



温州大學  
WENZHOU UNIVERSITY

# 深度学习-深度卷积神经网络

黄海广 副教授

2023年04月

- 01** 经典网络
- 02** 深度残差网络
- 03** 其它现代网络
- 04** 卷积神经网络使用技巧

# 1.经典网络

3

## 01 经典网络

**02** 深度残差网络

**03** 其它现代网络

**04** 卷积神经网络使用技巧

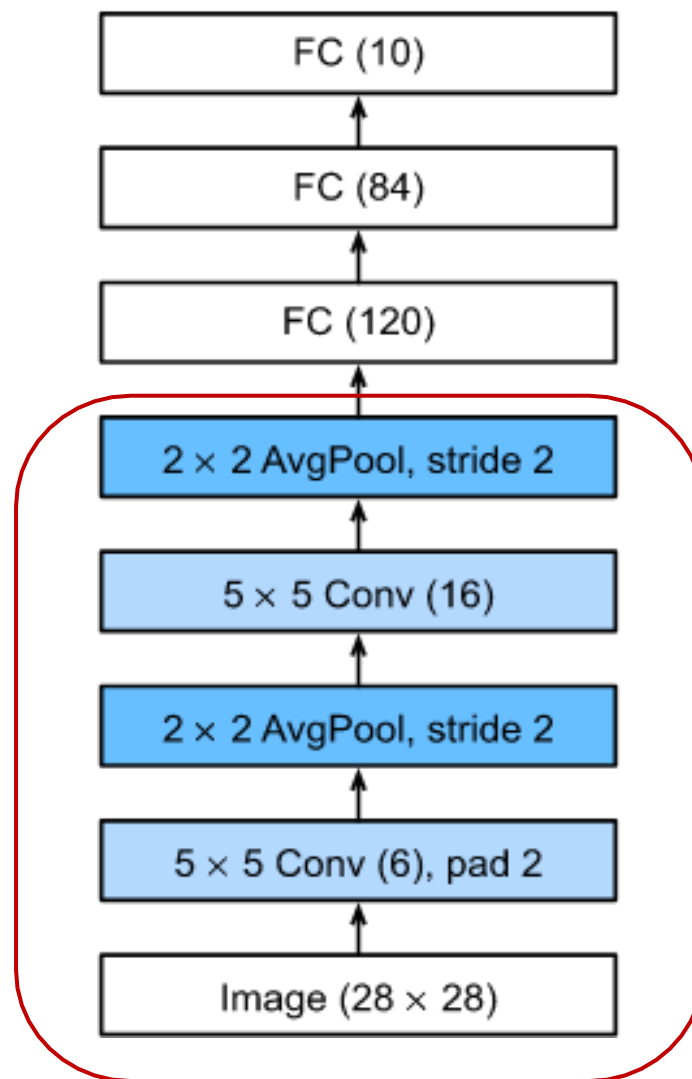
# 经典网络-LeNet-5

4

- LeNet 分为两个部分组成：
  - 卷积层块：由两个卷积层块组成；
  - 全连接层块：由三个全连接层组成。

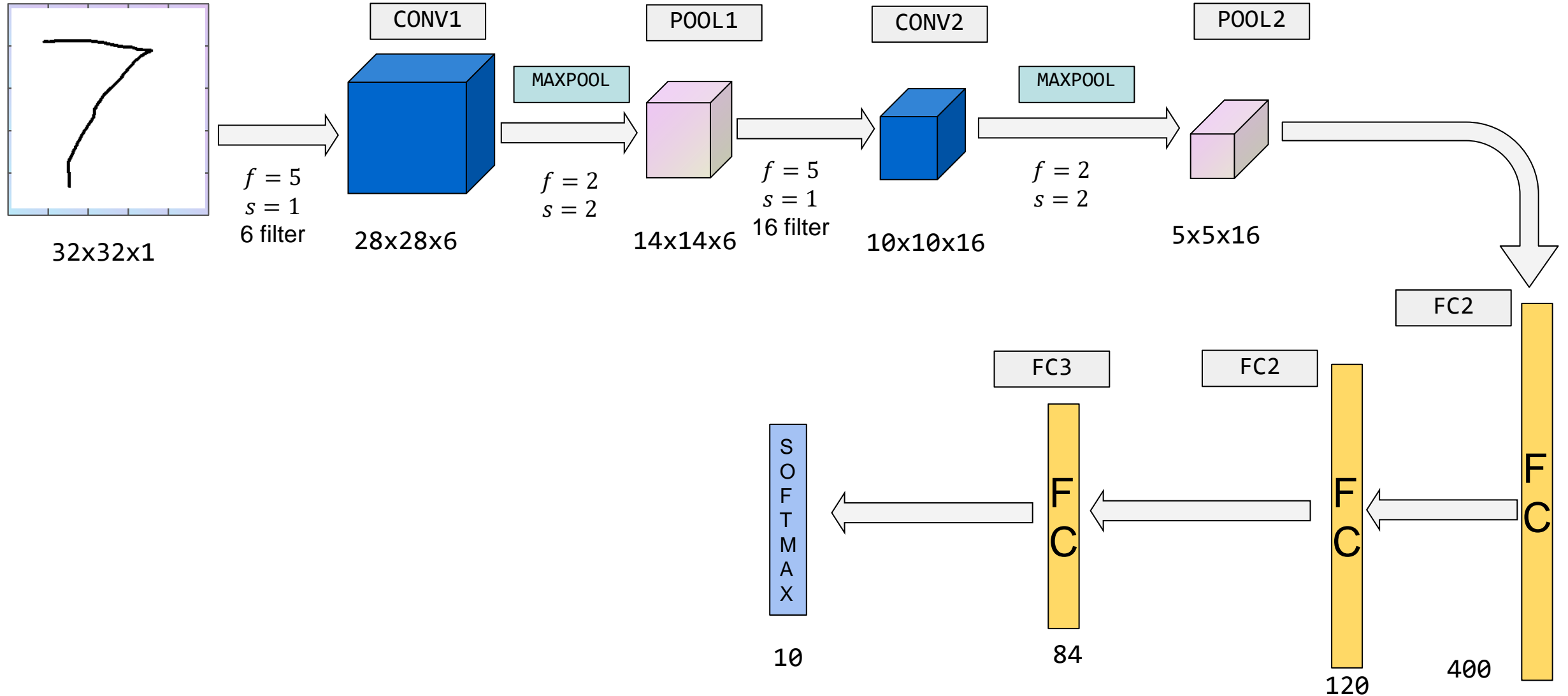
```
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, padding=2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.avg_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.avg_pool2d(x, 2)
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



# LeNet-5

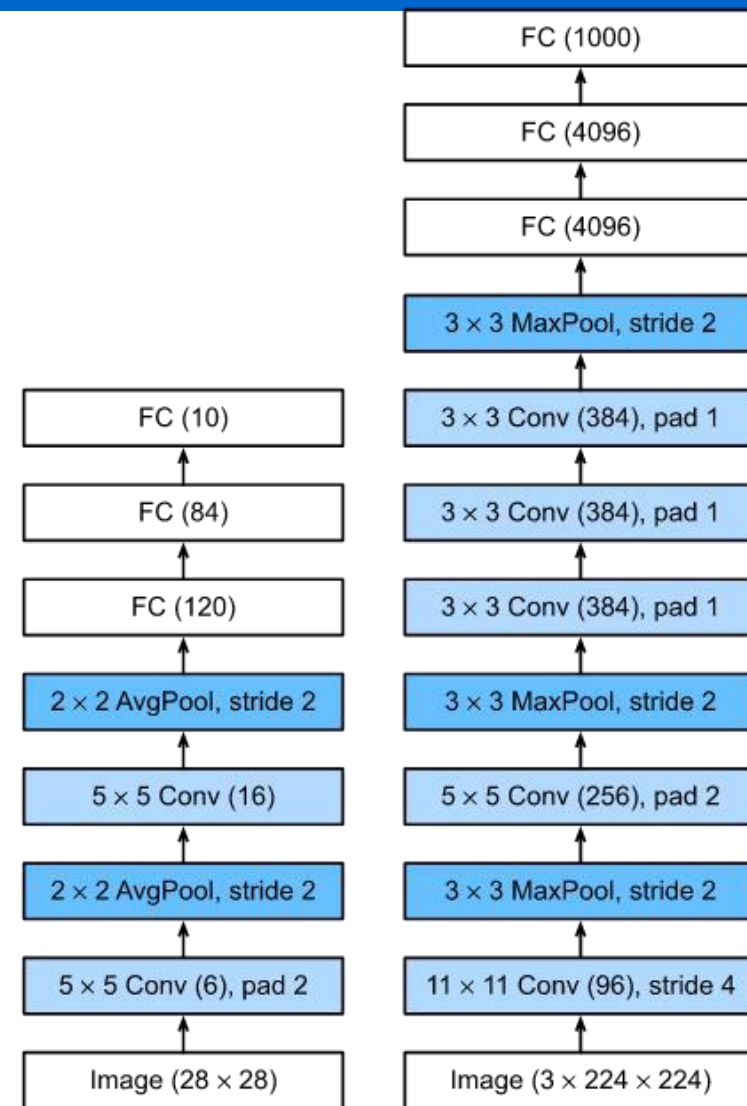
5



# AlexNet

6

- 2012年, AlexNet 横空出世。它首次证明了学习到的特征可以超越手工设计的特征。它一举了打破计算机视觉研究的现状。 AlexNet 使用了8层卷积神经网络, 并以很大的优势赢得了2012年 ImageNet 图像识别挑战赛。



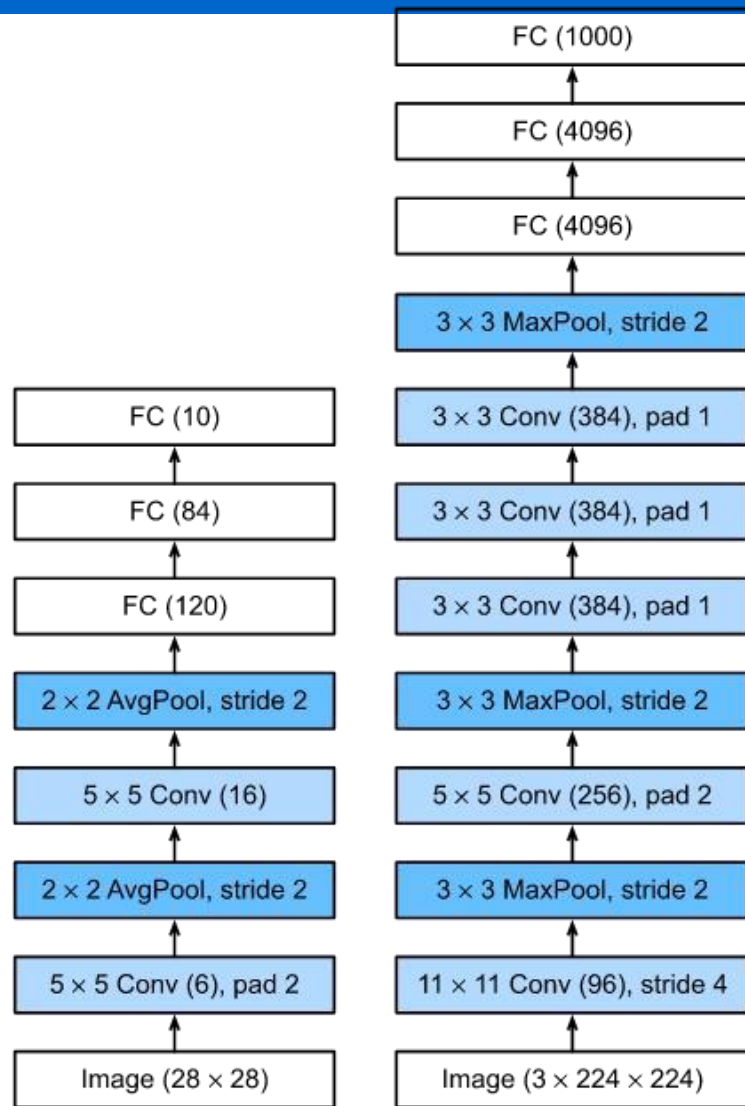
LeNet (左), AlexNet (右)



# AlexNet

7

- 在 AlexNet 的第一层，卷积窗口的形状是  $11 \times 11$ 。由于大多数 ImageNet 中图像的宽和高比 MNIST 图像的多10倍以上，因此，需要一个更大的卷积窗口来捕获目标。第二层中的卷积窗形状被缩减为  $5 \times 5$ ，然后是  $3 \times 3$ 。此外，在第一层、第二层和第五层之后，加入窗口形状为  $3 \times 3$ 、步幅为 2 的最大池化层。此外，AlexNet 的卷积通道是 LeNet 的10倍。
- 在最后一个卷积层后有**两个全连接层**，分别有4096个输出。这两个巨大的全连接层拥有将近 1GB 的模型参数。由于早期 GPU 显存有限，原版的 AlexNet 采用了双数据流设计，使得每个 GPU 只负责存储和计算模型的一半参数。幸运的是，现在GPU显存相对充裕，所以我们现在很少需要跨 GPU 分解模型 (因此，我们的AlexNet模型在这方面与原始论文稍有不同)。



LeNet (左), AlexNet (右)

# AlexNet

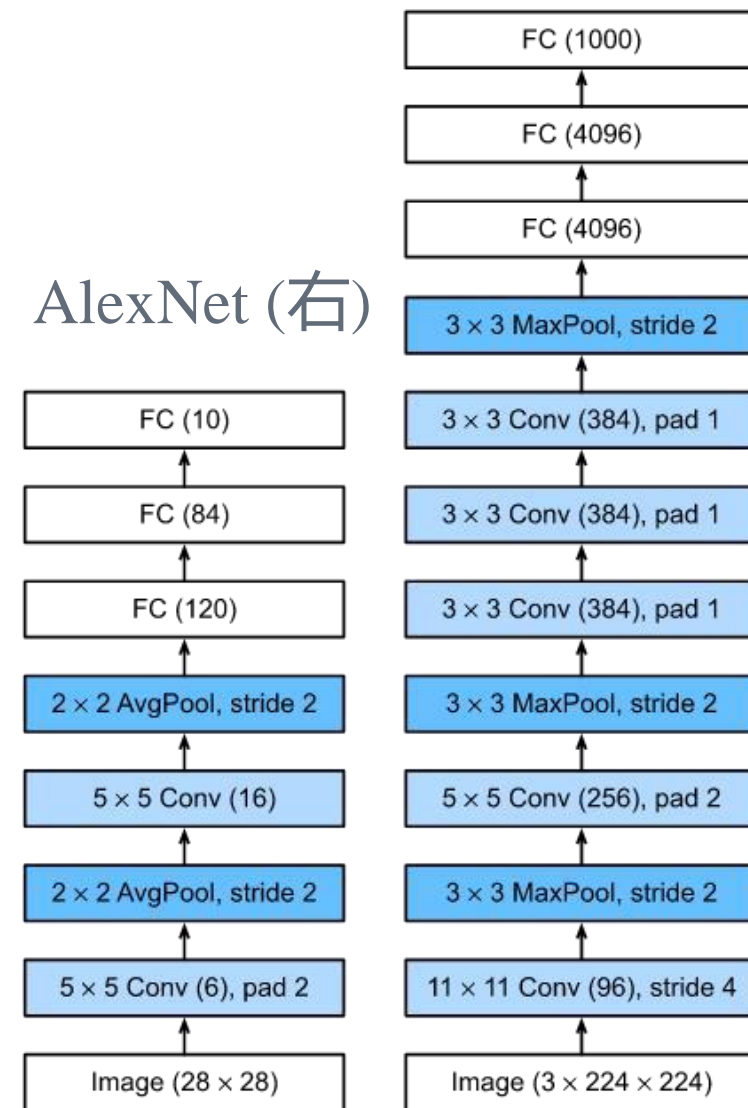
8

- AlexNet 将 sigmoid 激活函数改为更简单的 ReLU 激活函数。

```
class AlexNet(nn.Module):
    def __init__(self):
        super(AlexNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 96, 11, stride=4, padding=2)
        self.pool1 = nn.MaxPool2d(3, stride=2)
        self.conv2 = nn.Conv2d(96, 256, 5, padding=2)
        self.pool2 = nn.MaxPool2d(3, stride=2)
        self.conv3 = nn.Conv2d(256, 384, 3, padding=1)
        self.conv4 = nn.Conv2d(384, 384, 3, padding=1)
        self.conv5 = nn.Conv2d(384, 256, 3, padding=1)
        self.pool3 = nn.MaxPool2d(3, stride=2)
        self.fc1 = nn.Linear(256 * 6 * 6, 4096)
        self.fc2 = nn.Linear(4096, 4096)
        self.fc3 = nn.Linear(4096, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool1(x)
        x = F.relu(self.conv2(x))
        x = self.pool2(x)
        x = F.relu(self.conv3(x))
        x = F.relu(self.conv4(x))
        x = F.relu(self.conv5(x))
        x = self.pool3(x)
        x = x.view(-1, 256 * 6 * 6)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

LeNet (左), AlexNet (右)

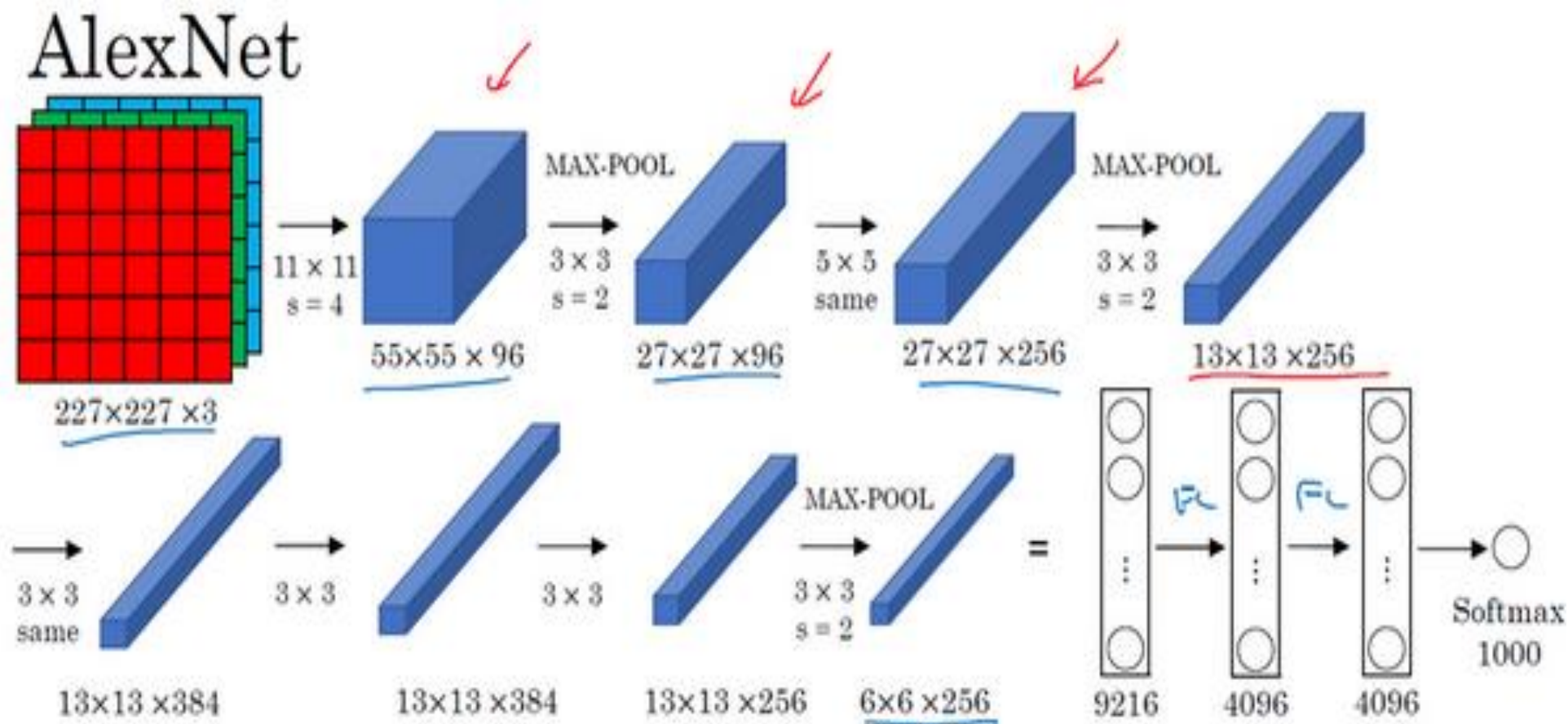


“Talk is cheap. Show me the code.”



# AlexNet

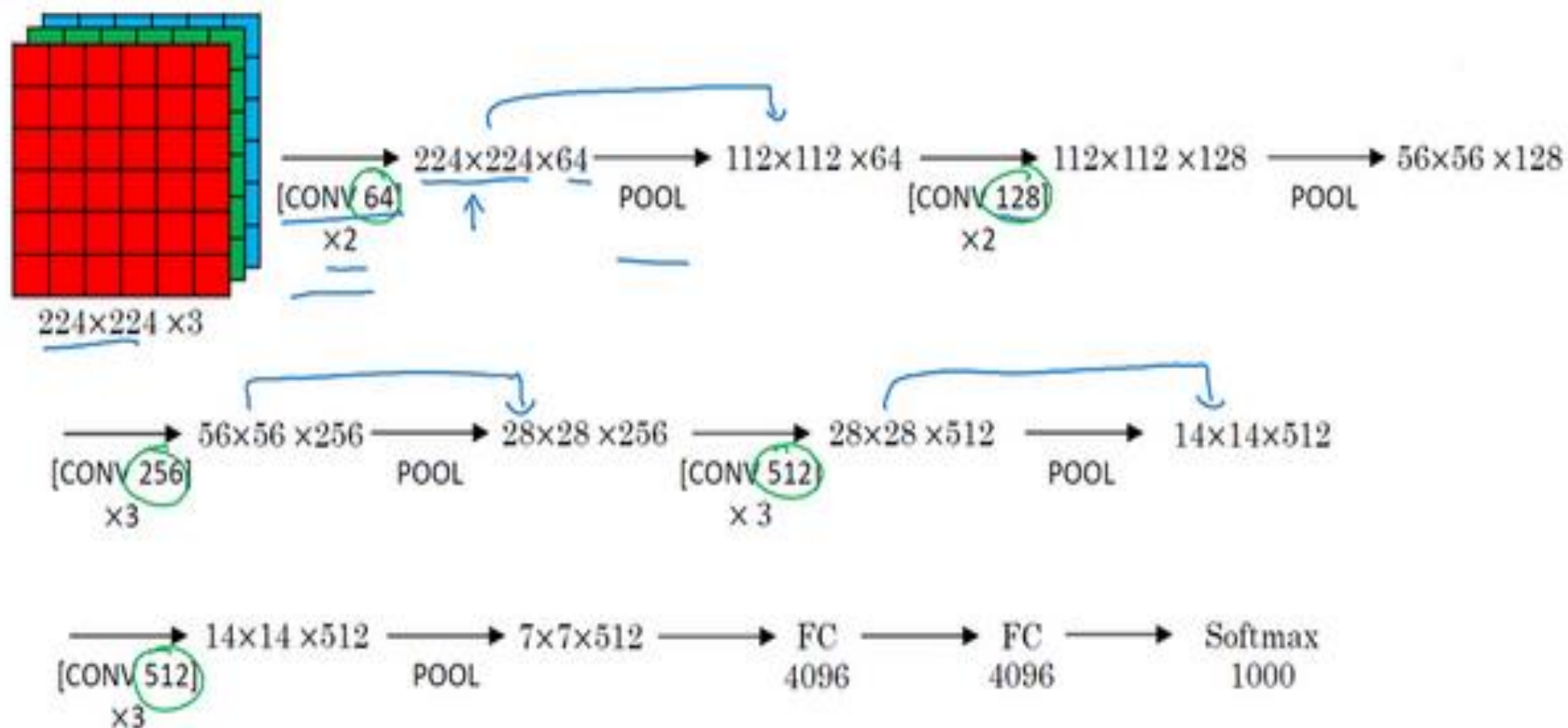
9



# VGG16

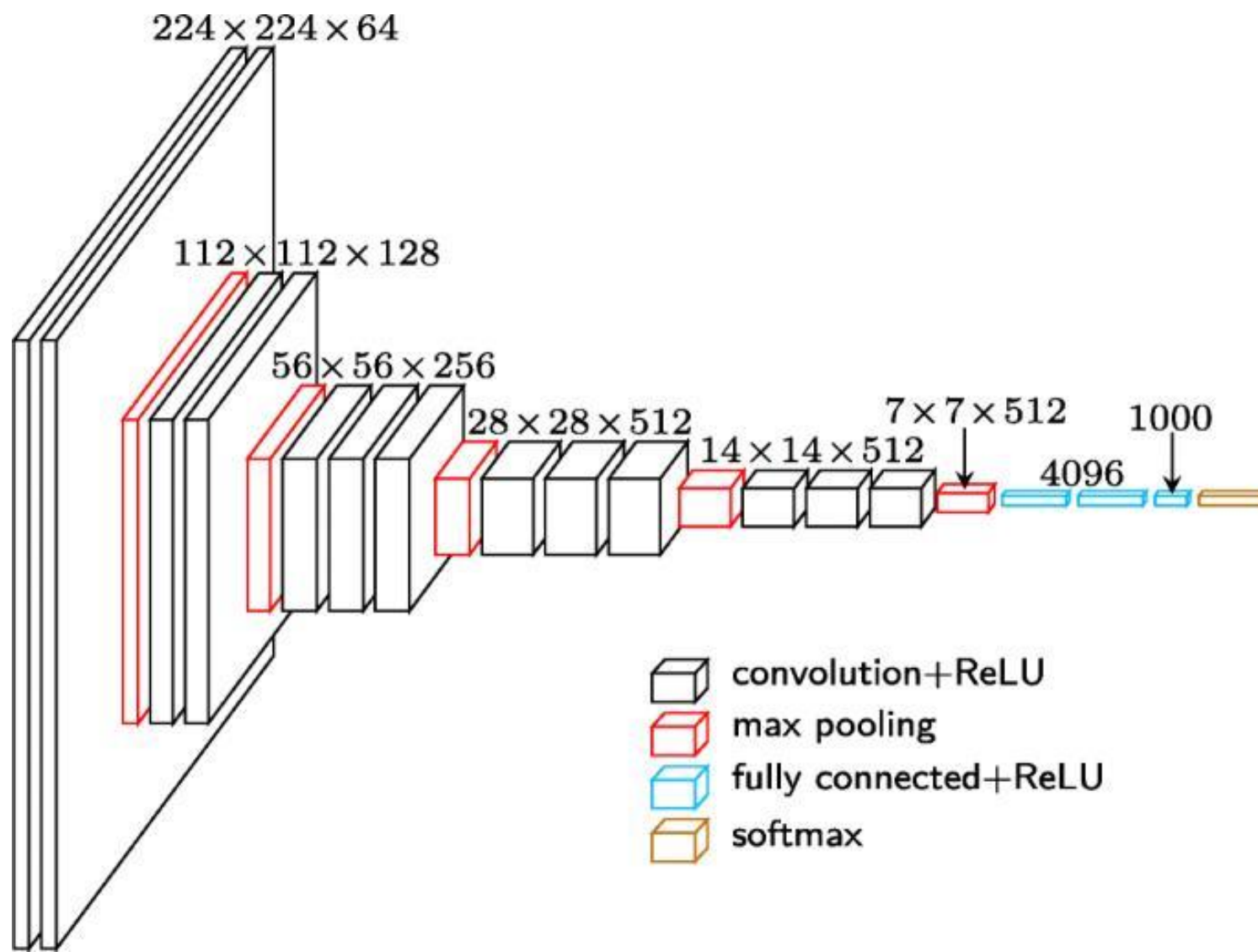
10

## VGG16



# VGG16

11



## 2.深度残差网络

12

**01** 经典网络

**02** 深度残差网络

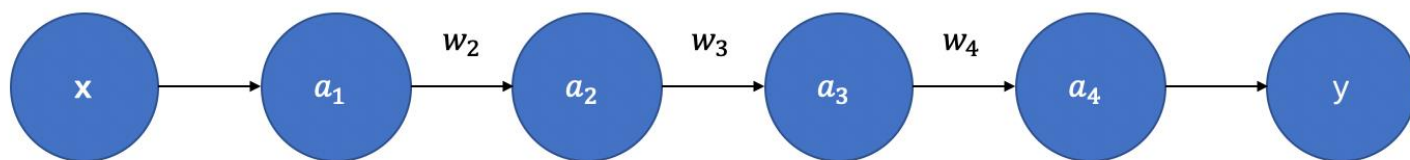
**03** 其它现代网络

**04** 卷积神经网络使用技巧

## 2.深度残差网络

13

### 梯度消失和梯度爆炸问题



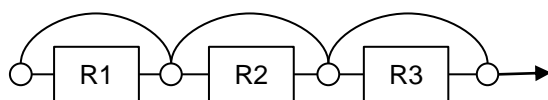
$$\frac{\partial y}{\partial a_1} = \frac{\partial y}{\partial a_4} \frac{\partial a_4}{\partial z_4} \frac{\partial z_4}{\partial x_4} \frac{\partial x_4}{\partial z_3} \frac{\partial z_3}{\partial x_3} \frac{\partial x_3}{\partial z_2} \frac{\partial z_2}{\partial x_2} \frac{\partial x_2}{\partial z_1} \frac{\partial z_1}{\partial a_1} = \frac{\partial y}{\partial a_4} \sigma'(z_4) w_4 \sigma'(z_3) w_3 \sigma'(z_2) w_2 \sigma'(z_1)$$



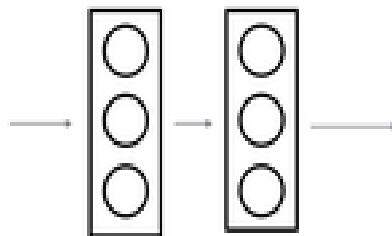


## 2.深度残差网络

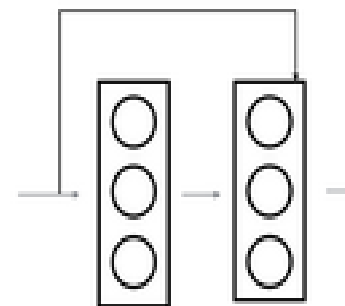
15



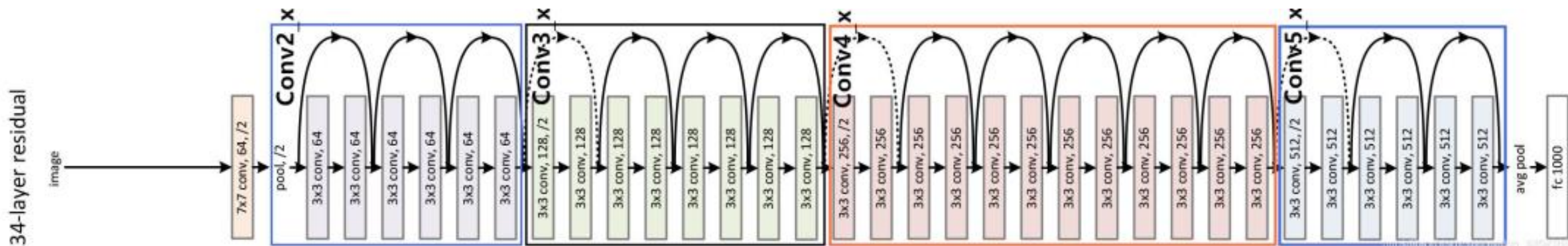
without skip connection



with skip connection



ResNets使用了许多same卷积



# 3.其它现代网络

16

**01** 经典网络

**02** 深度残差网络

**03** 其它现代网络

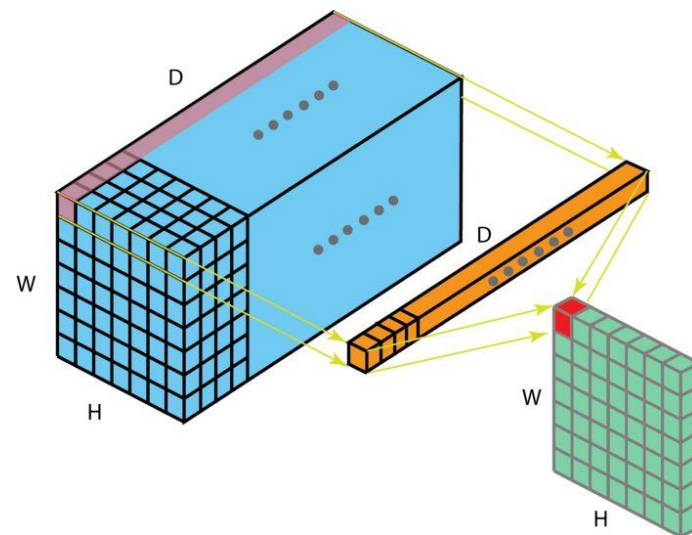
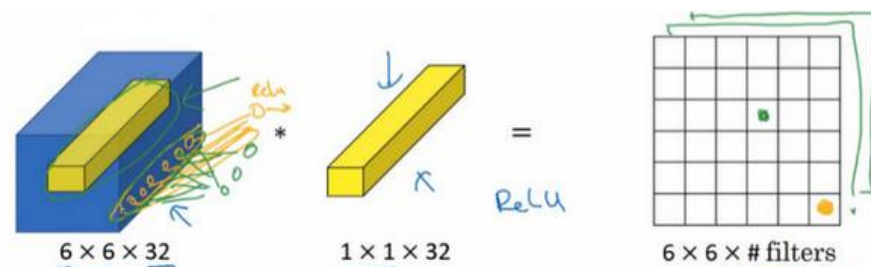
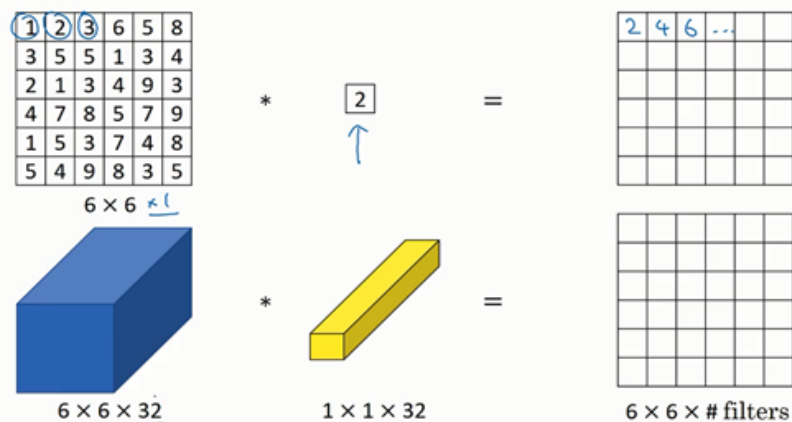
**04** 卷积神经网络使用技巧

# 3.谷歌Inception网络

17

## 1×1 卷积 (Network in Network)

Why does a  $1 \times 1$  convolution do?

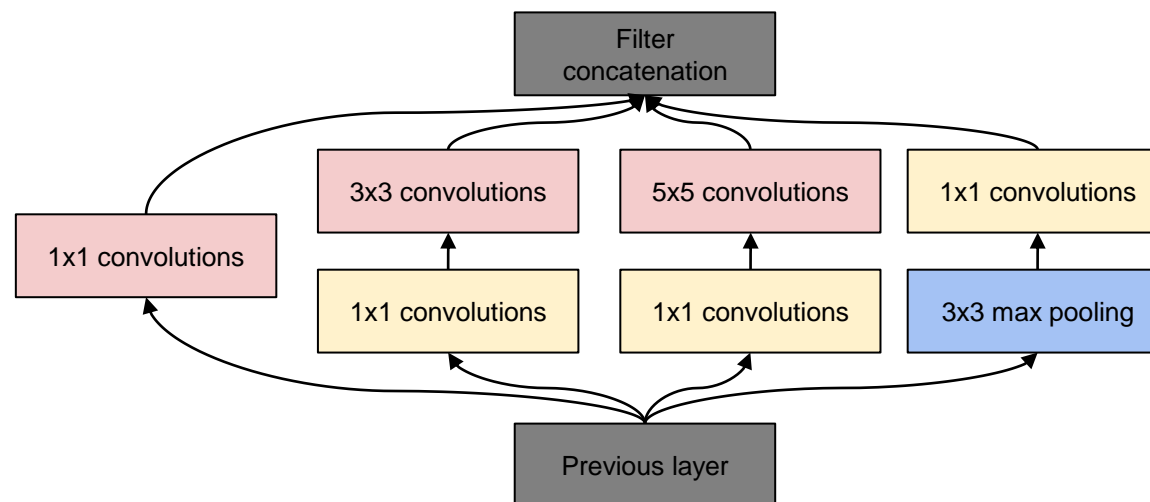
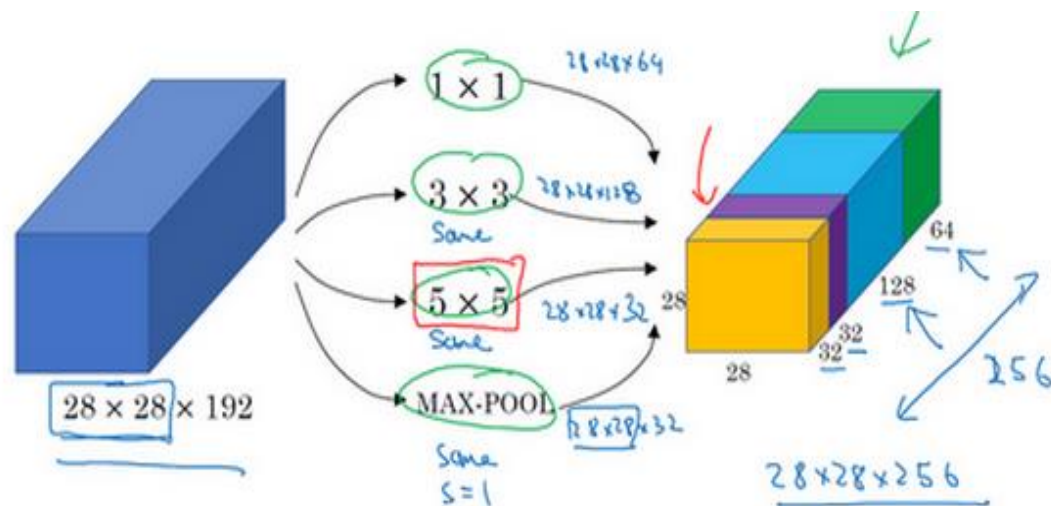


$1 \times 1$ 卷积层就是这样实现了一些重要功能的 (**doing something pretty non-trivial**)，它给神经网络添加了一个非线性函数，从而减少或保持输入层中的通道数量不变，当然如果你愿意，也可以增加通道数量。

# 3.谷歌Inception网络

18

## Inception模块

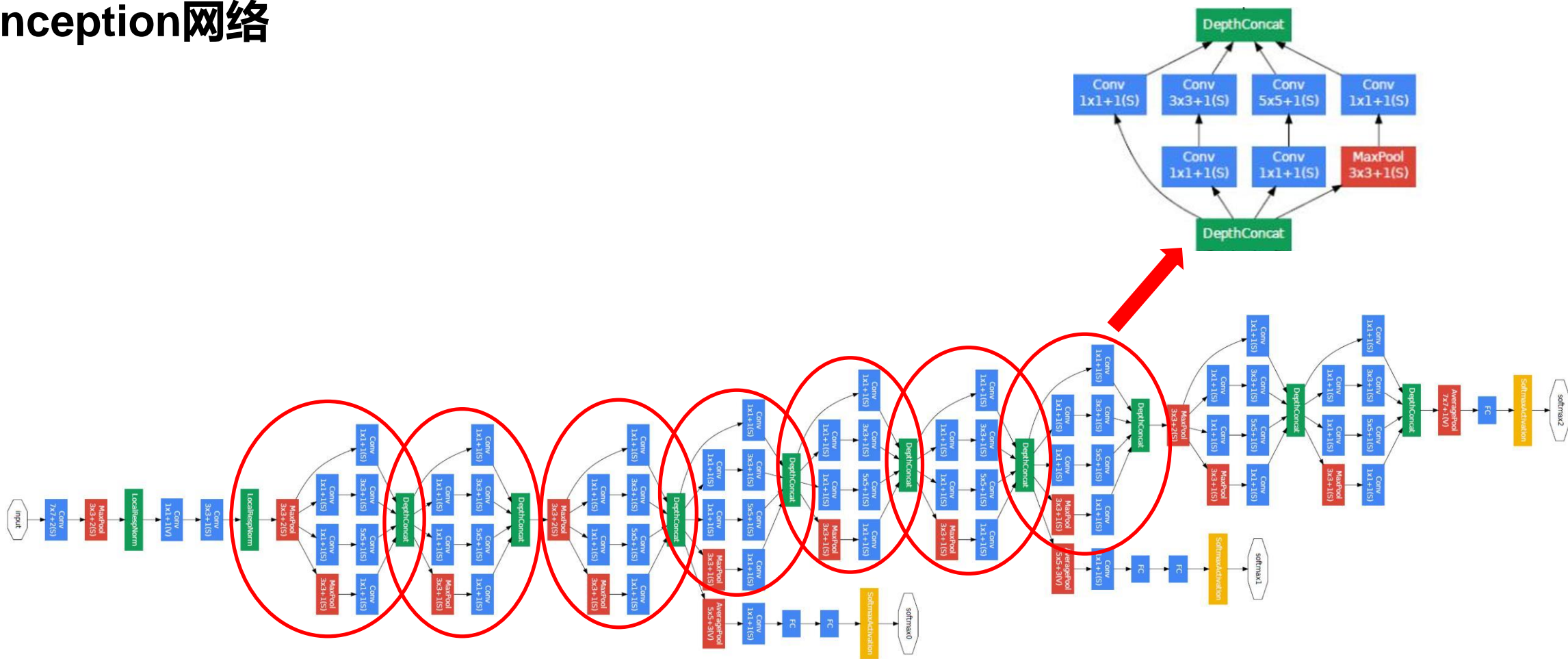


有了这样的**Inception**模块，你就可以输入某个量，因为它累加了所有数字，这里的最终输出为  $32+32+128+64=256$ 。**Inception**模块的输入为  $28 \times 28 \times 192$ ，输出为  $28 \times 28 \times 256$ 。

# 3.谷歌Inception网络

19

## Inception网络



# 3.其它现代网络

20

## DenseNet

**DenseNet**是2017年CVPR会议上提出的一种卷积神经网络结构，其名字来源于“密集连接网络（Densely Connected Network）”。

DenseNet的创新点在于在网络结构中引入了密集连接，使特征复用和梯度传播更加容易，在处理图像分类、目标检测、分割等问题中都取得了不错的结果。

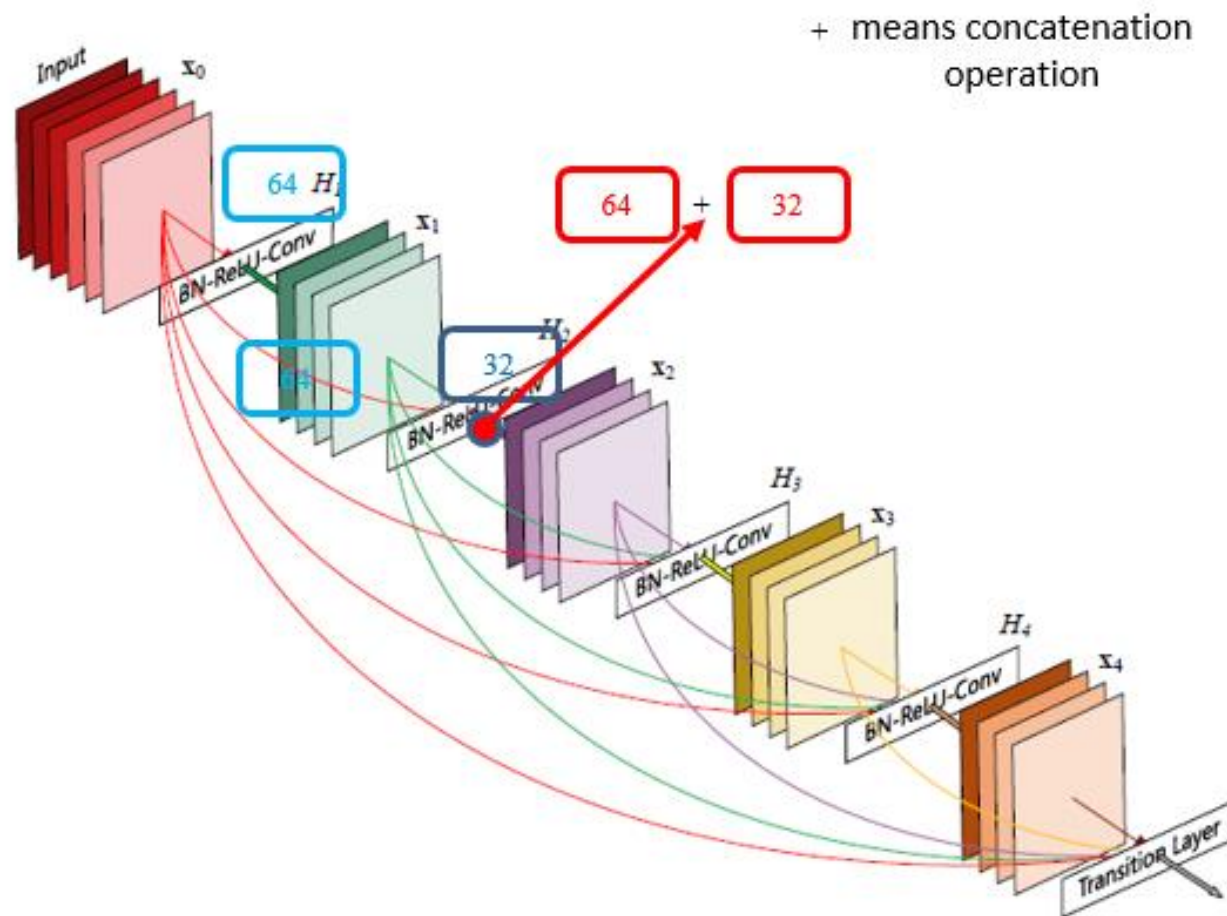


Figure 1. A 5-layer dense block with a growth rate of  $k = 4$ . Each layer takes all preceding feature-maps as input.

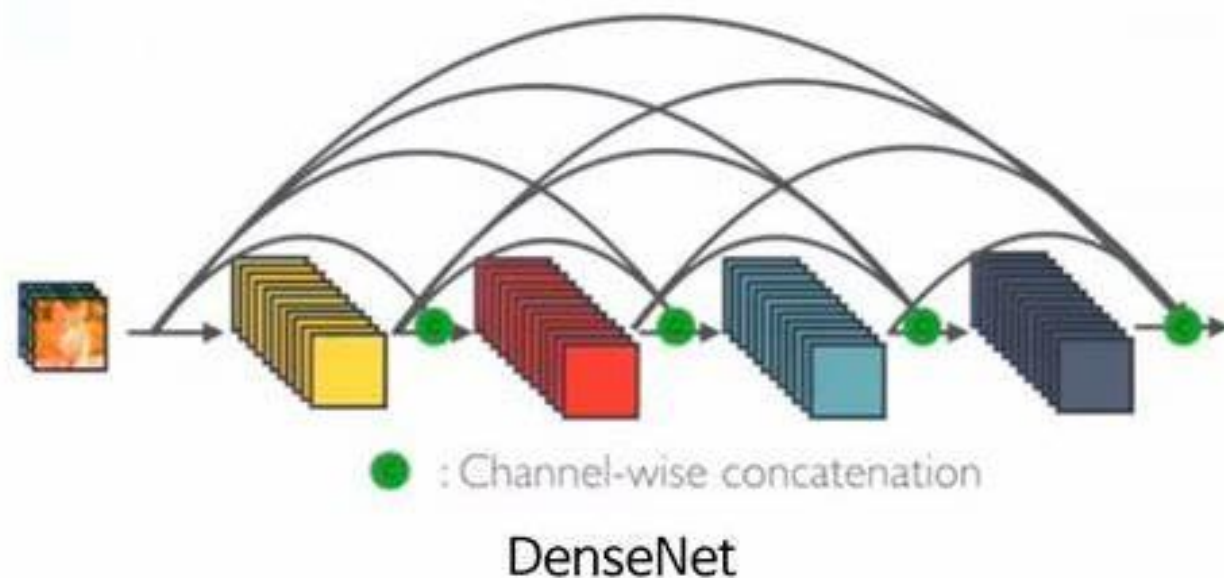
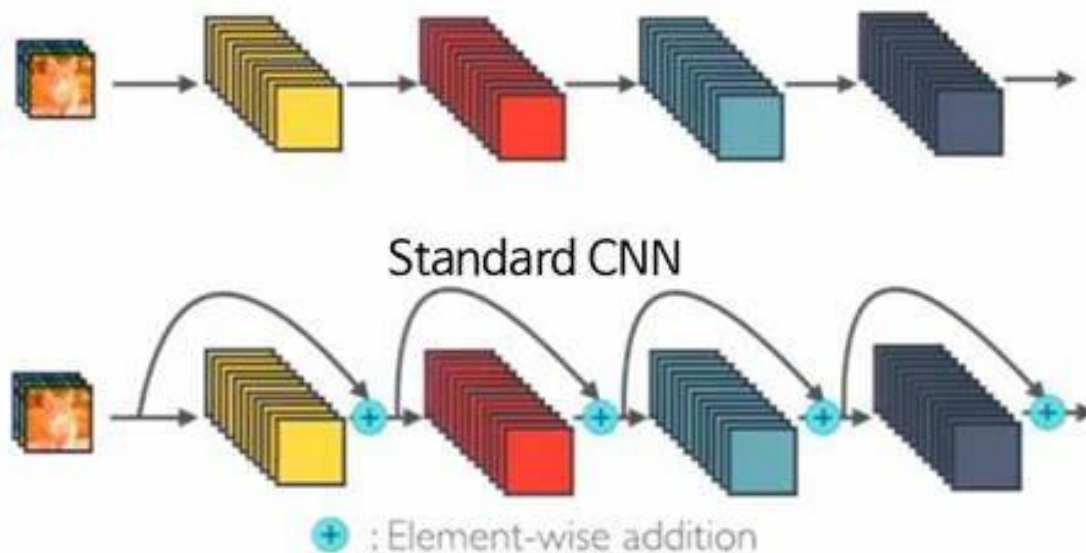


# 3.其它现代网络

21

## DenseNet

总的来说，DenseNet和ResNet都是很优秀的卷积神经网络结构，但DenseNet通过建立密集连接，使每一层都直接接收到多个之前层的特征图输出，增强了特征的流动和复用，从而在模型性能和训练稳定性上表现更好。



# 3.其它现代网络

22

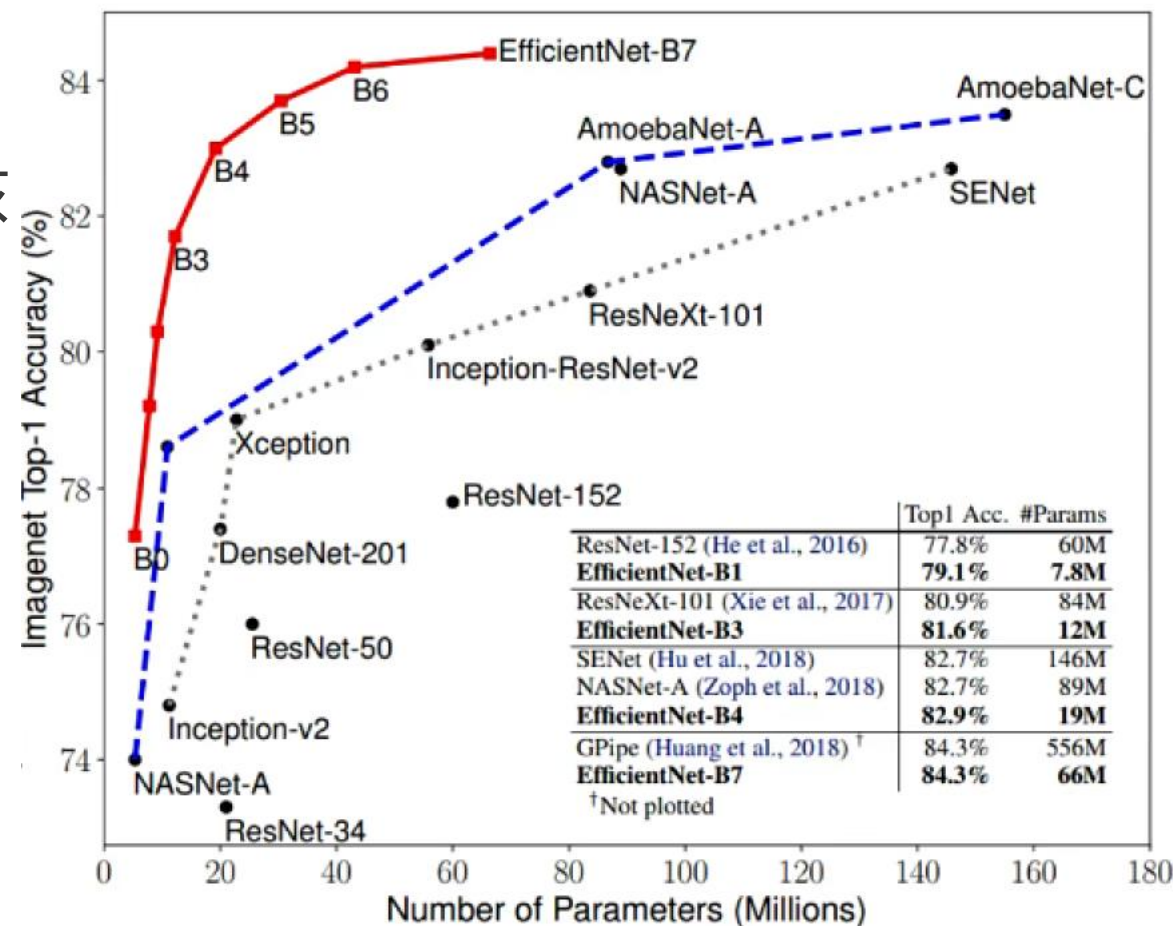
## EfficientNet

EfficientNet是一种基于自动模型缩放的神神经网络结构，由谷歌团队于2019年提出，该模型在图像分类、目标检测和图像分割等任务中取得了不错的结果。

EfficientNet的设计思路来源于模型优化的两个主要思想：

神经网络结构搜索（Neural Architecture Search, NAS）和模型融合。

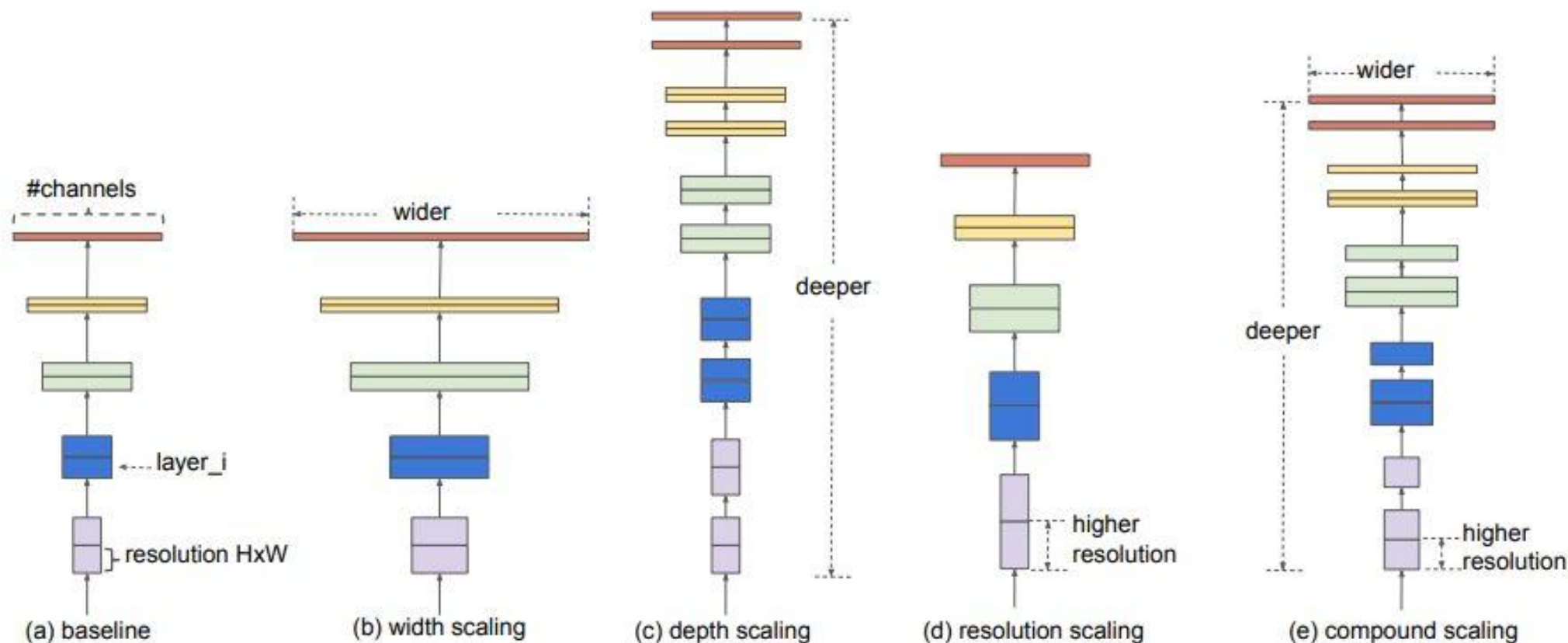
其主要贡献在于开创性地提出了通过均匀缩放（Accurate Scaling）来调整网络深度、宽度和分辨率的方法。



# 3.其它现代网络

23

## EfficientNet



**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.

# 4.卷积神经网络使用技巧

24

**01** 经典网络

**02** 深度残差网络

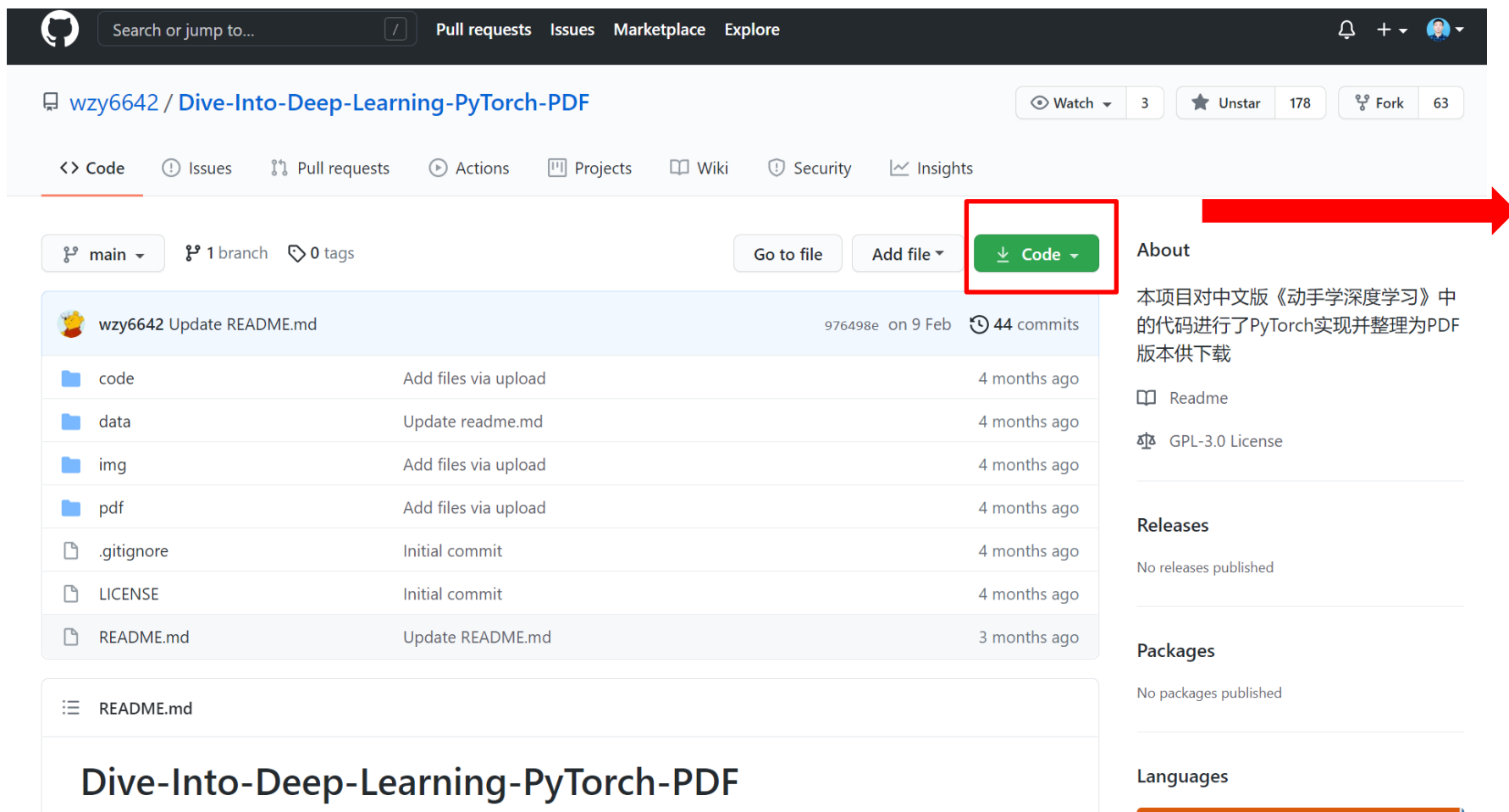
**03** 其它现代网络

**04** 卷积神经网络使用技巧

# 4.卷积神经网络使用技巧

25

## 使用开源的方案



GitHub repository page for `wzy6642 / Dive-Into-Deep-Learning-PyTorch-PDF`.

Repository structure:

File/Folder	Commit Message	Commit Date
code	Add files via upload	4 months ago
data	Update readme.md	4 months ago
img	Add files via upload	4 months ago
pdf	Add files via upload	4 months ago
.gitignore	Initial commit	4 months ago
LICENSE	Initial commit	4 months ago
README.md	Update README.md	3 months ago

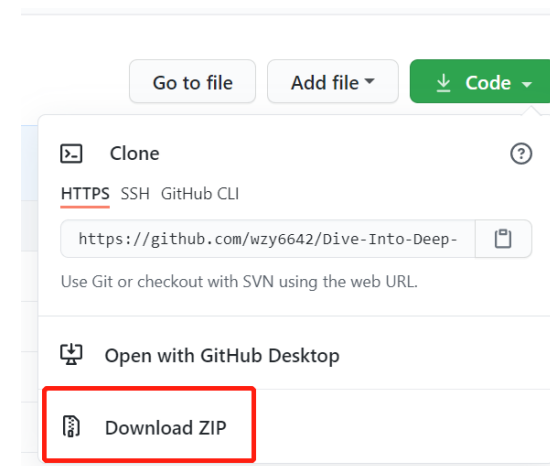
Repository description: 本项目对中文版《动手学深度学习》中的代码进行了PyTorch实现并整理为PDF版本供下载

License: GPL-3.0 License

Releases: No releases published

Packages: No packages published

Languages:



Code dropdown menu options:

- Clone (HTTPS, SSH, GitHub CLI)
- Open with GitHub Desktop
- Download ZIP

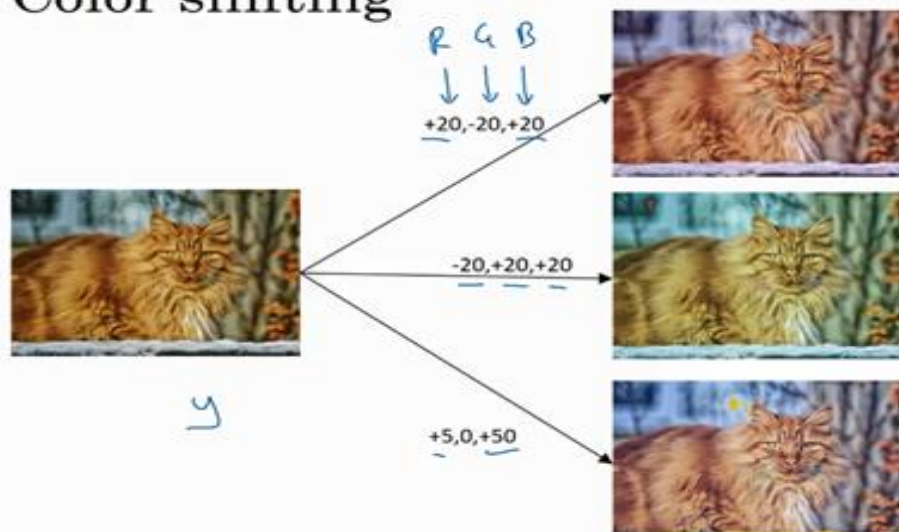


# 4.卷积神经网络使用技巧

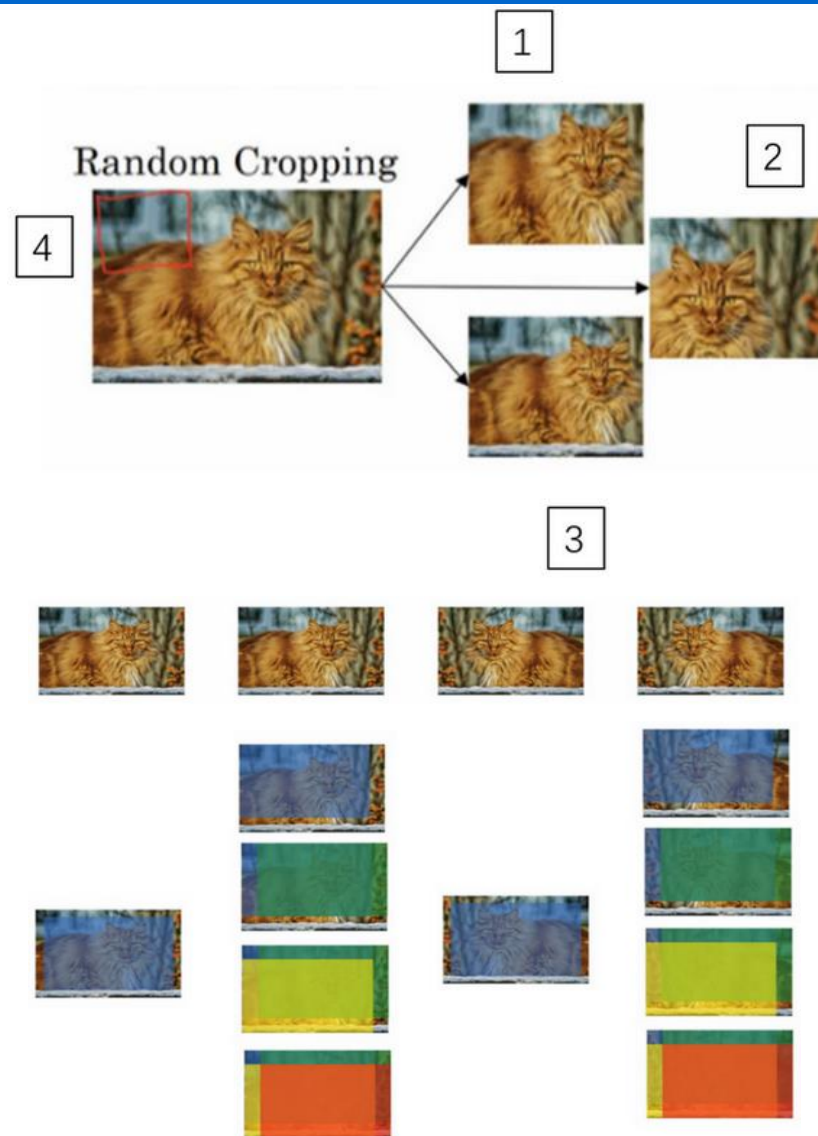
26

## 数据增强

### Color shifting



Advanced:  
PCA  
ml-class.org  
AlexNet paper  
"PCA color  
augmentation."  
R B G





# 4.卷积神经网络使用技巧

27

## 数据增强

```
import torch
from torchvision import transforms

# Random horizontal flip
transforms.RandomHorizontalFlip(p=0.5)

# Random vertical flip
transforms.RandomVerticalFlip(p=0.5)

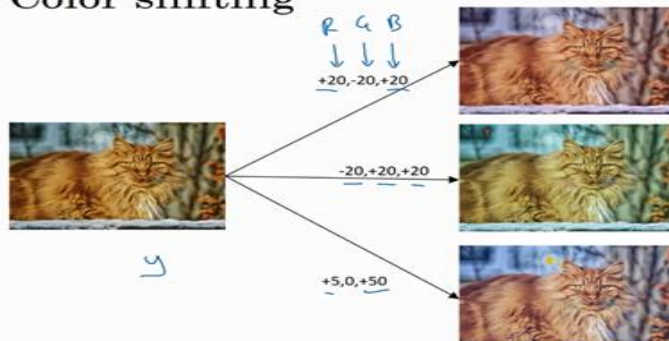
# Random rotation of angle between -degrees and degrees
transforms.RandomRotation(45)

# Random crop and return output of fix size
transforms.RandomCrop(224)

# Resize the image
transforms.Resize(256)

# Apply a linear affine transformation (translate, scale, shear, etc.)
transforms.Affine(matrix=None, resample=None, fillcolor=0)
```

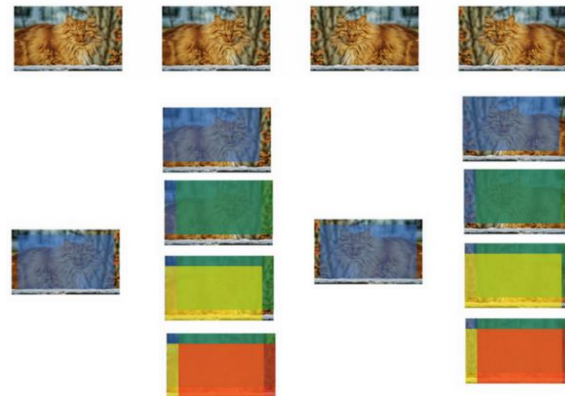
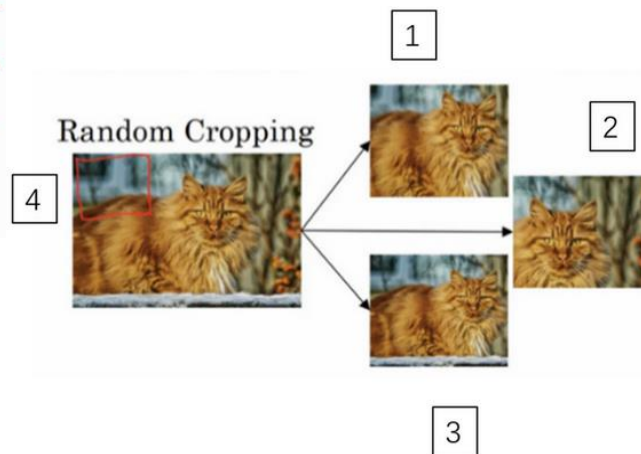
Color shifting



Advanced:

PCA  
ml-class.org  
AlexNet paper  
"PCA color  
augmentation"  
R B G

Random Cropping



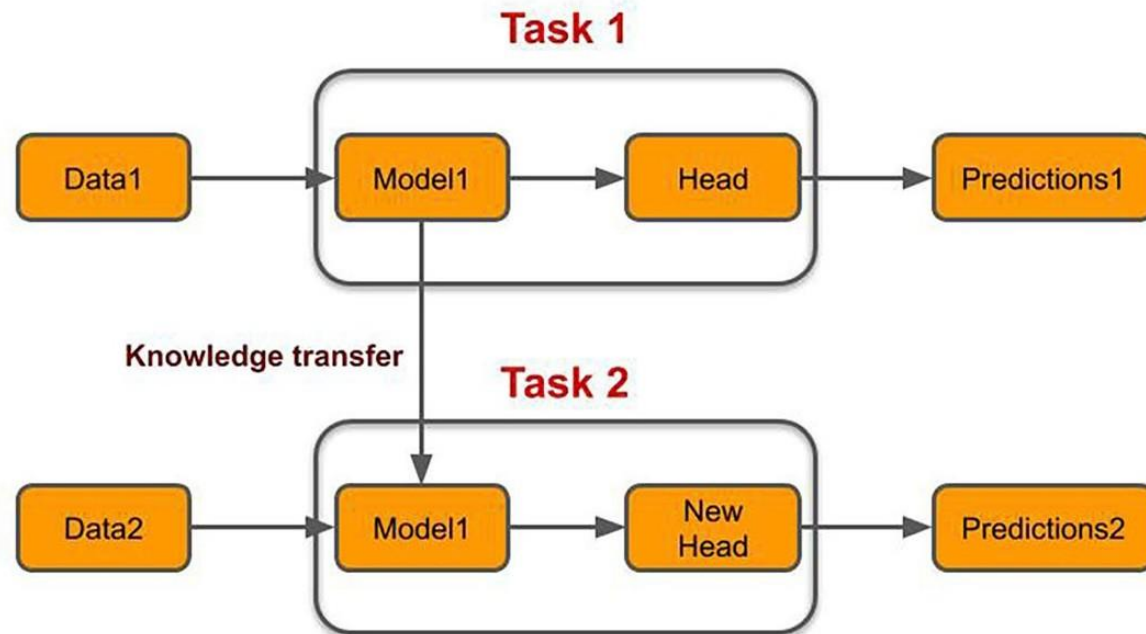
# 4.卷积神经网络使用技巧

28

## 迁移学习

迁移学习 (Transfer Learning) 是把已学训练好的模型参数用作新训练模型的起始参数。迁移学习是深度学习中非常重要和常用的一个策略。

### Transfer Learning



# 4.卷积神经网络使用技巧

29

## 迁移学习步骤

### 1.使用预训练的模型

```
net = models.resnet18(pretrained=True)
```

### 2.冻结模型权重

```
for param in net.parameters(): #遍历每个模型参数  
    param.requires_grad = False #参数梯度为False
```

### 3.替换全连接层

```
# 将最后的全连接层改成十分类  
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")  
net.fc = nn.Linear(512, 10)
```

## 常见模型

- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception
- GoogLeNet
- ShuffleNet
- MobileNet

- IAN GOODFELLOW等, 《深度学习》, 人民邮电出版社, 2017
- Andrew Ng, <http://www.deeplearning.ai>
- LeNet-5 : Gradient-Based Learning Applied to Document Recognition (Yann LeCun, 1998)
- AlexNet : ImageNet Classification with Deep Convolutional Neural Networks (Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012)

- VGG: Very Deep Convolutional Networks for Large-Scale Image Recognition (Karen Simonyan and Andrew Zisserman, 2015)
- GoogLeNet/Inception Net: Going Deeper with Convolutions (Christian Szegedy et al., 2015)
- ResNet: Deep Residual Learning for Image Recognition (Kaiming He et al., 2016)
- DenseNet: Densely Connected Convolutional Networks (Gao Huang et al., 2017)
- MobileNet: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (Andrew G. Howard et al., 2017)
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (Mingxing Tan and Quoc V. Le, 2019)

谢谢!

