卷积神经网络的工作原理

卷积神经网络用于计算机视觉（人工智能的一个分支），计算机视觉即告诉计算机如何理解图像：人脸识别、自动驾驶等，都是使用计算机视觉技术进行开发的。

卷积神经网络不仅可以处理图像，还可以处理音频、视频、文本等。卷积神经网络其实是一个函数，或者一个黑箱，有输入和输出，输入为二维的像素阵列（即图片），输出为标签（即图片是什么）。一张字母A的图片，输入到卷积神经网络后，得到输出标签“A”，即使对图片进行平移、缩放、旋转、局部加厚或整体加厚，依然可以得到输出标签“A”。即对于一个合格的卷积神经网络，它的鲁棒性和抗干扰性都比较强。

2010年，计算机视觉领域开始广泛使用深度学习和卷积神经网络，从2012年的AlexNet，到2014年的GooglleNet，再到2015年的ResNet。这些网络的深度加深，参数量增大，图像分类的准确率提高，同时计算量增大，结构也变的复杂、臃肿。对于一款兼顾精度与速度、可部署在移动端、终端设备上的轻量化网络的需求日益迫切。边缘计算即模型在移动端、终端设备上本地运行，省去了网络传输、通信延迟，保证了数据安全。边缘计算有着广泛的需求，无人驾驶就是典型的边缘计算场景，它需要无人驾驶汽车在本地实时推理并决策。但是，移动端、终端设备的算力比较低，这对模型的轻量化与加速推理、软硬件协同设计提出了挑战。

轻量化卷积神经网络综述

由图可知，卷积层计算占用了大部分的前向计算耗时，批量大小（Batch Size）越大时，卷积层计算耗时越长，前向计算耗时中的占比也越大。因此，轻量化网络MobileNet主要对卷积层（卷积运算）进行了优化。批量大小比较小时，全连接层计算耗时也比较长。因此MobileNet用全局平均池化（Global Average Pooling）取代了全连接层。

MobileNet基于深度可分离卷积（Depthwise Separable Convolution）优化卷积层，减小计算量和参数量，加速计算，使网络轻量化。

以3通道为例，在标准卷积运算中，3通道卷积核在3通道输入特征图（feature map）上滑动。卷积核滑动到某位置时，感受野的部分输入张量与卷积核张量按元素相乘并求和，得到一个标量值，即此位置的输出张量值。3通道输入特征图与128个3通道卷积核进行卷积运算后可得到128通道输出特征图。

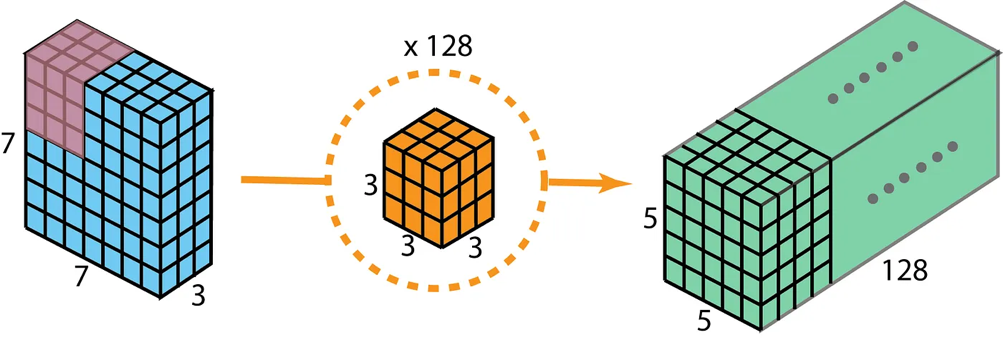


图 1 标准卷积

对比深度可分离卷积运算,深度可分离卷积由两个步骤组成：深度卷积（Depthwise Convolution）和1x1卷积（1x1 Convolution）。

深度卷积将3通道输入特征图分解为3个单通道特征图，用3个单通道卷积核分别与每一个单通道特征图进行卷积运算，得到3个单通道特征图。组合后得到的3通道特征图，作为1x1卷积的输入。

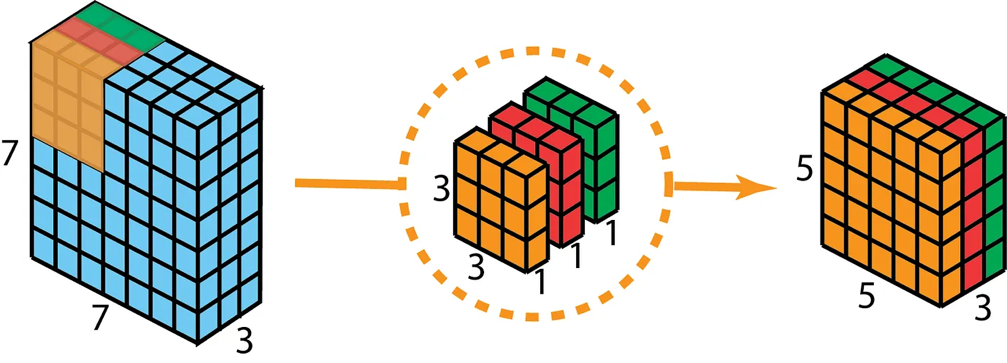


图 2 深度卷积（Depthwise Convolution）

1x1卷积用深度卷积得到的3通道特征图与128个3通道、大小为1x1的卷积核进行卷积运算，得到128个单通道特征图，组合后得到的128通道输出特征图，作为深度可分离卷积的输出。

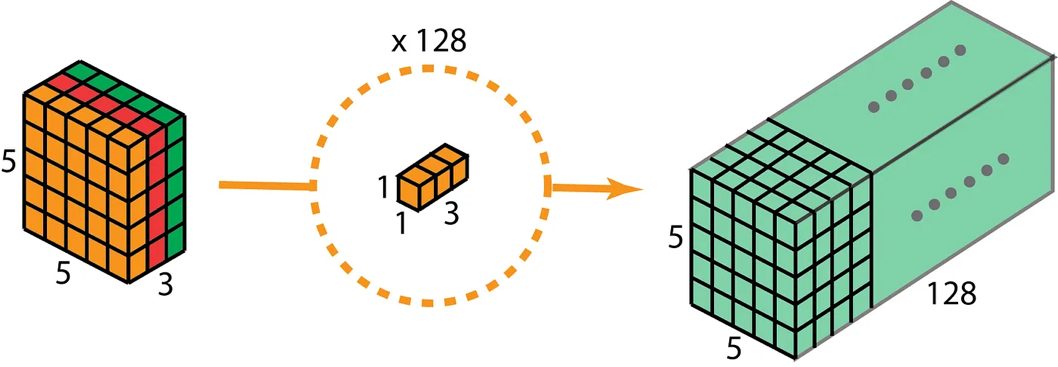


图 3 1x1卷积（1x1 Convolution）

总之，深度可分离卷积即先深度卷积后1x1卷积。深度卷积负责长宽方向的空间信息。1x1卷积负责跨通道信息。通过深度卷积和1x1卷积,将空间信息与跨通道信息解耦。深度可分离卷积可以看作分组卷积的特例。分组卷积即一个卷积核与一组通道的输入进行卷积运算，深度卷积即分组卷积中一组的通道数为1。

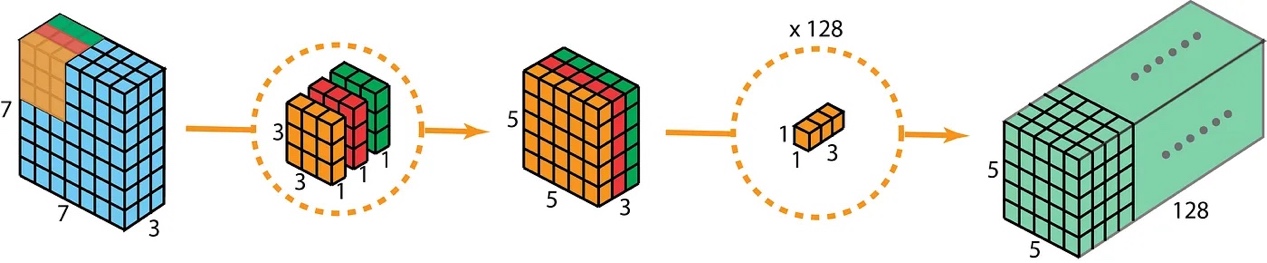
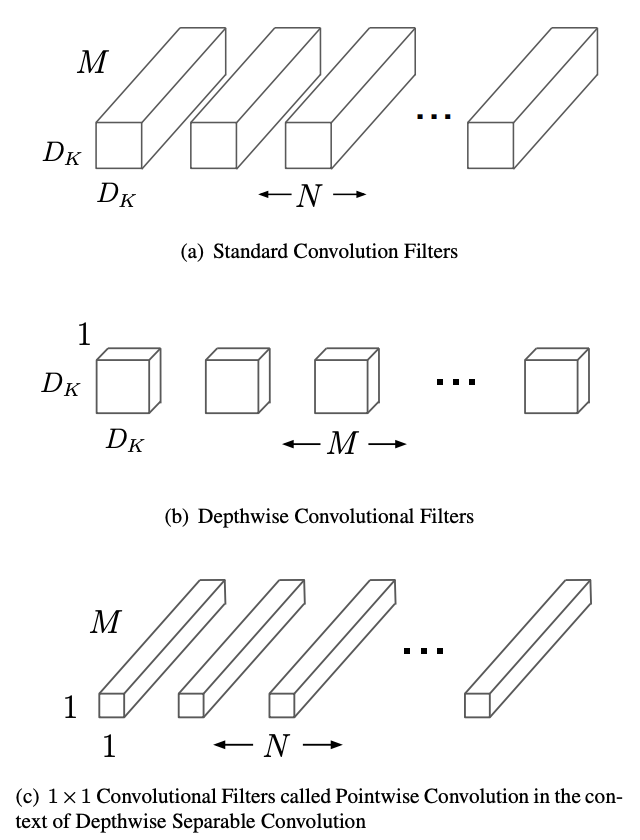


图 4 深度可分离卷积（Depthwise Separable Convolution）

深度可分离卷积参数量、计算量分析



* 标准卷积：一个卷积核M个通道、N个卷积核
* 深度可分离卷积

1. 深度卷积：一个卷积核一个通道、M个卷积核
2. 1x1卷积：一个卷积核一个通道、N个卷积核

标准卷积

|  |  |
| --- | --- |
| 一次卷积的乘法次数 |  |
| 输出特征图的元素个数 |  |
| 一次卷积的乘法计算量 |  |
| 参数量 |  |

深度可分离卷积

|  |  |  |
| --- | --- | --- |
|  | 深度卷积 | 1x1卷积 |
| 一次卷积的乘法次数 |  |  |
| 输出特征图的元素个数 |  |  |
| 一次卷积的乘法计算量 |  |  |
| 参数量 |  |  |

|  |  |
| --- | --- |
| 深度可分离卷积 | |
| 一次卷积的乘法计算量 |  |
| 参数量 |  |

此外，MobileNet还引入了两个用于控制网络大小的超参数

1. ：宽度超参数，控制深度卷积中卷积核的个数为
2. ：分辨率超参数，控制输入图像的分辨率，进而控制深度卷积输出特征图的尺寸为

由 ，得到深度可分离卷积乘法计算量为

当 时

其中，为输出通道数（1x1卷积的卷积核个数），为深度卷积的卷积核尺寸。

当 时， ，深度可分离卷积的计算量和参数量减小为标准卷积的约 。

MobileNet v1计算性能分析

|  |  |  |
| --- | --- | --- |
| 类型 | 乘-加计算量 | 参数量 |
| 1x11x1卷积 | 94.86% | 74.59% |
| 3x3深度卷积 | 3.06% | 1.06% |
| 3x3标准卷积 | 1.19% | 0.02% |
| 全连接层 | 0.18% | 24.33% |

表 1 MobileNet v1 乘-加计算量和参数量分布

图 5 1x1卷积

1x1卷积负责跨通道通信，引入额外的非线性，改变1x1卷积核的数量可以对输入特征图进行降维或升维。由表1可知，MobileNet v1计算量和参数量都主要集中在1x1卷积，为了方便MobileNet部署在移动端、终端设备上，足够轻量化，需要优化、加速1x1卷积。

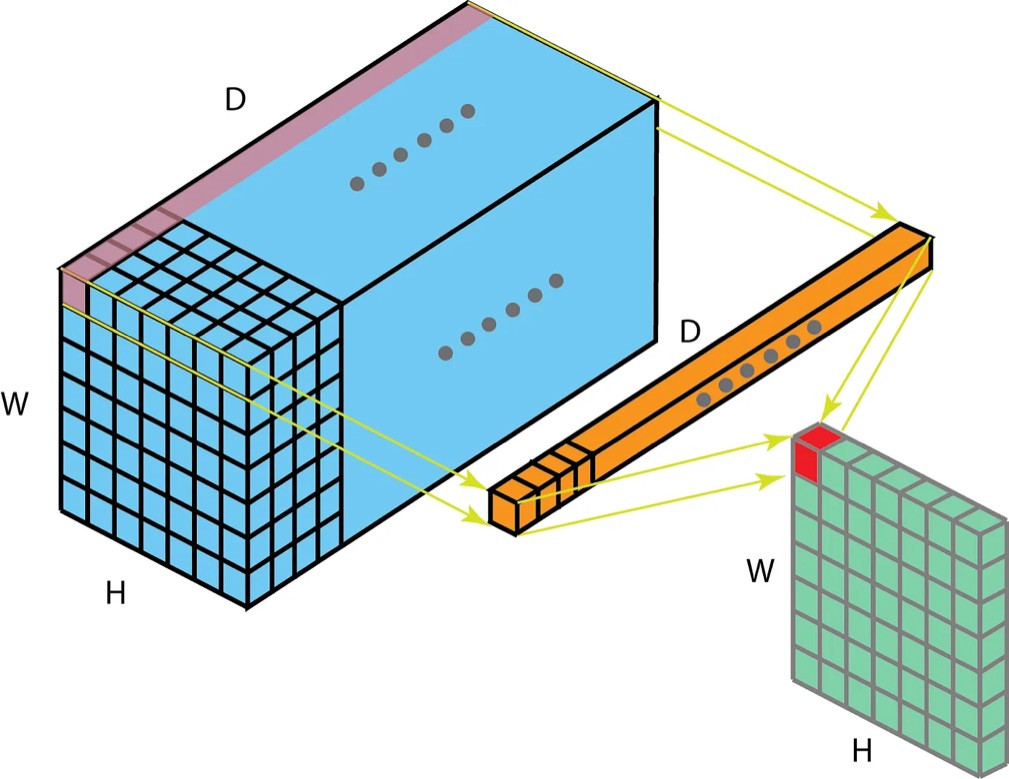


图 6 1x1卷积

为了方便，取填充padding = 0，步幅stride = 1。标准卷积有三层for循环，时间复杂度为 。

**def** corr3d(**in\_arr**, **kernel**):

c, ih, iw **=** in\_arr.shape

\_, kh, kw **=** kernel.shape

out\_arr **=** np.zeros(**shape=**(ih **-** kh **+** 1, iw **-** kw **+** 1))

for y in range(out\_arr.shape[0]):

for x in range(out\_arr.shape[1]):

for z in range(c):

out\_arr[y, x] **+=** (in\_arr[z, y:y **+** kh, x:x **+** kw] **\*** kernel[z]).sum()

return out\_arr

如何优化卷积运算？一种优化方法是基于im2col算法（image to columns），将感受野转化为行向量，卷积核转化为列向量，进而将卷积运算转化为矩阵乘法，矩阵乘法可以使用GEMM加速。

基于im2col算法的卷积有两层for循环，时间复杂度为 。

**def** im2col\_corr3d(**in\_arr**, **kernel**):

\_, kh, kw **=** kernel.shape

**def** im2col(**in\_arr**):

c, ih, iw **=** in\_arr.shape

oh **=** ih **-** kh **+** 1

ow **=** iw **-** kw **+** 1

out\_arr **=** np.zeros(**shape=**(c, kh, kw, oh, ow))

for y in range(kh):

for x in range(kw):

out\_arr[:, y, x, :, :] **=** in\_arr[:, y:y **+** oh, x:x **+** ow]

out\_arr **=** out\_arr.transpose(

(3, 4, 0, 1, 2)).reshape(oh **\*** ow, **-**1)

return out\_arr

in\_mat **=** im2col(in\_arr) # 感受野 -> 行向量

kernel\_mat **=** im2col(kernel).T # 卷积核 -> 列向量

out\_arr **=** np.matmul(in\_mat, kernel\_mat)

oh **=** in\_arr.shape[1] **-** kh **+** 1

return out\_arr.reshape(oh, **-**1)

1x1卷积有两层for循环，时间复杂度为 。

**def** pointwise(**in\_arr**, **kernel**):

\_, ih, iw **=** in\_arr.shape

out\_arr **=** np.zeros(**shape=**(ih, iw))

for y in range(ih):

for x in range(iw):

# np.matmul(in\_arr[:, y, x], kernel[:, 0, 0])

out\_arr[y, x] **+=** np.dot(in\_arr[:, y, x], kernel[:, 0, 0])

return out\_arr

对比标准卷积和基于im2col算法的卷积，1x1卷积的感受野、卷积核已经是行、列向量，相比基于im2col算法的卷积，1x1卷积不需要im2col操作，可以直接将卷积运算转化为矩阵乘法，使用高效GEMM加速。MobileNet V1中，计算量和参数量都主要集中在1x1卷积，因此得出结论：MobilNet V1速度很快。

MobileNet V1论文中，将使用深度可分离卷积的MobileNet与使用标准卷积的MobileNet、GoogleNet、VGG 16进行比较，在图像分类精度基本相同的条件下，MobileNet的乘-加计算量、参数量均明显减小。

并且，对于MobileNet的宽度超参数 、分辨率超参数 。实验结果表明：

* 不变时，随着 减小，图像分类精度大幅度下降，乘-加计算量、参数量均有明显减小。
* 不变时，随着 减小，图像分类精度只有小幅度下降，乘-加计算量有明显减小，而参数量保持不变。

计算量越大的网络，图像分类精度越高，但存在边际递减效应。使用者可以通过调整超参数 ，控制网络大小和图像分类精度。根据移动端设备硬件情况，权衡精度与速度。

实验证明，MobileNet在人脸特征分类、目标检测、人脸识别、Im2GPS（image to GPS）中都有优秀的表现。说明MobileNet可用于各种计算机视觉应用。

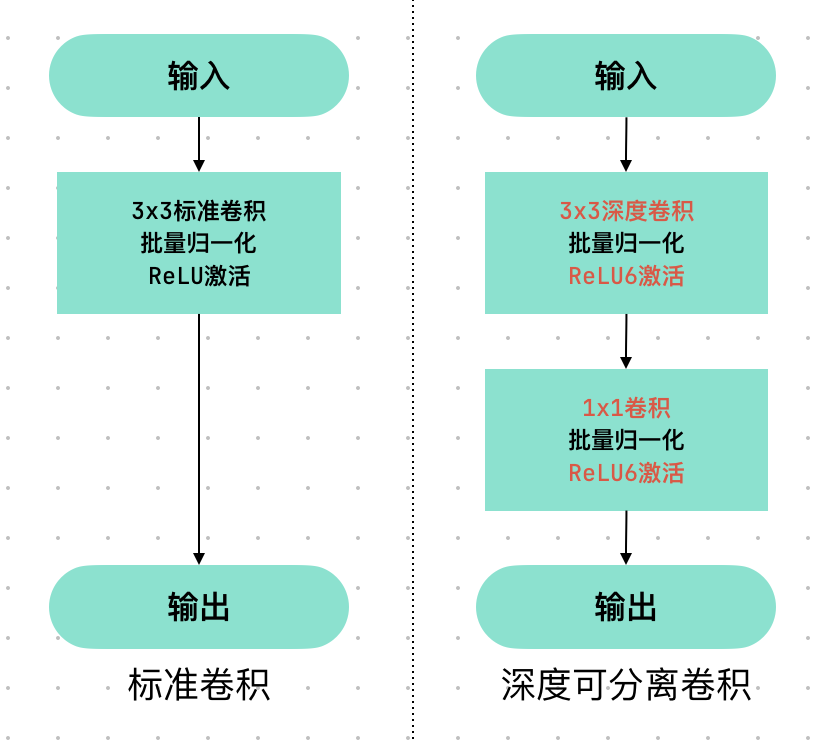


图 7 标准卷积模块、深度可分离卷积模块

标准卷积模块  
**def** conv2d(**filter\_in**, **filter\_out**, **kernel\_size**, **groups=**1, **stride=**1):

pad **=** (kernel\_size **-** 1) **//** 2 if kernel\_size else 0

return nn.Sequential(OrderedDict([

("conv", nn.Conv2d(filter\_in, filter\_out,

**kernel\_size=**kernel\_size,

**stride=**stride,

**padding=**pad,

**groups=**groups, **bias=**False)), # 3x3标准卷积

("bn", nn.BatchNorm2d(filter\_out)), # 批量归一化

("relu", nn.ReLU6(**inplace=**True))])) # ReLU6激活

深度可分离卷积模块

**def** conv\_dw(**filter\_in**, **filter\_out**, **stride=**1):

return nn.Sequential(

nn.Conv2d(filter\_in, filter\_in,

**kernel\_size=**3,

**stride=**stride,

**padding=**1,

**groups=**filter\_in,

**bias=**False), # 3x3深度卷积

nn.BatchNorm2d(filter\_in), # 批量归一化

nn.ReLU6(**inplace=**True), # ReLU6激活

nn.Conv2d(filter\_in, filter\_out, 1, 1, 0, **bias=**False), # 1x1卷积

nn.BatchNorm2d(filter\_out), # 批量归一化

nn.ReLU6(**inplace=**True)) # ReLU6激活

nn.ReLU6(**inplace=**True)) # ReLU6激活

轻量化神经网络技术主要分为四类：

1. 压缩已训练的神经网络：例如剪枝（权重剪枝、通道剪枝）、权值量化、知识蒸馏、注意力迁移等
2. 直接训练轻量化神经网络：例如EfficientNet、MobileNet、MnasNet、ShuffleNet、SqueezeNet、Xception等
3. 加速卷积运算：例如im2col + GEMM、Winograd、低秩分解
4. 硬件部署：例如FPGA、Jetson Nano、OpenVINO、Tensorflow-slim、Tensorflow-lite、TensorRT等

MobileNet是用于移动端计算机视觉应用的轻量化、高性能卷积神经网络。

MobileNet V1的局限

1. 没有残差连接
2. 很多深度可分离卷积核训练得到的权重为0
   1. 卷积核权重数量和通道数量太少。
   2. ReLU激活函数将输入张量小于0的部分全部算作0，当学习率比较大时可能出现“神经元死亡”现象：前向传播的张量为0，反向传播的梯度也为0，遍历整个训练集后，权重并未得到更新。
   3. 低精度的浮点数表示。

MobileNet V2的改进

已知通过改变1x1卷积核的数量可以对输入特征图进行降维或升维，ResNet中的标准残差模块通过1x1卷积先降维后升维，残差连接高维Tensor。模块中均使用ReLU激活，表现出瓶颈（BottleNeck）结构

MobileNet V2不但沿用了V1中提出的深度可分离卷积，还参考了ResNet。但与ResNet不同的是，MobileNet V2的倒残差模块通过1x1卷积先升维后降维，残差连接低维Tensor，模块中使用ReLU6非线性激活，降维时使用Linear线性激活，表现出线性瓶颈（Linear BottleNeck）结构

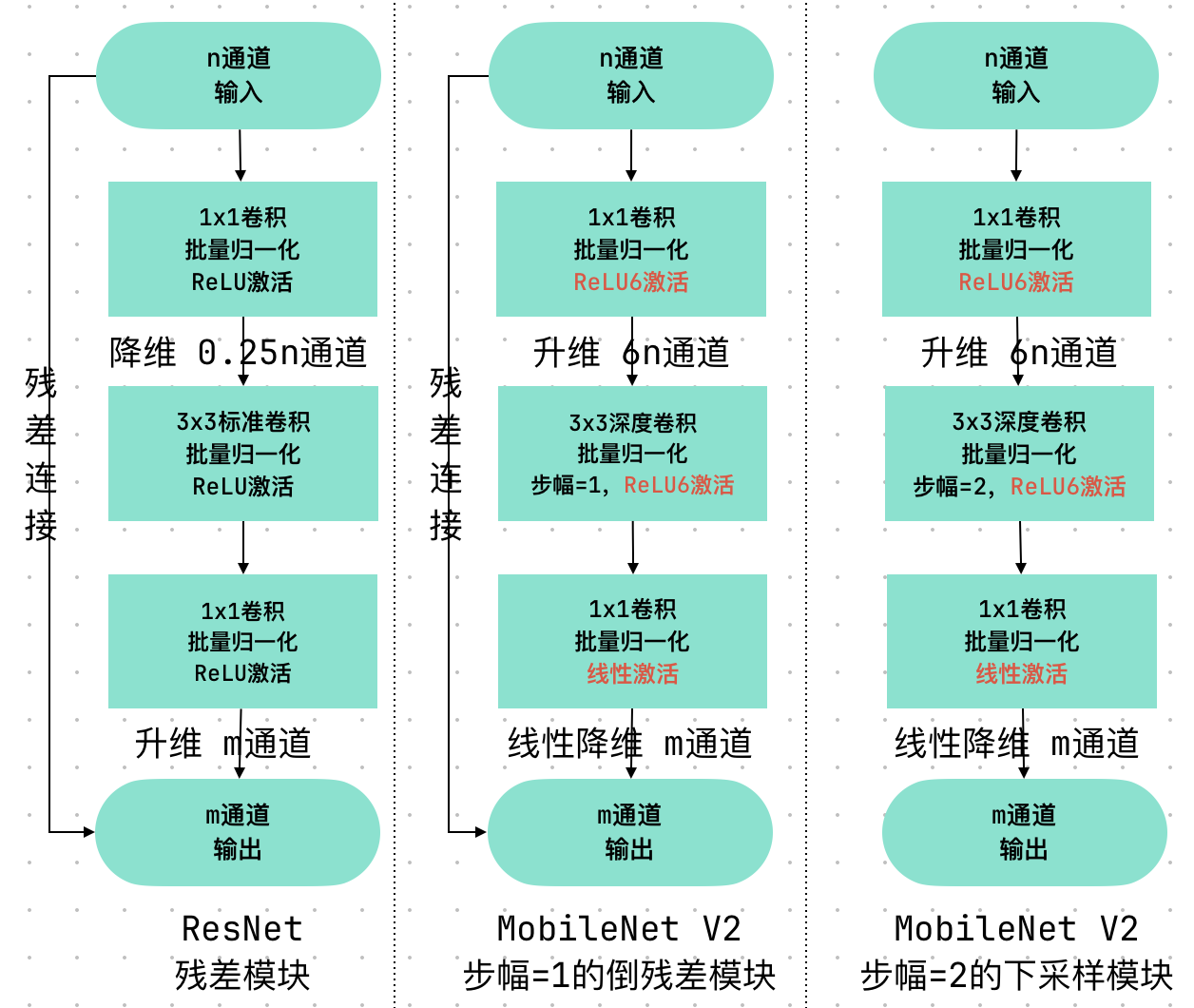


图 8 ResNet残差结构与MobileNet V2倒残差结构对比

通过pytorch和matplotlib绘制ReLU、ReLU6激活函数

**def** relu(**in\_arr**):

arr0 **=** torch.zeros\_like(in\_arr)

out\_arr **=** torch.max(in\_arr, arr0)

return out\_arr

**def** relu6(**in\_arr**):

arr0 **=** torch.zeros\_like(in\_arr)

arr6 **=** torch.ones\_like(in\_arr) **\*** 6

out\_arr **=** torch.min(torch.max(in\_arr, arr0), arr6)

return out\_arr

x **=** torch.arange(**-**4, 11, 0.1)

plt.plot(x, relu(x), "red", **linewidth=**2, **label=**"ReLU")

plt.plot(x, relu6(x), "blue", **linewidth=**2, **linestyle=**"dotted", **label=**"ReLU6")

plt.grid(True)

plt.xlabel("input")

plt.ylabel("output")

plt.legend(**loc=**"upper right")

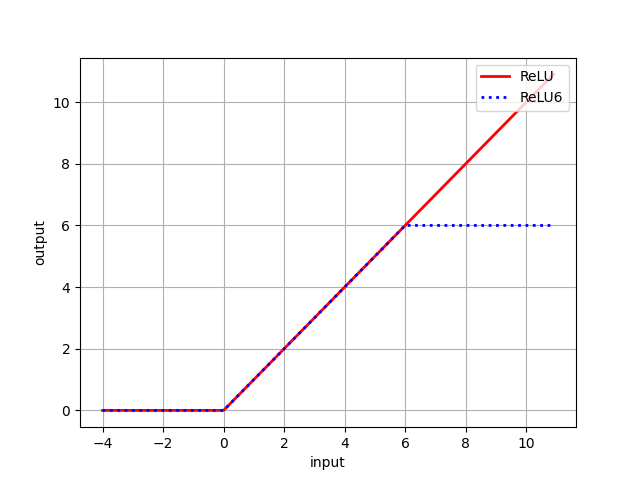


图 9 ReLU激活函数与ReLU6激活函数对比

低精度浮点数表示下（例如8 bit或16 bit），表示范围比较小。ReLU激活函数y轴值域为[0, +∞)。较少的比特数无法表示较大的数。相比ReLU6激活函数y轴值域为[0, 6)，在低精度浮点数表示下可以得到较好的表示性能。

