秒传输帧数(Frames per second，FPS)约为54.318，并且模型通过FP16量化后，其权重大小从21M缩减为13M；而经过TensorRT加速和参数量化后的YOLOv5-s，其处理时间也仅为21.22 ms，FPS约为47.125，权重大小从29.6M减小至17.5M。可见，经过加速后的Ghost-YOLO与YOLOv5-s在检测上都有提升，并且经过参数量化后，模型占用空间也得到进一步减小，减少了算法在嵌入式设备上的对性能存储空间的需求。

表4 Ghost-YOLO与YOLOv5-s经过TensorRT加速后的性能表现及权重大小对比

Tab. 4 The comparison between Ghost-YOLO and YOLOv5-s when using the TensorRT acceleration

网络结构	TensorRT	单图片处理时间	FPS	权重大小
Ghost-YOLO	Yes	18.41 ms	54.318	13.0M
	No	24.72 ms	40.453	21.0M
YOLOv5-s	Yes	21.22 ms	47.125	17.5M
	No	24.42 ms	40.950	29.6M

4 结论

本文提出了一种面向嵌入式设备的Ghost-YOLO口罩人脸检测算法。先通过所设计的高激活性鬼影(High Active Ghost，HAG)模块来减少特征图中的冗余，再利用HAG实现的高激活性鬼影跨段部分(High Active Ghost Cross Stage Partial，HAG-CSP)模块对YOLOv5-s模型进行轻量化改造，从而构造出口罩人脸检测器。最后的实验结果显示，本文提

出方法在NVIDIA Jetson NX嵌入式设备上，对尺寸为640×640的单图片处理时间为24.72 ms，相比于其他轻量型目标检测模型有更快推理速度且有更高精度的优势。本文不足之处在于HAG模块对前一组特征利用线性转换得到的后一组特征，若前一组特征存在冗余信息，则后一组特征难免相应存在冗余。后续的改进思路可以考虑在HAG模块线性转换得到的特征之间设计约束项，降低特征之间相关性，减少特征冗余。

参考文献

[1] 毛秀萍. 基于DCNN和faster RCNN的人脸检测方法[D]. 东北大学，2017.
MAO Xiuping. Face detection method based on DCNN and faster RCNN [D]. Northeast University, 2017. (in Chinese)

[2] WANG Y, ZHENG J. Real-time face detection based on YOLO[C]//2018 IEEE International Conference on Knowledge Innovation and Invention (ICKII). IEEE, 2018: 221-224.

[3] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: Unified, real-time object detection[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA. IEEE, 2016: 779-788.

[4] LIU Wei, ANGUELOV D, ERHAN D, et al. SSD: Single shot multibox detector[C]//European Conference on Computer Vision (ECCV). Amsterdam, Netherlands. Springer, Cham, 2016: 21-37.

[5] GIRSHICK R, DONAHUE J, DARRELL T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbus, OH, USA. IEEE, 2014: 580-587.

[6] GIRSHICK R. Fast R-CNN[C]//2015 IEEE International Conference on Computer Vision (ICCV). Santiago, Chile. IEEE, 2015: 1440-1448.

[7] REN Shaoqing, HE Kaiming, GIRSHICK R, et al. Faster R-CNN: Towards real-time object detection with region proposal networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137-1149.

[8] HOWARD A G, ZHU Menglong, CHEN Bo, et al. MobileNets: Efficient convolutional neural networks for mobile vision applications [EB/OL]. https://arxiv.org/abs/1704.04861, 2017.

[9] HOWARD A, SANDLER M, CHEN Bo, et al. Searching for MobileNetV3[C]//2019 IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea (South). IEEE, 2019: 1314-1324.

[10] IANDOLA F N, HAN Song, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size[EB/OL]. https://arxiv.org/abs/1602.07360, 2016.

[11] CHOLLET F. Xception: deep learning with depthwise separable convolutions[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA. IEEE, 2017: 1800-1807.

[12] RANGILYU, et al. NanoDet[EB/OL]. 2021. URL https://github.com/RangiLyu/nanodet.

[13] WANG R J, LI X, LING C X. Pelee: A real-time object detection system on mobile devices[J]. Advances in Neural Information Processing Systems (NeurIPS), 2018, 31.

[14] BOCHKOVSKIY A, WANG C Y, LIAO H Y MARK. YOLOv4: optimal speed and accuracy of object detection [EB/OL]. https://arxiv.org/abs/2004.10934, 2020.

[15] REDMON J, FARHADI A. YOLOv3: an incremental improvement [EB/OL]. https://arxiv. org/abs/1804.02767, 2018.

[16] JOCHER G, STOKEN A, BOROVEC J, et al. Yolov5: v3.0[EB/OL]. 2020. DOI: 10.5281/zenodo.3983579. (accessed on 13 August 2020)

[17] LIN T Y, MAIRE M, BELONGIE S, et al. Microsoft coco: Common objects in context[C]//European Conference on Computer Vision (ECCV). Zurich, Switzerland. Springer, Cham, 2014: 740-755.

[18] HAN Kai, WANG Yunhe, TIAN Qi, et al. GhostNet: more features from cheap operations [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA. IEEE, 2020: 1577-1586.

[19] WANG C Y, MARK LIAO H Y, WU Y H, et al. CSPNet: A new backbone that can enhance learning capability of CNN [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Seattle, WA, USA. IEEE, 2020: 1571-1580.

[20] RAMACHANDRAN P, ZOPH B, LE Q V. Swish: a self-gated activation function[EB/OL]. https://arxiv.org/abs/1710.05941v1, 2017.

[21] GLOROT X, BORDES A, BENGIO Y. Deep sparse rectifier neural networks[C]//Proceeding of the 14th International Conference on Artificial Intelligence and Statistics.

Fort Lauderdale, Florida, USA. JMLR Workshop and Conference Proceedings, 2011: 315-323.

[22] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, 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.

[23] 阮海涛, 曾焕强, 朱建清, 等. 基于特征金字塔融合表征网络的跨模态哈希方法[J]. 信号处理, 2021, 37(7): 1252-1259.
RUAN Haitao, ZENG Huanqiang, ZHU Jianqing, et al. Feature pyramid fusion representation network for cross-modal hashing[J]. Journal of Signal Processing, 2021, 37(7): 1252-1259. (in Chinese)

[24] LIN T Y, DOLLÁR P, GIRSHICK R, et al. Feature pyramid networks for object detection[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA. IEEE, 2017: 936-944.

[25] LIU Shu, QI Lu, QIN Haifang, et al. Path aggregation network for instance segmentation[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA. IEEE, 2018: 8759-8768.

[26] REDMON J, FARHADI A. YOLO9000: better, faster, stronger[C]//2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA. IEEE, 2017: 6517-6525.

[27] 李泽琛, 李恒超, 胡文帅, 等. 多尺度注意力学习的Faster R-CNN口罩人脸检测模型[J]. 西南交通大学学报, 2021, 56(5): 1002-1010.
LI Zechen, LI Hengchao, HU Wenshuai, et al. Fast R-CNN mask face detection model based on multi-scale attention learning[J]. Journal of Southwest Jiaotong University, 2021, 56(5): 1002-1010. (in Chinese)

[28] FAN X, JIANG M. RetinaFaceMask: A single stage face mask detector for assisting control of the COVID-19 pandemic[C]// 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2021: 832-837.

[29] DENG J, GUO J, VERVERAS E, et al. Retinaface: Single-shot multi-level face localisation in the wild[C]// 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Seattle, WA, USA. IEEE, 2020: 5203-5212.

[30] FAN X, JIANG M, YAN H. A deep learning based light-weight face mask detector with residual context attention and Gaussian heatmap to fight against COVID-19[J]. IEEE Access, 2021, 9: 96964-96974.

[31] HE Kaiming, ZHANG Xiangyu, REN Shaoqing, et al. Deep residual learning for image recognition[C]//2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA. IEEE, 2016: 770-778.

[32] RAJEEV R, KEVIN C, LEI M, et al. TensorRT[EB/OL]. 2020. URL https://github.com/NVIDIA/TensorRT.

[33] 蔡灿辉, 朱建清. 采用Gentle AdaBoost和嵌套级联结构的实时人脸检测[J]. 信号处理, 2013, 29(8): 956-963.
CAI Canhui, ZHU Jianqing. Real-time face detection using Gentle AdaBoost and nested cascade structure[J]. Journal of Signal Processing, 2013, 29(8): 956-963. (in Chinese)

作者简介

陈继平 男,1996年生,广东惠州人。华侨大学工学院硕士研究生,主要研究方向为图像目标检测与识别。
E-mail: emjp@stu. hqu. edu. cn

陈永平 男,1998年生,湖南长沙人。华侨大学工学院硕士研究生,主要研究方向为图像目标检测与识别。
E-mail: 20014084003@stu. hqu. edu. cn

谢 懿 男,1996年生,广东河源人。华侨大学工学院硕士研究生,主要研究方向为模式识别与机器视觉。
E-mail: yixie@stu. hqu. edu. cn

朱建清(通讯作者) 男,1987年生,福建莆田人。华侨大学工学院教授,博士,主要研究方向为图像处理、模式识别与机器视觉。
E-mail: jqzhu@hqu. edu. cn

曾焕强 男,1984年生,福建惠安人。华侨大学工学院、信息科学与工程学院教授,工学院院长,博士,IEEE高级会员,中国电子学会信号处理分会委员。主要研究方向为图像处理、视频编码、计算机视觉。
E-mail: zeng0043@hqu. edu. cn