

基于卷积神经网络的图像分类

【摘要】 在本次实验中, 基于 MNIST 数据集, 采用了多种 CNN 架构进行图像分类任务, 包括五种经典 CNN 架构和五种现代轻量级 CNN, 并尝试对比分析, 解释他们结果的异同, 对结果进行可视化.

【关键词】 人工智能, 计算机视觉, 卷积神经网络, 图像分类, MNIST 数据集

Project 3: CNN-based Image Classification

Abstract: In this experiment, we conducted image classification tasks utilizing multiple Convolutional Neural Network (CNN) architectures based on the MNIST dataset. Specifically, the research encompassed ten distinct CNN models, comprising five classical CNN architectures and five contemporary lightweight CNN designs. The investigation systematically compared and analyzed their respective performance outcomes, subsequently presenting the results through comprehensive visualization techniques.

Key Words: AI, CV, CNN, Convolutional Neural Network, Image Classification, MNIST Dataset

1 MNIST 数据集

MNIST^[1]是一个用于训练各种数字图像处理系统的数据库, 涵盖手写数字的图像, 包含 60,000 张训练图像和 10,000 张测试图像, 尺寸为 28×28 像素, 也被广泛运用于机器学习领域的训练与测试当中.

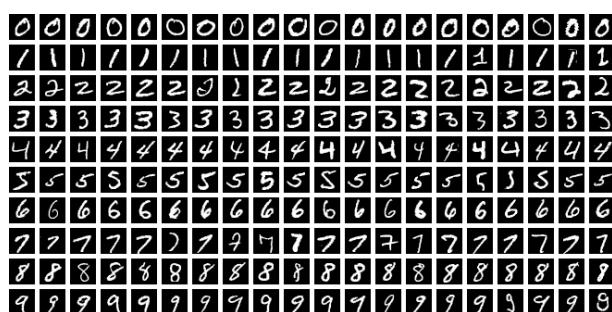


Figure 1 Samples from MNIST Dataset

2 ILSVRC 与经典 CNN 架构

ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 是计算机视觉历史上的重要竞赛, 它基于 ImageNet 的大型图像数据集, 包括了多个 CV 任务赛道, 如图像分类与目标定位, 目标检测, 分割等. 追寻其发展过程, 我们可以探究多个

经典的 CNN 模型.

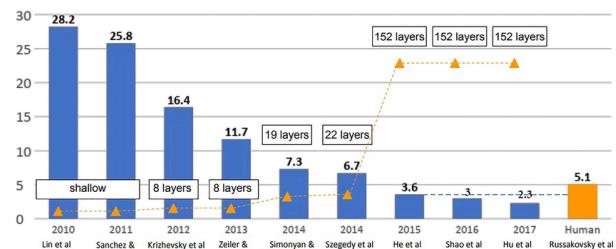


Figure 2 ILSVRC Models

2.1 LeNet

不过在此之前, 我们需要首先追溯 CNN 架构的鼻祖-LeNet^[2]. 该架构经历了一系列迭代, 最终在 1995 年形成了今天熟知的 LeNet-5 架构, 在 1998 年得到了实际应用.

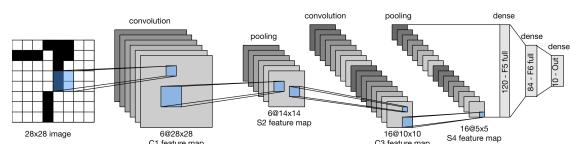


Figure 3 LeNet-5, Overview

它的具体实现上,有很多在现在看来已经 outdated 的技术,例如它使用了 Sigmoid 激活而不是现在流行的 ReLU 类型,采用了平均池化而不是最大池化,但他基本奠定了现在的 CNN 的框架. 具体来说,它依次进行了两次卷积-激活-池化,在过程中减少了宽高维度的尺寸,用卷积核数量提高了通道维度. 然后进行了展平(有时也被认为是第三次卷积),然后接上了类似 MLP 的线性全连接层,最后分类输出. 可以看到它的结构已经很全面,并且在 MNIST 这样的小数据集上取得了不错的效果.

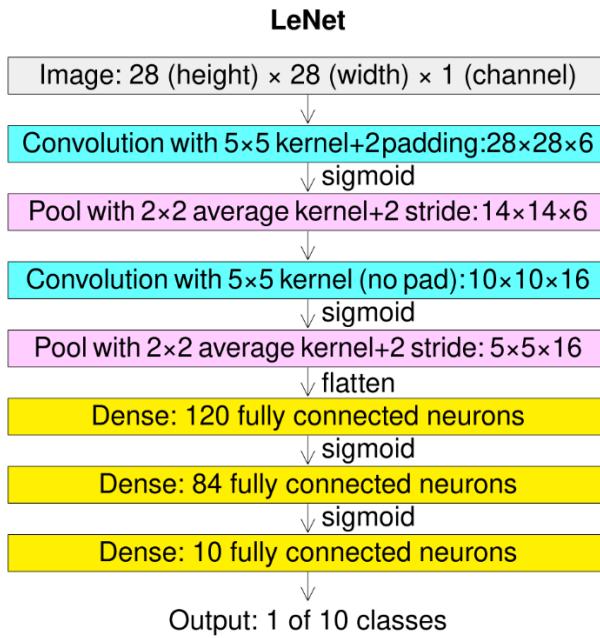


Figure 4 LeNet, Architecture

2.2 AlexNet

AlexNet^[3]是 ILSVRC 的第一个基于 CNN 架构的冠军,相比于先前的模型在 Top5 错误率上取得了巨大提升. 由于当时 GPU 显存的限制,AlexNet 被分成了两个部分分在两个 GPU 上训练,仅在部分层互通数据.

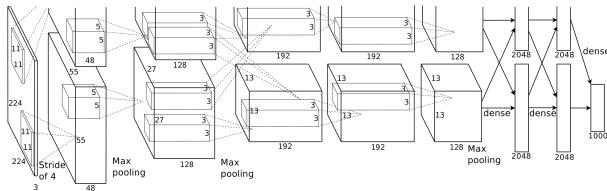


Figure 5 AlexNet, Overview

在具体架构上,AlexNet 仍然是卷积层,池化层和全连接层,但具体细节上有很多了现代技术,例

如 ReLU 激活函数,最大池化,以及全连接层中的 Dropout 等.

AlexNet

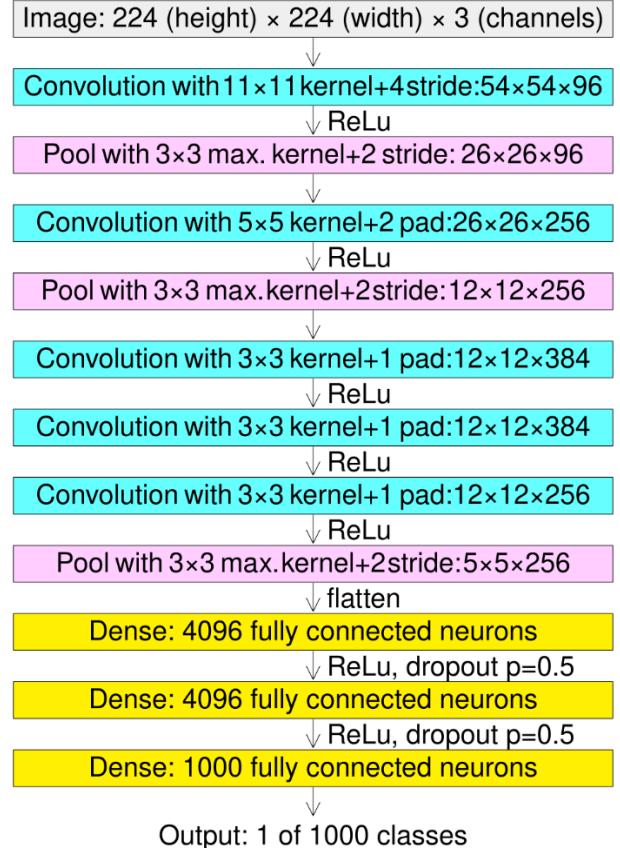


Figure 6 AlexNet, Architecture

在次年提出了 ZFNet^[4], 基于可视化方法, 对 AlexNet 中的卷积核大小和步长等进行了调优, 进一步提升了性能表现.

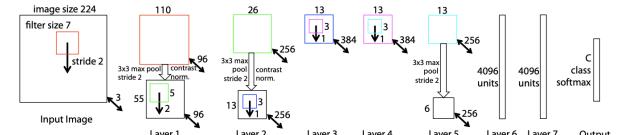


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \times 6 \times 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

Figure 7 ZFNet

2.3 VGG

VGG^[5]于 AlexNet 的两年后提出, 是对其架构的进一步挖掘, 它的核心思路是采用更小的卷积核, 但堆叠更多层数, 维持感受野.VGG 还提供了多个不同层数的版本, 从最小的 VGG11 到最大的 VGG19.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
		maxpool			
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128	conv3-128	conv3-128
		maxpool			
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv1-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256
		maxpool			
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv1-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
		maxpool			
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv1-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
		maxpool			
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 8 VGG, Overview

本次实验中我们会聚焦于最小的 VGG11

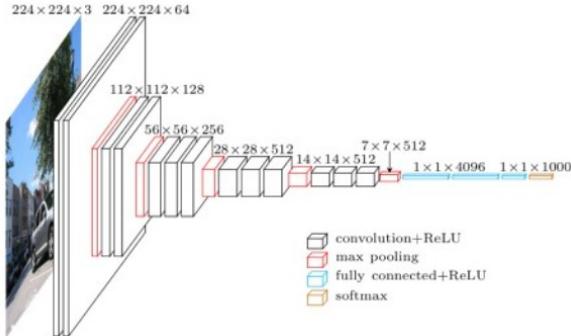


Figure 9 VGG11, Architecture

2.4 GoogLeNet

同年的 GoogLeNet^[6]则带来了更大的创新, 它提出了 Inception 模块, 堆叠不同大小的卷积核来提供多种层次上的信息, 同时用 1*1 的卷积核进行降维

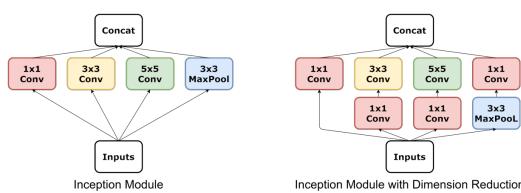


Figure 10 Inception

在网络中则是在一开始的普通卷积操作之后堆叠 Inception 模块. 在最后的分类器中, 它减少了

线性层的数量, 从而极大减少了整个网络的参数数量. 他还在中间提供了两个辅助分类器, 在训练过程中会参与损失计算.

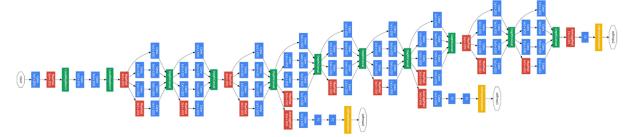


Figure 11 GoogLeNet

在后续还提出了 Inception V2, V3^[7]

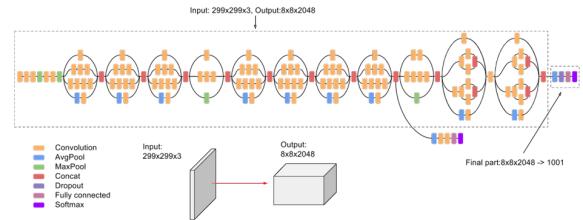


Figure 12 Inception V3

V4 以及 Inception-ResNet 结合体^[8]等.

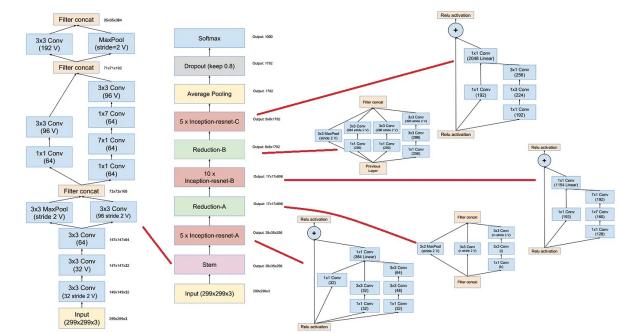


Figure 13 Inception-ResNet V2

2.5 ResNet

先前的 CNN 架构层数都很有限, 最多也就是 20 层左右, 而 2015 年的 ResNet^[9]改变了这一点, 提供了最多 152 层的深层架构, 其关键在于残差结构. 它通过让模型学习「残差」抑制了模型对先前数据的遗忘, 从而使得深层网络成为可能.

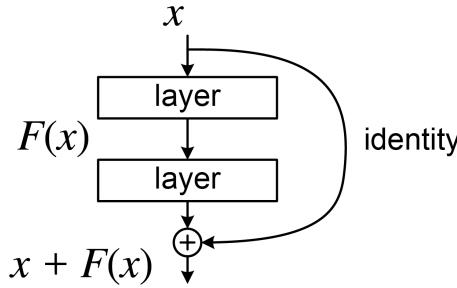


Figure 14 Residual Block

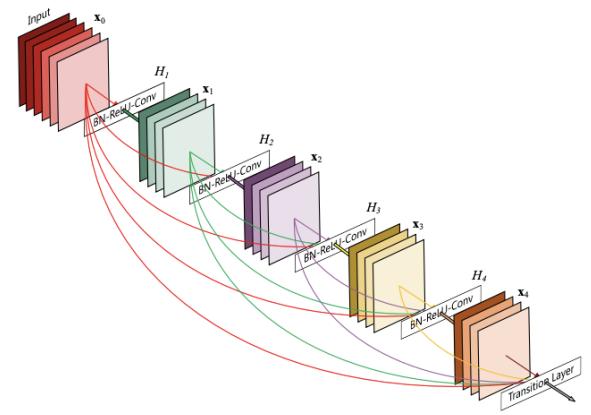


Figure 17 DenseNet

此外, 它还提供了 Bottleneck 块, 借鉴了 Inception 的思想, 通过 $1*1$ 卷积先降维在升维, 提高了运算效率.

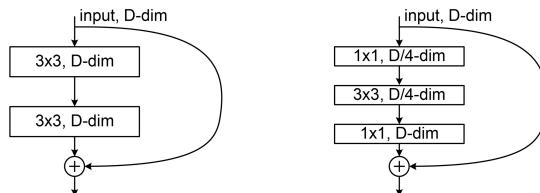


Figure 15 Bottleneck Block

在网络上, 它应用了 Batch Normalization^[10], 进行残差模块的堆叠, 提供了多种深度, 最低 18 层. 它还用全局平均池化代替了部分全连接层的工作以减少参数量.

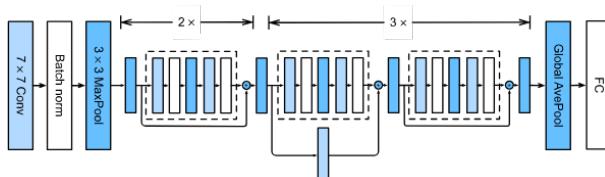


Figure 16 ResNet

ResNeXt^[13] 将先前提到的 Inception 模块融入了 ResNet

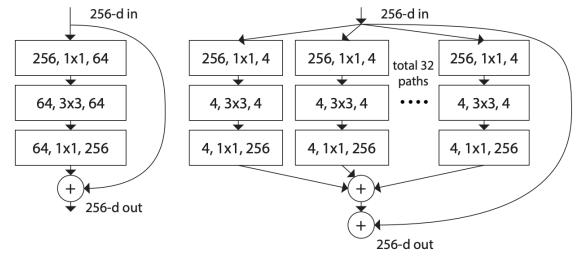


Figure 18 ResNeXt

Wide ResNet^[14] 则是提高了「宽度」(即卷积核的数量, 通道数), 并且引入了 Dropout(ResNet 没有采用), 从而实现了用更少的层数获得同等效果.

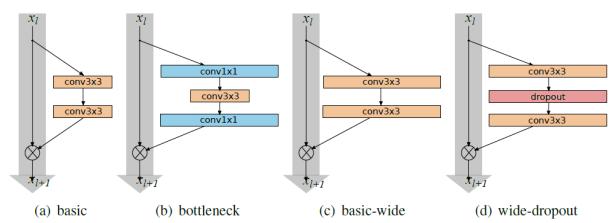


Figure 19 Wide ResNet

随后的工作包括 DenseNet^[11], 将每一层的输出都连接到后面的层里和 Stochastic Depth^[12], 随机丢弃一些层, 提供正则化效果, 提高深层网络的泛化能力.

3 现代轻量级 CNN 架构

在探究了五种历史上的经典 CNN 架构之后, 我们再来研究一些轻量化的 CNN 架构.

3.1 SqueezeNet

SqueezeNet^[15]则是设计了一种 fire 模块作为网络的基本构建单元, 由 squeeze 卷积层和 expand 卷积层组成, 前者使用 1×1 卷积限制输入通道数, 后者包含 1×1 和 3×3 卷积, 增加特征表达能力. 它推迟了池化操作的位置, 以保持更高分辨率的特征映射. 最终用仅 AlexNet $\frac{1}{50}$ 的模型大小实现了相近的准确率.

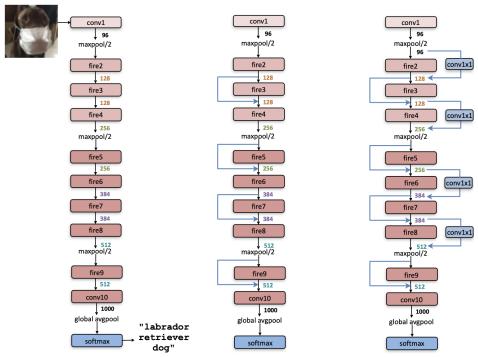


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

Figure 20 SqueezeNet

3.2 MobileNet

MobileNet 系列专门为移动和嵌入式视觉识别应用设计，包含了 V1^[16], V2^[17], V3^[18]三次迭代。V1 的核心创新是引入深度可分离卷积 (Depth-wise Separable Convolution)，将标准卷积分解为深度卷积和逐点卷积两个步骤，从而显著减少计算量和模型参数。

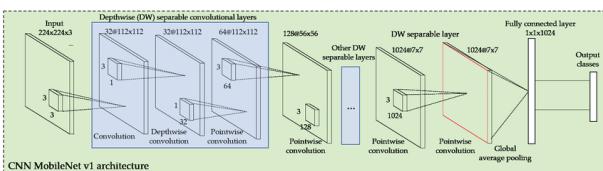


Figure 21 MobileNet V1

V2 引入线性瓶颈 (Linear Bottleneck) 结构, 采用反向残差 (Inverted Residual) 模块和扩展卷积来增加特征表达能力

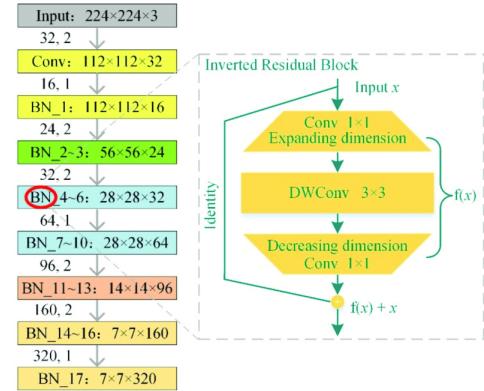


Figure 22 MobileNet V2

V3 结合了网络架构搜索 (NAS), 引入序列化注意力 (Squeeze-and-Excitation) 模块, 优化了激活函数, 并提供了 Small 和 Large 两个版本

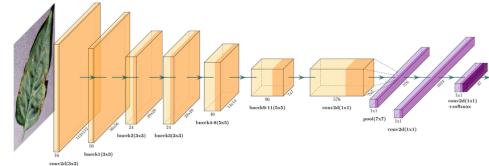


Figure 23 MobileNet V3

SE 模块在 SENet^[19]里提出, 也是 ILSVRC 的最后一届冠军. 它是一种注意力机制, 核心思想是通过自适应地重新校准通道间的特征响应, 动态地增强信息丰富的特征, 抑制不太重要的特征, 分为压缩, 激励, 重标定三个阶段.

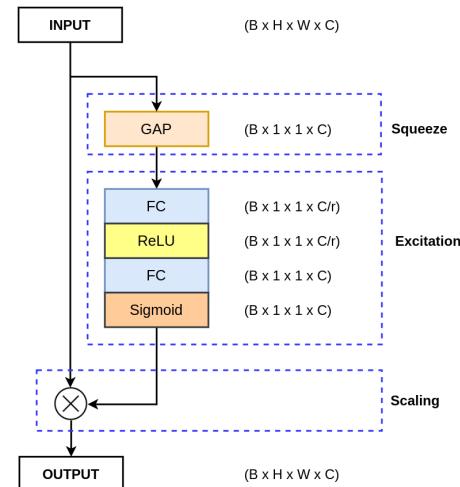


Figure 24 SE

3.3 ShuffleNet

ShuffleNet 主要有 V1^[20] 和 V2^[21]. 主要实现了分组卷积, 即将标准卷积分割成多个分组和通道重排允许不同组之间的信息交流, 解决分组卷积信息流受限的问题.

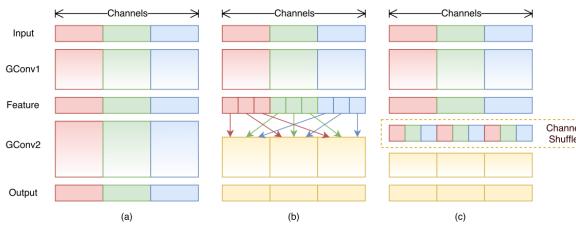


Figure 25 Channel Shuffle

而 V2 提出基于实际 MAC(内存访问成本) 的模型评估标准, 不仅关注理论计算量, 更重视实际硬件性能, 来优化通道分配策略, 并提供多种计算复杂度的版本 (0.5x, 1x, 1.5x).

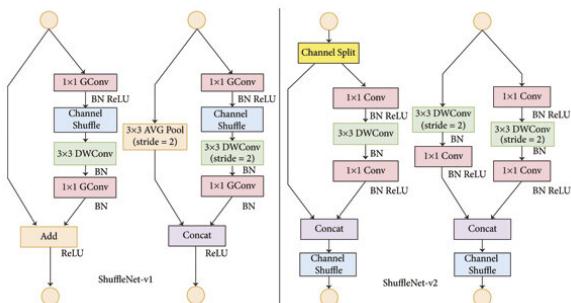


Figure 26 ShuffleNet Unit

3.4 EfficientNet

EfficientNet 也有 V1^[22] 和 V2^[23]. 它提出了复合缩放方法, 引入了三个维度的缩放系数深度 (d), 宽度 (w) 和分辨率 (r) 进行协同优化, 遵循经验公式 $depth = \alpha^\phi, width = \beta^\phi, resolution = \gamma^\phi$, 通过网络架构搜索 (NAS) 确定最佳系数. 采用 MBCConv(移动反向残差块) 作为基本构建单元, 提高计算效率.

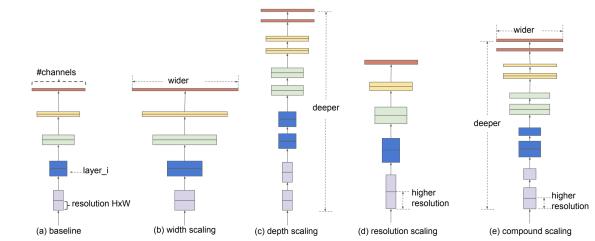


Figure 27 Model Scaling

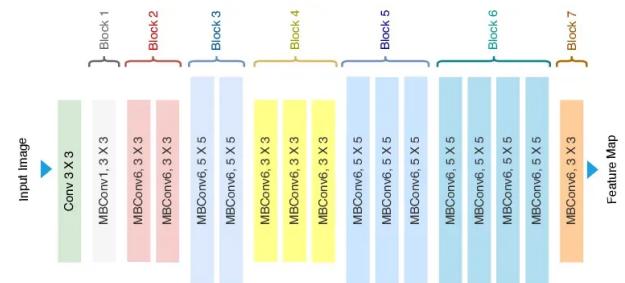


Figure 28 EfficientNet Architecture

V2 则引入了自适应正则化技术和渐进学习策略, 以及 Fused-MBConv 模块, 进一步减少计算开销

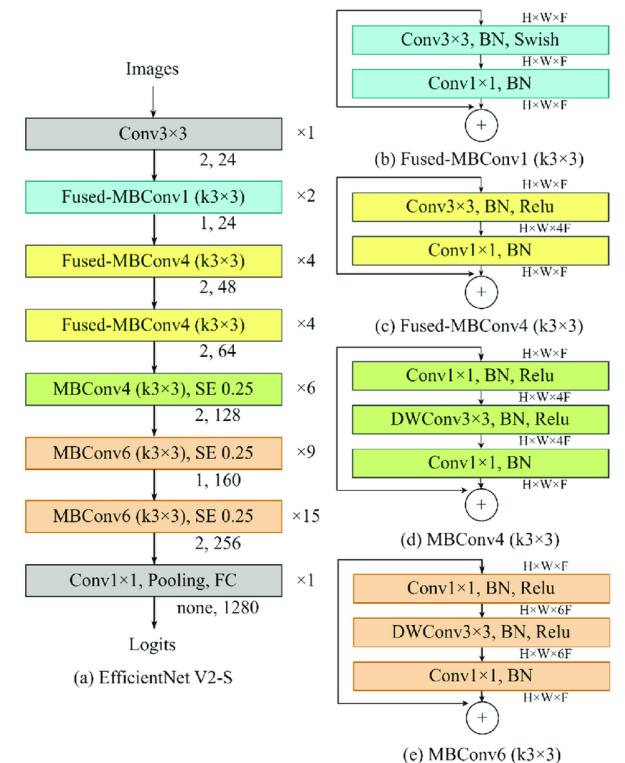


Figure 29 EfficientNet V2

3.5 MNASNet

MNASNet^[24]引入基于强化学习的神经架构搜索方法,实现了自动化架构搜索.采用多目标优化策略,同时考虑模型准确率和计算延迟,使用可区分的硬件延迟代理作为约束条件.它定义了灵活的网络构建块搜索空间,包括卷积类型,核大小,通道数等多个维度.

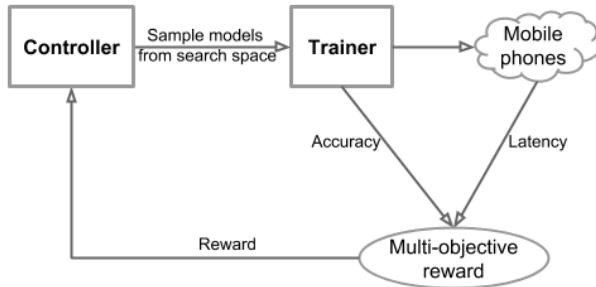


Figure 30 MNASNet Search

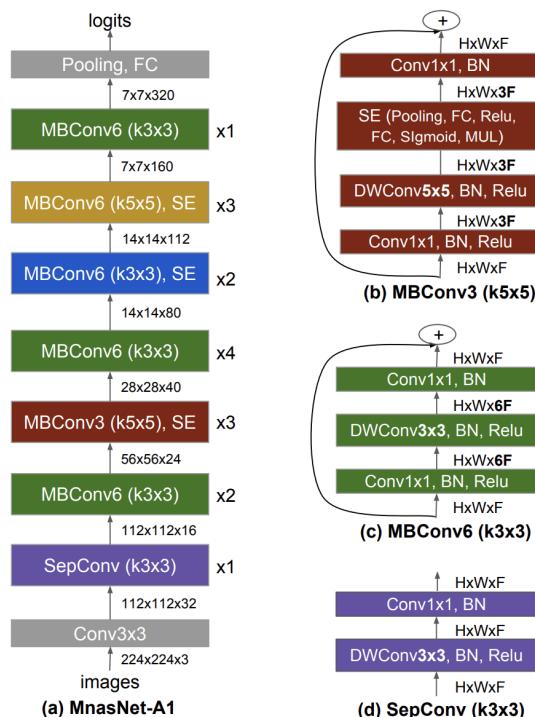


Figure 31 MNASNet Architecture

4 实验

4.1 训练

4.1.1 数据增强

在视觉任务中,一般为了提高模型的范围能力,我们可以做数据增强,例如对原始图像做随机

旋转,平移,缩放,擦除以及亮度,对比度的调整等.

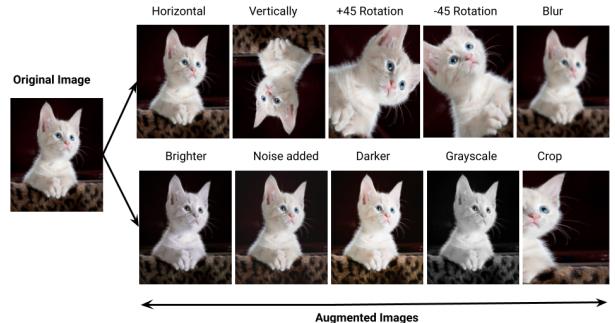


Figure 32 Data Augmentation

本次实验中就应用了这样的一些策略,不过由于 MNIST 数据集本身很简单,在加上权值衰减,Dropout,Batch Normalization 等其他的正则化技术的共同作用,导致了在实验中出现了验证集准确率可能高于训练集的情况.

4.1.2 学习率调度

学习率调度是训练过程中的重要 technique,在本次实验中,我在前几个 epoch 应用了线性的学习率 warmup,在之后则提供了两种学习率下降策略,分别是余弦模式和当验证集准确率不再下降时缩减学习率两种策略.LeNet 在两种模式下的结果差异是比较大的:

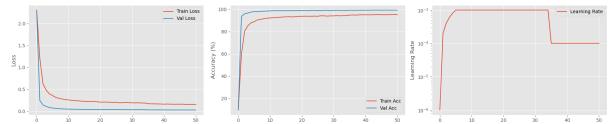


Figure 33 LeNet, Plateau

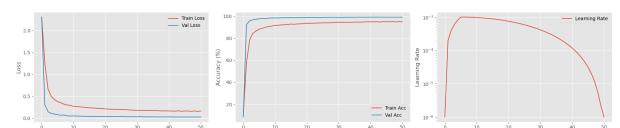


Figure 34 LeNet, Cosine

这个图里可能无法清晰展示,事实上 LeNet 训练过程有比较大的问题,它的稳定性较差,在后期验证集准确率会出现微小的反复波动,导致早停策略和 Plateau 的学习率调整都难以发挥效果,这恐怕也是导致了它受学习率调度策略影响较大的原因.

ResNet 和 Plateau 调度策略非常适配,这也是

原论文使用的方法,在验证集准确率不再上升时将学习率除以 10,最多三次.

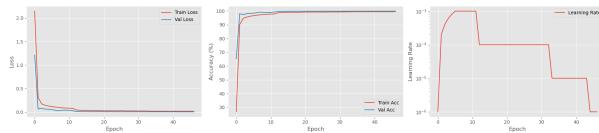


Figure 35 ResNet18, Plateau

AlexNet 搭配余弦调度策略在不到 30 个 epoch 就早停终止训练了, 并且最后的准确率也不低, 时间效率比较高.

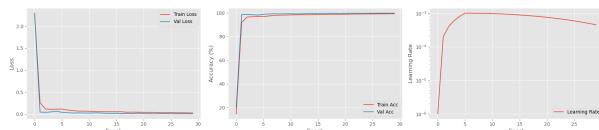


Figure 36 AlexNet, Cosine

4.1.3 模型展示

用 torchsummary 库可以帮助我们清晰看到模型的层次结构, 参数量和大小, 以 LeNet 为例

Layer (type)	Output Shape	Param #
<hr/>		
Conv2d-1	[-1, 6, 24, 24]	156
ReLU-2	[-1, 6, 24, 24]	0
MaxPool2d-3	[-1, 6, 12, 12]	0
Conv2d-4	[-1, 16, 8, 8]	2,416
ReLU-5	[-1, 16, 8, 8]	0
MaxPool2d-6	[-1, 16, 4, 4]	0
Dropout-7	[-1, 256]	0
Linear-8	[-1, 120]	30,840
ReLU-9	[-1, 120]	0
Dropout-10	[-1, 120]	0
Linear-11	[-1, 84]	10,164
ReLU-12	[-1, 84]	0
Linear-13	[-1, 10]	850
<hr/>		
Total params:	44,426	
Trainable params:	44,426	
Non-trainable params:	0	
<hr/>		
Input size (MB):	0.00	
Forward/backward pass size (MB):	0.08	
Params size (MB):	0.17	
Estimated Total Size (MB):	0.26	

Figure 37 LeNet Model Summary

本次实验不是做历史还原的, 而是参考其架构进行实现, 例如我并没有采用 LeNet 原版的 Sigmoid 激活和平均池化, 原版 LeNet 也没有 Dropout. 此外, 原版 AlexNet 包含了 LRN 归一化, 但这项技术后来证明效用较低, 因此也没有实现.

4.2 结果对比

这里为了方便起见, 我统一基于 Plateau 调度来对比. 在测试集准确率上, 其实个模型差别不算太大. 相对来说, EfficientNet 的准确率最高, ResNet 的 loss 最低. LeNet 的表现相对较差.

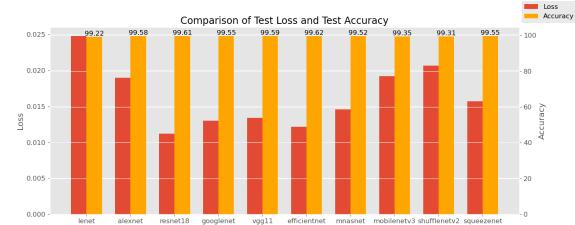


Figure 38 Test Loss and Accuracy

对比测试集, 训练集, 验证集准确率, 可以发现基本没有出现过拟合现象. 训练集的准确率由于数据增强和 Dropout 等略低.

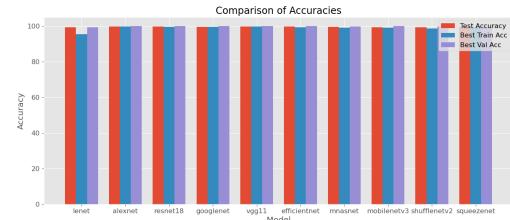


Figure 39 Train, Val and Test Accuracy

在训练时间上,LeNet 凭借参数量有实毫无疑问很低, 这里反常的是 VGG11 很低, 这是因为我实现的 VGG11 是魔改版本的, 在最后我们会解释. ResNet 我们实现了层数最少的 18, 因此也较低. 四种轻量级架构对比 AlexNet 都有降低.

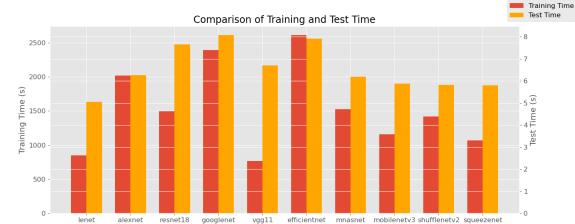


Figure 40 Train and Test Time

这里还有一个反常的是 EfficientNet 很高. 这是因为 EfficientNet 虽然是强调效率的, 但他其实不是一个真正意义上的轻量级模型. Torchvision 提供了一些预训练模型权重

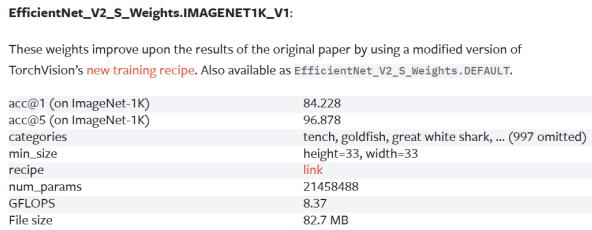


Figure 41 EfficientNet Model

可以看到它的计算需求 GFLOPS 并不低, 它的「轻量」只是相较于更重的模型而言的, 在大型数据集上才能体现出其优势. 测试时间上, 后四种轻量级模型比较有优势, 意味着他们在移动设备上使用时可以获得更低的延迟. 模型大小和参数量上, LeNet 太小了图片上无法展示. 后面的模型相较于 AlexNet 都优化了很多, 特别是几种轻量级模型. 不过在这里我们发现了一个重要事实—参数量小不代表运行性能高, 例如 EfficientNet.

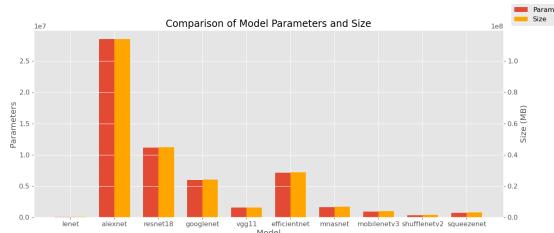


Figure 42 Model Size

修改后的 VGG11 较容易收敛, 能够有效利用早停机制减少实际训练的 epoch 数. 每个 epoch 需要的时间上后四种轻量级架构也比较有优势.

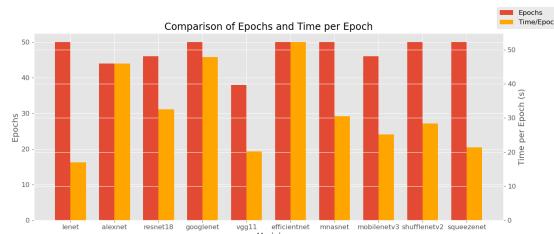


Figure 43 Epochs and Time/Epoch

通过观察训练过程, 我还想比较一个指标, 就是收敛速度. 因为训练过程中后面的很多 epoch 的提升是很小的, 我们可能回想达到一个阈值之后就终止训练, 例如 99% 的验证集准确率

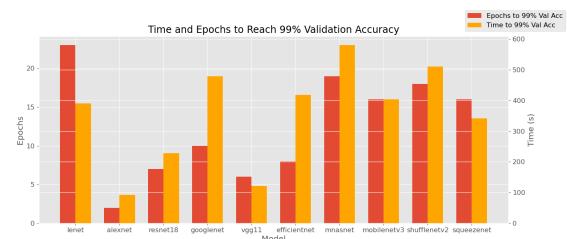


Figure 44 Epoches and Time to Get 99% Accuracy

在这个方面反而是比较简单的 AlexNet 表现比较好, 它的训练是比较容易的. 但由于 99% 这个阈值选取得相对较大, 本身上限就低的 LeNet 表现就很差了.

4.3 可视化

刚才的准确率数据太过冰冷, 有没有一种更直观的方式来展示模型的学习效果呢? t-SNE^[25]就是这样一个技术, 通过它, 我们可以把模型学到的特征向量降至二维并在平面上展示. 好的模型的结果应当有较高的分离度, 例如如下是 LeNet 的结果:

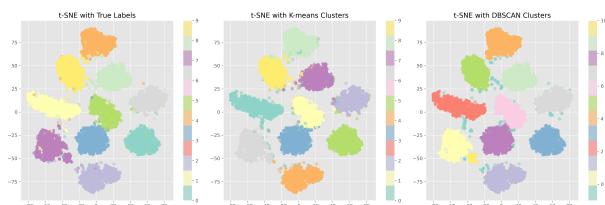


Figure 45 LeNet Feature Visualization

左侧图展示的是正确标签的结果, 中间是 K-means 聚类的结果, 右侧是 DBSCAN 的结果. 我们发现 LeNet 的结果就不算太好, 在正确标签上就没有实现很好的分离, 有较多混杂的点, 相应的聚类效果也不是很好. 那么哪个模型在这方面效果最好呢? 我才用了 ARI 来衡量

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]}{\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]} / \binom{n}{2}$$

并且用最大化 ARI 来搜索 DBSCAN 的超参数(eps 和 MinPts). 用 KNN 曲线拐点对应的 eps 作为基准, 网格搜索其附近的 eps 取值和 MinPts 取值.

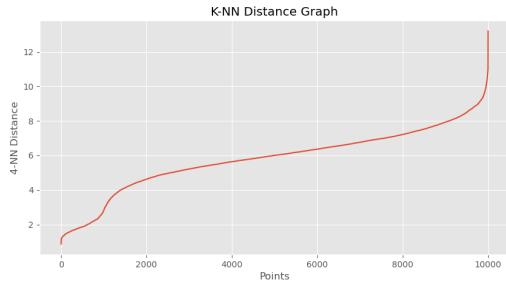


Figure 46 KNN, LeNet

各模型我选择了两种学习率调度效果较好的一个来执行 t-SNE 降维和聚类, 结果如下:

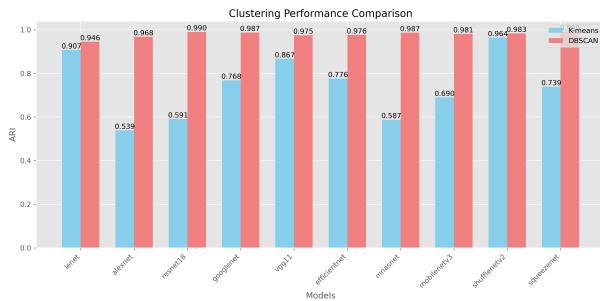


Figure 47 ARI Comparison

K-means 的 ARI 普遍较低,DBSCAN 上最好的是 SqueezeNet, 效果如下:

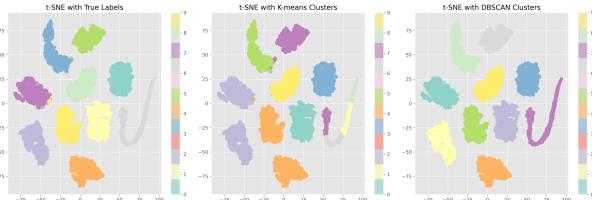


Figure 48 SqueezeNet Feature Visualization

可以看到分离效果确实非常好. 但是像 DBSCAN 的第 7 类, 它虽然有很好的密度连通, 但是它不是圆形的, 导致 K-means 的聚类较差. 而 ShuffleNet 相对来说每一类的分布形状更近似于圆形, 因此 K-means 的表现也很好

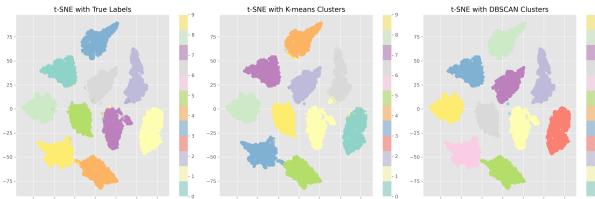


Figure 49 ShuffleNet Feature Visualization

4.4 模型对比

对比 AlexNet 和 ResNet18 的结果差异:

- ResNet18 的准确率更高, 测试 loss 更低, 这主要得益于它的残差连接机制支撑的深层网络, 信息可以跨层传递, 有更强的特征提取和非线性表达能力, 能够更深入地学习图像的层次化特征表示, 缓解了深层网络的退化问题. 此外, ResNet 的卷积核数量更多, 可以学到更广泛的特征.t-SNE 降维后的特征分离度也较高.
- ResNet18 的训练过程更稳定, 在搭配 Plateau 学习率调度时有清晰的三次学习率下降. 恒等映射分支显著改善了梯度在深层网络中的反向传播, 缓解梯度消失问题. 并且训练速度明显更快, 每 Epoch 需要的时间更短. 但 AlexNet 早期收敛较快, ResNet18 需要更多轮次才能达到最优, 因为残差结构在优化过程中需要较长时间来充分利用深层特征.
- ResNet18 在性能领先的情况下, 参数量比 AlexNet 大大降低. 并且 ResNet 的可扩展性很强, 其「家族」有多种层数的不同架构, 最高 152 层. 在简单数据集上参数量低意味着过拟合风险也会更低.

5 问题

在 VGG11 的实现中, 我本来基本是套用了类似原始的卷积参数设置, 不过实验中发现, 原版是针对 ImageNet 数据集, 原始图像的宽高足有 224, 而 MNIST 数据集只有 28. 在多层卷积后, 原始的参数设计会让 feature map 的尺寸太小. 因此我根据 MNIST 数据集的特性调整了卷积层数和卷积核数的配置. 这种针对 MNIST 数据集的定制调整让他在 MNIST 数据集上表现很好, 效率极高, 但是如果是更大的数据集, 准确率可能就会降低. MNIST 数据集太小太简单, 用它来对比模型不是一种全面, 公平的方式, 这是本次实验的局限性.

在 GoogLeNet 里, 我一开始的实现类似于原版, 但在训练过程中出现了梯度爆炸现象, 即 loss 突然暴增; 并且我发现它的训练速度比较缓慢, 在前几个 epoch 里准确率的提升略低. 为了解决这个问题, 我加入了 Batch Normalization, 发现收敛速度大大提高了, 并且也不再出现梯度爆炸.

REFERENCES

- [1] DENG L. The mnist database of handwritten digit images for machine learning research [best of the web][J]. IEEE signal processing magazine, 2012, 29(6): 141-142.
- [2] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
- [3] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25.
- [4] ZEILER M. Visualizing and understanding convolutional networks[C]//European conference on computer vision/arXiv: Vol. 1311. 2014.
- [5] SIMONYAN K. Very deep convolutional networks for large-scale image recognition[A]. 2014.
- [6] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
- [7] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.
- [8] SZEGEDY C, IOFFE S, VANHOUCKE V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C]//Proceedings of the AAAI conference on artificial intelligence: Vol. 31. 2017.
- [9] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
- [10] IOFFE S. Batch normalization: Accelerating deep network training by reducing internal covariate shift[A]. 2015.
- [11] HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.
- [12] HUANG G, SUN Y, LIU Z, et al. Deep networks with stochastic depth[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer, 2016: 646-661.
- [13] XIE S, GIRSHICK R, DOLLÁR P, et al. Aggregated residual transformations for deep neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1492-1500.
- [14] ZAGORUYKO S. Wide residual networks[A]. 2016.
- [15] IANDOLA F N. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size[A]. 2016.
- [16] HOWARD A G. Mobilenets: Efficient convolutional neural networks for mobile vision applications[A]. 2017.
- [17] SANDLER M, HOWARD A, ZHU M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.
- [18] HOWARD A, SANDLER M, CHU G, et al. Searching for mobilenetv3[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1314-1324.
- [19] HU J, SHEN L, SUN G. Squeeze-and-excitation networks [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7132-7141.
- [20] ZHANG X, ZHOU X, LIN M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856.
- [21] MA N, ZHANG X, ZHENG H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]// Proceedings of the European conference on computer vision (ECCV). 2018: 116-131.
- [22] TAN M, LE Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105-6114.
- [23] TAN M, LE Q. Efficientnetv2: Smaller models and faster training[C]//International conference on machine learning. PMLR, 2021: 10096-10106.
- [24] TAN M, CHEN B, PANG R, et al. Mnasnet: Platform-aware neural architecture search for mobile[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 2820-2828.
- [25] VAN DER MAATEN L, HINTON G. Visualizing data using t-sne.[J]. Journal of machine learning research, 2008, 9(11).