江南大学

大 作 业 报 告 书

题　　目：基于 YOLOv3 的模型压缩方法研究

院（系）：人工智能与计算机学院

专　　业：数字媒体技术

班　　级：数媒 2004

姓　　名：任真

学　　号：119200427

授课老师：钱琨

实验时间：22-23　学年　1　学期

2022 年 12 月

完成内容或创新点说明：

（1）附实验源码，代码完整，关键处注释可循，详见附件 1。

```
shortcut_prune.py
46
47      #将所有要剪枝的BN层的α参数，拷贝到bn_weights列表
48      bn_weights = gather_bn_weights(model.module_list, sort_prune_idx)
49
50      #torch.sort返回二维列表，第一维是排序后的值列表，第二维是排序后的值列表对应的索引
51      sorted_bn = torch.sort(bn_weights)[0]
52
53
54      #避免剪掉所有channel的最高阈值(每个BN层的gamma的最大值的最小值即为阈值上限)
55      highest_thre = []
56      for idx in sort_prune_idx:
57          #.item()可以得到张量里的元素值
58          highest_thre.append(model.module_list[idx][1].weight.data.abs().max().item())
59      highest_thre = min(highest_thre)
60
61      # 找到highest_thre对应的下标对应的百分比
62      percent_limit = (sorted_bn==highest_thre).nonzero().item()/len(bn_weights)
```

（2）附检测报告，经 PaperYY 官网查重，查重率<15%(11.1%)，检测报告单详见附件 2。


检测文献：基于YOLOv3的模型压缩方法研究（免费版）
文献作者：任真
报告时间：2023-01-10 11:03:52
段落个数：2
报告编号：YY202301101103501640
检测范围： 中国期刊库 中国图书库 硕士论文库 博士论文库 会议论文库 报纸库
网友专利库 网友标准库 网友共享库 个人对比库 网页库 百科库
相似度 11.1%
总文字复制比：11.1%
去除引用文献复制比：11.1%
去除本人已发表文献复制比：11.1%
单篇最大文字复制比：2.5%

（3）阅读大量 YOLO 系列的资料文献，在文中介绍了 YOLO 系列实时目标检测模型的思想及发展历程，总结了 YOLO 系列模型存在的不足。

（4）采用剪枝、蒸馏、量化等方式对模型加以缩小，显著降低 YOLOv3 中产生的数据冗余，使得网络对物理存储空间的要求大大降低。

（5）基于 YOLOv3 模型的结构特点，有大量的卷积层和 BN 层进行直连，利用 BN 层的 γ 系数来判断对应通道的重要性，移除掉不重要通道，再进行微调从而提高模型的 mAP。

基于 YOLOv3 的模型压缩方法研究

任真

（江南大学，人工智能与计算机学院，无锡 214122）

摘 要：近年来，卷积神经网络（CNN）已经成为各种计算机视觉工作的基础技术，例如，语义分割、图像分类、物体检测等。学术界新提出的神经网络模型越来越大，例如从LeNet到AlexNet再到ResNet和DenseNet，ImageNet分类挑战赛冠军模型已由八层发展到了一百多层。但在现实应用中许多模块都要被部署到移动终端。移动端的存储和计算能力有限，无法运行太大的模型，所以需要对模型进行压缩。本文对YOLOv3目标检测模型进行剪枝、蒸馏、量化，极大地减小了网络模型规模。实验结果表明，压缩后模型的mAP提高了0.4%，模型体积压缩了95%，对不同硬件平台的适用性大大提高。

关键词：目标检测；YOLO；通道剪枝；知识蒸馏；MNN量化

0 引言

伴随深度学习技术的发展，采用深度学习的方法已全面主导了目标检测领域的发展。在当前，采用深度学习的目标检测器一般包括双步(Two-Stage)目标检测算法和单步(One-Stage)目标检测算法二大类。Two-Stage目标检测器在第一步提取目标信息，在第二步中将所提出的目标区域做出了划分并将边界框回归，如：R-CNN (Region-based Fully Convolutional Neural Network)[1]、Fast R-CNN (Fast Region-based Convolutional Neural Network)[2]、Faster R-CNN(Faster Region-based Convolutional Neural Network)[3]等，

因此推理效率较低，且实时性也较差；One-Stage检测器主要是用于一次推理中发现的边框和类概率问题，如：SSD(Single Shot Detector)[4]、YOLO(You Only Look Once)[5]等，因其检测速率高，更适用于实时检测。

虽然One-Stage目标检测器取得了良好的性能，然而他们学习的参数过多，导致模型过大，消耗的计算资源过多，在存储空间和算力有限的移动端部署上存在困难。因此，有必要对模型进行压缩。

本文综合了网络剪枝、知识蒸馏方法，以及量化的方式，对YOLOv3模型加以完善，并对其进行了压缩，以便获得更轻量化的目标检测模块。

1 已有的研究工作

1.1 模型压缩

低秩分解法：E.L.Denton等人[6] 采取奇异值分解法（Singular Value Decomposition, SVD）等技术对神经网络中的权重矩阵进行近似处理。在全连接层上效果特别好，可以产生3倍模型大小的压缩，但是由于大量计算存在于卷积层，加速效果不够显著。

权重量化：W.Chen等人[7]提出对网络权重进行量化。只需对共享权重和哈希指数进行存储，可以大大节省存储空间。

权重修剪/稀疏化：S.Han等人[8]提出，在训练的神经网络中可以使用较小权重修剪不重要的信息连接，并且训练量化和霍夫曼编码。结果网络的权重大部分是零，从而能够以稀疏格式保存模型，进而减少了存储空间，并借助专门的稀疏矩阵计算库/硬件来实现加速。但其实际运行的效果仍不够乐观。

1.2 目标检测

2014年，Girshick R等[9]给出了R-CNN模型，使用候选区域(Region Proposal)生成新的边界范围框，而后再在这些域上应用分类器。同年，Girshick R等[10]在先前提出的算法基础上，改进并实现了Fast R-CNN算法，克服了先前提取特征步骤冗余、训练过程复杂导致训练效率低等问题，采用对整幅图像执行卷积操作，再从特征映射中确定候选范围，极大地提高了训练效率。2015年，Ren等[11]实现了Faster R-CNN算法，对候选区域提取网络设计、候选区域提取网络训练、共享卷积特征等步骤进行改进，较先前的Fast R-CNN算法节省了约10倍的运行时间。

2 YOLO目标检测算法

相较于传统Two-Stage目标检测算法R-CNN、Fast R-CNN、Faster R-CNN等，Redmon等[12]于2016年提出One-Stage的YOLO方法把目标检测问题转变为回归问题，使用卷积式神经网络来来快速实现对目标类型的判断及边界的检测。使得目标的检测速度大大加快，达到了实时检测的程度。

2.1 YOLOv1

2.1.1 训练阶段

如图1所示，将一幅图像分成S*S个网格(Grid Cell)，每个grid_cell包含B个bbox的位置信息以及置信度信息，以及C个类别的条件概率信息。从物体标准框(GroundTruth)中心落在的grid cell负责对目标进行一定的预测，从gird cell预测的B个bbox以及GroundTruth的IOU最大的bbox中作为预测物的bbox。损失函数对负责预测物体的bbox，物体中心落在的girdcell中不负责预测物体的bbox，以及物体中心不落在的girdcell中产生的bbox分别进行处理。

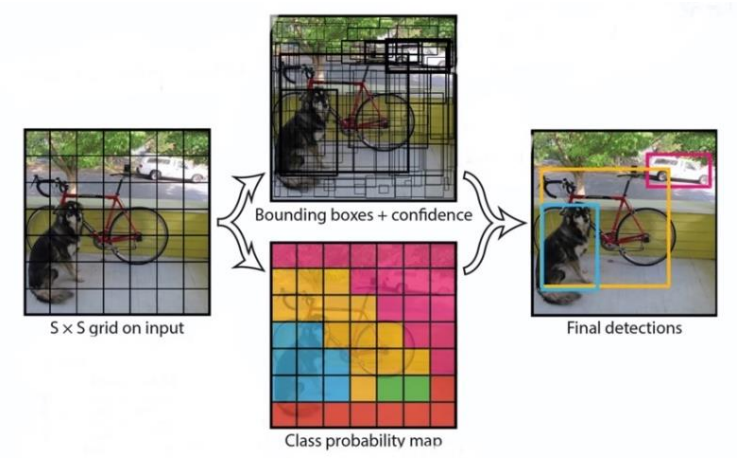

图 1 YOLOv1 训练阶段

Figure 1 The training phase of YOLOv1

2.1.2 预测阶段

直接获得S*S*（5*B+C）维的向量，再根据阈值去掉可能性较小的bbox，最后NMS去除冗余窗口，得到目标检测的结果。

2.1.3 模型结构

在PASCAL VOC数据集中，图像输入为448x448，取S=7，B=2，所以一共是二十个类别(C=20)。其输出就是7x7x30的张量(Tensor)。整个网络结构如图2所示：

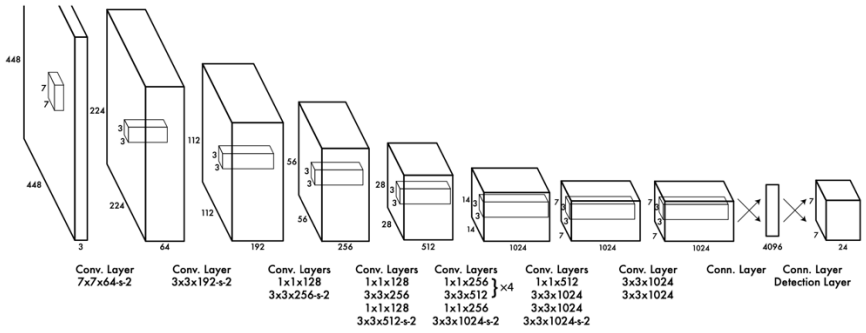

图 2 YOLOv1 模型结构

Figure 2 The model structure of YOLOv1

2.1.4 损失函数

公式如图3所示，其损失函数一般包括以下三个组成部分：（1）Objectness Loss，描述在一个bbox中对一个object的损失；（2）Classification Loss，表示一个bbox的分类损失；（3）Regression Loss，表示一个bbox的坐标回归损失。

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$

$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+\sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

图 3 YOLOv1 损失函数

Figure 3 The loss function of YOLOv1

其中，对负责预测物体的bbox使用更大的权重进行位置以及置信度信息的惩罚，并且对width和height取平方根对小框的偏差进行额外的惩罚。对不负责预测物体使用更小的权重且只对置信度进行惩罚。

2.2 YOLOv2

初代的YOLO虽然检测速度快，但是在定位方面却不够精准，并且召回率也不高。为提高定位准确率，提高召回率，YOLOv2在YOLOv1的基础上，加了一些长宽比预先设置好的先验anchor，BN层，高分辨率分类器，passthrough层以及多尺度训练。其相对于YOLOv1的改进如下：

2.2.1 相对于YOLOv1的改进

（1）先验框(Anchor)

借助Faster RCNN的经验，YOLOv2也试图过采用先验框。在每个grid里设置一个不同尺寸和长宽高比例的界限，以包括整个图形的各个区域和各种尺寸。YOLOv2中通过对训练集中设置的界限使用K-means聚类分析，对IOU和模型复杂的进行权衡，选择最优的聚类数。

（2）约束预测边框的位置(Direct Location Prediction)

初版YOLO对预测的边框取值并无任何限制，中心可以出现在任意地方，训练早期阶段会出现不稳定的状态。如图4所示，YOLOv2改进了预测公式，使用sigmoid函数将预测边框anchar的位置限制在了一个网格中。

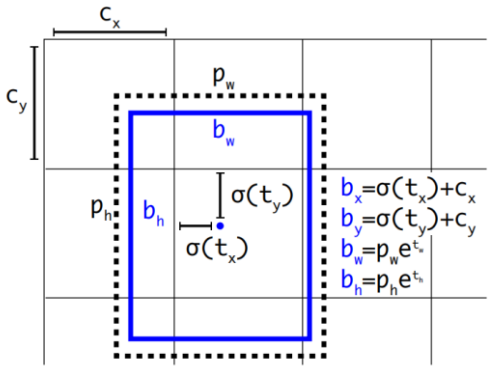

图 4 约束预测边框的位置

Figure 4 Direct location prediction

（3）批标准化(Batch Normalization)

对于训练阶段：某层的某个神经元会输出batch_size个响应值。如图5所示，对这batch size个响应值求均值$\mu_{\mathcal{B}}$和标准差$\sigma_{\mathcal{B}}$，再做归一化。把归一化后的响应值乘γ再加上β。对每个神经元都训练一组γ和β。

$$\hat{z} = \frac{z_{in} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}; z_{\text{out}} = \gamma \hat{z} + \beta$$

图 5 批标准化

Figure 5 Batch Normalization

对于测试阶段：测试阶段均值、方差、γ 和 β 都用训练阶段全局求出的。批标准化相当于做线性变换。

（4）高分辨率图像分类器(High resolution classifier)

YOLOv1在ImageNet图像分类与数据集训练时，选用224*224大小的图片作为输入；在目标检测数据集训练时，采用更高分辨率的448*448的图像作为输入。这样切换分辨率对模型性能有一定影响。

在YOLOv2中选择224*224的数据对分类建模预训后，再选择448*448的高分辨率样本对分类建模进行训练（10个epoch），然后在选择448*448的目标样本进行微调，从而减少了高分辨率突然转换带来的干扰。

2.2.2 模型结构

在PASCAL VOC数据集中，图像信号输入类型是416x416，S=7，B=2，一共有二十个分类(C=20)。其输出是7x7x30的张量。整个网络结构如图6所示：

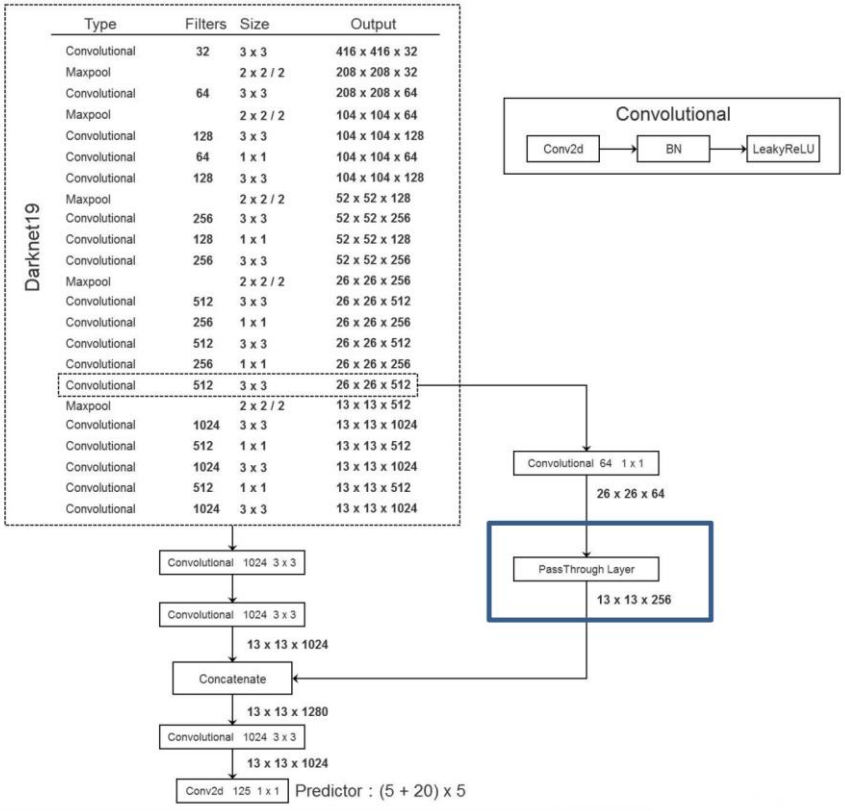

图 6 YOLOv2 模型结构

Figure 6 The model structure of YOLOv2

2.2.3 损失函数

YOLOv2的损失函数与YOLOv1非常相似，同样包含Objectness loss、Classification loss、Regression loss 3个部分：

$$L(\text{ object }) = \sum_{i=0}^{K\times K}\sum_{j=0}^{M} I_{\text{MaxIOU}<\text{Thresh}}\,\lambda_{\text{noobj}} * (-b_{ij}^{o})^2$$
$$+I_{t<12800}\lambda_{\text{prior}} * \sum_{r\in(x,y,w,h)} (\text{ prior }_{j}^{r} - b_{ij}^{r})^2$$
$$+I_{j}^{\text{truth}}(\lambda_{\text{coord}} * \sum_{r\in(x,y,w,h)} (\text{ truth }^{r} - b_{ij}^{r})^2$$
$$+\lambda_{\text{obj}} * (\text{IOU}_{\text{truth}}^{j} - b_{ij}^{o})^2$$
$$+\lambda_{\text{class}} * (\sum_{c\in \text{ classes}} (\text{ truth }^{c} - b_{ij}^{c})^2))$$

图 7 YOLOv2 损失函数

Figure 7 The loss function of YOLOv2

对负责预测物体的anchar使用更大的权重，进行位置信息、置信度信息、分类信息的惩罚。对不负责预测物体的anchar使用更小的权重，只对置信度信息进行惩罚。在模型训练的早期（迭代次数小于12800），对anchar位置与预测框位置的偏差进行惩罚。

2.3 YOLOv3

三代YOLO在二代YOLO的基础上，采用层数更深的DarkNet-53获得了更细粒的特征信号，同时通过对特征金字塔(Feature Pyramid Networks, FPN)结构进行了多尺寸估计，并通过1×1卷积和Sigmoid激活函数替代了Softmax分类层，从而更高效地进行了数据拟合。YOLOv3在各项性能上都得到了很大的改进。

2.3.1 相对于YOLOv2的改进

（1）多尺度预测 （引入FPN）

YOLOv3采用了FPN的概念，在各种尺寸上提取特征。对比YOLOv2，YOLOv3获取最后三层特征图，不但可以在各个小特征图上分别独立做出进一步预测，同时可以通过从小特征图上采样到和更大的小特征图一样尺寸，然后与大的特征图拼合，做出进一步预测。和YOLOv2一样，通过对训练集中标注的边框进行K-mean聚类分析，并选取了最优的聚类值9，从而可以把九种尺度的anchor box均衡分派给三种尺寸的小特征图。在基础网络结构图上增加了多尺度特征提取方面的示意图，如图7所示。

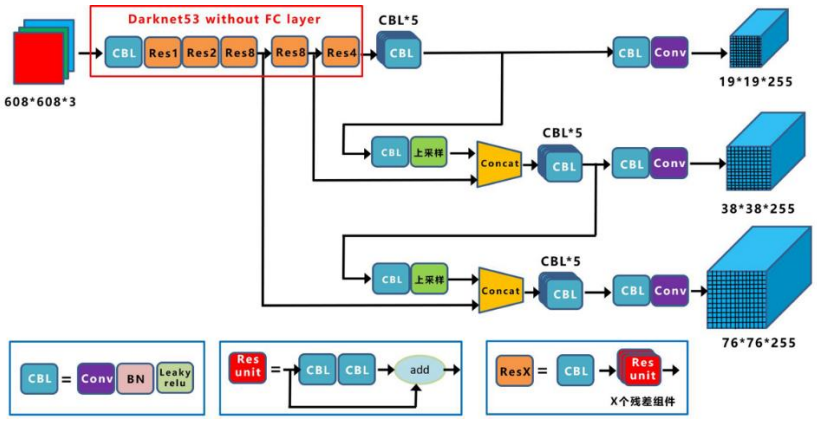

图 8 YOLOv3 模型结构

Figure 8 The model structure of YOLOv3

（2）更好的基础分类网络

在YOLOv2模型的基础上，进一步加深了网络，并且类似于ResNet引入残差结构。

（3）分类器的改进

分类器不再使用Softmax，使用1×1卷积和Logistic激活函数来替换Softmax层，对分类损失采取二分类交叉损失熵(Binary Cross-entropy Loss)的策略。

2.3.2 损失函数

YOLOv3中使用的损失函数公式如图9所示，包括了如下三方面：一个是xywh部分带来的误差，也是

由bbox产生的loss；二是置信度所造成的偏差，也就是obj带来的loss；三是类别带来的误差，也就是class带来的loss。

$$lbox = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} (2 - w_i \times h_i)[(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 + (w_i - \hat{w_i})^2 + (h_i - \hat{h_i})^2]$$

$$lcls = \lambda_{\text{class}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{\text{obj}} \sum_{c \in \text{ classes}} p_i(c) \log(\hat{p_i}(c))$$

$$lobj = \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{noobj} (c_i - \hat{c_i})^2 + \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} (c_i - \hat{c_i})^2$$

$$loss = lbox + lobj + lcls$$

图 9 YOLOv3 损失函数

Figure 9 The loss function of YOLOv3

3 模型压缩

3.1 剪枝

3.1.1 剪枝流程

Zhuang Liu[13]提出了针对通道进行剪枝的方法，流程如下图所示：

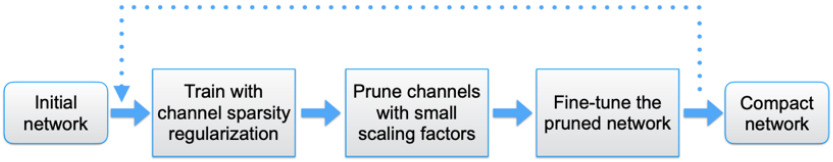

图 10 剪枝流程

Figure 10 The pruning process

如图11所示，在初始网络的基础上，给每个通道都加上一条缩放因子γ（见BN层），与通道的输出值相乘。然后结合训练网络权重和通道对应的缩放因子，并对后者施加稀疏性正则化进行稀疏训练。接着再对那些系数较小的通道进行修剪，之后再对修改过的模型使用微调的网络剪枝策略。

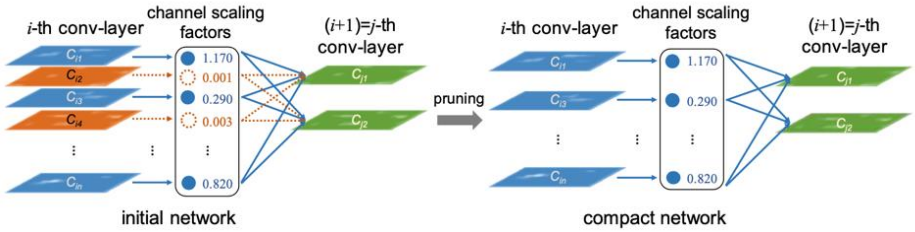

图 11 模型剪枝前后对比

Figure 11 Comparison of model before and after pruning

其中，损失函数如图12示：

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

图 12 损失函数

Figure 12 The loss function after pruning

其中(x，y)表示训练输入和目标，W表示可训练的权重，第一个和项对应于CNN的正常训练损失，g(s)是L1-norm用于实现γ因子稀疏性。

本文基于YOLOv3模型的结构特点，有大量的卷积层和BN层进行直连，利用BN层的γ系数来判断对应通道的重要性，移除掉不重要通道，再进行微调从而提高模型的mAP。

3.1.2 实验结果

（1）稀疏训练

使用tensorboard记录了正在进行稀疏训练的BN层的γ值情况，图11左发现了正常训练时γ总体上分布为在1左右的正态分布，而右侧则可以发现稀疏过程γ的大部分会被压至近零，而靠近零的通道因其输出值近似于常量，因此可将其剪去。

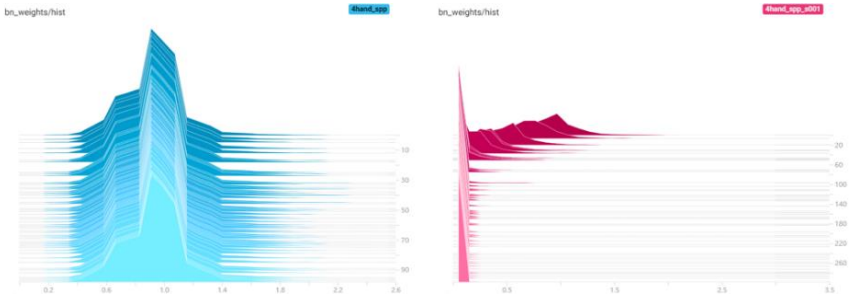

图 12 稀疏训练

Figure 12: Comparison of indexes before and after pruning

（2）剪枝/微调

剪枝/微调前后指标对比如表1所示：

表 1 剪枝前后指标对比

Table 1 Comparison of indexes before and after pruning

模型	参数量	模型体积	压缩率	前向推断耗时	mAP
Baseline	61.5M	246.4MB	32.8%	15.0 ms	0.7692
Prune	10.9M	43.6MB	9.6%	7.7 ms	0.7722
Finetune	10.9M	43.6MB	9.6%	7.7 ms	0.7750

3.2 知识蒸馏

3.2.1 分类模型蒸馏

Hinton[14]提出了先训练一个大的Teacher模型，运用Teacher模型预测出的soft label代替hard label作为训练网络的标签，使Student模型往Teacher模型的方向去训练。并且提出了可以采用提高蒸馏温度的方法，使得soft label变得更加soft的技巧。模型结构如图13所示：

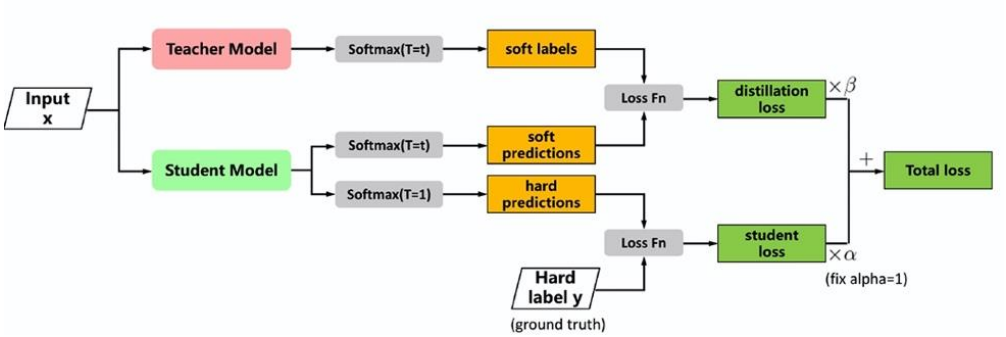

图 13 分类模型蒸馏

Figure 13 Classification model distillation

因目标函数是2种不同目标函数的加权平均值。前者目标函数是学生预测和软目标之间的交叉熵损失，而第二个目标函数是学生输出和正确标签之间的交叉熵损失。

3.2.2 YOLOv3 One-stage目标检测模型蒸馏

One-stage目标检测任务相对于分类问题的训练目标难度更大，因为teacher网络会预测出更多的bbox/anchar，如果直接用teacher的预测输出作为student学习的soft label会有严重的类别不均衡问题。

Rakesh Mehta[15]提出了采取了FM-NMS进行teacher model输出的重复框进行去除，并针对YOLOv3中分类、回归、物体检测使用不同的损失函数，基于预测框是否包含物体的概率调整损失函数权重，解决了预测框类别不均衡问题。模型结构如图14所示：

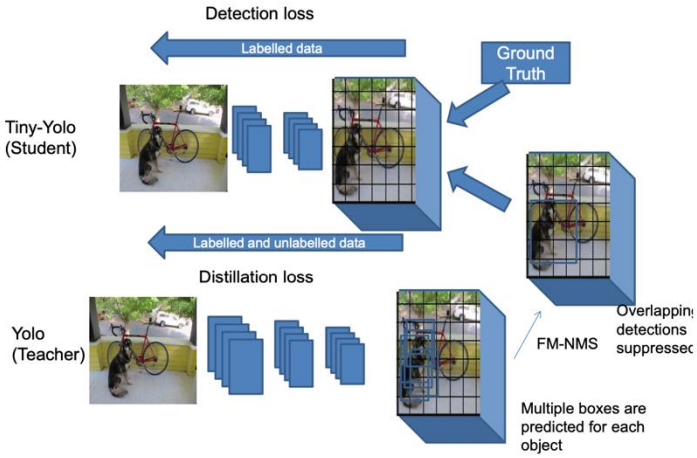

图 14 FM-NMS 模型结构

Figure 14 Model structure of FM-NMS

损失函数如图15所示：

$$L_{\text{final}} = f_{bb}^{Comb}(b_i^{gt}, \hat{b}_i, b_i^T, \hat{o}_i^T) + f_{cl}^{Comb}(p_i^{gt}, \hat{p}_i, p_i^T, \hat{o}_i^T) + f_{obj}^{Comb}(o_i^{gt}, \hat{o}_i, o_i^T)$$

图 15 FM-NMS 损失函数

Figure 15 Loss function of FM-NMS

3.2.3 实验结果

蒸馏方法在此阶段只能进行微调，若要着重考虑精度，而且剪枝时也要尽量剪不出点的比例，这时蒸馏的效果也就不明显；因为注重速度，且剪枝数量很大，使得模型的精度降低较多，建议结合蒸馏提高精度。如表2所示，有个极端例子，全局修剪为0.95，层修剪54个，mAP降到了0，但是微调50epoch后依然回到了0.75。

表 2 蒸馏前后指标对比

Table 2 Comparison of indexes before and after distillation

模型	参数量	模型体积	压缩率	前向推断耗时	mAP	mAP 蒸馏后
Prune	10.9M	43.6MB	9.6%	7.7 ms	0.7722	0.7872
Finetune	1.07M	2.8MB	89.8%	5 ms	0	0.7510

3.3 量化

3.3.1 量化过程

量化的作用是使系统中的算符由原先的浮点编码运算改为低精度的int8计算，通过较小的比特位来表达主要参数值，降低模型质量并提高性能。

YOLOv3模型使用MNN量化推理框架，将PyTorch的模型先转为ONNX模型再转为MNN自定义的模型格式(mnn)进行量化。如图15所示：

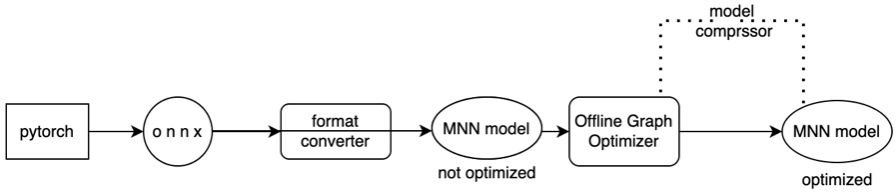

图 15 MNN 量化过程

MNN中通过最小化kl散度，找到最优的阈值T，将[-T,+T]映射为[-127,+127]，大于±T的直接映射为阈值±127，从而将float32转换为int8的数据，使得原本32bit表示的tensor变为只需要8bit表示，从而达到模型压缩、算子加速的目的。

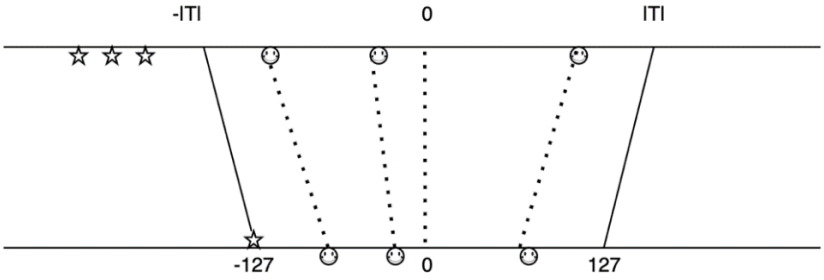

图 16 MNN 量化推理框架

3.3.2 实验结果

量化前的YOLOv3网络大小约为5.7M，使用MNN量化推理框架去压缩tensor精度，将float32转换为int8的数据，模型大小减少为1.7M，模型体积进一步下降了约70%。

4 结论

文章首先阐述了YOLO实时目标检测的原理及其演变进程，指出了YOLO系统模型具有冗余参数等缺点。之后又通过深度神经网络压缩技术对YOLOv3进行了压缩。虽然单独采用了剪枝量化算法，但压缩后的模型中依然具有一定的信息冗余，并且精度下降明显。因此提供了一个能够结合剪枝、蒸馏和量化的方式对模型加以缩小。设置了模型稀疏程度评价的手段，从而能够在缩小后更直观清晰地看到目标层在整体神经网络中的冗余状况。实验结果证实，通过采用深度神经网络压缩的混合压缩方式，能够有效降低YOLOv3模型的参数冗余率，从而减少了网络模型对物理存储设备的要求。未来，随着物理存储的不断优化，YOLO系列模型将会在解决实时目标检测的任务中大放光彩。

参考文献

[1] GUPTA S, GIRSHICK R, ARBELEZRBELÁEZ P, et al. Learning rich features from RGB-D images for object detection and segmentation[C]// European conference on computer vision. Springer, Cham, 2014: 345-360.

[2] GIRSHICK R. Fast R-CNN[C]// IEEE International Conference on Computer Vision and Pattern Recognition, 2015: 1440-1448.

[3] REN S, HE K, GIRSHICK R B, et al. Faster R-CNN: towards real- time object detection with region proposal networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 39(6): 1137-1149.

[4] HUANG J, RATHOD V, SUN C, et al. Speed/accuracy trade-offs for modern convolutional object detectors[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, 2017: 7310-7311.

[5] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real- time object detection[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2016: 779-788.

[6] E.L Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In NIPS, 2014.

[7] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick[J]. ICML, 2015.

[8] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[C]//Advances in Neural Information Processing Systems. 2015: 1135-1143.

[9] Girshick R,Donahue J,Darrell T,et al.Rich feature hierarchies for accurate object detection and semantic segmentation[G]//Proc of ImageNet Large-Scale Visual Recognition Challenge Workshop. [s.l.]:ICCV Press,2013:10-15.

[10] Girshick R. Fast R-CNN[C]//ICCV.[s. l.]:[s. n.], 2015.

[11] Ren S, Girshick R, Girshick R, et al. Faster R-CNN:towards real-time object detection with region proposal networks[J]. IEEE Transactions on Pattern Analysis &.Machine Intelligence,2015,39(6):1137-1149.

[12] REDMON J, DIVVALA S, GIRSHICK R, etal. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[13] Zhuang L, Li J, Shen Z, et al. Learning Efficient Convolutional Networks through Network Slimming[J]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[14] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network[J]. Computer Science, 2015, 14(7):38-39.

[15] Rakesh Mehta and Cemalettin Ozturk. Object detection at 200 Frames Per Second.[J]. CoRR, 2018, abs/1805.06361