# EfficientDet论文笔记

## 前言

作者：Mingxing Tan Ruoming Pang Quoc V. Le Google Research, Brain Team  
论文链接：[EfficientDet: Scalable and Efficient Object Detection](https://arxiv.org/abs/1911.09070)  
代码链接：https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch (非官方复现)

EfficientDet是Google的大作。在分类任务上有一篇[EfficientNet](https://arxiv.org/abs/1905.11946)，从名字看就知道，它是EfficientNet的在目标检测任务上的延伸。这篇文章的重点有两个：首先是BiFPN结构(*weighted bi-directional fea- ture pyramid network*) ，可以更快更好地融合特征。其次是提出一种*compound scaling method*，在EfficientNet那篇论文里也有提过。本质上，就是把NAS需要搜索优化的很多参数，基于一些insight和经验，用少量的参数关联起来，这样就可以减小减小搜索空间，实现更快更高效地搜索。EfficientDet使用的是SSD+FPN的one-stage检测架构，所以需要搜索的网络结构参数，包含backbone、feature网络(FPN)、bbox/cls 网络的width、height以及输入的resolution。

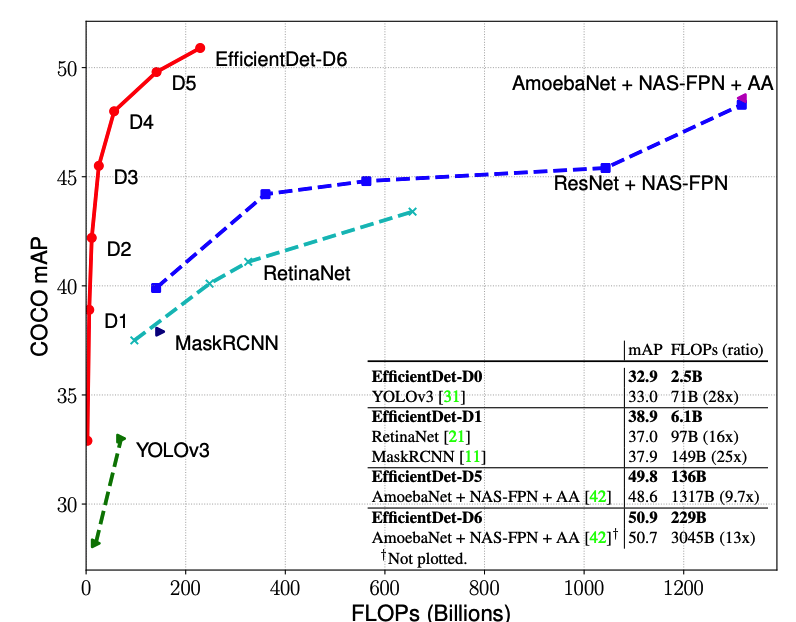
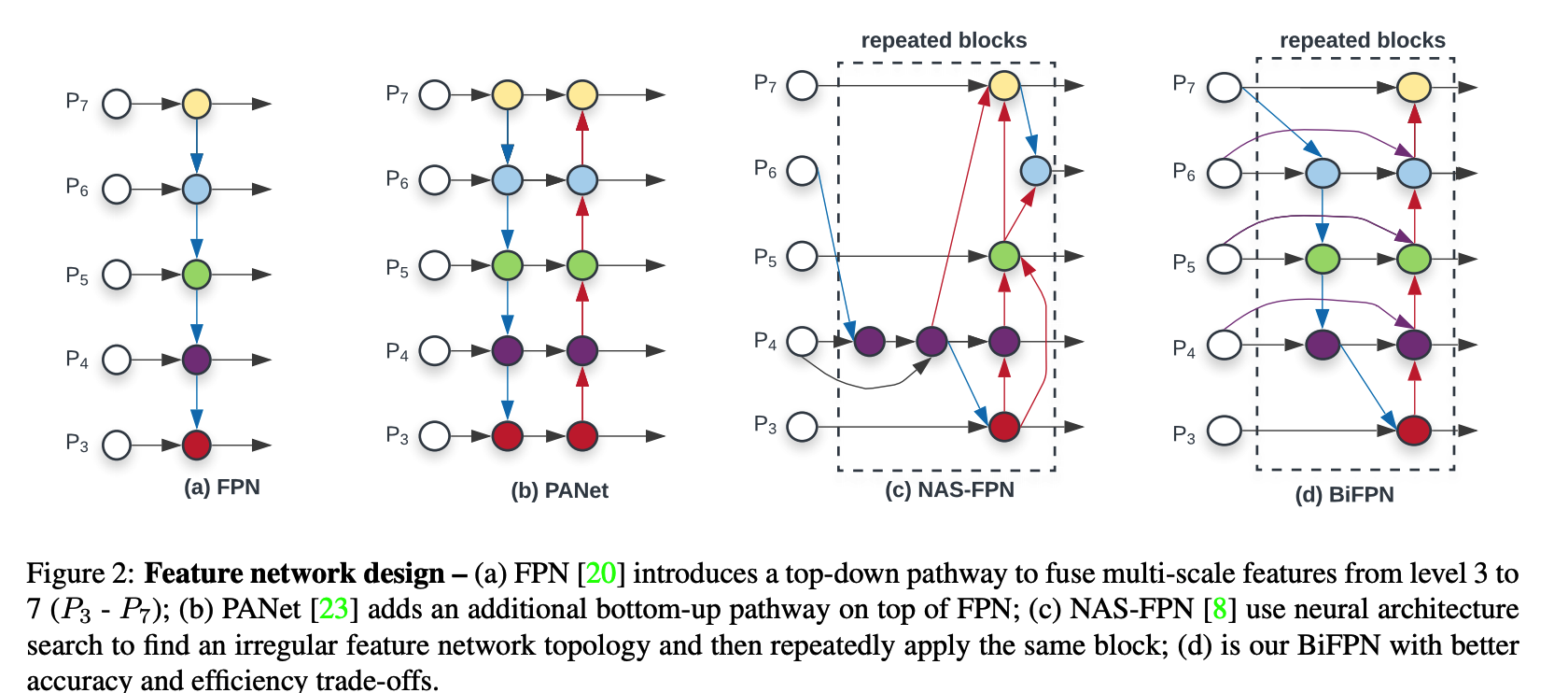
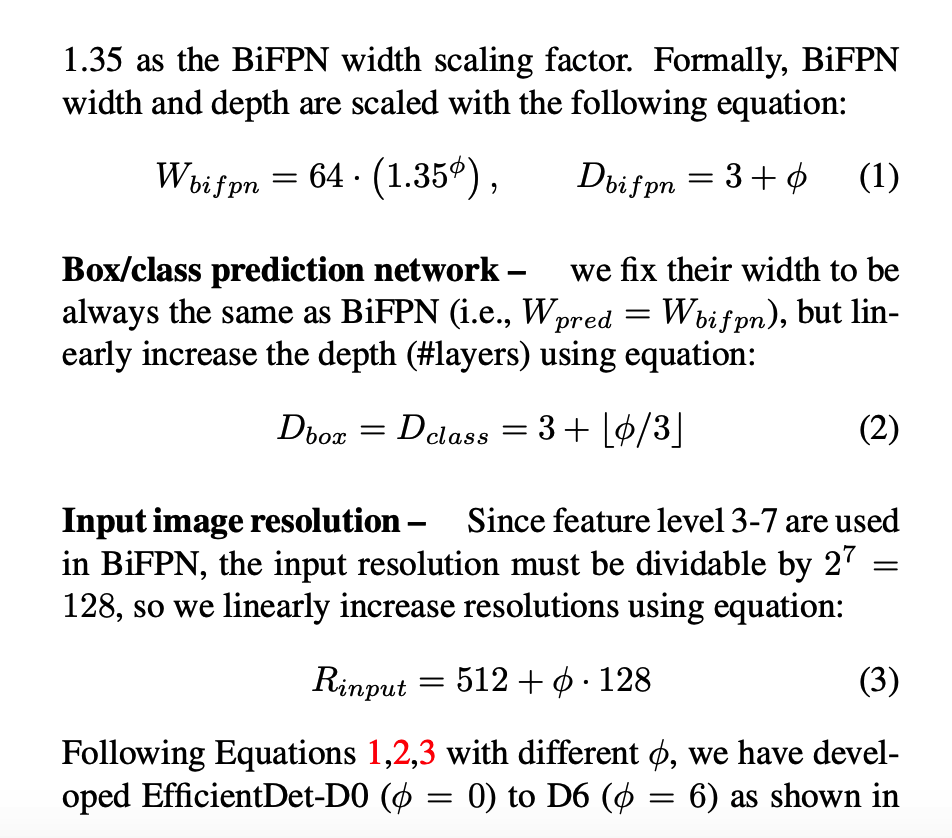
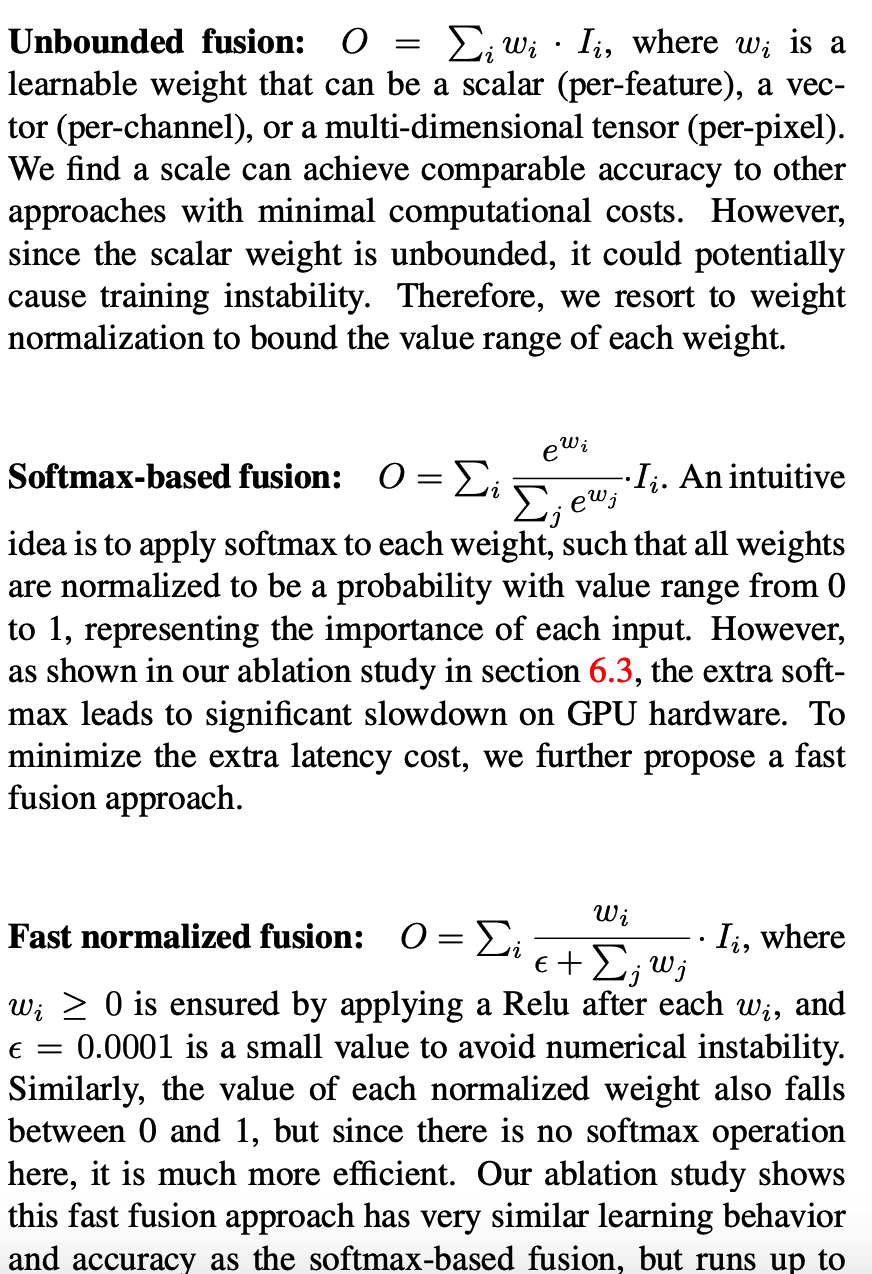


image-20200415214342327

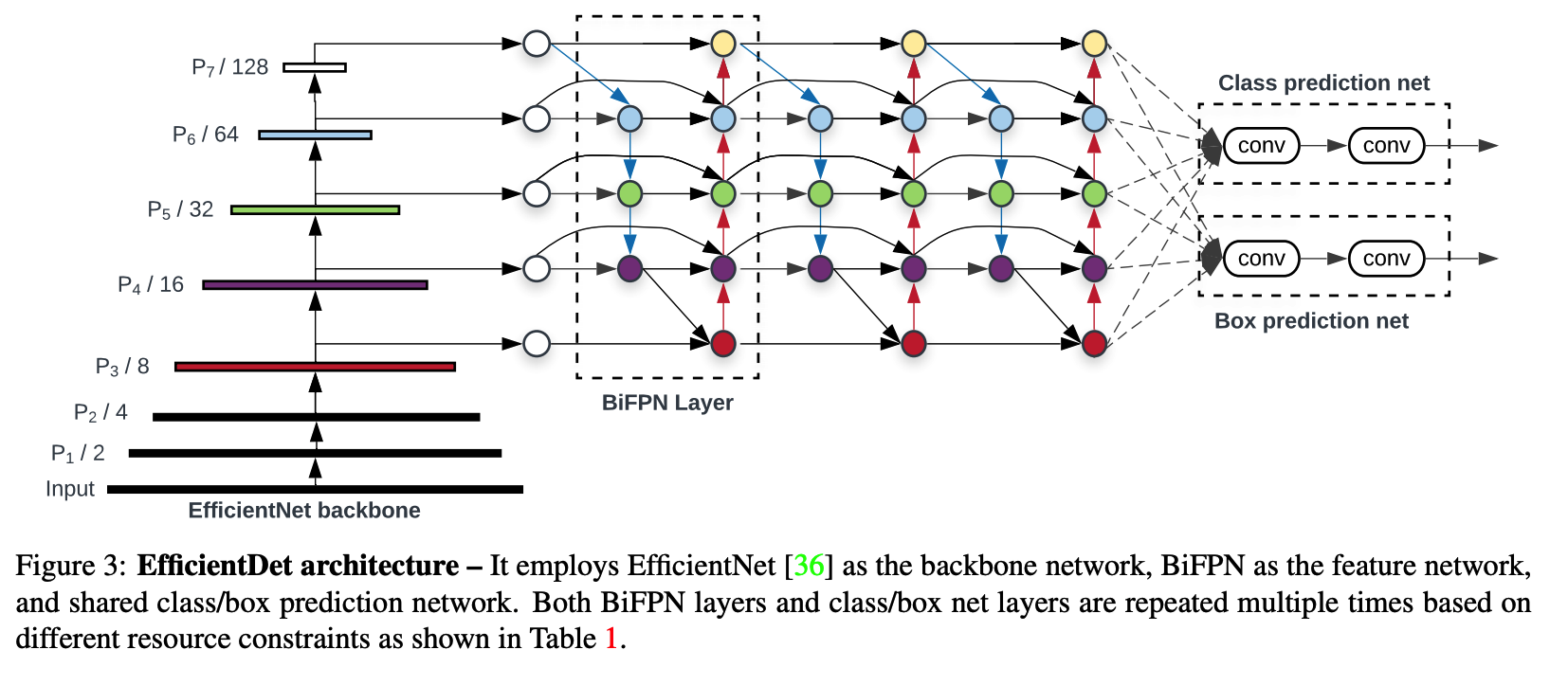
## 本文贡献

* BiFPN (双向FPN)
* FPN只有bottom-2-up的path；PANet使用了双path的结构；NAS-FPN通过神经架构搜索得到网络结构，但是结构的可解释性很差。我们参考PANet，增加了skip connection和weighted fusion，以便更好地融合特征。
* 
* Weighted feature fusion
* 

1) 在FPN部分，每个节点都是由多个节点融合而来的，我们发现，不同深度的feature map对结果的贡献是不同的。因此，我们给每个节点的输入节点添加learnable权重。为了更好地学习降低计算效率，不适用sigmoid归一化，而使用均值归一化。

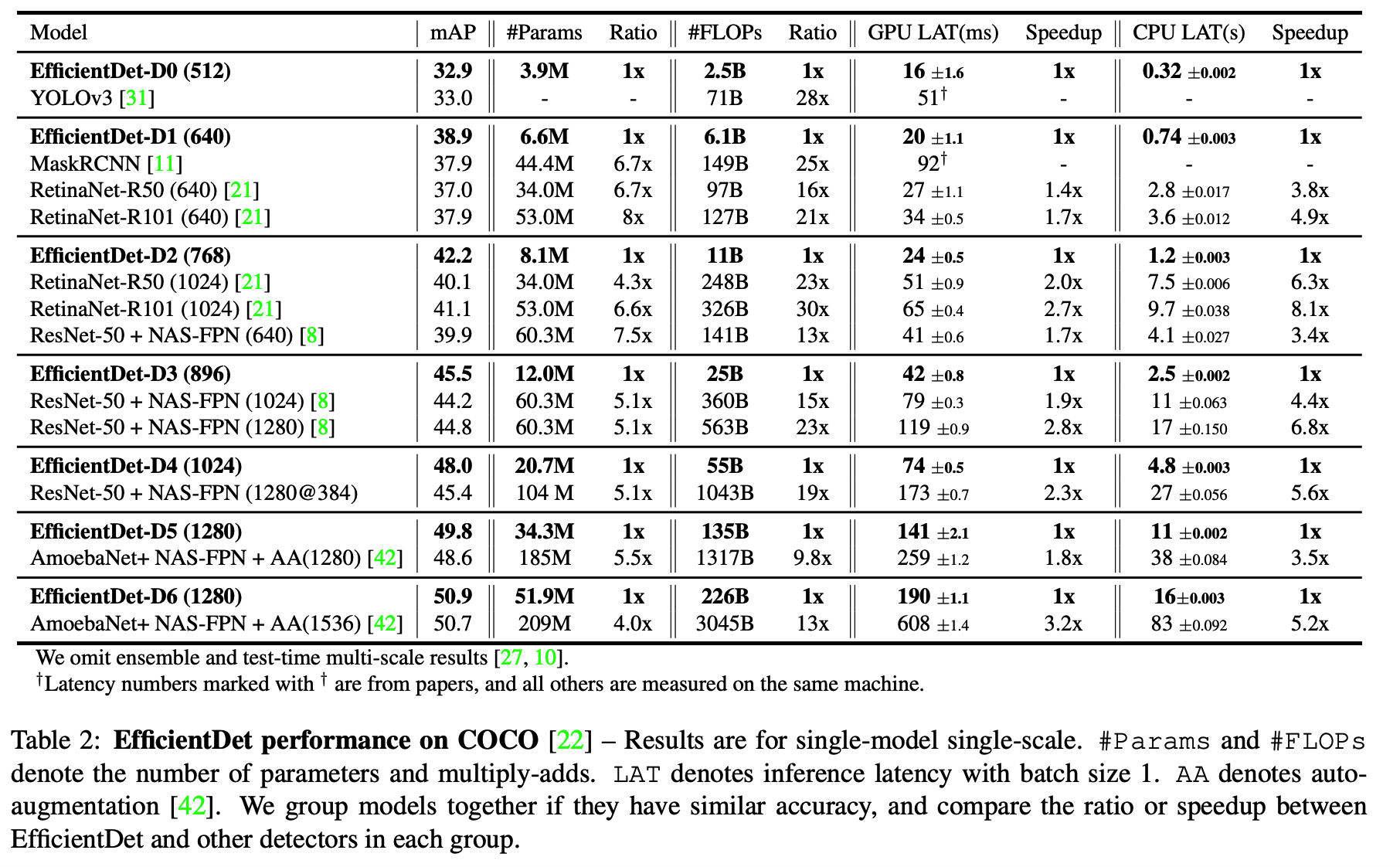
* compound scaling method
* 目标检测中需要考虑的参数比分类任务更多。分类任务中只考虑了width，depth和resolution（input)，目标检测任务中，还需要考虑cls/bbox net。
* 与EfficientNet相同，在架构搜索阶段。我们用一个参数关联所有需要搜索优化的参数，比如width，bbox/cls 的depth, 以及input resolution。通过优化，我们搜索得到最优的网络架构。这就是compond scaling method。

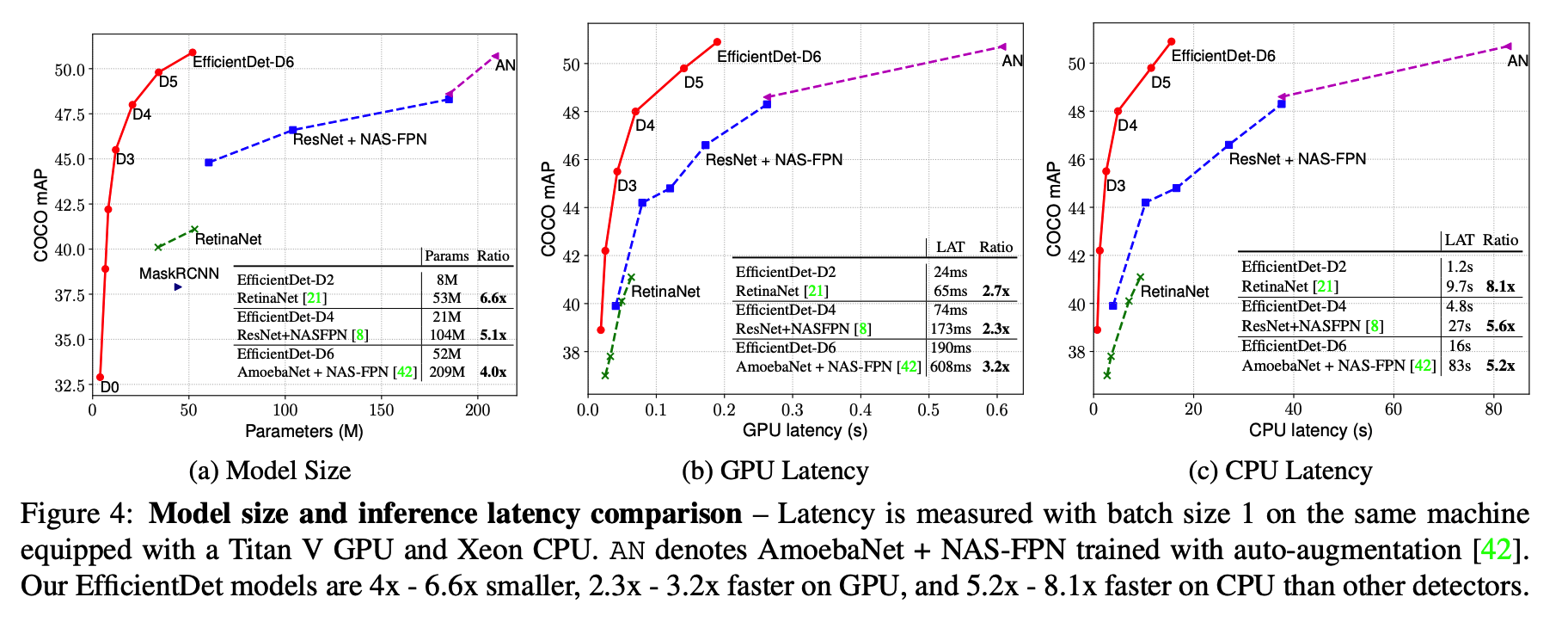
## 网络结构



基于一阶段SSD+FPN结构改造。以EfficientNet为backbone，然后接上3个(bottom-up & up-down)的结构，最后的特征用于预测bbox和cls。

## 实验





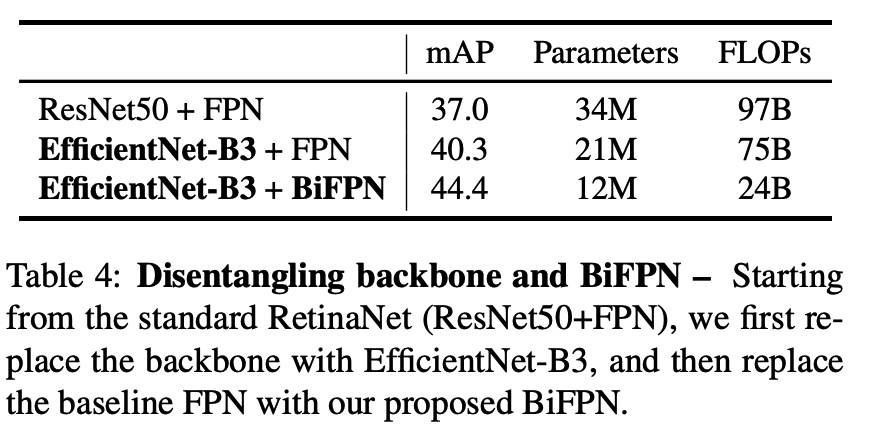


image-20200325220947891

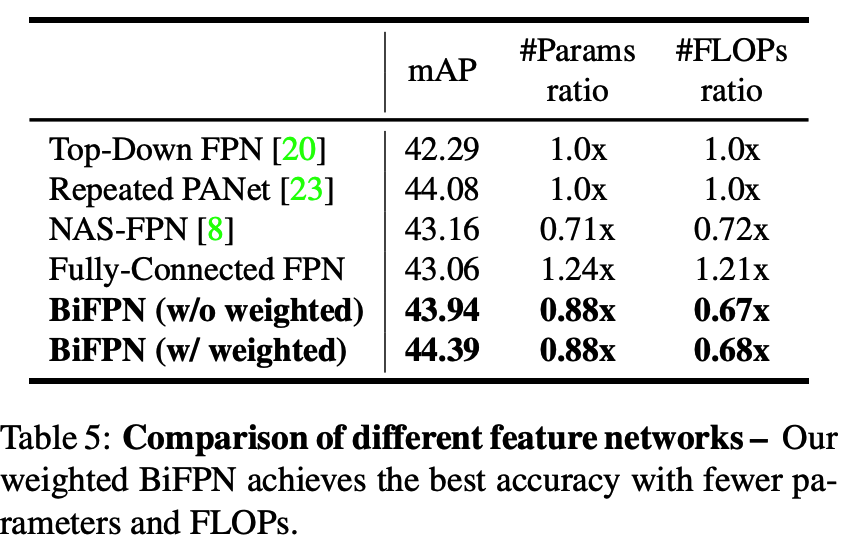


image-20200325221023949

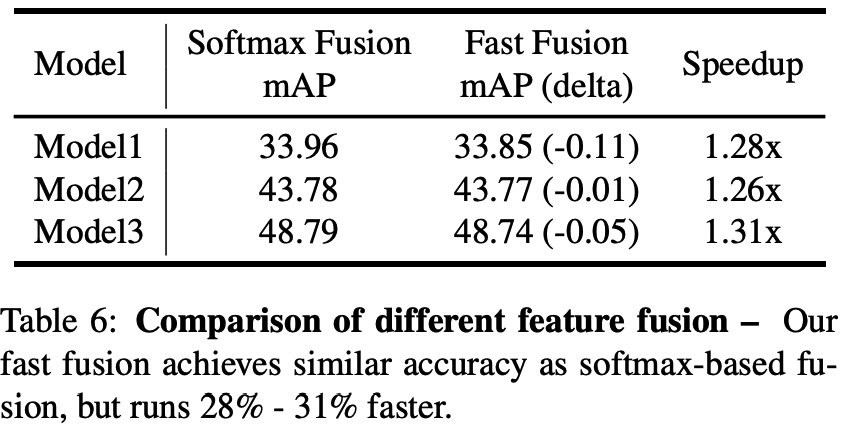


image-20200325221112045

## 其他

下图是[paperwithcode](https://paperswithcode.com/sota/object-detection-on-coco-minival)的cocominival的排行榜，可以看到，前四名都被EfficientDet霸占了。由于Google一直没有放出开源代码，大家开始自力更生。最近比较火的，是[zylo117](https://github.com/zylo117)的工作[全网第一SoTA成绩却朴实无华的pytorch版efficientdet](https://zhuanlan.zhihu.com/p/129016081),代码在这里[Yet-Another-EfficientDet-Pytorch](https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch)。

