

CVPR 第四次作业-Caltech256 数据集图像分类

强基数学

吴天阳 2204210460, 陈开来 2205110920, 王瑞恒 2202113454,
马煜璇 2204220461, 申宇环 2201422097

1 实验目的

基于 Caltech256 数据集的深度网络分类性能比较.

1. 以 Caltech256 为试验数据集, 按照原始数据集的要求进行训练、测试集的划分.
2. 比较不少于三种深度神经网络结构, 全部平均精度不低于 85%.
3. 对算法从处理速度、正确率方面给出性能评估.

我们选择了 VGG-19, Inception-ResNet, MoblieNet, EfficientNet 对 Caltech256 数据集进行测试, 并对其进行评估. 首先介绍卷积神经网络基本原理和 6 大经典模型的特点.

2 实验原理

2.1 卷积神经网络基础

这里的卷积指的是离散型的卷积形式.

2.1.1 一维卷积

设 $\{w_i\}, \{x_i\}$ 为两个数列, $k \in \mathbb{R}$, 定义 $\{w_i\}$ 与 $\{x_i\}$ 的有限卷积为以下数列

$$y_t = \sum_{k=1}^K w_k x_{t-k+1}, \quad (t \geq K) \quad (2.1)$$

其中 $\{w_i\}$ 称为滤波器 (Filter) 或卷积核 (Convolution Kernel), $\{x_i\}$ 为信号序列, K 为滤波器长度.

如果我们将数列记为对应的函数值: $w(i) = w_i (1 \leq i \leq K)$, $x(i) = x_i (1 \leq i)$, $y(t) = y_t (K \leq t)$. 则上述定义可视为: 数列 $\{w_i\}, \{x_i\}$ 在 \mathbb{R} 上的零延拓, 即
 $w(i) = \begin{cases} w_i, & 1 \leq i \leq K, \\ 0, & \text{otherwise.} \end{cases}$ 用更形象的方式将其列出如下

$$\begin{aligned} i &= \dots, -1, 0, 1, 2, \dots, K, K+1, K+2, \dots \\ w(i) &= \dots, 0, w_1, w_2, w_3, \dots, w_K, 0, 0, \dots \\ x(i) &= \dots, 0, x_1, x_2, x_3, \dots, x_K, x_{K+1}, x_{K+2}, \dots \end{aligned}$$

定义两个离散数列 $\{w_i\}, \{x_i\}$ 的卷积如下:

$$w * x := \sum_{i=-\infty}^{\infty} w_i x_{t-i+1} \stackrel{i=t-j+1}{=} \sum_{j=-\infty}^{\infty} x_j w_{t-j+1} = x * w \quad (2.2)$$

$$= \sum_{i=1}^K w_i x_{t-i+1} \quad (2.3)$$

通过 (2.2) 式可知卷积具有可交换性, (2.3) 式表明 (2.1) 式中定义的有限卷积其实就是在数列零延拓下的卷积, 再截取 $t \geq K$ 这一段的结果.

卷积操作在信号处理方面有不错的效果, 可以通过不同的卷积核, 对不同的信号进行提取. 下面是几个简单例子.

1. 简单移动平移: $w = [1/k, 1/k, \dots, 1/k]$ (用于时间序列中消除数据的随机波动)
2. 二阶微分近似: $w = [1, -2, 1]$, 由数值分析的知识可知, 连续二阶可微函数 $x(t)$, 有如下近似式

$$x''(t) \approx \frac{x(t-h) - 2x(t) + x(t+h)}{h^2} \stackrel{h=1}{=} x(t-1) - 2x(t) + x(t+1)$$

2.1.2 二维卷积

常用于图像处理, 设图像 $x \in \mathbb{R}^{M \times N}$, 卷积核 $w \in \mathbb{R}^{U \times V}$, 一般有 $U \ll M, V \ll N$, 类比一维卷积定义, 二维卷积定义如下:

$$y_{st} = \sum_{i=1}^U \sum_{j=1}^V w_{ij} x_{s-i+1, t-j+1} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} w'_{ij} x'_{s-i+1, t-j+1} =: w * x \quad (2.4)$$

其中 w'_{ij}, x'_{ij} 分别为 w_{ij}, x_{ij} 的零延拓, 记 $y = w * x \in \mathbb{R}$. 图1是几种不同卷积核作用在一张图片上的效果.

2.1.3 互相关

在机器学习和图像处理中, 卷积的作用主要是通过在一个图像上滑动一个卷积核, 通过卷积操作得到一个新的图像. 在计算卷积过程中, 需要对卷积核进行反转操作, 即对卷积核旋转 π 大小. 这个操作就显得多余了, 所以在计算机中经常将卷积视为互相关 (Cross-Correlation) 操作, 即直接对卷积核和原图进行点积操作 (对应位相乘).

设图像 $x \in \mathbb{R}^{M \times N}$, 卷积核 $w \in \mathbb{R}^{U \times V}$, 则它们的互相关为:

$$y_{st} = \sum_{i=1}^U \sum_{j=1}^V w_{ij} x_{s+i-1, t+j-1} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} w'_{ij} x'_{s+i-1, t+j-1} =: w \otimes x \quad (2.5)$$

和 (2.4) 式对照可知, 互相关和卷积的区别仅仅在于卷积核是否需要翻转, 即 $w \otimes x = \text{rot}(w) * x$, $\text{rot}(w)$ 表示将矩阵 w 旋转 π 以后的结果. 因此互相关也称为不翻转卷积.

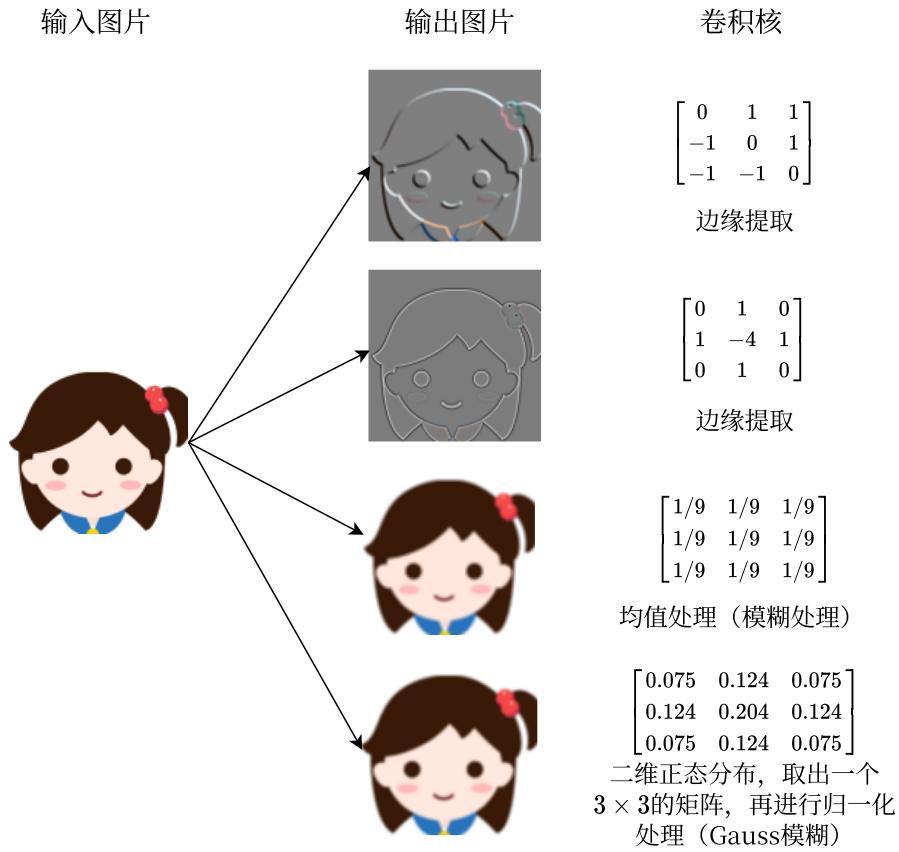


图 1: 不同卷积核处理效果

2.1.4 卷积的变种

在卷积的基础上，还可以引入步长和零填充增加卷积的多样性，以便更灵活地提取图像特征。

- **步长 (Stride)** 指卷积核在滑动时的时间间隔。如图2(a).
- **零填充 (Zero Padding)** 指对输入矩阵的边缘进行零填充。如图2(b).

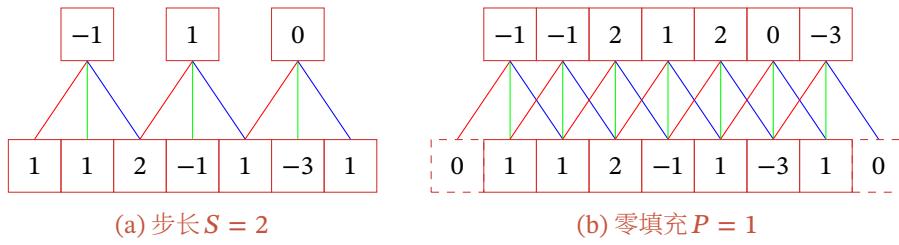


图 2: 步长和零填充

设卷积层的输入向量维数为 M ，卷积大小为 K ，步长为 S ，在输入两端各填补 P 个 0，则输出向量维度为 $(M - K + 2P)/S + 1$ ，

常用卷积有以下三种：

1. 窄卷积 (Narrow Convolution): $S = 1, P = 0$, 输出维度为 $M - K + 1$. (普通卷积)
2. 宽卷积 (Wide Convolution): $S = 1, P = K - 1$, 输出维度为 $M + K - 1$.
3. 等宽卷积 (Equal-Width Convolution): $S = 1, P = (K - 1)/2$, 输出维度为 K . 如图2(b) 就是一种等宽卷积.

2.2 卷积神经网络结构

2.2.1 卷积层

卷积层的作用是提取局部区域的特征，将输入卷积层的矩阵称为输入特征，将通过卷积层后的输出称为输出特征，也称特征映射（Feature Map）。

一般的图片每个像素由RGB三原色（颜色通道数为3）构成，假设图片的宽度和高度分别为 N, M ，颜色通道数为 D ，则一张图片 $x \in \mathbb{R}^{N \times M \times D}$ ，由于图片的像素值一般为无符号8位整型，即 $x_{ijk} \in [0, 255]$ ，所以也有 $x \in [0, 255]^{N \times M \times D}$ ，当我们对图片进行归一化处理后，即 $x \leftarrow x/256$ ，就有 $x \in [0, 1]^{N \times M \times D}$ 。

卷积层中，假设每个卷积核大小为 $U \times V$ ，且每个颜色通道上都对应有 P 个卷积核，则卷积核 $w \in \mathbb{R}^{U \times V \times P \times D}$ ，令第 d 个颜色通道上的第 p 个卷积核为 $w_{d,p}$ 。由于每个卷积核 $w_{d,p}$ 作用在图片 x 上都会得到一个输出 y_p ，所以一共有 P 个输出特征，所以特征映射 $y \in \mathbb{R}^{N' \times M' \times P}$ ， $N' \times M'$ 为卷积核 $U \times V$ 作用在 $N \times M$ 矩阵后的维度。可以参考下图更好地理解。

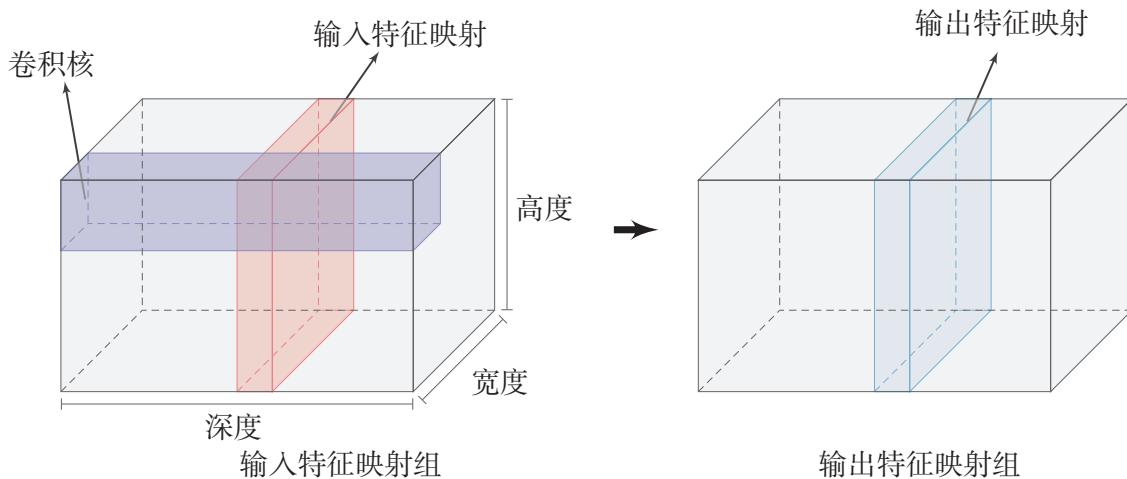


图 3: 卷积层的三维结构

2.2.2 汇聚层

汇聚层（Pooling Layer）也称池化层，子采样层（Subsampling Layer）。其作用是对卷积层输出的特征映射进一步进行特征选取，降低特征数量，减少参数数量。

设汇聚层的输入特征 $x \in \mathbb{R}^{N \times M \times D}$ ，对于其中每一个颜色通道中的图像 x^d ，划分为很多的区域 $\{R_{ij}^d\}$ ，满足 $\bigcup_{ij} R_{ij}^d \subset \{x_{ij}\}$ ，这些区域可以是不交的，也可以有交集。汇聚

(Pooling) 是指对每个区域进行下采样(Down Sampling)操作得到的值，作为该区域的概括。

常用的汇聚操作有以下两种：

1. **最大汇聚** (Maximum Pooling): 对于一个区域 R_{ij}^d ，选择这个区域内所有神经元的最大活性值作为这个区域的表示，即

$$y_{ij}^d = \max_{x \in R_{ij}^d} x \quad (2.6)$$

2. 平均汇聚 (Mean Pooling): 取该区域内的所有活性值的平均值作为该区域的表示, 即

$$y_{ij}^d = \frac{1}{|R_{ij}^d|} \sum_{x \in R_{ij}^d} x \quad (2.7)$$

其中 $|R_{ij}^d|$ 表示集合 R_{ij}^d 的基数, 即该集合中所包含元素的个数.

2.2.3 卷积网络的一般结构

一个经典卷积网络由卷积层、汇聚层、全连接层堆叠而成, 常用卷积神经网络结构如图4所示. 一个卷积块为一组连续 M 个卷积层和 b 个汇聚层构成 (M 取值通常为 $2 \sim 5$, 且卷积核大小逐层增大, 个数逐层增多, b 通常取为 0 或 1), 卷积神经网络堆叠 N 个连续的卷积块, 然后连接 K 个全连接层 (N 通常取为 $1 \sim 100$ 或更大, K 一般取为 $0 \sim 2$).

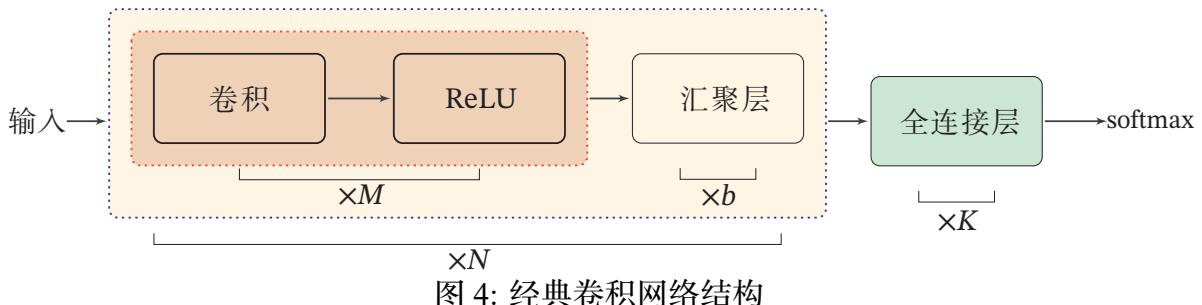


图 4: 经典卷积网络结构

卷积网络的卷积核大小一般取为 2×2 或 3×3 , 以及更多的数量如 32 个或更多. 由于卷积可以设置步长减少输出特征的大小, 所以汇聚层的作用并不显著了, 可以通过增加步长来替代.

2.3 经典图像分类网络原理

2.3.1 AlexNet

AlexNet 夺得了 2012 年 [ImageNet 比赛](#)第一名, 是最经典的卷积神经网络模型, 使用 227×227 的图像输入, 论文中画的是 224×224 , 后来指出应该是通过 padding 增加到了 227×227 的输入. 5 个卷积层, 3 个最大池化层, 3 个全连接层, 前两个激活函数使用 ReLU, 最后一层使用 softmax 转化为概率值进行分类. 具体网络结构见图5.

参考论文: [ImageNet Classification with Deep Convolutional Neural Networks - 2012](#).

AlexNet 特点具有的特点:

1. **ReLU 激活函数:** 在 AlexNet 之前, 神经网络使用的一般为 tanh 作为激活函数, 而 AlexNet 中使用的全部为 ReLU 激活函数, 论文中显示, 达到相同的 75% 正确率下, ReLU 比 tanh 快接近 6 倍的时间.

2. **GPU 多线程处理:** 当时训练所使用的显卡为 NVIDIA GTX 580 GPU, 仅有 3GB 显存, 为了避免显存超出, 在神经网络框架上可以看出, 图像被划分为两条路径分别进行卷积操作, 但并没有实现速度上的提升. 现在的显卡控制好 batch 大小, 已经不会再出现显存溢出问题了.

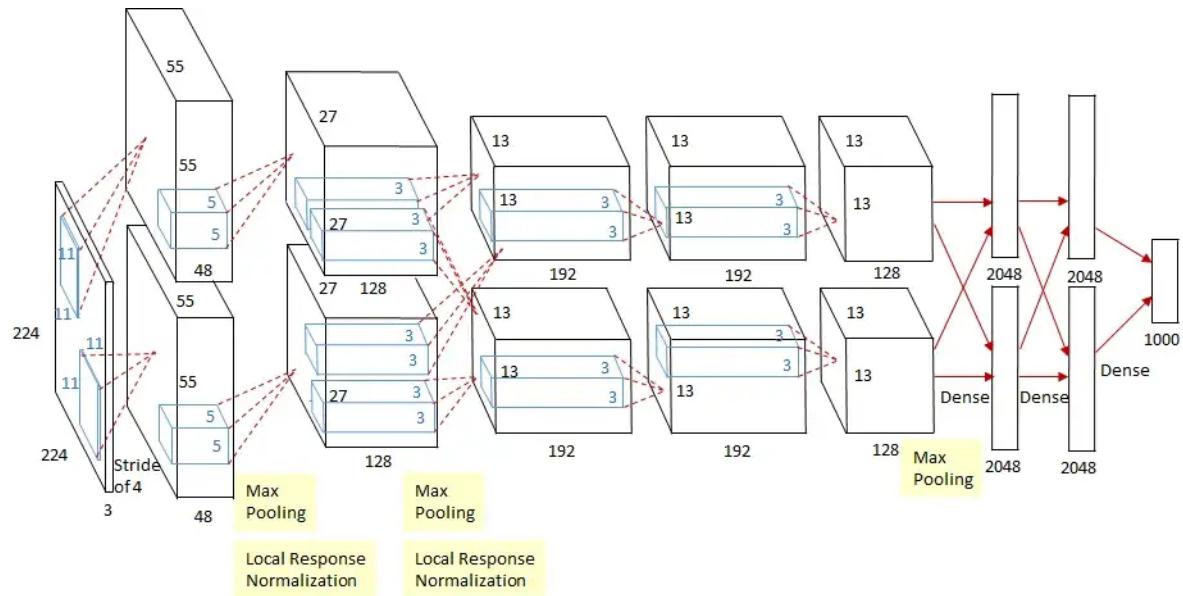


图 5: AlexNet 基本结构

3. 局部响应归一化: 对局部值的特征进行了归一化操作，归一化操作可以提升神经网络收敛速度，而现在神经网络使用的更多的是以 batch 作为一个单位进行归一化操作。

4. 数据增强: 主要是使用了如下两种数据增强方法：图像平移与水平翻转，改变光照强度。

5. Dropout: Dropout 操作是指每次训练过程中会随机丢弃一定比例的神经元，并且不进行梯度更新；在测试集上运行时，则不会调用 Dropout 操作。该操作是一种正则化操作，使得每个神经元都有更大的概率去更新权重，可以一定程度上避免过拟合发生。

2.3.2 VGG

VGG 神经网络是 2014 年 ImageNet 比赛中的第二名。参考论文：[Very Deep Convolutional Networks for Large-Scale Image Recognition - 2014](#).

VGG 是在 AlexNet 上的进一步改进。在 VGG 神经网络中，卷积层持续使用大小为 3×3 ，步长为 1，padding 为 1 的卷积核；所有池化层均为最大池化层，大小为 2×2 ，步长为 2，在通过池化层之后通道数加倍。

VGG 网络共包含 5 个卷积阶段：

阶段 1: conv-conv-pool

阶段 2: conv-conv-pool

阶段 3: conv-conv-pool

阶段 4: conv-conv-conv-[conv]-pool

阶段 5: conv-conv-conv-[conv]-pool

注：在 VGG-19 中阶段 4,5 中包含 4 个卷积层。

VGG 的设计思路主要是：相同的卷积核大小，在使用更小更多的卷积核与一个大卷积核的感受野形同，但具有更少的参数和计算量，所以 VGG 中全部使用了 3×3 的卷积核代替 5×5 的卷积核。

缺点：VGG-16 的内存开销整体上比 AlexNet 大了 25 倍，参数增多了 2.3 倍，整体训练难度提高。

2.3.3 GoogleNet

GoogleNet 在 2014 年 ImageNet 比赛的第一名，在随后的两年后一直改进，产生了 Inception V2、Inception V3、Inception V4(Inception-Resnet) 众多版本，最后的一个版本我们还会对其进行介绍，并用其做本次图像识别任务。参考论文：[Going Deeper with Convolutions - 2014](#)

提升网络识别图像的性能一般是增加网络的深度和宽度，但是直接增加会产生非常多问题：

1. 参数太多，若训练集有限，容易过拟合；
2. 网络越大，训练难度增加，计算复杂度增大，难以应用；
3. 网络越深，容易发生梯度消失问题，难以优化模型。

希望在增加网络深度和宽度的同时减少参数，通过将全连接转化为稀疏连接从而减少参数个数，这里稀疏连接指的就是池化层，在 GoogleNet 中使用了平均池化层代替输出层前的多个全连接层，大大减少了参数数目；并提出 Inception 方法就是把多个卷积或池化操作组装为一个网络模块，设计神经网络时，以模块形式进行组装（GoogleNet 中一共组装了 9 个 Inception 模块）。

下面主要分析 Inception 原理，图6展示了 Inception 模块的基本结构：

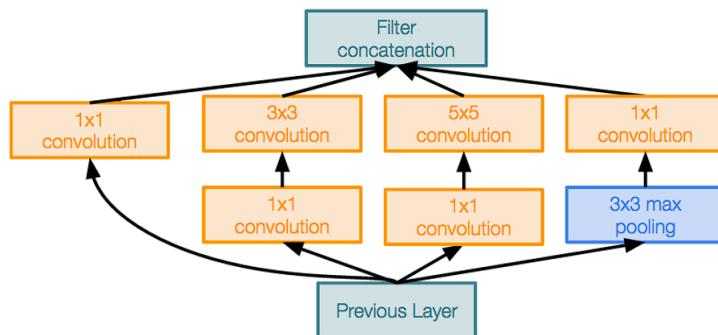


图 6: Inception 模块基本结构

在一般的神经网络中，一层只能表示卷积层或者池化层，而且卷积层的卷积核具有固定的大小。但是，实际情况下，对于不同尺度的图片，使用不同大小的卷积核，识别的效果可能各不相同，因为感受野都不同，所以我们希望让神经网络自动去选择效果好的卷积核大小，于是一个 Inception 模块中并列了多种不同的卷积核操作，神经网络可以通过调节参数自适应选择卷积核大小。

但是直接并列卷积核会导致参数数目大大增加，特征图厚度（即通道数个数）太大，为解决该问题，加入 1×1 的卷积核减少特征图厚度（也称瓶颈层，Bottleneck）。这样就可以大大减少参数数量，并且增加的 1×1 卷积在降维之后可以更加有效、直观地进行数据的训练和特征提取。

为了避免梯度消失，GoogleNet 使用了两个辅助分类器接受中间结点的 loss 值，但是后面发明的 batch 归一化操作可以代替这个部分。

GoogleNet 总层数为 22 层，而 VGG 总层数为 19 层，但是 GoogleNet 的模型使用内存仅为 VGG 的 1/10.

2.3.4 ResNet

残差神经网络（ResNet）是 2015 年 ImageNet 比赛的第一名，其层数直接增加到了 152 层。参考论文[Deep Residual Learning for Image Recognition - 2015](#).

ResNet 主要成就在于发现了“退化现象（Degradation）”，针对退化现象发明了“快捷连接（Shortcut connection）”极大程度的缓解了深度过大的神经网络训练问题。

如果一个神经网络在浅层就能完成图像分类问题，那么一定可以利用更深层的神经网络完成相同任务，主要将浅层神经网络后面加入多层恒等变换层即可。

但实验发现事实并不如此，使用深层神经网络在准确率达到某一个饱和值后迅速下降，该过程被 ResNet 团队称为“退化现象”，他们将发生这种退化现象的原因归结为深层神经网络难以实现“恒等变换”操作。这里是因为，神经网络使用大量非线性转换将数据映射到高维空间一遍完成更好的分类，导致数据变得更加离散，此时数据已经难以回到原始状态（恒等变换），所以发生“退化现象”。

解决“退化现象”的方法就是将线性变换与非线性变换进行平衡，于是 ResNet 中引入 ResNet 模块增加了“快捷连接”分支，期望让网络自动地在线性变换和非线性变换之间寻找平衡。ResNet 模块如图 7 所示。

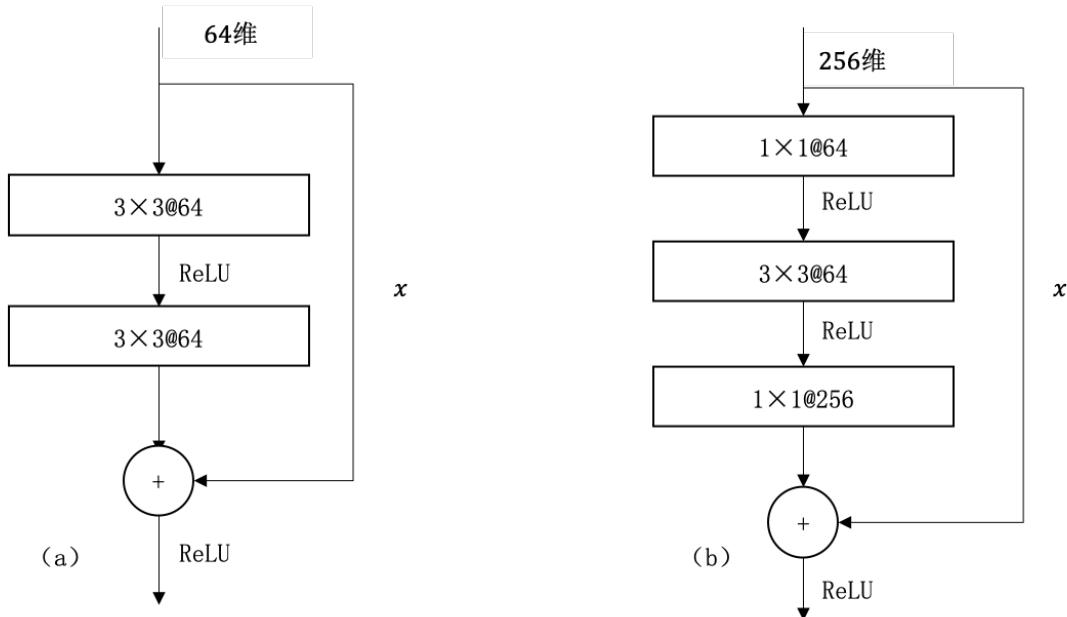


图 7: ResNet 模块

ResNet 模块分为两种：ResNet 构建块 (a)，降采样的 ResNet 构建块 (b)。降采样的构建块即在主干分支上增加了一个 1×1 的卷积核（瓶颈层），和 GoogleNet 在 Inception 模块中使用的相同，减少参数个数同时增加卷积层，增强提取特征的能力。

2.3.5 MobileNet

MobileNet 是 Google 于 2017 年提出的神经网络，经典轻量级神经网络，参考论文：[MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications - 2017.](#)

轻量级神经网络是专注于在移动设备上运行的神经网络，主要思路有两个方向：1. 对训练好的复杂模型进行压缩得到较小的模型；2. 直接设计较小而模型进行训练。其目的是保持模型性能（Accuracy）的前提下降低模型大小（Parameters Size），同时提升模型速度（Speed, Low Latency）。MobileNet 牺牲了少量的精度和但是极大地提升了的速度和降低了模型大小。

MobileNet 的核心思想就是将标准卷积转化为深度可分离卷积，具体原理参考了[知乎-轻量级神经网络 MobileNet](#)中的解释：将原有卷积操作分为两个部分，深度卷积和逐点卷积，如图8所示

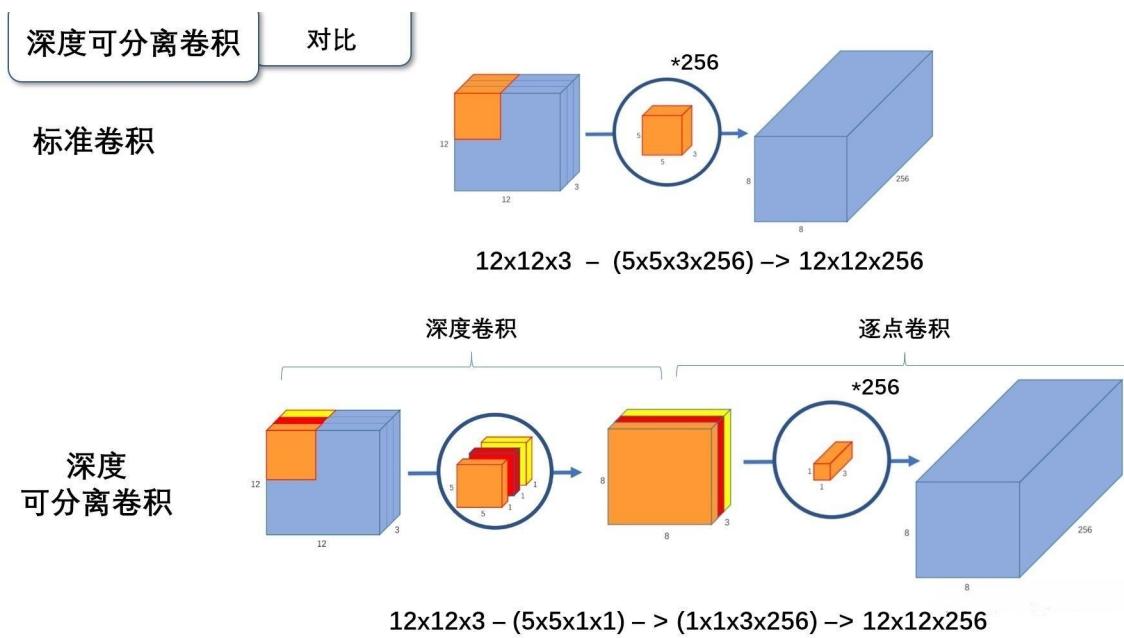


图 8: 标准卷积和深度可分离卷积对比

深度可分离卷积与标准卷积不同的是：我们将卷积拆分为多层单通道形式，不改变图像深度的情况下，先对每一通道进行卷积操作，这样特征图像保持和输入图像相同的通道数目；然后使用逐点卷积，也就是 1×1 卷积（就是之前在 GoogleNet、Resnet 中都是用过的瓶颈层，只不过这次是可以用于升维），我们使用多通道的 1×1 的卷积核作用在刚才输出的特征图像上，即可得到和标准卷积相同大小的特征图像结果。

与之不同的是，深度可分离卷积具有更小的量和计算量，设原始图像通道数为 M ，卷积核大小为 $D_k \times D_k \times M \times N$ ，输出的特征图像大小为 $D_W \times D_H \times N$ 。则标准卷积的参数数目和计算次数分别为： $D_K \times D_K \times M \times N$, $D_K \times D_K \times M \times N \times D_W \times D_H$ ，而可分离卷积的参数数目和计算次数分别为： $(D_K \times D_K + N) \times M$, $(D_K \times D_K + N) \times M \times D_W \times D_H$ ，于是我们发现深度可分离卷积在参数数目和计算速度上都减少了 $\frac{1}{N} + \frac{1}{D_K \times D_K}$ (N 一般较大，所以主要由第二项控制大小)。如果我们将卷积取为 3×3 ，则整体速度提升大约 9 倍！

经过实验发现，MobileNet 在 ImageNet 数据库上的准确率仅比 VGG16 低 0.9%，而参数量减少了 30 倍。

2.3.6 EfficientNet

EfficientNet 是 Google 于 2019 年提出的神经网络，参考论文：[EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks - 2019](#)。

在之前 ImageNet 比赛中我们可以看出，通过对网络的不断扩展可以提高网络的性能，具体扩展方向有如下三个（如图9所示）：

1. 增加网络层数
2. 增加网络宽度
3. 高分辨率

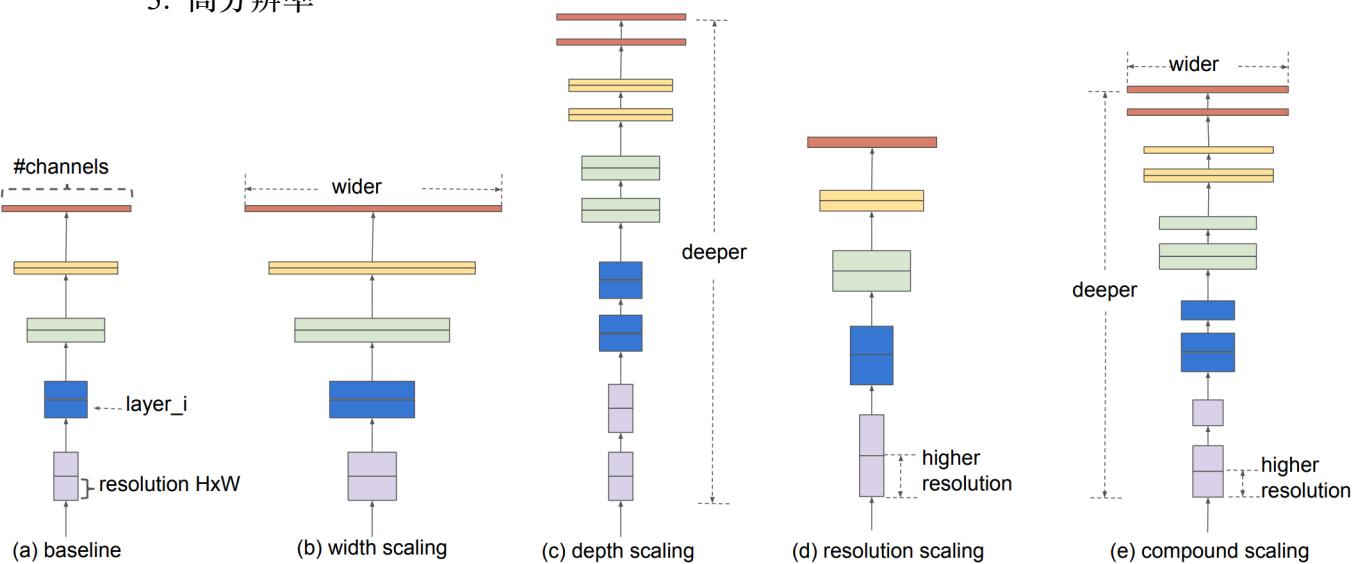


Figure 2. Model Scaling. (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

图 9: 不同的卷积神经网络扩展方向

具体思路是通过神经网络求解平衡三个参数之间的关系，使精度和效率最大化，具体实现较为复杂不再详细分析。实验表明 EfficientNet 在迁移学习上也具有很好的表现，在实际效果上优于之前的经典网络，如附录 A 中图11所示。

3 实验步骤与结果分析

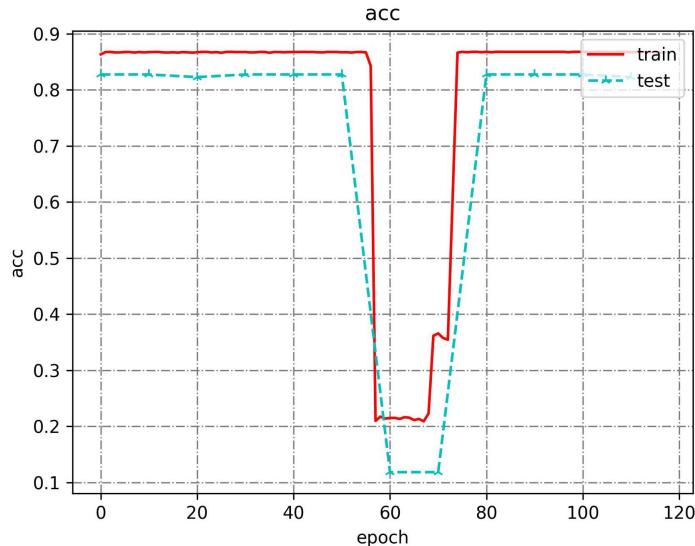
Caltech256 数据集总共有 30607 张图片，257 个类别，我们将其按照 4 : 1 划分为训练集和验证集。我们代码均在服务器上完成训练，服务器显卡配置为 NVIDIA GeForce RTX3090，将使用该显卡做模型速度测试，为统一测速标准：测速时均测试整个数据集，batch 大小均设置为 32。

模型实现方面：第一个 VGG-19 模型为 Pytorch 实现，没有使用预训练模型；后面三个模型 Inception-ResNet, MoblieNet, EfficientNet 均为 TensorFlow 实现，使用迁移学习完成（模型数据均来自 [TensorFlow Hub](#)，初始训练集均为 ImageNet 数据集）。

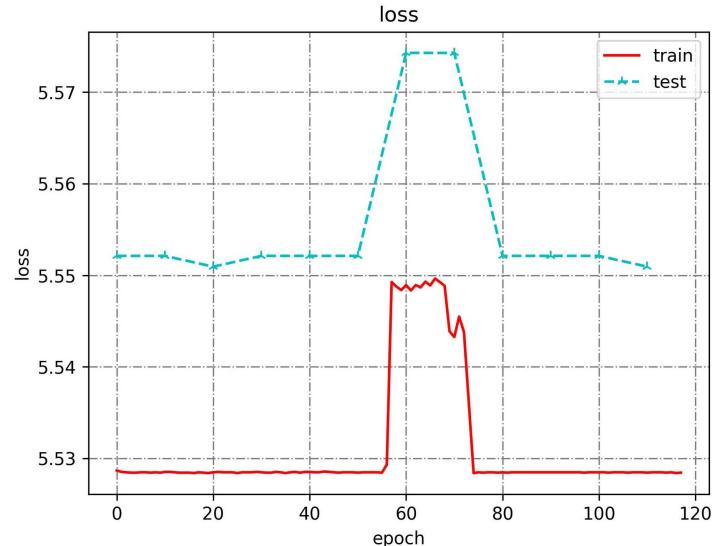
3.1 VGG-19

我们使用 Pytorch 对 VGG-19 模型进行搭建，并进行了训练，模型搭建部分代码请见[GitHub-model.py](#)，完整代码请见[GitHub-VGG_torch](#).

模型约为大小 485MB，训练集准确率为 86.7%，验证集准确率为 82.27%，平均准确率达到了 85.814%，整个训练集上运行速度约为 250 秒。训练过程图如下图所示（中间准确率突然下降原因，小组成员解释可能是发生了梯度消失的问题）。



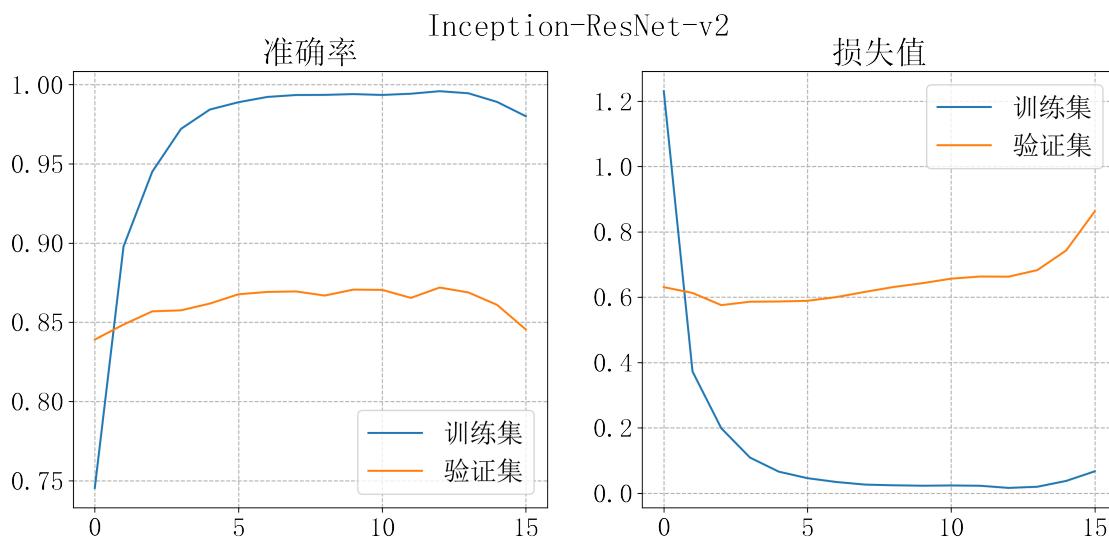
(a) 准确率



(b) 损失值

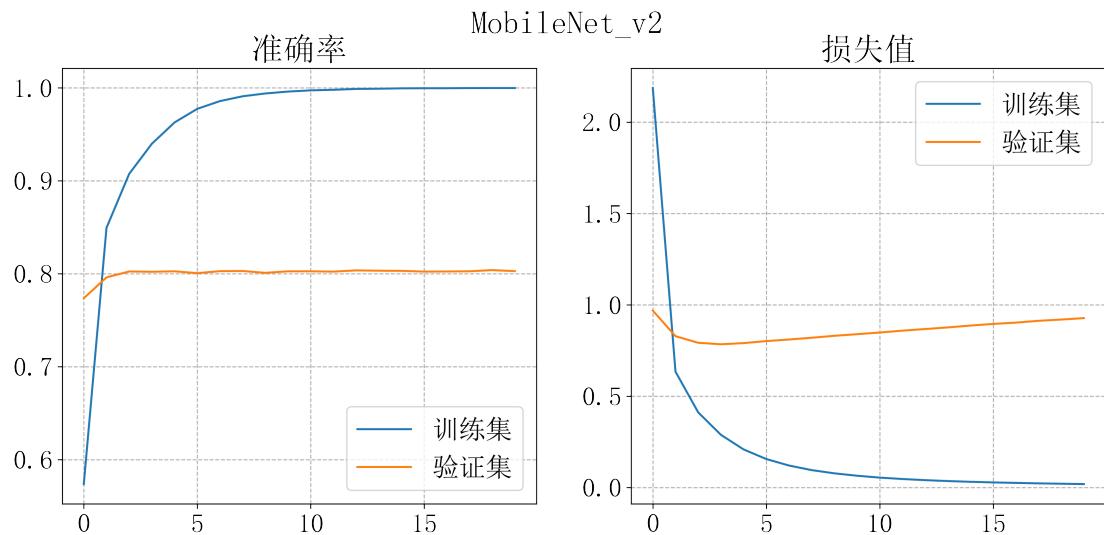
3.2 Inception-ResNet

模型来自[TensorFlow Hub-inception_resnet_v2](#)，迁移学习后模型大小约为 227MB，在原始模型后加入 1000 个神经元的全连接层和 257 个神经元的输出层，激活函数分别为 ReLU 和 softmax，batch 大小为 256。训练集准确率为 99.58%，验证集准确率为 87.19%，平均准确率为 96.97%，整个训练集上运行速度约为 100 秒。训练过程图如下图所示（训练速度非常快，在第 12 个 epoch 时达到最大准确率，后面就已经开始发生过拟合）



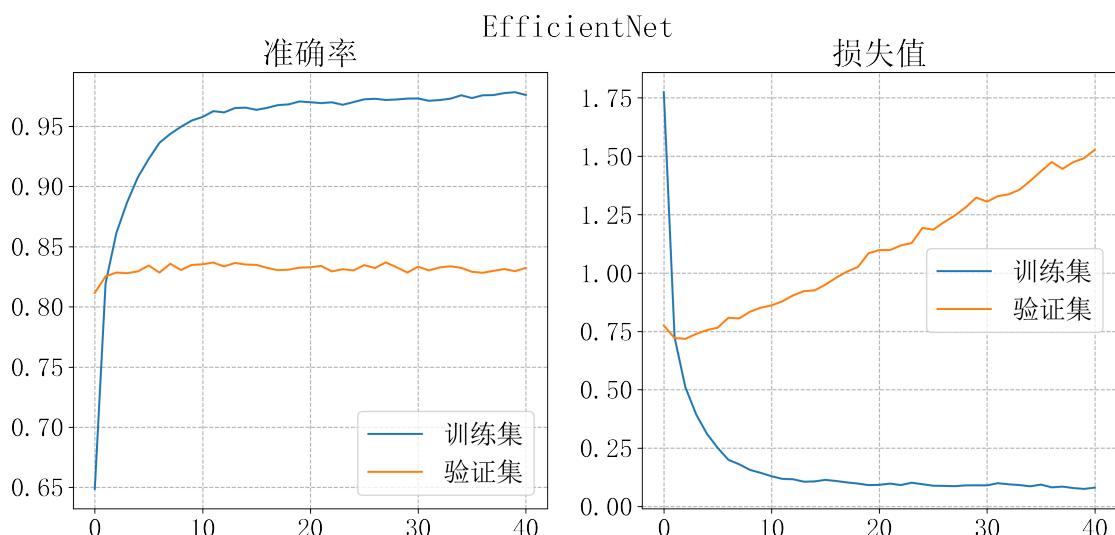
3.3 MoblieNet

模型来自[TensorFlow Hub-mobilenet_v2](#), 迁移学习后模型大小约为 16.5MB, 在原始模型后直接加入 257 个神经元的输出层, 激活函数 softmax, batch 大小为 32. 训练集准确率为 99.98%, 验证集准确率为 80.41%, 平均准确率为 95.76%, 整个训练集上运行速度约为 23 秒. 训练过程图如下图所示 (在第 3 个 epoch 时验证集上 loss 值达到最小, 后面模型就已经发生轻微过拟合)



3.4 EfficientNet

模型来自[TensorFlow Hub-efficientnet_lite4](#), 具体架构使用的是 EfficientNet-B4, 迁移学习后模型大小约为 53.1MB, 在原始模型后加入 1000 个神经元的输出层, 加入一层 0.5 的 Dropout 层, 最后连接 257 个神经元的输出层, 激活函数分别为 ReLU 和 softmax, batch 大小为 256. 训练集准确率为 98.45%, 验证集准确率为 83.79%, 平均准确率为 96.7%, 整个训练集上运行速度约为 41 秒. 训练过程图如下图所示 (准确率有波动就是因为加入了 Dropout 层导致的, 由于验证集上准确率较低, 我们考虑对其加入 Dropout 避免过拟合)



4 结论与讨论

我们制作了下表便于比较模型性能：

模型名称	模型大小	平均准确率	运行速度
VGG-19	485MB	85.81%	250s
Inception-ResNet-v2	227MB	96.97%	100s
MoblieNet-v2	16.5MB	95.76%	23s
EfficientNet-B4	53.1MB	96.70%	41s

可以看出来 EfficientNet 在模型大小和速度上都优于传统深层神经网络； MoblieNet 模型大小上有非常大的优势，而且准确率仅低不到 1%； Inception-ResNet 在大小和速度上都不占优势，但是达到了最高的平均准确率；经典神经网络 VGG-19 与新型神经网络相比差距就比较大了。

通过这次图像识别实验，我们学习到了卷积神经网络原理和用于图像识别的神经网络发展历程，并详细分析了每种神经网络在何处进行了优化，开拓了我们在卷积神经网络方面的视野，以后再科研当中可以结合这些优化技巧对模型进行优化。

在图像分类模型上，由于我们第一次使用服务器训练神经网络，在服务器配置上花费了不少时间，而且对神经网络框架并不熟悉，通过本次学习学到很多模型训练技巧，使得我们以后可以更高效的训练模型。迁移学习模型均先使用 TensorBoard 可视化输出效果如图10，然后选择训练效果较好的结果进行展示（ResNet50 可能是参数问题，准确率较低，故没有展示）

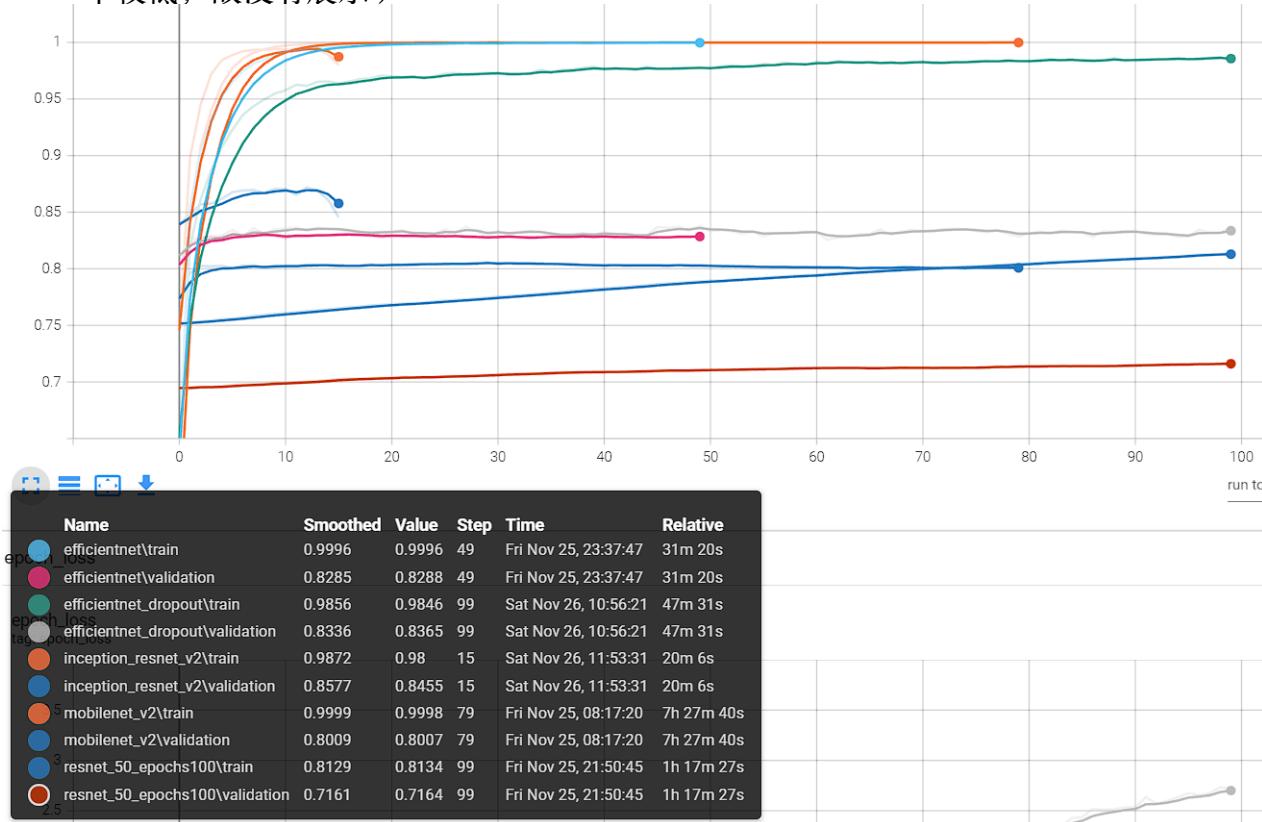


图 10: TensorBoard 可视化效果

A EfficientNet 与其他网络的比较

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	77.1%	93.3%	5.3M	1x	0.39B	1x
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	79.1%	94.4%	7.8M	1x	0.70B	1x
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	80.1%	94.9%	9.2M	1x	1.0B	1x
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	81.6%	95.7%	12M	1x	1.8B	1x
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	82.9%	96.4%	19M	1x	4.2B	1x
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	83.6%	96.7%	30M	1x	9.9B	1x
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	84.0%	96.8%	43M	1x	19B	1x
EfficientNet-B7	84.3%	97.0%	66M	1x	37B	1x
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models ([Hu et al., 2018](#)), or models pretrained on 3.5B Instagram images ([Mahajan et al., 2018](#)).

图 11: EfficientNet 与其他网络在 ImageNet 数据集上的比较