Web 信息挖掘与分析课程报告

论文题目	MobileNet 系列论文阅读分析报告
学　　院	计算机科学与技术学院
专　　业	软件工程
班　　级	专硕
学　　号	2019021932
学生姓名	崔虎
指导教师	黄瑞章

目录

MobileNet 系列论文阅读分析

崔虎*

2020 年 9 月 12

摘　要

神经网络的快速发展使得人工智能领域产生了革命性的变化，比如图像分类和识别的精度已经超越了人类本身的识别精度，语义分隔，物体检测，图像生成等等有意思而又有意义的工作都会在每年的计算机顶级会议中出现，而这一切的基础都是因为我们计算机的计算能力在近些年有了极大的提高，然而，我们也经常发现，很多计算模型虽然很有意思，却过分依赖于硬件基础。而本文所分析的三篇 MobileNet 系列论文，就是针对如何使得原本重量型的模型应用于移动和嵌入端的问题，来展开讨论和分析。

Abstract

The rapid development of neural network makes a revolutionary change in the field of artificial intelligence. For example, the accuracy of image classification and recognition has exceeded the recognition accuracy of human beings. Interesting and meaningful work will appear in the top computer conference every year, and the foundation of all these is because of our computer Computing power has been greatly improved in recent years. However, we often find that many computing models, although interesting, rely too much on hardware. The three papers in this paper are about how to apply the original weight model to the mobile and embedded end.

*贵州大学，计算机科学与技术学院，软件工程 2019021932

1 MobileNetV1

该论文是 2017 年由谷歌团队发表 [12]，提出了一类高效的移动和嵌入式视觉应用模型 MobileNets。MobileNets 基于一种流线型结构，它使用深度可分离卷积来构建轻量化的深度神经网络。引入了两个简单的全局超参数，它们可以有效地在延迟和准确性之间进行权衡。这些超参数允许模型生成器根据问题的约束为其应用程序选择适当大小的模型。

我们以下就论文中的两个重点进行分析：**流线型结构（深度分离卷积）与全局超参数**。

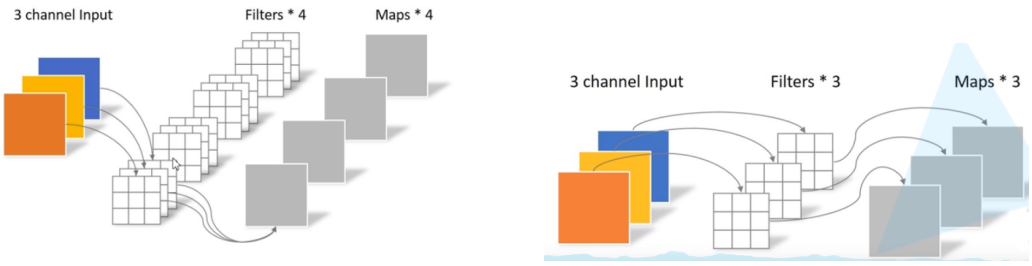
1.1 工作基础

mobilenet 主要由深度可分离卷积构建，最初在 [27] 中引入，随后在初始模型 [15] 中使用，以减少前几层的计算量。Flattened networks[18] 利用全因子卷积构建网络，并展示了极大因子分解网络的潜力。随后，Exception network[3] 演示了如何扩展深度可分离滤波器，以超越初始 V3 网络。另一个小型网络是 Squeezenet[14]，它使用 bottleneck 方法来设计一个非常小的网络。其他减少计算量的网络包括 structured transform network[28] 和 deep fried convnets[34]。

另一种获得小网络的方法是收缩、分解或压缩预训练网络。基于乘积量化的压缩 [30], 基于哈希 [2]，霍夫曼编码 [7]；此外，此外，还有各种因子分解来加速预训练网络 [17, 19]。另一种训练小网络的方法是蒸馏 [10]，用较大的网络来 teach 小网络。还有一种方法比较新兴，称为 low bit networks[4, 13, 23]。

1.2 深度可分离卷积

深度可分离卷积是 MobileNet 的基础，它与普通卷积的区别主要在于结构，而正式这种结构，导致了网络参数的大幅度下降与模型推理速度的大幅度提高，



(a) 传统卷积　　(b) DP 卷积

图 1: 传统卷积与 depthwise 卷积对比

从图上我们可以看出传统卷积与深度可分离卷积（Depthwise Separable Convolution）的不同。假设我们的输入图像为 3 通道，尺寸是假设为 $D \times D$ 经过 stride=1 卷积后，假

设卷积核为 $K \times K$ 要产生 n 通道卷积，那么我们传统卷积计算量是：

$$C1 = 3 \times D \times D \times K \times K \times n \tag{1}$$

那么以相同的尺度，要输出相同的通道和尺寸的特征图，Depthwise Separable Convolution 的计算量又是多少呢？

从图1(b) 可以看出，dp 卷积相对于传统卷积，是分两步走的，按原论文中的意思是，第一进行图像滤波，第二部进行特征生成，第一步先对原来三通道图像的三个通道分别进行卷积，生成的中间结果我们可以看出依然是三通道的。第二部对生成的三通道的中间结果进行 1×1 卷积核，n——outchannel, 生成输出特征图。

第一步的计算量：

$$D \times D \times K \times K \times 3 \tag{2}$$

第二部计算量：

$$D \times D \times 1 \times 1 \times n \tag{3}$$

我们将式2和式3相加，得到：

$$C2 = D \times D \times K \times K \times 3 + D \times D \times 1 \times 1 \times n \tag{4}$$

然后我们对比以下 dp 卷积和传统卷积的计算量：

$$d = \frac{3 \times D \times D \times K \times K \times n}{D \times D \times K \times K \times 3 + D \times D \times 1 \times 1 \times n} \tag{5}$$

从式5 可以看出，当我们假设卷积核等于 3 的时候，这个比值十分等于 $\frac{1}{n} + \frac{1}{27}$ 的倒数，可以看到，这个计算量和参数量的区别是十分巨大的，而得到的却都是同样通道的特征图。

1.3 网络结构

我们现在来具体看看实际的 Mobilnet 的网络结构：如图2所示，右图即为即为深度可分离卷积的模块，我们称其为 Conv dw 卷积。

我们再来看看在 Conv dw 基础上实现实现的整个 Mobilenet Net 结构：我们可以从3看到网络主要有四种结构，Conv1-1,Conv dw 3-3,Conv 3-3, Full Connected。

论文中分析了网络中这四种结构的参数和计算量对比：

从图4可以看出计算资源和参数大都几种到了 conv1-1 和全连接层了。

1.4 超参 $\aleph$ 和 ρ

1.4.1 窄网络

该论文的另一个亮点，则是提出两个超参数 α 和 ρ, 我们不禁要问这两个参数是做什么的呢？论文中给出的解释是 Tinner MOdell 和 Reduced Representation, 大致意思就是

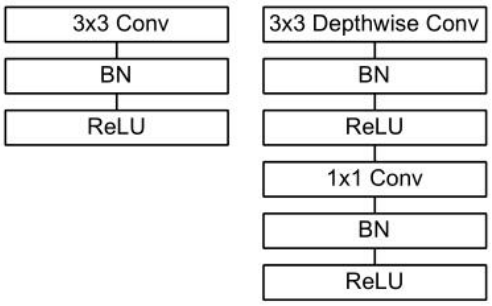


图 2: 左:batchnorm 和标准卷积层线性整流函数（Rectified Linear Unit）右: 深度可分离的卷积，深度和点态层，然后是批范数和 ReLU。

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

图 3: MobileNet 主体架构

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

图 4: 每层网络资源情况

为了更‘瘦’的模型，和‘多分辨率’效果。
我们从图5可以看出，在 conv MobileNet 和 conv dw Mobilenet 的对比中，采用了 conv dw 卷积的网络的参数和计算量 (Mult-adds) 下降了接近八倍之多。

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Table 5. Narrow vs Shallow MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.75 MobileNet	68.4%	325	2.6
Shallow MobileNet	65.3%	307	2.9

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

图 5: 参数和效果对比图

那么如果如果遇到一种情况，也就是将即便参数下降了这么多，还是显得太大，还是无法适应我们的目标平台，怎么？这也就两个超参数 α 和 ρ 的作用了。

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F \tag{6}$$

式6中，$\alpha \in (0, 1]$，一般设为 1，0.75，0.5，0.25，它是用来空值通道数 M,N 的，M 为输入通道，N 为输出通道，我们从图5可以看出，$\alpha = 0.75$ 和 $\alpha = 1.0$ 的时候，精度从 70.6% 降低到了 68.4%, 而参数和计算量降低了近乎一半。论文中将这种降低通道的数的方法成为**窄网络**。而将减少网络深度的网络成为**浅网络**。

同样我们可以从图5中可以看出，0.75 的窄网络和浅网络的对比，在参数和计算量相当的情况下，宅网络要比浅网络准确度表现更好。

1.4.2 多分辨率效果

α 用控制卷积通道数的方法来控制参数，使得在准确度下降不高的情况下，进一步降低网络的计算量和参数量，而 ρ 则是用来控制图像输入尺度的。

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F \tag{7}$$

式7就是完整的超参控制公式。当然超参数 ρ 一般是直接手动的来进行控制输入图像的大小尺度即可，从图5最后一表可以看出，不同尺度的准确率和相应的参数和计算量情况。

1.5 与其他卷积网络框架对比

论文中，将 MobileNet 作为 bacbone 骨干网络部署到目标检测框架中，并且根据当年 COCO 挑战赛的报告数据进行了对比，MobileNet 在 faster-RCNN[25] 和 SSD[21] 框架下与 VGG 和 Inception V2[16] 进行了比较。，SSD 被评估为 300 输入分辨率（ssd300），faster-RCNN 与 300 和 600 输入分辨率（FasterRCNN 300，FasterRCNN 600）进行了比较。更快的 RCNN 模型评估每个图像 300 个 RPN 建议框。

对于这两种框架，MobileNet 在计算复杂度和模型大小上与其他网络相比具有可比性。

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

图 6: 使用不同框架和网络架构的 COCO 对象检测结果比较

图 7: 以 MobileNet 为骨干网络的 SSD 示例

我们从图6可以看出，MAP 指标 (AP at IoU=0.50:0.05:0.95) 相比其他骨干网络的目标检测框下降了一到三个百分点，但是计算量和参数量大小却比他们少 5-10 倍，这显然是值得的。

1.6 小结

经过我们仔细的阅读这篇论文，我们得知了作者是如何通过引入新的卷积 DW 卷积来代替传统卷积，同时引入两个超参数来对模型的参数和计算量进行控制，在将计算量和参数量与准确度之间进行权衡，以达到可以将模型运用于嵌入式或者移动平台。

然而，方法虽然有效，但是经过测试，有人发现 dw 网络中的图2conv1×1 卷积的梯度很容易便程 0，导致损失损失很多信息。

2 MobileNetV2

MobileNetv2 网络模型是 2018 年谷歌团队在 v1 基础上提出的 [26], 主要引入了两个改动，Linear Bottleneck 和 Inverted Residual Blocks. 也就是线性 Bottleneck 和反残差网络。所以，我们分析的重点也就几种在这两个部分。

2.1 Linear BottelNeck 的提出

在分析 LinearBottelNeck 的时候，我们需要先分析以下 Relu 激活函数8的作用，一般用它来代替 sigmod 函数，因为 sigmod 激活函数很容易引起梯度消失或者梯度爆炸。然而 Relu 的激活张量如果应用于低纬度可能会使得原本的输入信息出现丢失。

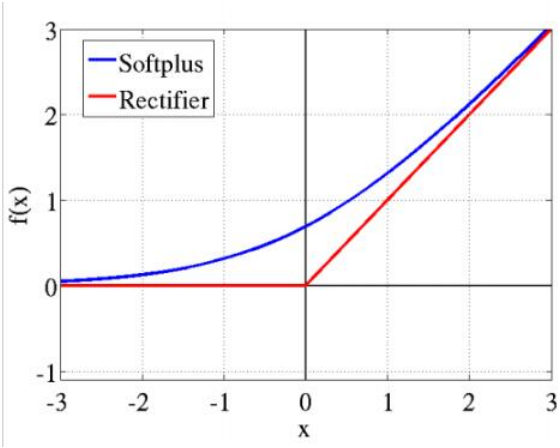


图 8: Relu 函数

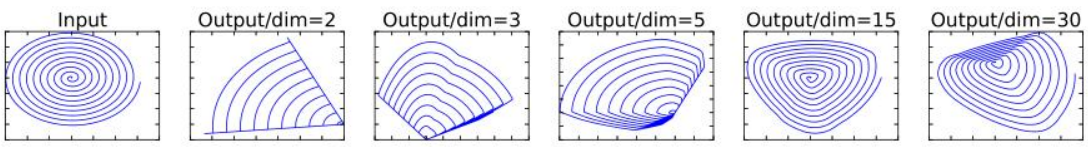


图 9: Relu 不同维度应用后还原后得到的信息量对比

从图9可以看出，当 relu 函数应用于低纬度特征下后，经过信息还原，信息丢失了很多，而当特征维度越来越高，应用 relu 激活张量后，还原后，保留的信息就越多，论文中称这种高纬度的 relu 激活近似为线性，因为信息没有或者丢失的很少。说的直白点，就是通道数数越多，经过 relu 后，在一个通道上丢失的信息可能在别的通道上保留着，这样将最后的结果进行叠加，那么丢的信息也就微乎其微了。

而瓶颈层又于 rule 激活张量的这些特性有什么关系呢？我们从上边的分析得出，越是高维，利用 rule 激活保留的信息越接近于线性，那么我们如果遇到这种情况：输入的感兴趣的信息是低维的，该怎么办？不能用 relu 激活张量了嘛？很简单，将输入信息提升到高维，然后使用 relu，之后再进行降维到我们的目标维度即可。论文中将这种方法成为 BottleNeck。这相对于 mobilenetv1 来说是一个很明显的提升。

2.2 反残差 Inverted residuals

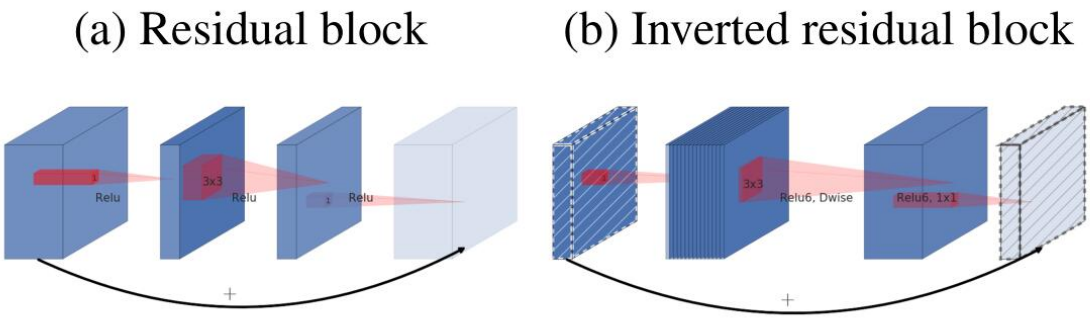


图 10: 残差于反残差对比

从图10可以看出，残差 [8, 32] 于反残差结构的对比，残差块是在多通道上进行叠加，反残差正好相反，是在低通道上进行叠加，通过残差链接，特征图可以总和不同尺度的特征，那么反残差块呢？通道数降低了是否也能达到综合不同尺度特征的效果呢？这就要结合我们之前所说的 BottleNeck，bottleneck 的结构正好就是中间维度高，两端维度低，并且可以线性的保留信息。正好于反残差完美结合。

将 relu 的高纬特性，和反残差结构组合起来，就得到了 Bottleneck residual block ，结构网络结构如图11所示，与 MobileNetV1 中的 dp 卷积块不同，这里先使用 1×1 的卷积对输入图像尺度为 $h \times w \times k$ 的张量进行维度的提升，以方便 relu 激活张量可以保持线性而不丢失信息，其中的 t 是维度扩展因子，然后第二步使用 3×3dw 卷积，对 tk 个通道分别使用 $k = 3 \times 3, stride = s$ 的卷积，对张量进行滤波处理，然后使用 1×1 的卷进行降维度处理。

我们还注意到，前两部卷积后使用的都是 Relu6 激活函数，relu6 与 relu 激活函数的不同在于后者的范围是 $0 \to \infty$, 而前者的激活范围是 $0 \to 6$, 在移动端或者嵌入式端，

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

图 11: 反残差瓶颈块

float16 下，可以精确描述数据精度，否则范围太大，无法描述大范围的数值，效果反倒不好。

对于扩展因子 t，论文中描述，一般取值在 $5 \rightarrow 10$ 即可。

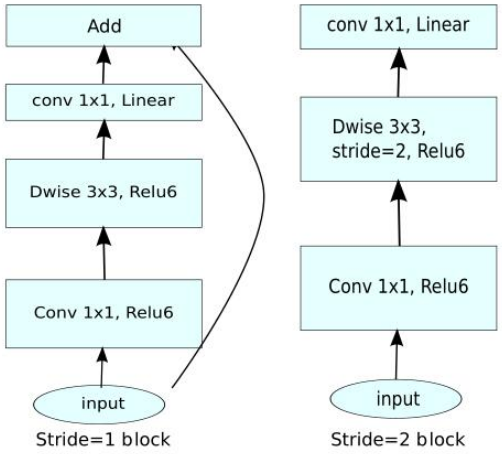


图 12: MobilenetV2

最后的 BottleNeck 卷积块，如图12所示，左边为有残差链接的块，右边为无残差链接的块，结合图13可以看出，只有在每个上层卷积组的时候，stride=2，每个卷积块层的内部图像的尺度是不变的，stride=1，也就是图12中的右图是为了缩放张量尺度，而残差层都是在每个卷积块的内部的。

从图13可以看出，标准输入是 224，通道数为 3 的图像，其中 t 表示膨胀系数，c 表示卷积输出通道，n 表示每个卷积层内部的 bottleneck 块重复次数，s 表示每个卷积层第一个 bottleneck 块的卷积 stride 大小。

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

图 13: MobilenetV2 整体网络结构

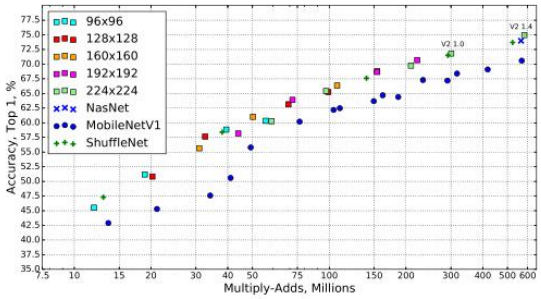


图 14: 不同网络在图像分类方面的比较

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

图 15: 不同网络在图像分类方面的比较

2.3 效果比较

图14对不同设置不同的超参数1.4的 mobileNetv2 网络与 NasNet[36],ShuffleNet[35], MobileNetV1[12] 在图像分类效果上的比较。横轴表示计算量，纵轴表示准确率，可以看出计算量相当情况下 MobileNetV1 的效果已经有所落后了，但是我们可以看出，MobileNetV2 的设置不同超参 $\aleph$ 和 ρ 的情况下，效果均好于其他网络。从图15也可以看出，相比于其他移动网络，MobileNetV2 的推理速度更块。

该论文实验部分另一个亮点，个人认为是将 SSD[21] 网络进行修改部分，不仅是用 MobileNetV2 来替换其骨干网络，而且将其预测部分的卷积网络替换为 BottleNet 卷积，使得模型更加轻量化，论文中称该版本为 SSDLite。

	Params	MAdds
SSD[34]	14.8M	1.25B
SSDLite	**2.1M**	**0.35B**

(a)

Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

(b)

图 16: SSDlite 效果比较

从图16(a)中，我们可以看出，SSDlite 相比于 SSD[21], 参数量下降了近七倍，计算量下降了近四倍。从图16(b)中，看出，无论是参数量还是计算量或者推理速度，效果也比 YOLOV2[24] 好。

2.4 小结

MobileNetV2 最难理解的其实是 Linear Bottlenecks，论文中用很多公式表达这个思想，但是实现上非常简单，就是在 MobileNetV2 微结构中第二个 PW 后无 ReLU6。对于低维空间而言，进行线性映射会保存特征，而非线性映射会破坏特征，而 Bottelnecks 就是用维度提升然后 relu，然后降维的方法实现本来是低纬度的 relu 线性转换。

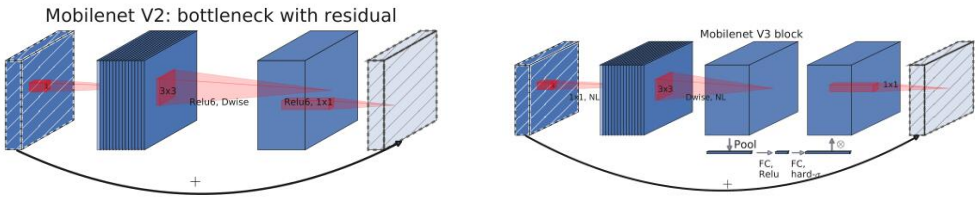
3 MobileNetV3

最后一篇论文同样是谷歌团队 19 年发表的题名为《Searching for MobileNetV3》[11] 的论文，顾名思义，V3 网络是利用搜索技术来得到模型。该论文的两大创新点：

1. 互补搜索技术组合：由资源受限的 NAS[36] 执行模块级搜索，NetAdapt[33] 执行局部搜索。

2. 网络结构改进：将最后一步的平均池化层前移并移除最后一个卷积层，引入 h-swish 激活函数。

3.1 结构特点



(a) (b)

图 17: MobileNetV3 block

从图17可以看出 MobileNetV3 block 相对 MobileNetV2 block 的变化，它用了 MobileNetV1[12] 的深度可分离卷积（depthwise separable convolutions）、MobileNetV2[26] 的具有线性瓶颈的逆残差结构 (the inverted residual with linear bottleneck) 和 MnasNet[29] 的基于 squeeze and excitation 结构的轻量级注意力模型。MobileNetV3 正式综合以上三种网络设计结构而完成的。

3.2 互补网络搜索

个人认为网络搜索是本论文的基础，因为从论文中我们得知，网络的各个模块都是应用 NAS[29] 搜索得到的，那么网络搜索究竟是什么东西呢？之前有甚多新颖的网络，比如 ResNet[8, 32],ShuffleNet[22, 35] 等，都是通过研究人员手工设计出来的，但是随着神经网络越来越复杂，设计网络的时间成本和试错成本越来越难以让人接受，而网络搜索就是解决这个难题的，简单来说就是应用类似强化学习的方法，在给定的搜索空间中，依照某种搜索策略进行网络的部件搜索，产生一个网络，然后对这个网络进行评估，循环不断的进行，直到生成最优的网络。

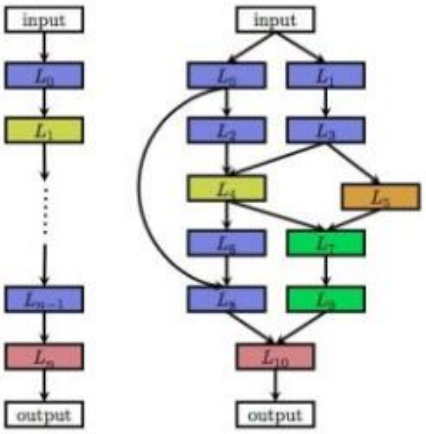


图 18: 网络搜索示例

从18可以看出，网络搜索的简单例子，可以将搜索空间设定为组成网络的各种基本层，也可以设定称基本块，甚至还允许网络进行分割和残差链接。

而在改论文中，用的就是一种互补网络搜索的方法，作者结合了两种策略，资源受限的 NAS(platform-aware NAS)[29],NetAdapt[33]，前者用于计算和参数量受到限制的前提下搜索网络的各个模块，所以称之为模块级搜索，后者用于对各个模块确定之后的网络，进行微调，也就是参数和层级级别搜索。从而达成互补性搜索。

3.3 网络的改进于 **h-swish**

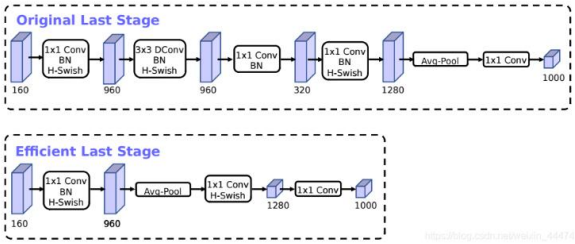


图 19: V2 与 V3 网络最后部分

图19中，上边的是 v2 网络的尾部结构，下边的则是 v3 的尾部，因为作者发现 (NAS 发现？) 这部分的计算量很大，所以进行了简化，移除了 33 卷积层，将 avg-pool 层前移。

同时我们也可以从图中看出，每层卷积的最后用的是 H-Swish 激活函数，因为作者团队也发现了新出的激活函数 swish[5, 9] 可以有效的提高网络精度，其定义如下：

$$\text{swish } x = x \cdot \sigma(x) \tag{8}$$

虽然这种非线性函数提高了精度，但在嵌入式环境中代价是非零的，因为在移动设备上计算 sigmoid 函数要昂贵得多，于是作者提出了解决方式：

$$\text{h} - \text{swish}[x] = x\frac{\text{ReLU}\,6(x+3)}{6} \tag{9}$$

相近似的激活函数也在另一篇论文中提出 [1], 可以达到计算量小，而准确率相近，并且延迟还大大降低。

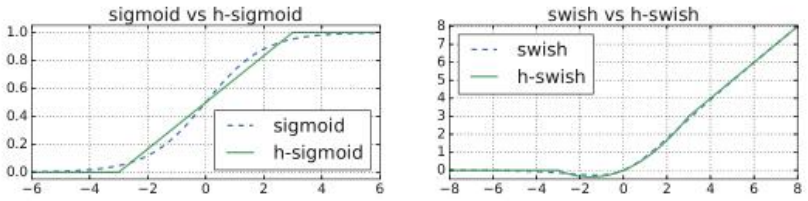


图 20: 两中激活函数的比较

从图中可以看出，swish 和 h-swish 激活函数是十分相近的，而计算量和延迟却下降了。

3.4 网络结构

MobileNetV3 被定义为两个模型：MobileNetV3 大的和 MobileNetV3 小的。这些模型分别针对高资源和低资源用例。这些模型是通过将支持平台的 NAS[29] 和 NetAdapt[33] 应用于网络搜索并结合本节中定义的网络改进而创建的。

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

(a) big

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

(b) small

图 21: MobileNetV3-Large and MobileNetV3-Small

其中的 bneck 就是第二节中2中应用 BottleNecks。SE 表示是否应用了挤压和激励[29],NL 表示应用的非线性激活类型，RE 表示 ReLU,HS 表示 hard-swish 激活。NBN 表示不应用 batch normalization。

3.5 效果对比

我们主要看看目标检测方面的效果对比。作者使用 MobileNetV3 作为 SSDLite[26] 中主干特征提取器的替代品，并与 COCO 数据集上的其他主干网进行了比较 [20]。

图22中给出了 COCO 测试集的结果。随着信道的减少，MobileNetV3 Large 比具有几乎相同 mAP 的 MobileNetV2 快 27%。MobileNetV3Small（信道缩减）map 稍微低于 V2 和 MnasNet[29], 但是速度却快了 35%。

3.6 小结

当看到 MobileNetV3 应用强化学习的方法来进行网络搜索来构建网络的时候，真的感觉很惊奇，因为在学习深度学习的过程中，各种网络纷繁复杂，有时候不禁会想作者

Backbone	mAP	Latency (ms)	Params (M)	MAdds (B)
V1	22.2	228	5.1	1.3
V2	22.1	162	4.3	0.80
MnasNet	23.0	174	4.88	0.84
V3	22.0	137	4.97	0.62
V3†	22.0	119	3.22	0.51
V2 0.35	13.7	66	0.93	0.16
V2 0.5	16.6	79	1.54	0.27
MnasNet 0.35	15.6	68	1.02	0.18
MnasNet 0.5	18.5	85	1.68	0.29
V3-Small	16.0	52	2.49	0.21
V3-Small†	16.1	43	1.77	0.16

图 22: 不同骨架的 SSDLite 在 COCO 测试集上的目标检测结果

到底是怎么构建这些网络的，有时候可以找到理由，但是大部分还是无法得到其中的道理，而网络搜索技术的应用，或许真的会成为以后神经网络构建的主流吧。

4 简单的实验

实验方面暂时只是进行了以 MobileNetV2 与 ResNet50[8] 作为 faster-RCNN[25] 目标检测框架的骨干网络。

4.1 数据集

数据集应用的并不是 pascal voc 数据集 [6]。而是华南理工大学人机智能交互实验室的提供的 SCUT-Ego-Gesture Dataset[31] 数据集。

数据集地址：http://www.hcii-lab.net/data/scutegogesture/

SCUT-Ego-Gesture 数据集是 SCUT-Ego-Finger 数据集的扩展。该数据集包含 59111 个 RGB 图像，这些图像包含 16 种不同的手势（11 种单手手势和 5 种双手手势）。单手 11 种手势：单手（3374 帧）、单 2 帧（3763 帧）、单三帧（3768 帧）、单四帧（3767 帧）、单五帧（3755 帧）、单六帧（3757 帧）、单七帧（3773 帧）、单八帧（3380 帧）、单九帧（3769 帧）、SinleBad（3761 帧）、SingleGood（3769 帧）；双手 5 种手势：PairSix（3681 帧）、PairSeven（3707 帧）、PairEight（3653 帧）、PairNine（3653 帧）和 PairTen（3536 帧）。数据集样本如图23：

本次实验我们仅仅提取 11 个单手手势数据作为训练集和验证集，训练集集为百分之 70%，验证集为剩下的百分之 30%.

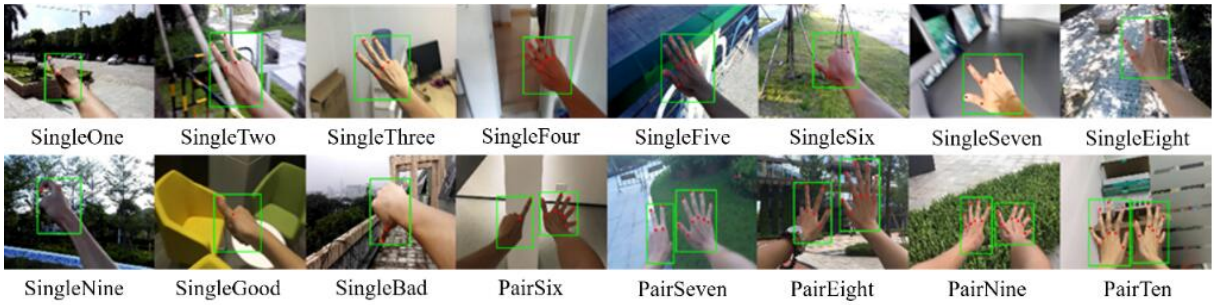


图 23: SCUT-Ego-Gesture 数据集样本

4.2 实验内容

实验内容方面，我们主要想要验证 mobilenetV2 与 resnet50 为骨干网络时，fasterCNN 的检测结果差别，Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100] 指标和 Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100] 指标。

表 1: 以 mobilenetv2 和 resnet50 作为骨干网络的 fasterrcnn 结果比较

backbone 指标	AP	RC
ResNet50	0.846	0.892
MobileNetV2	0.833	0.878

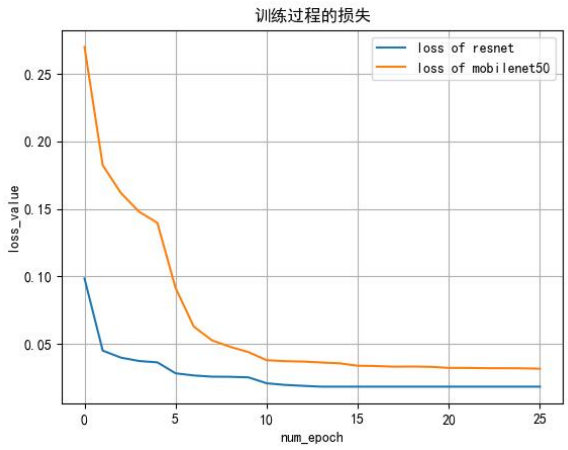


图 24: 训练过程的 loss 变化曲线

我们从数据集上最终训练的结果表1可以看出，resNet50 的效果要比用 MobileNetV2 的结果要好一些。

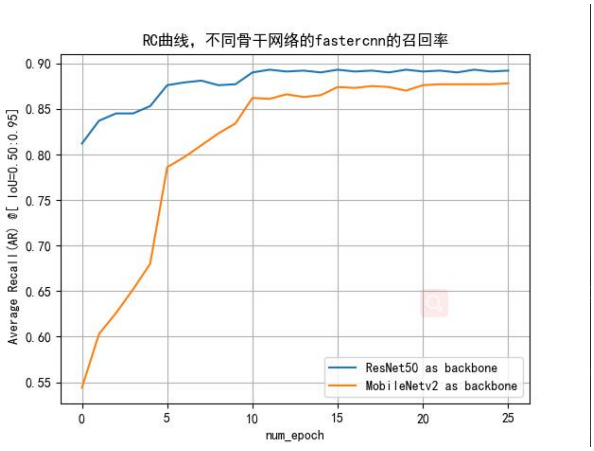


图 25: 训练过程的召回率曲线

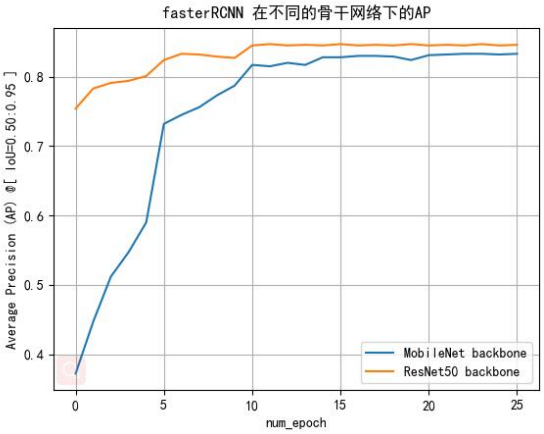
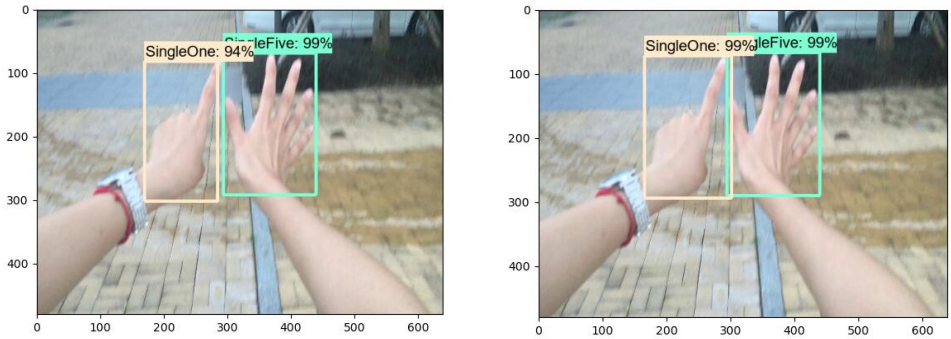


图 26: 训练过程的 AP 曲线



(a) mobileNet 测试结果　　(b) resnet50 测试结果

图 27: 简单的效果展示

4.3 还未完成的工作

虽然做了一些相关的工作，但是很明显，这点实验对论文中的验证还是十分不足的：

- 无法测试 mobilnetv2 和 resnet50 在 fasterrcnn 上推理时间差别，因为 fasterrcnn 本身并不是十分快的网络，而且运算也并非几种在骨干网络，RPN，于 ROIpooling 层的计算量也十分大的，所以就无法测试骨干网络的推理时间的快慢了。
- 实验只是在测试了 mobilenetV2，并未测试 V3。

所以，接下来，还会计划实行以下计划：

- 用 ssd300，ssdlite 来代替代 fasterRcnn 进行手势检测实验。
- 用 mobilenetV3 替换 V2 骨干网络，并对比效果。

参考文献

[1] R. Avenash and P. Vishawanth. Semantic segmentation of satellite images using a modified cnn with hard-swish acti-vation function. 2019.

[2] Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. *Computer Science*, pages 2285–2294, 2015.

[3] Franois Chollet. Xception: Deep learning with depthwise separable convolutions. 2016.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean Pierre David. Training deep neural networks with low precision multiplications. *Computer ence*, 2014.

[5] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw*, 2018.

[6] Mark Everingham and John Winn. The pascal visual object classes challenge 2007 (voc2007) development kit. *International Journal of Computer Vision*, 111(1):98–136, 2006.

[7] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2016.

[9] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. 2016.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *Computer ence*, 14(7):38–39, 2015.

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, and Vijay and Vasudevan. Searching for mobilenetv3. 2019.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017.

[13] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18, 2016.

[14] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[17] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *Computer ence*, 4(4):XIII, 2014.

[18] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *Computer Science*, 2014.

[19] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *Computer ence*, 2014.

[20] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. 2014.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. 2016.

[22] Ningning Ma, Xiangyu Zhang, Hai Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. 2018.

[23] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. 2016.

[24] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 6517–6525, 2017.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(6):1137–1149, 2017.

[26] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[27] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for texture classification. *Computer ence*, 3559:501–515, 2014.

[28] Vikas Sindhwani, Tara N Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. 2015.

[29] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018.

[30] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] Wenbin Wu, Chenyang Li, Zhuo Cheng, Xin Zhang, and Lianwen Jin. Yolse: Egocentric fingertip detection from single rgb images. In *IEEE International Conference on Computer Vision Workshop*, 2018.

[32] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] Tien Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. 2018.

[34] Zichao Yang, Marcin Moczulski, Misha Denil, Nando De Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. *Computer Science*, 2015.

[35] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. 2017.

[36] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2016.

A 附录：数据集与代码地址

实验数据集：http://www.hcii-lab.net/data/scutegogesture/
实验代码：https://github.com/smiledinisa/-web_mining_final_work