背景

通常我们去评价一个模型时，首先看的应该是它的精确度，当精确度达不到要求的时候，模型预测的速度再快，在后续模型部署的时候内存占用量再小，都无济于事。但如果模型达到一定的精确要求时，就需要更进一步的评价指标来评价模型，例如，前向传播时所需的计算力，它反应了对硬件如 GPU 性能要求的高低；参数量大小，它反应所占内存大小。

现在很多网络模型的设计都是以 FLOPs 为指标，浮点运算量是实际运算过程中的加减乘除计算过程中的计算次数，描述计算力。但这是一种非直接的度量网络计算复杂度的方式，还有许多直接度量的方式，例如模型的速度，但是速度又与其他因素有很大的关系，比如内存访问成本、并行度、硬件条件，FLOPs 没有考虑这几个对速度有相当大影响的重要因素。

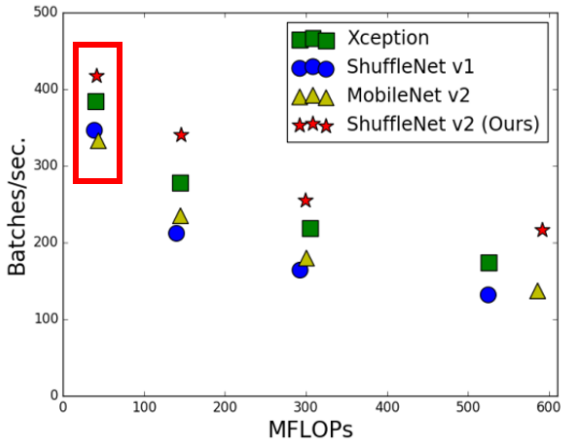

图 1

内存访问成本（MAC）是指计算机在进行计算时候要加载到缓存中，然后再计算，这个加载过程是需要时间的。其中，分组卷积是对 MAC 消耗比较多的操作（例如 AlexNet 将卷积分到多个 GPU 上）；在相同的 FLOPs 下，具有高并行度的模型可能比具有低并行度的另一个模型快得多，如果网络的并行度较高，那么速度就会有显著的提升；有的硬件平台会对卷积操作进行优化，比如：cudnn 加强了对 3×3 卷积计算的优化，这样一来，不同平台下的 FLOPs 指标大小没有什么可比性。通过图 1 的红色框中可以看出，即使是有相同的 FLOPs，在速度上也是有差异的，因此选用 FLOPs 作为计算复杂度的评价指标是非常片面的。

除此之外，通过将几个网络的运行时间拆分，可以看出卷积操作占据了大部

分时间，而 FLOPs 指标只考虑了卷积部分的运行时间，其他的操作，例如数据 I/O 操作、数据打乱操作、逐元素操作等，也会占用大量的时间，因此从这个角度看 FLOPs 指标并不能够反应实际的运行时间，很片面。

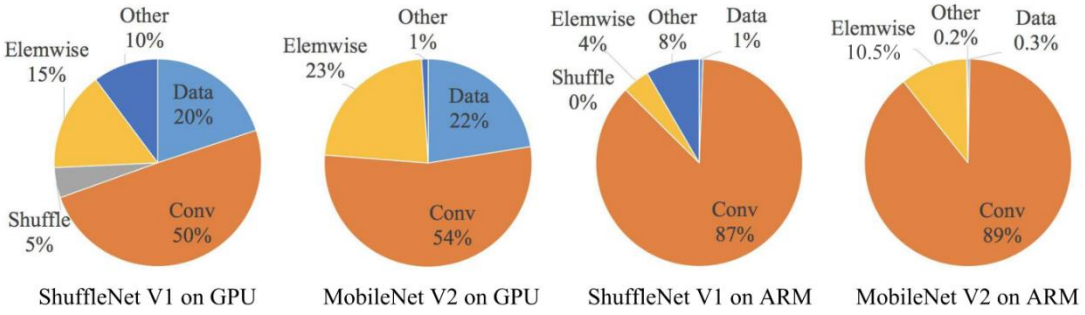

图 2

本文通过从不同的角度分析网络的运行时间，得出了几条高效网络的设计指南，与此同时也提出了 ShuffleNetV2 网络模型。

方法

高效的模型设计指南

①输入输出通道数相同时，内存访问成本最小

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 3

图 3 是对 MobileNetV1 各个操作所占用的计算量和参数量的分析，可以看出 1×1 卷积占总计算量的 94.86%，占总参数量的 74.59%，可见 1×1 卷积导致计算量很大。

ShuffleNetV1 是提出了逐点分组卷积来解决上述问题，在本文中是通过分析 1×1 卷积的输入输出通道数，来设计使得 MAC 最小的卷积核形状。

从数学角度证明一下：

以 1×1 卷积为例，假设输入通道数为c_1，输出通道数为c_2，输出特征图的大小为$h \times w$，那么该操作的计算量为：

$$1 \times 1 \times c_1 \times h \times w \times c_2 = hwc_1c_2 \tag{1}$$

内存访问成本为：

输入部分：$h \times w \times c_1$，输出部分：$h \times w \times c_2$，权重：$1 \times 1 \times c_1 \times c_2$，所以 MAC 为三部分之和：$h \times w \times c_1 + h \times w \times c_2 + 1 \times 1 \times c_1 \times c_2 = h \times w \times (c_1 + c_2) + c_1 \times c_2$，根据基本不等式“两个正实数的算术平均数大于或等于它们的几何平均数”，可以得出：

$$h \times w \times (c_1 + c_2) + c_1c_2 \geq 2hw\sqrt{c_1c_2} + c_1c_2 \tag{2}$$

设$hwc_1c_2 = B$，那么$c_1c_2 = \frac{B}{hw}$，则代入(2)式中可得：

$$h \times w \times (c_1 + c_2) + c_1c_2 \geq 2hw\sqrt{c_1c_2} + c_1c_2 = 2\sqrt{Bhw} + \frac{B}{hw} \tag{3}$$

可以看出当且仅当$c_1 = c_2$时等号成立，因此当输入通道数等于输出通道数时，MAC 最小。

实验证明一下：

		GPU (Batches/sec.)				ARM (Images/sec.)		
c1:c2	(c1,c2) for ×1	×1	×2	×4	(c1,c2) for ×1	×1	×2	×4
1:1	(128,128)	1480	723	232	(32,32)	76.2	21.7	5.3
1:2	(90,180)	1296	586	206	(22,44)	72.9	20.5	5.1
1:6	(52,312)	876	489	189	(13,78)	69.1	17.9	4.6
1:12	(36,432)	748	392	163	(9,108)	57.6	15.1	4.4

图 4

图 4 是在 FLOPs 相等的前提下进行的一系列对照实验，可以看出当输入输出通道数相等时，网络运行的最快。

②过多的分组会增加内存访问成本

使用分组卷积，当 FLOPs 固定的前提下，可以使用更多的特征通道，从而提高精度，但是更多的特征通道就意味着 MAC 的增加。

依然是从两方面进行证明。

从数学角度证明一下：

在 ShuffleNetV1 中已经分析过分组卷积和普通卷积在计算量上的差异，可以得出一个一般性的结论，就是分组卷积的计算量是普通卷积计算量的$\frac{1}{g}$，其中g为分组的组数，那么(1)式就改写为：

$$\frac{1\times1\times c_1\times h\times w\times c_2}{g}=\frac{hwc_1c_2}{g} \tag{4}$$

MAC 为：

$$h\times w\times c_1+h\times w\times c_2+\frac{1\times1\times c_1\times c_2}{g}=h\times w\times(c_1+c_2)+\frac{c_1\times c_2}{g} \tag{5}$$

设$\frac{hwc_1c_2}{g}=B$，那么$\frac{c_1c_2}{g}=\frac{B}{hw}$，$hwc_2=\frac{Bg}{c_1}$，则代入(5)式中可得：

$$h\times w\times(c_1+c_2)+\frac{c_1\times c_2}{g}=hwc_1+\frac{Bg}{c_1}+\frac{B}{hw} \tag{6}$$

从(6)式中可以看出，MAC 与分组数 g 成正比。

实验证明一下：

		GPU (Batches/sec.)				CPU (Images/sec.)		
g	c for ×1	×1	×2	×4	c for ×1	×1	×2	×4
1	128	2451	1289	437	64	40.0	10.2	2.3
2	180	1725	873	341	90	35.0	9.5	2.2
4	256	1026	644	338	128	32.9	8.7	2.1
8	360	634	445	230	180	27.8	7.5	1.8

图 5

图 5 是通过改变通道数使得所有实验是在相同 FLOPs 下进行的，可以看出分组的组数越多，网络的运行速度就越慢。

③模型中的一些碎片化操作不利于网络并行化

这里的碎片化操作指的是在模型搭建时，每个 block 里面的卷积操作次数或者是池化操作的次数。虽然这些碎片化操作有利于精度的提升，但是不利于像

GPU 这样的有很强并行处理能力的硬件设备，因此导致了在 GPU 设备上效率不高。

	GPU (Batches/sec.)			CPU (Images/sec.)		
	c=128	c=256	c=512	c=64	c=128	c=256
1-fragment	2446	1274	434	40.2	10.1	2.3
2-fragment-series	1790	909	336	38.6	10.1	2.2
4-fragment-series	752	745	349	38.4	10.1	2.3
2-fragment-parallel	1537	803	320	33.4	9.1	2.2
4-fragment-parallel	691	572	292	35.0	8.4	2.1

图 6

图 6 是在相同 FLOPs 下进行的对比实验，可以看出在 GPU 设备上随着 block 中串行的分支或是并行的分支数增加，网络运行速度会大幅下降，但是，同样的情况在 CPU 上网络的运行速度影响并不是很大。

④逐元素操作对 FLOPs 和 MAC 的影响是不可忽略的

这里的逐元素操作指的是 ReLU、张量之间的相加操作、深度方向的卷积操作这类对 FLOPs 影响不大，但是对 MAC 影响很大的操作都称为逐元素操作。

		GPU (Batches/sec.)			CPU (Images/sec.)		
ReLU	short-cut	c=32	c=64	c=128	c=32	c=64	c=128
yes	yes	2427	2066	1436	56.7	16.9	5.0
yes	no	2647	2256	1735	61.9	18.8	5.2
no	yes	2672	2121	1458	57.3	18.2	5.1
no	no	2842	2376	1782	66.3	20.2	5.4

图 7

图 7 是通过将 ResNet 中瓶颈结构做了一系列对照实验，说明逐元素操作对网络速度有不可忽视的影响。

ShuffleNetV2 的单元结构

遵循以上四条高效网络的设计指南，一个高效的网络模型应该是：

①使用输入输出通道数相同的卷积操作；

②不过度依赖分组卷积操作来减少网络的计算量；

③网络的碎片化程度低；

④逐元素操作较少。

那么现在以这四个条件为准则，反观 ShuffleNetV1（结构如图 8a、b 所示），其为了在不增加 FLOPs 的前提下增加特征通道数，使用了分组卷积以及瓶颈结构，现在看来这样的设计不符合条件 1 和 2。

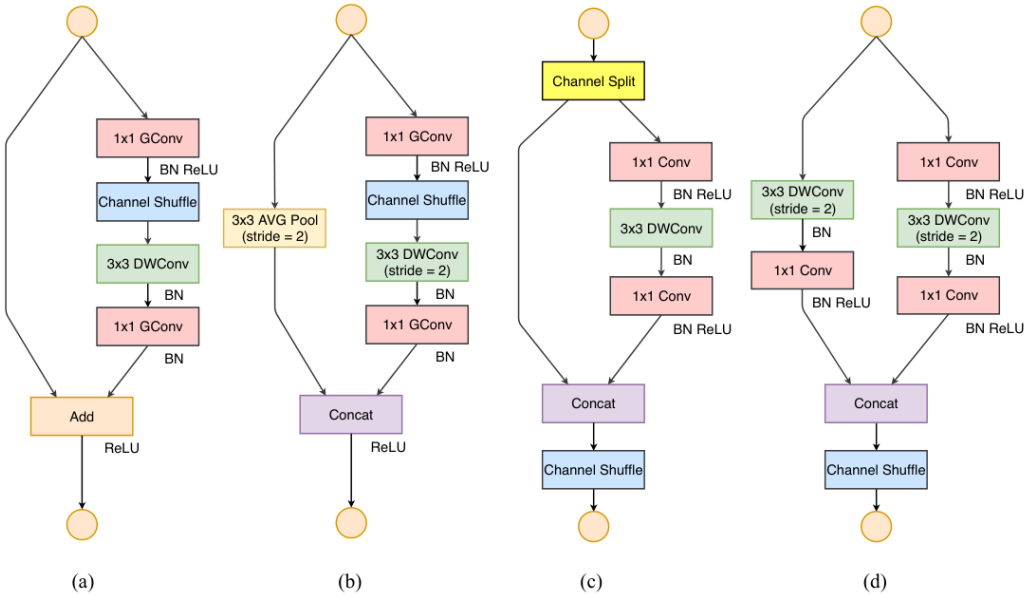

图 8

因此为了得到高效的模型，设计的关键就在于如何在保持特征通道数足够多的前提下既不增加 block 中卷积的次数也不增加分组的数量。在这样的准则下设计了 ShuffleNetV2，其结构如图 8(c)、(d)所示。

图 8(c)中，首先是一个通道分离操作，在遵循设计指南 3 的前提下，将输入的通道数 c 一分为二。这样一分为二可以起到特征复用的效果。

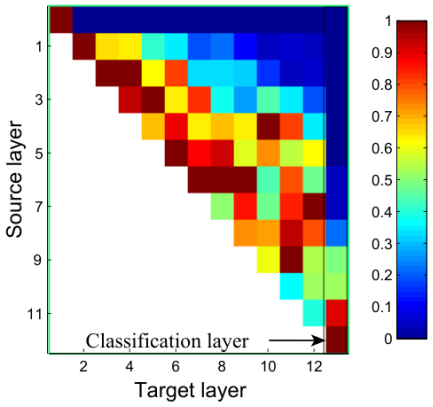

图 9

从图 9 可以看出相邻两层之间的关联性更强，这就意味着将所有层之间进行密集的连接是冗余的。

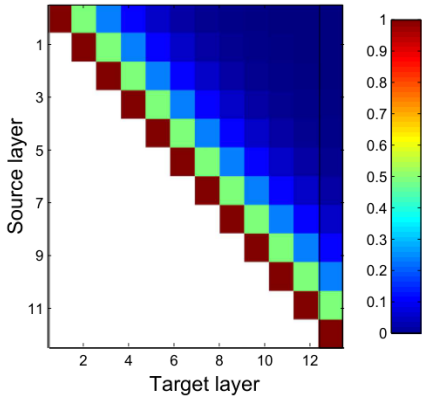

图 10

从图 10 可以看出 block 之间的间隔越多，其特征的相关性就越弱。

再看接下来的右侧分支，右侧分支是 3 个卷积操作，而且这 3 个卷积都是输入输出通道数相等的，也没有使用分组操作，因此符合指南 1 和 2。最后将左侧的恒等映射和右侧分支进行 concat 拼接，没有使用逐元素相加操作，因此符合指南 4。两部分内容拼接后进行了通道重排操作。

图 8(d)是 ShuffleNet2 的下采样模块，和图 8(c)不同的是将深度方向的卷积步长调整为 2，并删除了一开始的通道分离操作，这样做的目的就是下采样，使特征图尺寸减半，通道数翻倍。

Model	mmAP(%)				GPU Speed (Images/sec.)			
FLOPs	40M	140M	300M	500M	40M	140M	300M	500M
Xception	21.9	29.0	31.3	32.9	178	131	101	83
ShuffleNet v1	20.9	27.0	29.9	32.9	152	85	76	60
MobileNet v2	20.7	24.4	30.0	30.6	146	111	94	72
ShuffleNet v2 (ours)	22.5	29.0	31.8	33.3	**188**	**146**	**109**	**87**
ShuffleNet v2* (ours)	**23.7**	**29.6**	**32.2**	**34.2**	183	138	105	83

图 11

图 11 是在 COCO 目标检测任务上轻量级网络进行的对比，发现 ShuffleNetV2 网络更胜一筹。其中 ShuffleNetV2*是在每个 block 的第一个 1×1 卷积之前使用了额外的 3×3 深度方向的卷积，以此来增加感受野大小。