# MobileNetv2

2018

Inverted Residuals and Linear Bottlenecks

## 论文出发点和背景

现在的很多高精度网络需要比较高的计算资源，超出了移动设备和嵌入式设备的能力

## 论文创新思路

我们的主要贡献是一个新的层模块：具有线性瓶颈的反向残差。该模块将一个低维压缩表示作为输入，首先扩展到高维，并使用轻量级的深度卷积进行过滤。特征随后被投影回一个具有线性卷积的低维表示形式。

如果当前激活空间内兴趣流行完整度较高，经过ReLU，可能会让激活空间坍塌，不可避免的会丢失信息。

如果经过ReLU变换输出是非零的，那输入和输出之间是做了一个线性变换的，即将输入空间中的一部分映射到全维输出，换句话来说，ReLU的作用是线性分类器

我们想要兴趣流行存在低维空间中，即想要提升效果，维度是要低一点。但是维度如果低的话，激活变换ReLU函数可能会滤除很多有用信息，而ReLU对于没有滤除的部分，即非零的部分的作用是一个线性分类器。

既然在低维空间中使用ReLU做激活变换会丢失很多信息，论文针对这个问题使用linear bottleneck(即不使用ReLU激活，做了线性变换)的来代替原本的非线性激活变换

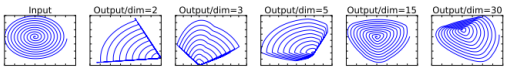



Figure 1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an $n$-dimensional space using random matrix $T$ followed by ReLU, and then projected back to the 2D space using $T^{-1}$. In examples above $n = 2, 3$ result in information loss where certain points of the manifold collapse into each other, while for $n = 15$ to 30 the transformation is highly non-convex.

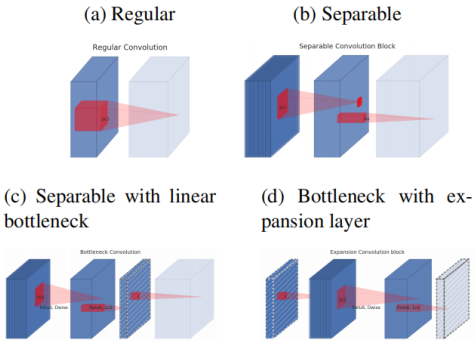



Figure 2: Evolution of separable convolution blocks. The diagonally hatched texture indicates layers that do not contain non-linearities. The last (lightly colored) layer indicates the beginning of the next block. Note: 2d and 2c are equivalent blocks when stacked. Best viewed in color.

# 论文方法介绍

深度可分离卷积能够作为卷积核的替代运用在神经网络中，同时其参数量较于标准卷积大大降低

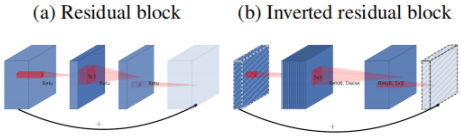
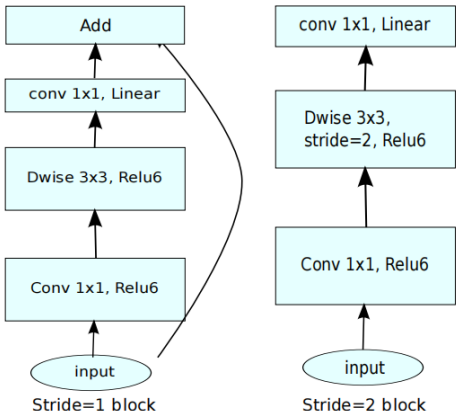
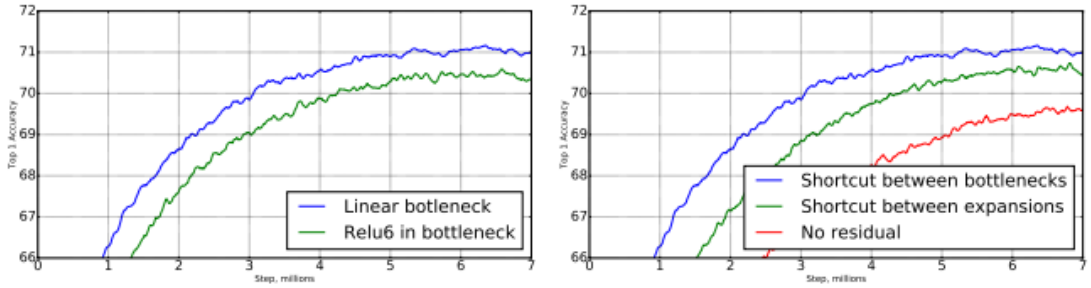



Figure 3: The difference between residual block [8, 30] and inverted residual. Diagonally hatched layers do not use non-linearities. We use thickness of each block to indicate its relative number of channels. Note how classical residuals connects the layers with high number of channels, whereas the inverted residuals connect the bottlenecks. Best viewed in color.




(d) Mobilenet V2




(a) Impact of non-linearity in the bottleneck layer.

(b) Impact of variations in residual blocks.

Figure 6: The impact of non-linearities and various types of shortcut (residual) connections.

## 论文实际效果

| Network | OS | ASPP | MF | mIOU | Params | MAdds |
|---|---|---|---|---|---|---|
| MNet V1 | 16 | ✓ | | 75.29 | 11.15M | 14.25B |
| | 8 | ✓ | ✓ | 78.56 | 11.15M | 941.9B |
| MNet V2* | 16 | ✓ | | 75.70 | 4.52M | 5.8B |
| | 8 | ✓ | ✓ | 78.42 | 4.52M | 387B |
| MNet V2* | 16 | | | **75.32** | **2.11M** | **2.75B** |
| | 8 | | ✓ | 77.33 | 2.11M | 152.6B |
| ResNet-101 | 16 | ✓ | | 80.49 | 58.16M | 81.0B |
| | 8 | ✓ | ✓ | 82.70 | 58.16M | 4870.6B |

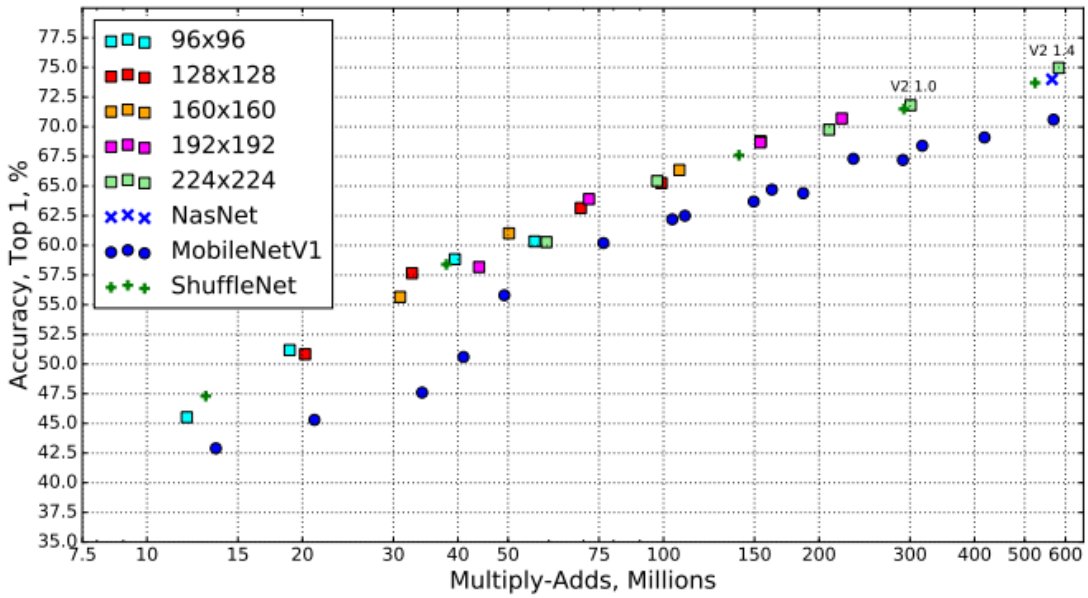



Figure 5: Performance curve of MobileNetV2 vs MobileNetV1, ShuffleNet, NAS. For our networks we use multipliers 0.35, 0.5, 0.75, 1.0 for all resolutions, and additional 1.4 for for 224. Best viewed in color.

# 个人理解

论文中关于流形学习中的一些内容没有看懂，但是主要的话文章提出的只有两点：

1.低维空间中使用ReLU做激活变换会丢失很多信息，因而在bottleneck末尾使用linear bottleneck代替原本的ReLU非线性激活函数。

2.倒残差结构

ResNet是两头大、中间小的结构；mobilenet则是中间大两头小，同时采用了新的激活函数ReLU6

另外看过一篇博客说是，之所以使用ReLU6（将ReLU的最大输出限制为6）就是为了在移动端设备float16/int8的低精度的时候也能有很好的数值分辨率。如果对ReLU的激活范围不加限制，输出范围为0到正无穷，如果激活值很大，分布在一个很大的范围内，则低精度的float16/int8无法很好地精确描述如此大的数值，带来精度损失。