



## 基于敏感度的YOLO网络集成剪枝算法

张江永<sup>1,2</sup>, 徐智勇<sup>1</sup>, 张建林<sup>1</sup>, 许 涛<sup>3</sup>

(1. 中国科学院光电技术研究所, 成都 610209; 2. 中国科学院大学 电子电气与通信工程学院, 北京 100049;  
3. 中国船舶工业系统工程研究院, 北京 100036)

**摘 要:** 深层卷积神经网络所需的计算量和存储空间严重制约了其在资源有限平台上的应用与部署。针对基于单一参数重要性评价或者特征重建的剪枝算法泛化能力较差的问题, 提出基于敏感度的集成剪枝算法, 利用BN层的缩放因子稀疏YOLO网络中卷积核个数较多的冗余层, 结合3种参数重要性评价方法对卷积核做重要性排序, 并根据敏感度确定每一层的剪枝比率。实验结果表明, 该剪枝算法对于YOLOv3和YOLOv3-tiny网络分别缩减80.5%和92.6%的参数量, 并且相比基于网络轻量化方法的剪枝算法提升了网络模型压缩后的检测精度和泛化能力。

**关键词:** 卷积神经网络; 敏感度; 集成剪枝算法; YOLO网络; 重要性评价

开放科学(资源服务)标志码(OSID):



中文引用格式: 张江永, 徐智勇, 张建林, 等. 基于敏感度的YOLO网络集成剪枝算法[J]. 计算机工程, 2021, 47(9): 59-68.

英文引用格式: ZHANG J Y, XU Z Y, ZHANG J L, et al. Sensitivity-based integrated pruning algorithm for YOLO network[J]. Computer Engineering, 2021, 47(9): 59-68.

## Sensitivity-based Integrated Pruning Algorithm for YOLO Network

ZHANG Jiangyong<sup>1,2</sup>, XU Zhiyong<sup>1</sup>, ZHANG Jianlin<sup>1</sup>, XU Tao<sup>3</sup>

(1. Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu 610209, China;

2. School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China;

3. China State Shipbuilding Corporation Systems Engineering Research Institute, Beijing 100036, China)

**[Abstract]** Deep Convolutional Neural Network(CNN) require considerable storage space and operation times, which hampers their application and deployment on platforms with limited resources. The pruning algorithms can alleviate this problem, but the ones based on a single method for parameter importance evaluation or feature reconstruction have poor generalization performance. To address the problem, an integrated pruning algorithm based on sensitivity is proposed. The algorithm employs sparse scaling factor of BN layer to reduce the density of the layers with many convolutional kernels in YOLO. Then three methods for parameter importance evaluation are used to sort the convolutional kernels by importance. The ratio of to-be-pruned parts of each layer is determined according to the sensitivity. Experimental results show that the proposed algorithm reduces the parameter number of YOLOv3 by 80.5% and YOLOv3-tiny by 92.6%. Compared with the pruning algorithm based on network lightweight method, the proposed algorithm can better improve the detection accuracy and the generalization performance of the pruned model.

**[Key words]** Convolutional Neural Network (CNN); sensitivity; integrated pruning algorithm; YOLO network; importance evaluation

DOI: 10.19678/j.issn.1000-3428.0058784

### 0 概述

自从2012年AlexNet问世以来, 深度卷积神经

网络(Convolutional Neural Network, CNN)在图像处理 and 自然语言处理等任务中取得了重要成果。为增强卷积神经网络的非线性拟合能力, 网络通常被设

基金项目: 国家重点研发计划(G158207)。

作者简介: 张江永(1995—), 男, 硕士研究生, 主研方向为机器学习、深度学习、模型压缩; 徐智勇(通信作者)、张建林, 研究员; 许 涛, 高级工程师。

收稿日期: 2020-06-29 修回日期: 2020-08-17 E-mail: zhang709478621@163.com

计得更宽和更深,并且需要大量的存储空间和计算资源,导致其不适用于算力有限的硬件设备。因此,针对大型卷积神经网络进行压缩,在确保精度损失较少的前提下,设计占用存储量、前向推理计算量和能源消耗量均减少的轻量化网络模型就显得十分重要,而轻量化方法主要包括剪枝<sup>[1-2]</sup>、低秩分解<sup>[3]</sup>、量化<sup>[4]</sup>、知识蒸馏<sup>[5-6]</sup>等。由于卷积神经网络中包含卷积层和全连接层等人为设定的训练参数模块,这种凭借经验通过重复实验得到的局部最优超参数不能代表网络的实际需求。根据深度卷积神经网络的新双U型偏差-方差风险曲线<sup>[6]</sup>可知,在不考虑资源限制的条件下,参数越多、模型越复杂的网络性能表现越好,但是过多参数的复杂模型没有权衡成本和性能间的关系并且模型存在冗余。剪枝的目的是通过剪除网络中冗余的模块,从过参数化的复杂模型中得到参数较少的轻量化模型,该模型处于双U型曲线的第一极小值点。

剪枝按照粒度的粗细可以分为非结构化剪枝和结构化剪枝两种。非结构化剪枝主要是对权重进行修剪,LECUN等<sup>[7]</sup>通过删除目标函数的参数二阶导较小的神经元以稀疏权重,HAN等<sup>[8]</sup>提出基于参数值的剪枝方法去除冗余权重,GUO等<sup>[9]</sup>提出动态网络剪枝方法,可恢复已被剪掉的重要神经元,避免了错误修剪导致的精度损失。结构化剪枝由于内部非规整的连接,需要依赖专门运行库和硬件设备,因此卷积核(指三维卷积核,即滤波器)层面的结构化剪枝更受青睐。在卷积核粒度层面的结构化剪枝中,一旦某个卷积核或者通道被视为不重要,则会被整个移除,因此需要得到整个卷积核或者通道的重要性。LI等<sup>[10]</sup>将卷积核权重的L1范数值作为得分判断卷积核的重要性。HU等<sup>[11]</sup>通过计算卷积核权重的稀疏程度,得出越稀疏的核越不重要的结论。WANG等<sup>[12]</sup>利用Group Lasso得到稀疏化的卷积核,以便于进一步剪枝。YE等<sup>[13]</sup>对“较小范数不重要”

准则进行重新思考,认为较小的范数可能是对信息的补充。

除了从参数层面评估卷积核的重要性以外,还有基于特征重建的剪枝方式,通过最小化重建误差获取轻量化模型。LUO等<sup>[14]</sup>使用贪心算法,在每一层的卷积核中寻找合适的子集代替原有的集合,如果可以得到类似于原来的特征,则可去除子集外的卷积核。ZHUANG等<sup>[15]</sup>在fine-tuning和剪枝阶段引入鉴别力感知的辅助损失,增强剪枝后所保留特征的鉴别能力。HE等<sup>[16]</sup>认为位于或者接近几何中位数的卷积核是冗余的,可以用边缘的剩余卷积核代替。LIU等<sup>[17]</sup>提出网络轻量化方法,通过在训练过程中将BN层的缩放因子的稀疏性损失加入网络本身的损失函数中,使网络在训练时自主稀疏缩放因子,基于缩放因子的大小决定卷积核是否删减,但如果层之间的稀疏不均匀,则会导致网络中的层间卷积核个数不均匀。本文提出针对YOLO网络的基于敏感度的集成剪枝算法,利用稀疏缩放因子方式稀疏YOLO网络中卷积核个数较多的层,综合3种参数重要性评价方法对卷积核做重要性排序,通过网络中每个剪枝层对精度下降的敏感度确定剪枝比率。

## 1 基于敏感度的YOLO网络集成剪枝

### 1.1 针对YOLO网络的结构化剪枝

结构化剪枝可表示为如下形式:

$$\min_{\mathbf{W}} L(D; \mathbf{W}) + \lambda \|\mathbf{W}\|_0. \quad (1)$$

其中: $L$ 是网络损失函数; $D$ 是预测值和真实值的差; $\mathbf{W}$ 是卷积核的权重参数; $\lambda$ 是平衡剪枝比率和损失的系数。剪枝的目标为在减少卷积核数目的前提下,尽量使损失函数最小化。本文采用结构化剪枝,剪去重要性低的卷积核,并通过fine-tuning得到具有较高网络精度的精简模型。YOLO网络剪枝流程如图1所示。

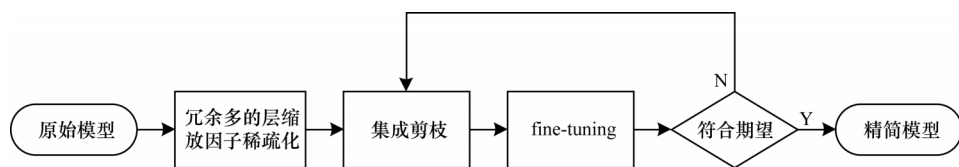


图1 YOLO网络剪枝流程

Fig.1 Procedure of YOLO network pruning

YOLOv3<sup>[18]</sup>采用darknet-53的backbone,同时借鉴ResNet<sup>[19]</sup>的残差结构,采用5个残差单元(共53个卷积层)进行5次降采样。在输入到yolo层之前,为避免深层特征提取的信息不足以判断类别,采用concat的方式进行特征叠加,提高识别精度。由于残差结构连接了不同层,如果剪除残差结构中某一层的卷积核,在进行剪枝时要维持输入输出的维度一致,需要对与其连接的所有层进行对应个数的卷积核剪除。为避免这一问题,在对YOLOv3进行剪枝时,仅对残差块中间的卷积层(共42个卷积层)进行卷积核筛选。YOLOv3残差结构如图2所示。

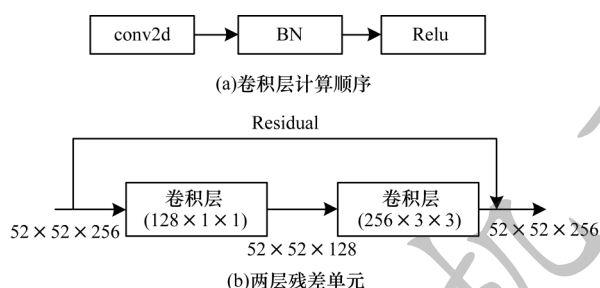


图2 YOLOv3残差结构

Fig.2 Residual structure in YOLOv3

YOLOv3-tiny采用简单的直筒式结构,为达到更快的检测速度,剪除了一些特征层和残差层,只采用2个yolo层检测不同尺度。YOLOv3-tiny网络共23层,其中可剪枝的卷积层共10个,其余的是pooling层和yolo层前的卷积层。

## 1.2 集成剪枝算法

### 1.2.1 冗余层稀疏

根据文献[20]研究,近些年来被提出的一些参数重要性评价方法有很强的相似性,依据的参数都近似符合类似高斯分布。如果按照这些准则进行剪枝,可以得到近似的卷积核重要性排序,但是当某一卷积层存在过多的卷积核时,网络训练会导致卷积层形成特殊的空间几何结构,参数重要性评价方法不能有效区分卷积核重要性。YOLOv3网络中的卷积核个数相差较大,可剪枝层中最少的层有32个卷积核,最多的层有1024个卷积核。YOLOv3-Tiny网络中最少的层只有16个卷积核。面对卷积核个数差异巨大的剪枝层,通过稀疏缩放因子方法可以解决上述问题。BN层<sup>[21]</sup>的作用是使每一层的激活输入值在训练时保持相同分布,避免内部协变量偏移问题,使得激活输入值始终处于敏感区,远离导数饱和区,解决了反向传播的梯度消失问题。

卷积层对输入特征图的处理如下:

$$x = Wu + B \quad (2)$$

其中: $u$ 是输入值; $B$ 是偏置。

对 $x$ 进行正则化,使其符合标准正态分布:

$$\hat{x} = \frac{x - \mu_{\zeta}}{\sqrt{\sigma_{\zeta}^2 + \varepsilon}} \quad (3)$$

其中: $\mu_{\zeta}$ 和 $\sigma_{\zeta}$ 代表输入 $x$ 的小批次的均值和方差; $\varepsilon$ 是一个极小值。 $\hat{x}$ 符合标准正态分布,使用式(4)保证非线性,而 $s_{scale}$ 和 $s_{shift}$ 是网络中的可学习参数,根据不同的实例将标准正态分布进行尺度变换和平移:

$$y = s_{scale} \times \hat{x} + s_{shift} \quad (4)$$

稀疏缩放因子方法针对的是网络中BN层的 $s_{scale}$ 参数,通过对缩放因子L1正则化将其不断向0推进,因此与接近0的缩放因子一一对应的卷积核对网络性能的贡献较小,可以将其剪除而不会导致过多的精度损失。

在训练过程中,对缩放因子进行稀疏性正则化处理:

$$L = \sum_{(x,y)} l(f(x,W),y) + \lambda \sum_{\lambda \in L'} g(\gamma) \quad (5)$$

其中: $W$ 是卷积核矩阵; $x$ 和 $y$ 是输入和标签; $g(\gamma) = |\gamma|$ 是稀疏诱导惩罚函数, $\lambda$ 将原始损失和进行稀疏平衡。

通过对卷积核多的层(实验中选择512和1024个卷积核的层)稀疏训练得到稀疏化模型,此时参数重要性评价方法可以判断卷积核重要性,再使用集成剪枝算法进行剪枝。

### 1.2.2 参数重要性评价方法

为避免一种参数重要性评价方法对于YOLO网络的卷积核重要性判断存在片面性,本文采用结合L1范数、激活值和梯度3种参数重要性评价方法的集成剪枝算法。首先分析上述3种参数重要性评价方法的剪枝原理,说明根据每种评价方法的剪枝合理性,然后比较基于3种参数重要性评价方法的剪枝算法和集成剪枝算法的剪枝效果,以展示集成剪枝在YOLO网络上的优越性。集成剪枝算法同时采用3种评价方法得到的3个卷积核排序,三者之间没有优先级关系。

1) L1范数。将卷积核的权重的绝对值相加得到L1范数 $\|W_{i,j}\|_1$ ,然后将所有卷积核的L1范数进行排序,由于在网络训练过程中不重要的神经元在优化时权重会逐渐趋近于0,因此权重的绝对值较小的卷积核对于最终结果的影响是相应较小的,可在剪枝过程中对其进行剪除。

2) 激活值。在网络训练过程中,为能够分辨不

同特征的重要性,会尽可能保留对于判定类别有重要影响的特征,其他特征则被视为噪声尽量剪除。这一行为体现在卷积后激活得到的特征图上,重要的特征对应的是高响应值,其他特征对应的是低响应值,因此根据激活值大小 $\|W_{i,j} \otimes X_i\|_1$ 对卷积神经网络进行剪枝。

3)梯度。卷积神经网络通过反向传播进行参数优化,在训练过程中根据式(6)更新参数。SUN等<sup>[22]</sup>提出的meProp方法仅更新梯度最大的 $k$ 个神经元,由于一些梯度为0的神经元会加速反向传播,梯度大小与神经元的重要性是相关的,因此仅保留梯度

大的卷积核。

$$W' = W - \eta \frac{\partial L}{\partial W} \quad (6)$$

图3是YOLOv3网络预训练后第41层和稀疏训练后第76层的3种参数的分布情况,选用这两层是因为它们的参数分布在稀疏前后更具代表性。L1范数、激活值和梯度在稀疏前都是类似正态分布,稀疏后的参数中心较为靠近0。稀疏前的参数差异是明显存在的,可以根据分布将正态分布中心前的一部分卷积核剪除,因为这些卷积核的3种参数值都较小,而稀疏后的参数差异更明显,可以将靠近0的参数代表的卷积核剪除。

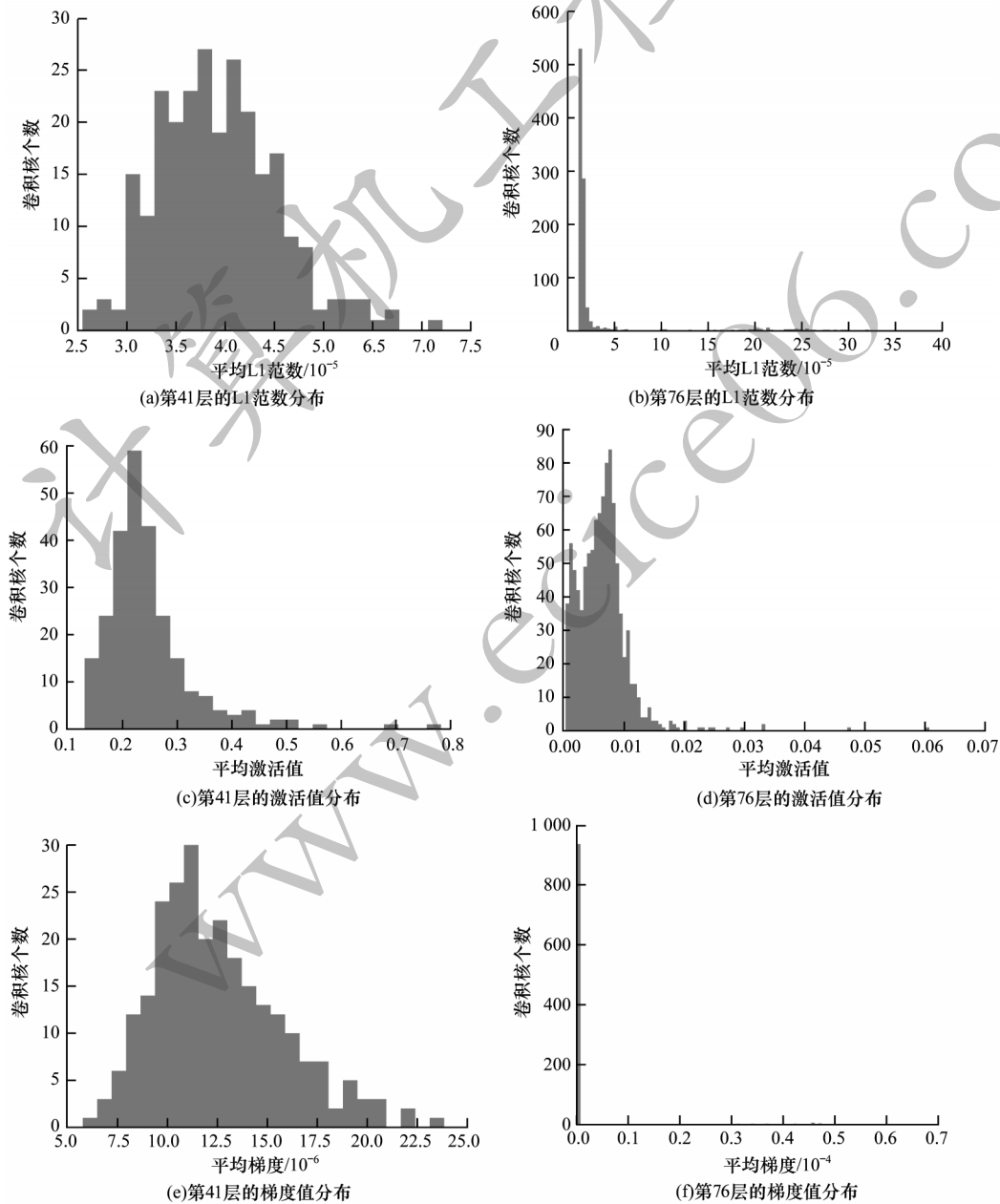


图3 3种参数稀疏前后的分布情况

Fig.3 Distribution of three parameters before and after sparse



通过比较集成剪枝算法和基于L1范数重要性评价方法的剪枝算法(简称为基于L1范数的剪枝算法)、基于激活值重要性评价方法的剪枝算法(简称为基于激活值的剪枝算法)、基于梯度重要性评价方法的剪枝算法(简称为基于梯度的剪枝算法)的剪枝效果,验证集成剪枝算法的优越性。选择YOLOv3-tiny网络进行实验,对网络每一剪枝层的卷积核按照上述4种重要性评价方法排序后剪枝50%,得到同等结构的不同剪枝策略下的精简模型,并且各自微调100代后得到100组微调的平均精度均值(mean Average Precision, mAP)数据,拟合后的曲线如图4所示。可以看出,利用基于3种单一参数重要性评价方法的剪枝算法得到的精简模型在训练时的曲线比较接近,说明它们的剪枝效果十分接近,而集成剪枝算法得到的曲线在训练5代以后一直处于前3种算法训练曲线的上方,表明集成剪枝算法的剪枝效果优于基于单一重要性评价方法的剪枝算法。

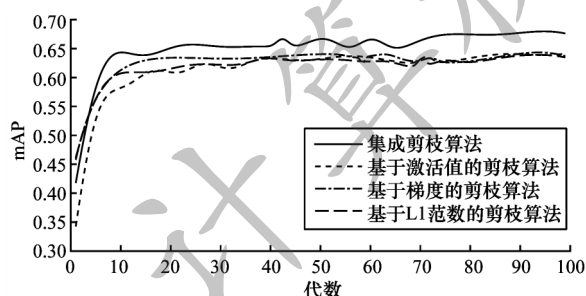


图4 4种剪枝算法的剪枝效果比较

Fig.4 Comparison of pruning effect of four pruning algorithms

4种集成剪枝算法的检测精度如表1所示。

表1 4种剪枝算法的检测精度比较

Table 1 Comparison of detection accuracy of four pruning algorithms

剪枝算法	检测精度
基于L1范数的剪枝算法	0.680
基于激活值的剪枝算法	0.682
基于梯度的剪枝算法	0.681
集成剪枝算法	0.690

为加快训练速度,在训练过程中采用的置信度阈值为0.1,以减少检测框的数量。在测试训练的最后一代时采用的置信度阈值为0.001,因此mAP在最后一代会有一个向上的跳变(在图4中采用的是前99代的数据拟合曲线,未体现出该跳变),最终集成

剪枝算法得到的模型检测精度最高。

### 1.2.3 集成剪枝的参数重要性评价方法

每种参数重要性评价方法都考虑每层的所有卷积核的不同参数,如果要根据某一参数对卷积核做筛选,需要计算每个卷积核对应参数的值,然后将同一层卷积核的该参数的对应值做排序。为使网络精度损失最小,需要剪除排序靠后的部分卷积核,即不重要的卷积核。

在同一层中,不同卷积核对该层在网络整体性能中的贡献是不同的,不同的参数重要性评价方法对同一卷积核在该层的排序也是不同的,因此不同的参数重要性评价方法剪枝后得到的精简模型有所差别。在理想情况下,为能够使不同的参数重要性评价方法相结合,将其写成一个数学表达式的形式,但是本文采用的L1范数、梯度和激活值3种参数之间的数值差可能有好几个数量级,而且在数值空间上的分布也不一致,因此单纯采用相加、相乘等方式得到的排序剪枝的效果可能还不如单一参数重要性评价方法。

由于每种参数重要性评价方法都能够得到每一层卷积核的重要性排序,因此将该排序当作新的评价参数,此时每一个剪枝层都有3组排序,把3组排序相加即可得到一组新的卷积核重要性排序。这组新的排序集成了3种方法对每一个卷积核的重要性判断,相比单一方法得到的判断更全面,可以得到更接近最优解的卷积核保留集合,减少了单一参数重要性评价方法的片面性。图4的实验结果也证实了集成剪枝算法得到的精简模型微调后的检测精度优于基于单一参数重要性评价方法的剪枝算法。

假设对第 $L$ 层(共 $m$ 个卷积核)进行剪枝,卷积核权重集合为 $\{W_{l,1}^0, W_{l,2}^0, \dots, W_{l,m}^0\}$ (图5中省略了卷积层参数 $l$ 和上标 $0$ ,其中 $0$ 表示卷积核原始权重)。从训练集中随机选择256张图片得到网络可剪枝层的每个卷积核的平均L1范数、该层的每个卷积核的平均激活值以及反向传播时的平均梯度(取绝对值的平均值),对该层所有卷积核按照3种方法对应的值分别进行从大到小的排序,得到每一个卷积核在序列中的对应位置,然后将每个卷积核的3个序号对应相加作为最终的重要性得分,据此去除得分低的 $m-n$ 个卷积核得到新的权重集合 $\{W'_{l,1}, W'_{l,2}, \dots, W'_{l,n}\}, n < m$ 。

	L层															
卷积核	$W_1$	$W_2$	$W_3$	$W_4$	$W_5$	$W_6$	$W_7$	$W_8$	$W_9$	$W_{10}$	$W_{11}$	$W_{12}$	$W_{13}$	$W_{14}$	$W_{15}$	$W_{16}$
L1范数	15	1	10	7	5	13	0	4	8	12	9	6	11	2	3	14
激活值	+															
梯度	13	3	14	5	6	11	2	7	12	8	4	9	10	0	1	15
	+															
最终得分	10	5	13	2	8	15	4	6	7	9	11	3	12	1	0	14
排序	38	9	37	14	19	39	6	17	27	29	24	18	33	3	4	43
剪枝50%	13	3	12	4	7	14	2	5	9	10	8	6	11	0	1	15
剪枝50%后的卷积核	$W_1$	$W_2$	$W_3$	$W_4$	$W_5$	$W_6$	$W_7$	$W_8$	$W_9$	$W_{10}$	$W_{11}$	$W_{12}$	$W_{13}$	$W_{14}$	$W_{15}$	$W_{16}$
	$W_2$	$W_4$	$W_5$	$W_7$	$W_8$	$W_{12}$	$W_{14}$	$W_{15}$								

图5 集成剪枝的卷积核选择过程

Fig.5 Convolution kernel selection process of integrated pruning

### 1.3 基于敏感度的剪枝比率确定

在剪枝实验时需要多个层进行剪枝,上述方法针对的是单层的卷积核排序,因此需要根据总体剪枝比率确定每一层的剪枝比率,在这种情况下需要考虑多层剪枝时的精度损失均衡问题,以避免对某一层剪枝比率过大出现短板效应,导致精度损失严重, fine-tuning 后精度恢复达不到理想效果。为解决这一问题,在所有层的重要性排序完成后,首先用验证集测试网络原始精度  $m_{\text{mAP}_0}$ ,然后从第一个待剪枝的层开始,剪除重要性较低的60%的卷积核(将BN的缩放因子设置为0),对验证集测试得到新的精度  $m_{\text{mAP}_i}$ ,根据  $m_{\text{mAP}_0} - m_{\text{mAP}_i}$

得到精度差值  $\Delta m_{\text{mAP}_i}$ ,再除以该层60%的卷积核个数近似得到这些卷积核对网络精度造成的平均损失  $\overline{\Delta m_{\text{mAP}_i}}$ ,对其他待剪枝层执行相同的操作得到每层对网络精度造成的平均损失的集合  $\{\overline{\Delta m_{\text{mAP}_1}}, \overline{\Delta m_{\text{mAP}_2}}, \overline{\Delta m_{\text{mAP}_3}}, \overline{\Delta m_{\text{mAP}_4}}, \dots, \overline{\Delta m_{\text{mAP}_p}}\}$ ,将该层每个卷积核的平均损失精度称为该层的剪枝敏感度。

首先根据平均损失集合和需要剪除的卷积核总数,求解式(7)得到每一层剪除的卷积核个数,其中式(7)的矩阵表示为式(8);然后执行剪枝操作,按照得到的剪枝数量逐层进行剪枝;最后通过 fine-tuning 恢复网络精度。

$$\begin{cases} k_1 + k_2 + k_3 + k_4 + \dots + k_p = N \times K\% \\ k_1 \overline{\Delta m_{\text{mAP}_1}} = k_2 \overline{\Delta m_{\text{mAP}_2}} = k_3 \overline{\Delta m_{\text{mAP}_3}} = k_4 \overline{\Delta m_{\text{mAP}_4}} = \dots = k_p \overline{\Delta m_{\text{mAP}_p}} \end{cases} \quad (7)$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ \overline{\Delta m_{\text{mAP}_1}} & -\overline{\Delta m_{\text{mAP}_1}} & 0 & 0 & \dots & 0 \\ \overline{\Delta m_{\text{mAP}_2}} & 0 & -\overline{\Delta m_{\text{mAP}_2}} & 0 & \dots & 0 \\ \overline{\Delta m_{\text{mAP}_3}} & 0 & 0 & -\overline{\Delta m_{\text{mAP}_3}} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{\Delta m_{\text{mAP}_p}} & 0 & 0 & 0 & \dots & -\overline{\Delta m_{\text{mAP}_p}} \end{bmatrix} \times \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ \vdots \\ k_p \end{bmatrix} = \begin{bmatrix} N \times K\% \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (8)$$

在式(7)中: $k$ 代表每一剪枝层通过计算得到的剪除的卷积核个数,下标代表的是剪枝层的序号,共  $p$  个待剪枝层; $N$ 是所有剪枝层的卷积核总数, $K$ 是本次剪枝期望的总体剪枝比率;第1个等式代表各剪枝层所有剪除的卷积核总数,与所有剪枝层的卷积核总数乘以期望的整体剪枝比率是相等的;第2个等式用来确定不同剪枝层的剪除卷积核个数间的比

率。为避免出现短板效应,假设每层在剪除一部分卷积核后剩下的卷积核对网络性能的贡献是相当的,不会出现明显差异,否则会导致精度微调后恢复不到理想精度。由于不同剪枝层中的每个卷积核对性能的贡献不同,逐次单独测算一个卷积核对性能贡献又不现实,因此在实际剪枝时采用测试每一层剪枝排序靠后的60%的卷积核,经过计算得到

$\Delta m_{\text{mAP}}$ , 该平均损失精度可以当作该层 60% 的卷积核的精度贡献。为避免出现短板效应, 需要使每一层剪枝时剪除的卷积核对网络造成的精度损失是相等的, 也就是剩下的卷积核对检测精度的贡献也是相等的, 如图 6 所示。式(8)将式(7)写成了矩阵形式, 计算得到每一个剪枝层应剪除的卷积核个数。

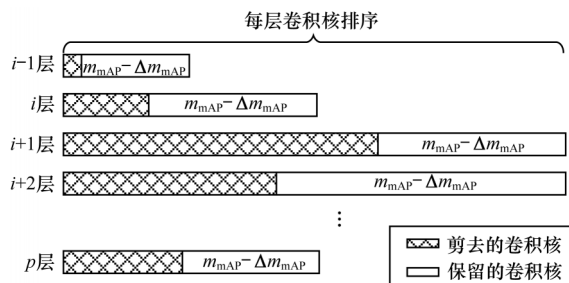


图6 基于敏感度的剪枝比率确定

Fig.6 Determination of pruning ratio based on sensitivity

在实际操作过程中, 每次剪除总数 50% 的卷积核, 确保检测得到的敏感度在适用范围内。为避免出现求得的解过大或者过小, 对每一层设置最高剪枝阈值为 60%、最低剪枝阈值为 25%, 经过多次剪枝得到精简模型。

#### 1.4 基于敏感度的YOLO网络集成剪枝步骤

通过稀疏卷积核多的层后进行集成剪枝, 可避免集成剪枝算法无法处理冗余多的层及区分卷积核重要性的问题, 使剪枝泛化能力更好, 同时有利于获得均匀模型和更高的剪枝精度。基于敏感度的YOLO网络集成剪枝步骤具体如下:

**输入** VOC数据集, YOLO原始网络的cfg配置文件和初始化权重, 期望剪枝比率K%

**输出** 剪枝的YOLO模型cfg配置文件及对应权重

1) 预训练YOLO模型。

2) 确定剪枝层, 对YOLO模型冗余多的剪枝层使用稀疏缩放因子方法进行稀疏训练。

3) 使用集成剪枝算法, 基于3种参数重要性评估方法对每一剪枝层的卷积核进行重要性排序, 将每个卷积核的3种排序相加得到卷积核的重要性得分。

4) 在验证集上测试得到原始精度, 将每一个剪枝层的重要性低的K%的卷积核剪除, 在验证集上测试得到新精度, 原始精度减去新精度, 除以剪掉的卷积核个数得到剪除的每个卷积核对精度的贡献。

5) 求解式(7)得到每一剪枝层应剪除的卷积核个数。

6) 对网络模型按式(5)中的每一剪枝层应减卷积核个数进行逐层剪枝, 得到精简模型。

7) 对精简模型进行fine-tuning操作恢复模型精度。

8) 如果卷积核剪枝数目达不到期望值, 则返回步骤4。

## 2 实验与结果分析

实验对YOLOv3网络使用VOC数据集剪枝, 但对从VOC2007和VOC2012的train+val数据中提取出的行人类别进行网络剪枝, 从模型大小、mAP、推理时间、参数量、卷积核个数和计算量6个方面, 对原始YOLOv3模型与本文模型和文献[17]模型进行对比。

### 2.1 YOLOv3实验效果对比

实验平台CPU为i5-9600KF, 显卡为RTX2060 Super。实验利用VOC2007和VOC2012的train+val数据进行训练, 采用VOC2007的test数据进行测试。原始YOLOv3模型与两种轻量化模型性能比较结果如表2所示, 其中计算量为每秒10亿次浮点运算。

表2 YOLOv3与两种轻量化模型性能比较

Table 2 Performance comparison among YOLOv3 and two lightweight models

模型	模型大小/MB	mAP	推理时间/s	参数量	卷积核个数	计算量
原始YOLOv3模型	246.8	0.770	0.015 5	61 626 049	13 376	20.496
文献[17]模型	48.7	0.716	0.009 3	12 110 248	2 007	2.553
本文模型	48.4	0.718	0.009 1	12 033 588	2 300	1.765

由表2可以看出, 文献[17]模型和本文模型相比原始YOLOv3模型, 在mAP上都有一定的下降, 原因在于检测网络相较分类网络不单需要识别, 还要回归以确定目标位置, 在针对20类目标时需要较高的网络性能, 剪枝在达到一定比率的阈值后精度会开始快速下降。两种轻量化模型的模型大小、推理时间、参数量、剪枝层卷积核个数、剪枝层计算量分别缩减为原始YOLOv3模型的19.7%和19.6%、60%和59%、19.7%和19.5%、15%和17.2%、12.5%和8.6%, 由此可得本文模型

的参数量缩减了80.5%, 并且检测精度相对文献[17]模型提升了0.2%。可见, 使用基于敏感度的集成剪枝算法得到的网络模型相比文献[17]基于网络轻量化方法的网络模型参数量、模型规模和计算量更小, 推理速度更快且mAP更高。

表3比较了原始YOLOv3模型与两种轻量化模型对于20个类别的检测精度, 原始YOLOv3模型的检测精度在所有类别中均是最高, 本文模型在各类别的检测精度上相比文献[17]模型更具优势。



表3 YOLOv3与两种轻量化模型精度比较  
Table 3 Accuracy comparison among YOLOv3  
and two lightweight models

类别	原始YOLOv3模型	文献[17]模型	本文模型
全部	0.770	0.716	0.718
飞机	0.807	0.741	0.745
自行车	0.871	0.822	0.831
鸟	0.762	0.664	0.684
船	0.671	0.578	0.573
瓶子	0.627	0.555	0.529
巴士	0.845	0.789	0.798
汽车	0.885	0.845	0.854
猫	0.875	0.817	0.82
椅子	0.552	0.492	0.495
牛	0.780	0.773	0.742
餐桌	0.713	0.662	0.676
狗	0.846	0.781	0.789
马	0.844	0.817	0.833
自行车	0.854	0.825	0.815
行人	0.843	0.817	0.806
盆栽	0.506	0.46	0.463
羊	0.817	0.762	0.745
沙发	0.682	0.604	0.616
火车	0.831	0.813	0.833
摩托车	0.783	0.696	0.719

## 2.2 YOLOv3-tiny实验效果对比

实验环境设置同YOLOv3,由于YOLOv3-tiny网络层数较少(23层),因此网络能力有限,实验仅针对从VOC2007和VOC2012的train+val数据中提取出的行人类别进行网络剪枝,利用VOC2007的test数据进行测试,并在模型大小、mAP、推理时间等方面,对YOLOv3-tiny原始模型与本文模型和文献[17]模型进行比较,如表4所示。

由于YOLOv3-tiny网络较浅,GPU的算力完全能够满足网络需求,而算力较弱的NVIDIA Jetson TX2开发平台也能较好地展现剪枝后模型的速度提升情况。由表4可以看出,文献[17]模型和本文模型在精度轻微下降的情况下,得到的轻量化模型大小、前向推理时间、参数量、剪枝层卷积核个数、剪枝层计算量分别缩减为原始YOLOv3-tiny模型的8.6%和7.5%、41.2%和28%、8.6%和7.4%、32.7%和21.9%、28.75%和13.47%,由此可得本文模型的参数量缩减了92.6%,并且检测速度相比文献[17]模型更快,检测精度提升了0.2%。

表4 YOLOv3-tiny与两种轻量化模型性能比较  
Table 4 Performance comparison among YOLOv3-tiny and two lightweight models

模型	模型大小/MB	mAP	前向推理时间/s	参数量	卷积核个数	计算量	TX2推理时间/ms
原始YOLOv3-tiny模型	34.7	0.736	0.001 8	8 669 876	2 048	5.843	51
文献[17]模型	3.0	0.689	0.001 7	743 153	669	1.680	21
本文模型	2.6	0.691	0.001 7	645 101	448	0.787	14

表5给出了原始YOLOv3-tiny模型与两种轻量化模型的各卷积层卷积核个数,可以看出相比文献[17]网络轻量化方法,本文集成剪枝算法对各剪枝层的卷积核都有相当程度的剪除,说明对于网络各剪枝层的能力贡献判断更加全面,得出的网络卷积核分布更加均匀。文献[17]网络轻量化方法由于

没有在训练中控制各剪枝层的稀疏比例,可能导致有些层稀疏程度较大,有些层稀疏程度较小,因此剪枝后各层之间的卷积核个数差异仍然较大,而卷积核多的层会对网络有较大的贡献,但是在卷积核总数一定时,可能会加重卷积核较少层的负担,导致精度下降严重。

表5 YOLOv3-tiny与两种轻量化模型结构比较  
Table 5 Structure comparison among YOLOv3-tiny and two lightweight models

剪枝层	原始YOLOv3-tiny模型卷积核个数	文献[17]模型		本文模型	
		卷积核个数	同层剪枝比率/%	卷积核个数	同层剪枝比率/%
0	16	15	6.25	9	43.80
2	32	32	0.00	17	46.90
4	64	63	1.50	34	46.90
6	128	110	21.90	66	48.40
8	256	111	56.60	131	48.80
10	512	70	86.30	117	77.10
12	1 024	512	50.00	218	78.70
13	256	39	84.80	58	77.30
14	512	71	86.10	140	72.70
21	256	47	81.60	32	87.50



### 2.3 网络剪枝层对网络性能的贡献分析

为进一步说明本文基于敏感度的集成剪枝算法对各剪枝层剪枝比率的选择合理性,对直筒式架构的YOLOv3-tiny网络进行网络性能贡献实验。YOLOv3-tiny共10个剪枝层,对每一个剪枝层利用集成剪枝算法剪去排序靠后的50%的卷积核,然后微调网络100代,得到最终的单层剪枝的精简模型并测试其精度,比较剪枝前后的精度差。共做10组实验,将得到的10个精度差作为该层对网络整体性能贡献的测度值,精度差越大说明该层对网络性能的贡献越大,反之越小。

在表6中对10个剪枝层分别做剪枝50%的实验,微调精度是剪枝后微调网络100代得到的模型的检测精度,使用原始精度剪去微调精度得到每一层剪枝50%后的损失精度。将每层的损失精度和同层的剪枝比率(本文基于敏感度的集成剪枝算法得到的最终剪枝模型的剪枝比率)进行分析,发现剪枝比率与损失精度呈负相关关系。因为前面的卷积层用于提取物体的边缘特征,且卷积核个数较少,所以每个卷积核承担的性能压力较大,后面的卷积层处理的是高维度信息,卷积核个数较多,相比较而言每个卷积核承担的性能压力较小。

表6 YOLOv3-tiny网络性能贡献和各层剪枝比率

Table 6 The contribution of each layer to network performance and the pruning ratio of YOLOv3-tiny

剪枝层	微调精度	损失精度	同层剪枝比率/%
0	0.698	0.038	43.80
2	0.702	0.034	46.90
4	0.699	0.037	46.90
6	0.702	0.034	48.40
8	0.709	0.027	48.80
10	0.718	0.018	77.10
12	0.715	0.021	78.70
13	0.711	0.025	77.30
14	0.712	0.024	72.70
21	0.708	0.028	87.50

剪枝比率与损失精度的关系如图7所示,其中,圆形点表示剪枝比率与损失精度呈负相关关系,方形点表示网络最后一个剪枝层的剪枝比率,也就是第二个yolo层前的第二个卷积层,该剪枝层前面一层将浅层和深层信息拼接后输入,进行特征融合后输入到这一层,在网络剪枝时对于融合特征的卷积核的剪枝比率较大。

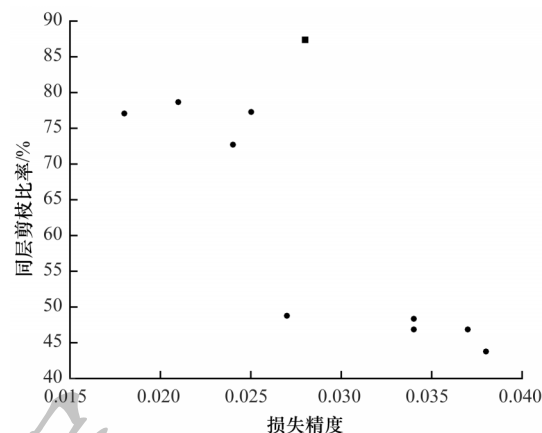


图7 剪枝比率与损失精度的关系

Fig.7 The relationship between pruning ratio and loss accuracy

### 2.4 YOLO网络集成剪枝算法耗时分析

本文基于敏感度的集成剪枝算法在YOLO系列网络的剪枝中,展现出优于基于单一参数重要性评估方法和文献[17]网络轻量化方法的剪枝算法的性能,并且在实际操作中时间成本并没有增加。为得到所需的L1范数、激活值、梯度这3种参数,对256张图片进行单张输入(batch size设置为1),网络训练1次即可获取梯度和激活值的平均值,在加载权重后得到卷积核的L1范数,3种参数的获取耗时较短。在获得每层的平均梯度损失后,需要对网络的每一层伪剪枝后(某些核BN层缩放因子置0)使用测试集测试获得新的精度,这一操作相对耗时。但是,本文基于敏感度的集成剪枝算法只需在第一次剪枝之前做稀疏化训练,解除特殊空间结构的限制,后面再重复剪枝、微调步骤时无需提前稀疏化训练,相比文献[17]网络轻量化方法的剪枝算法每次剪枝前都要稀疏化训练节省了很多时间。

## 3 结束语

为在算力有限的设备上部署YOLO检测网络,本文提出基于敏感度的集成剪枝算法,通过结合3种参数重要性评价方法得到每一层的卷积核重要性的最优排序,并使用稀疏缩放因子方式将冗余多的层稀疏化后进行网络剪枝。实验结果表明,该算法相比基于单一参数重要性评估方法和网络轻量化方法的剪枝算法有效提升了剪枝效果,解决了基于单一参数重要性评价方法的剪枝算法的片面性和泛化能力差问题。后续将针对检测类别较多或者特征较复杂的数据集,结合TensorRT对称饱和量化等方法进一步加速网络前向推理。

## 参考文献

- [1] ZHANG X Y, ZHOU X Y, LIN M X, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices[C]//Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C. , USA: IEEE Press, 2018: 6848-6856.
- [2] 杨民杰, 梁亚玲, 杜明辉. 基于参数子空间和缩放因子的 YOLO 剪枝算法[J]. 计算机工程, 2021, 47(2): 111-117.
- [3] YANG M J, LIANG Y L, DU M H. YOLO pruning algorithm based on parameter subspace and scaling factor[J]. Computer Engineering, 2021, 47(2): 111-117. (in Chinese)
- [4] JADERBERG M, VEDALDI A, ZISSERMAN A. Speeding up convolutional neural networks with low rank expansions[EB/OL]. [2020-05-11]. <https://arxiv.org/abs/1405.3866v1>.
- [5] ZHU C Z, HAN S, MAO H Z, et al. Trained ternary quantization[EB/OL]. [2020-05-11]. <https://arxiv.org/abs/1612.01064v3>.
- [6] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network[J]. Computer Science, 2015, 14(7): 38-39.
- [7] NAKKIRAN P, KAPLUN G, BANSAL Y, et al. Deep double descent: where bigger models and more data hurt[EB/OL]. [2020-05-11]. <https://arxiv.org/abs/1912.02292>.
- [8] LECUN Y, DENKER J S, Solla S A. Optimal brain damage[C]//Proceedings of 1990 International Conference on Neural Information Processing. Berlin, Germany: Springer, 1990: 598-605.
- [9] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural networks[C]//Proceedings of 2015 International Conference on Neural Information Processing. Berlin, Germany: Springer, 2015: 1135-1143.
- [10] GUO Y, YAO A, CHEN Y, et al. Dynamic network surgery for efficient DNNs[C]//Proceedings of 2016 International Conference on Neural Information Processing. Berlin, Germany: Springer, 2016: 1387-1395.
- [11] LI H, KADAV A, DURDANOVIC I, et al. Pruning filters for efficient ConvNets[EB/OL]. [2020-05-11]. <https://arxiv.org/pdf/1608.08710.pdf>.
- [12] HU H Y, PENG R, TAI Y W, et al. Network trimming: a data-driven neuron pruning approach towards efficient deep architectures[EB/OL]. [2020-05-11]. <https://arxiv.org/pdf/1607.03250.pdf>.
- [13] WANG H, ZHANG W H, WONG K Y M, et al. Encoding multisensory information in modular neural networks[C]//Proceedings of 2017 International Conference on Neural Information Processing. Berlin, Germany: Springer, 2017: 658-665.
- [14] YE J B, LU X, LIN Z, et al. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers[EB/OL]. [2020-05-11]. <https://arxiv.org/abs/1802.00124>.
- [15] LUO J H, WU J X, LIN W Y. ThiNet: a filter level pruning method for deep neural network compression[C]//Proceedings of 2017 IEEE International Conference on Computer Vision. Washington D. C. , USA: IEEE Press, 2017: 5068-5076.
- [16] ZHUANG Z, TAN M, ZHUANG B, et al. Discrimination-aware channel pruning for deep neural networks[C]//Proceedings of International Conference on Neural Information Processing. Berlin, Germany: Springer, 2018: 883-894.
- [17] HE Y, LIU P, WANG Z W, et al. Filter pruning via geometric median for deep convolutional neural networks acceleration[C]//Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Washington D. C. , USA: IEEE Press, 2019: 4335-4344.
- [18] LIU Z, LI J G, SHEN Z Q, et al. Learning efficient convolutional networks through network slimming[C]//Proceedings of 2017 IEEE International Conference on Computer Vision. Washington D. C. , USA: IEEE Press, 2017: 2755-2763.
- [19] REDMON J, FARHADI A. YOLOv3: an incremental improvement[EB/OL]. [2020-05-11]. <https://arxiv.org/pdf/1804.02767.pdf>.
- [20] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition[C]//Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Washington D. C. , USA: IEEE Press, 2016: 770-778.
- [21] HUANG Z Z, WANG X J, LUO P. Convolution-weight-distribution assumption: rethinking the criteria of channel pruning[EB/OL]. [2020-05-11]. <https://arxiv.org/abs/2004.11627>.
- [22] IOFFE S, SZEGEDY C. Batch normalization: accelerating deep network training by reducing internal covariate shift[C]//Proceedings of the 32nd International Conference on International Conference on Machine Learning. New York, USA: ACM Press, 2015: 448-456.
- [23] SUN X, REN X C, MA S M, et al. meProp: sparsified back propagation for accelerated deep learning with reduced overfitting[C]//Proceedings of the 34th International Conference on Machine Learning. Washington D. C. , USA: IEEE Press, 2017: 3299-3308.

编辑 陆燕菲