Lane Segmentation Week 1

主要内容

- 计算机视觉学习提速绝招
- 项目概述
- ResNet
- ResNeSt

朱利明

人工智能资深讲师，华为云AI社区达人

2002年毕业于中科院研究生院，近二十年软件和算法研发经验，主要研究领域为自动驾驶、人体姿态识别、3D计算机视觉，具有京东、华为等大厂AI资源，在人工智能课程研发和教学有丰富经验，已指导近千位同学进入计算机视觉领域。

微信：aiking2018

入行的最短时间

- 六个月-三个月

学习提速绝招

- 明确目标
- 确定最短学习路径
- 选择正确的参考书
- 必学必会

明确学习目标

- 求职
- 毕设
- 提高水平

确定最短学习路径

- AI是一个开放领域，没有明确的考试大纲
- 有所学有所不学

预备知识

- Python
- Machine Learning
- CNN and Image Classification
- Tensorflow or Pytorch
- English
- Math

你听到的

- 数学要求高（高数、线性代数、概率论）
- 要懂机器学习
- 要精通Python
- 要会Tensorflow/Pytorch
- 要会OpenCV……
- 要会C++

你看到的

$$
\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]
$$

$$
+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]
$$

$$
+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2
$$

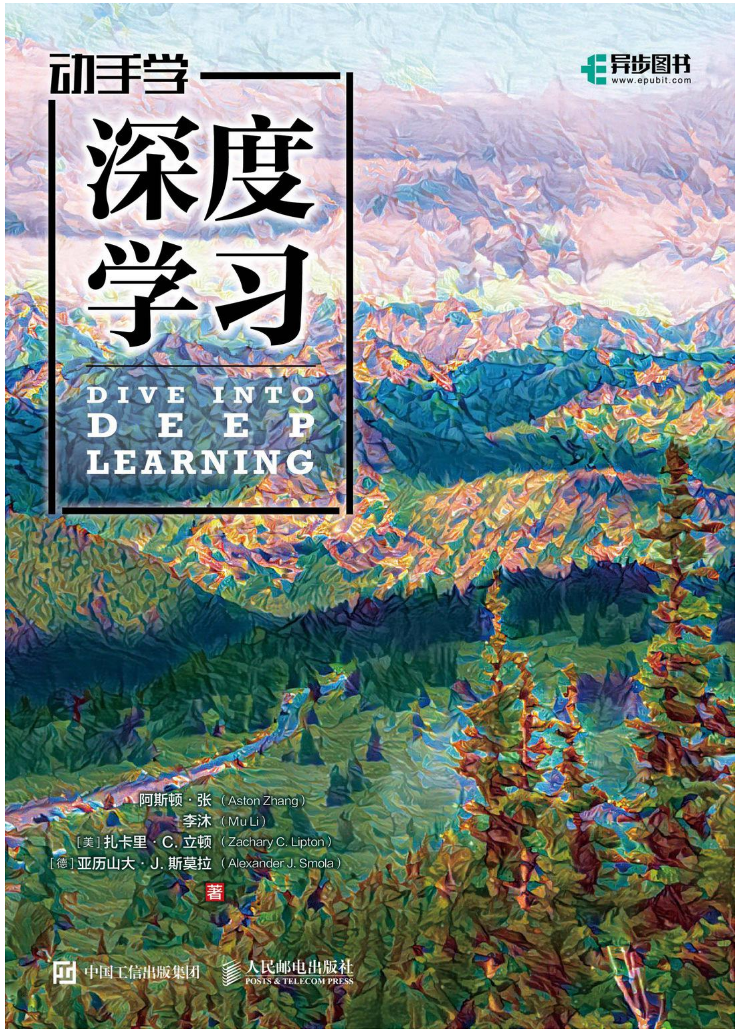
$$
+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2
$$

$$
+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
$$

其实可以这样

- 很少的数学知识
- 很少的机器学习
- 基本的Python

选择正确的参考书


异步图书
www.epubit.com
动手学
深度学习
DIVE INTO DEEP LEARNING
阿斯顿·张（Aston Zhang）
李沐（Mu Li）
[美] 扎卡里·C. 立顿（Zachary C. Lipton）
[德] 亚历山大·J. 斯莫拉（Alexander J. Smola）
著
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

Framework

TensorFlow

PyTorch

飞桨

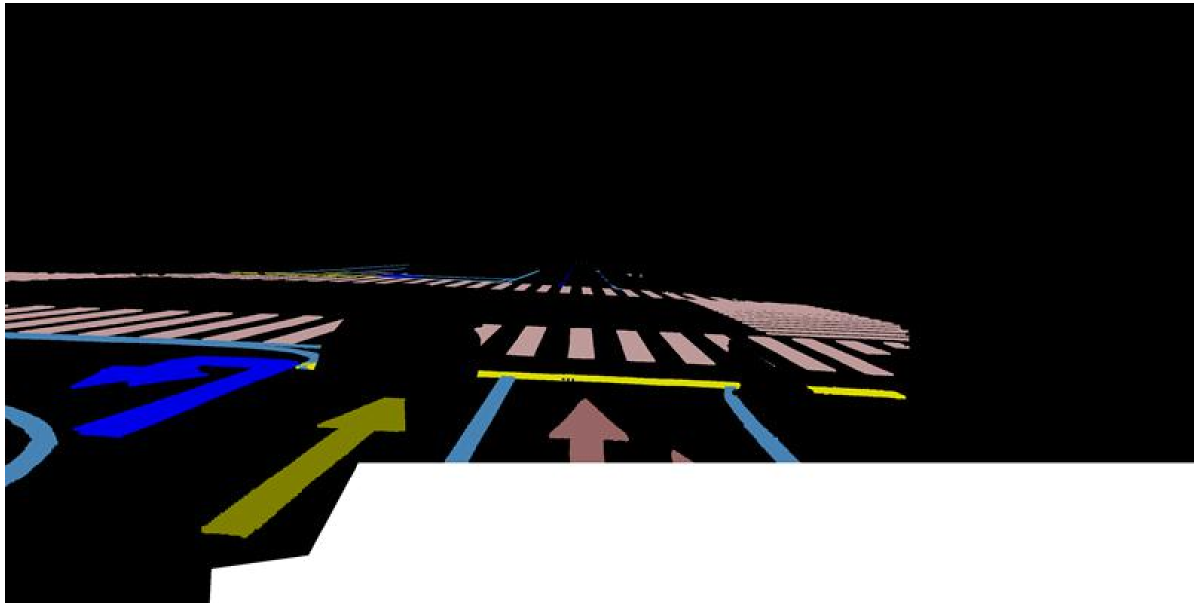
天元 MegEngine

MindSpore

Jittor

PyTorch

Lane Segmentation



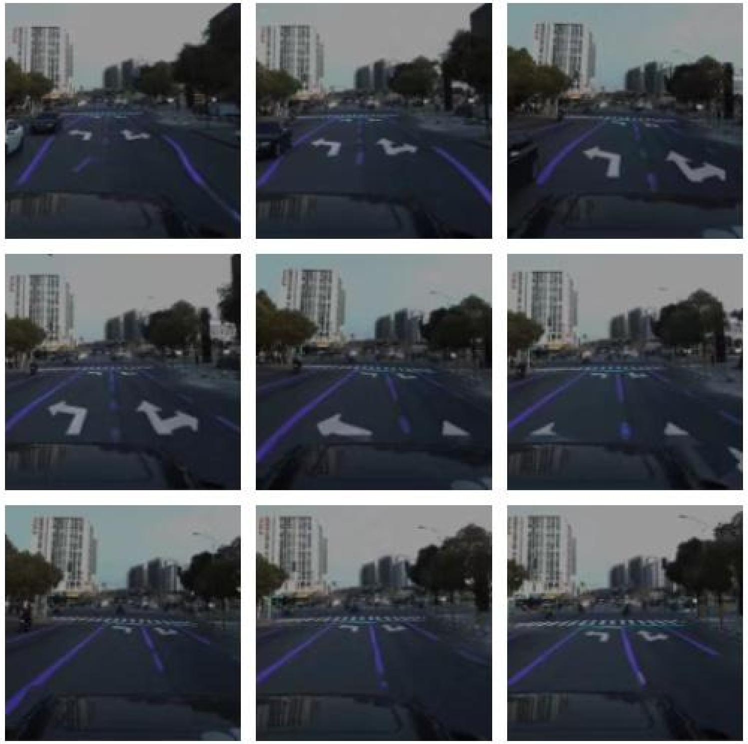
项目设计原则

- 紧扣前沿热门领域：无人驾驶
- 既重视特定项目实战
- 又重视算法的普适性
- 关注领域最新发展

Make your hand dirty

Howard Chow

✌️第一个完整手撸的工程。深度神经网络真的太神奇了，它咋知道那就是车道线呀😂



1小时前

♡ Alan Wang, AI-King

Rongfan Leo: 效果还挺好的

Lane Segmentation

- 骨干网络：ResNet
- 语义分割进入深度学习时代：FCN
- 简单实用：U-Net
- Google出品：Deeplab v3+

Lane Segmentation

- 项目框架实战
- 模型训练实战
- 模型部署实战
- 图像分割最新发展：全景分割

Notes

Backbone Network

- VGG
- ResNet

VGG


3x3 conv，64
3x3 conv，64
pool/2
3x3 conv，128
3x3 conv，128
pool/2
3x3 conv，256
3x3 conv，256
3x3 conv，256
pool/2
3x3 conv，512
3x3 conv，512
3x3 conv，512
pool/2
3x3 conv，512
3x3 conv，512
3x3 conv，512
pool/2
fc 4096
fc 4096
fc 1000
Size:224
Size:112
Size:56
Size:28
Size:14
Size:7

VGG

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
input (224 × 224 RGB image)					
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 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 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 conv3-256 **conv3-256**
maxpool					
conv3-512 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 **conv3-512**
maxpool					
conv3-512 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 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

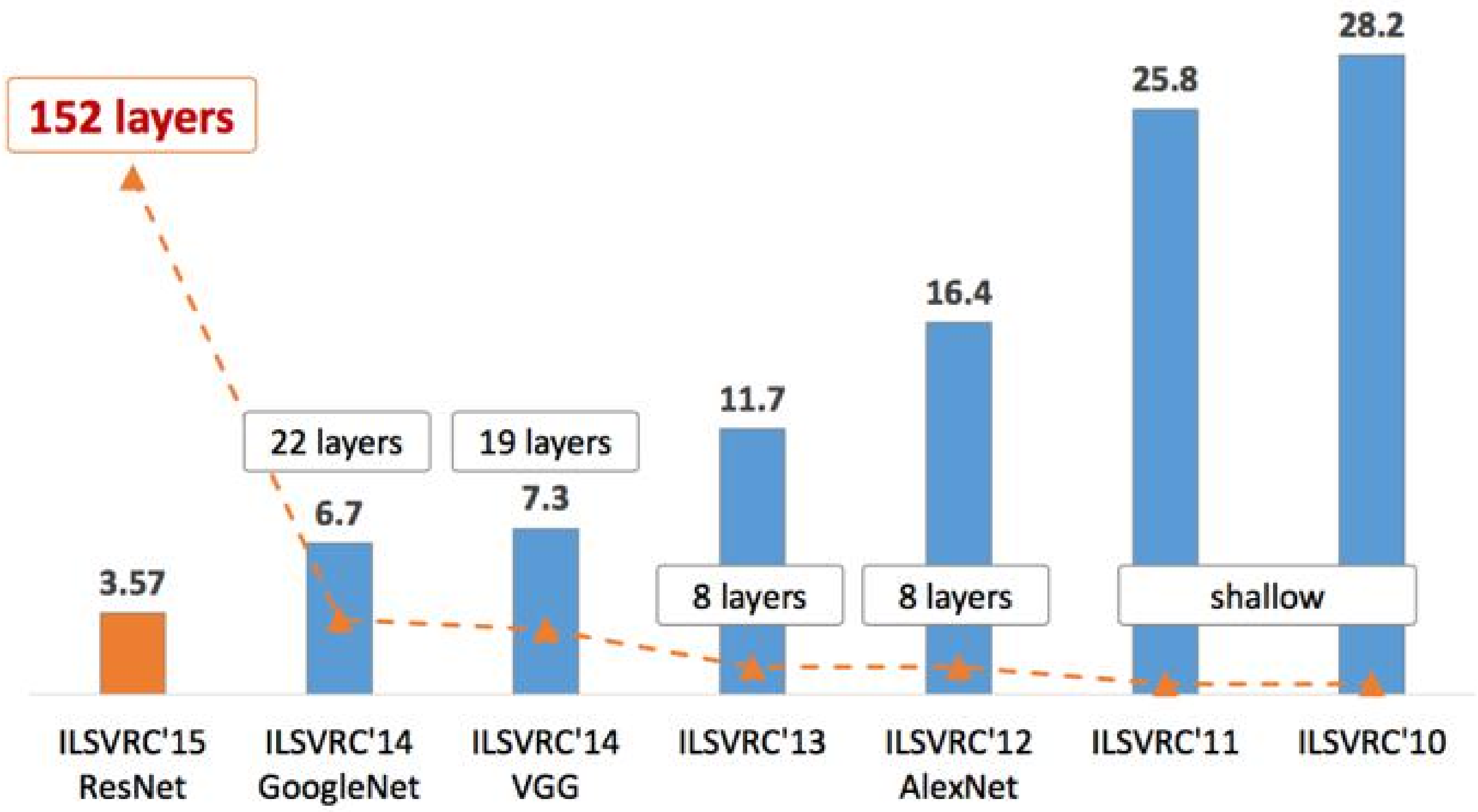
VGG-BN

```
def make_layers(cfg, batch_norm=False):
    layers = []
    in_channels = 3
    for v in cfg:
        if v == 'M':
            layers += [nn.MaxPool2d(kernel_size=2, stride=2)]
        else:
            conv2d = nn.Conv2d(in_channels, v, kernel_size=3, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
            else:
                layers += [conv2d, nn.ReLU(inplace=True)]
            in_channels = v
    return nn.Sequential(*layers)
```

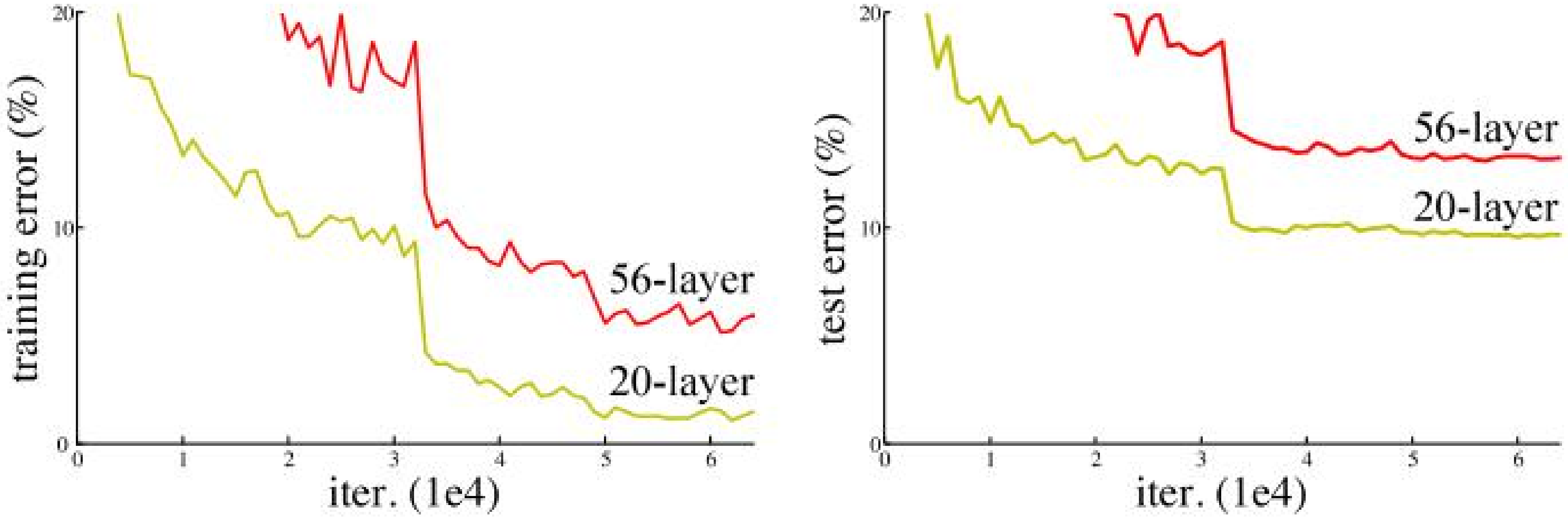
参数初始化

```
def _initialize_weights(self):
    for m in self.modules():
        if isinstance(m, nn.Conv2d):
            nn.init.kaiming_normal_(m.weight, mode='fan_out', nonlinearity='relu')
            if m.bias is not None:
                nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.BatchNorm2d):
            nn.init.constant_(m.weight, 1)
            nn.init.constant_(m.bias, 0)
        elif isinstance(m, nn.Linear):
            nn.init.normal_(m.weight, 0, 0.01)
            nn.init.constant_(m.bias, 0)
```

ResNet


152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10

Deep Network Degradation



Notes

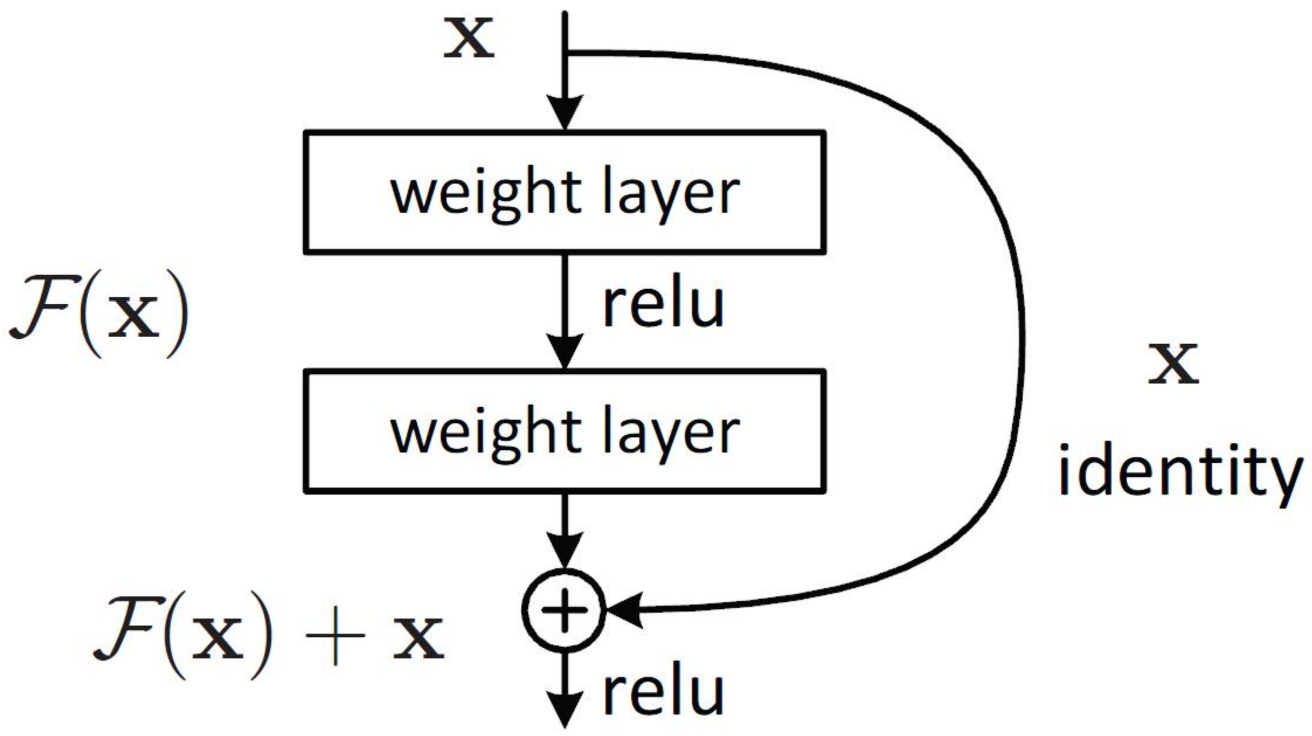
Residual Learning

$$\mathcal{H}(\mathbf{x})$$

$$\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}.$$

$$\mathcal{H}(\mathbf{x}) \quad \mathcal{F}(\mathbf{x}) + \mathbf{x}.$$

Residual Learning



Notes

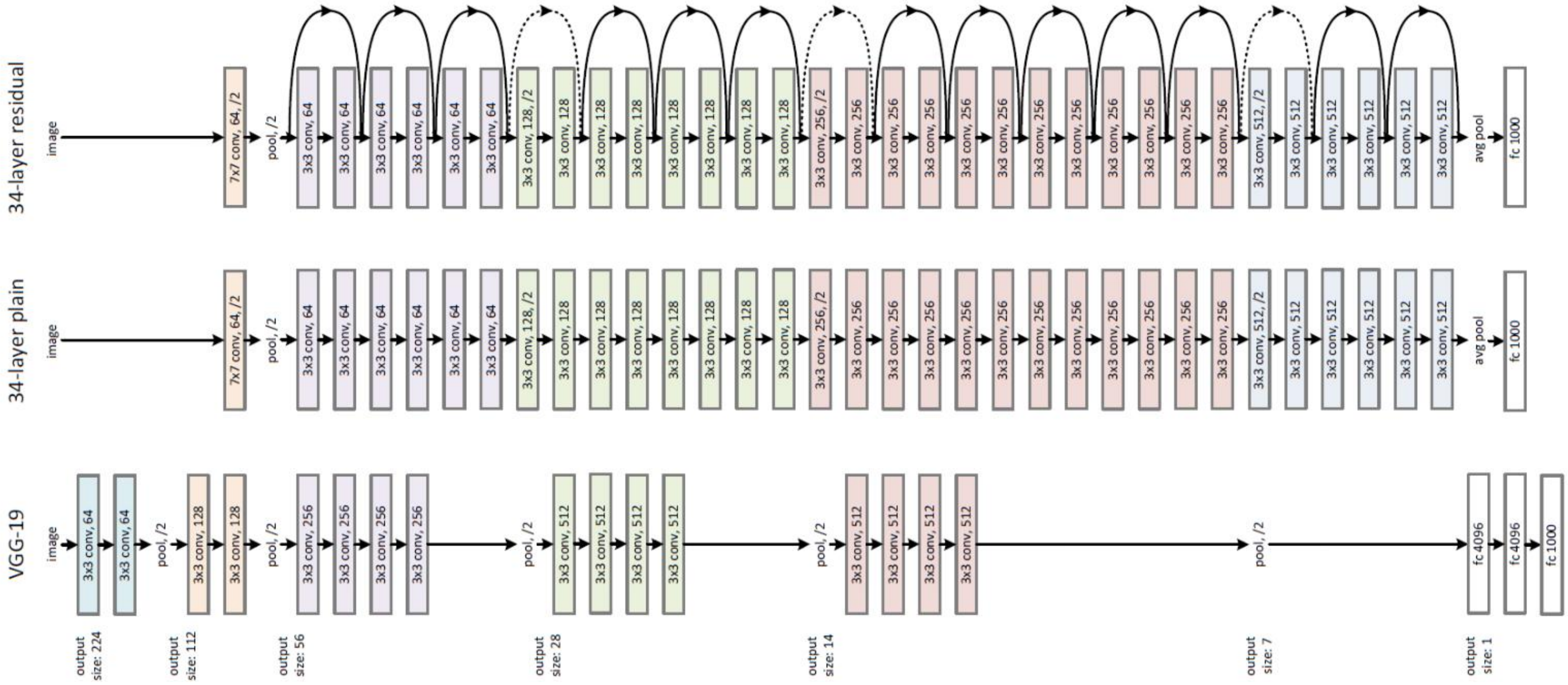
Identity Mapping Shortcut

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}.$$

- 1x1 Convolution

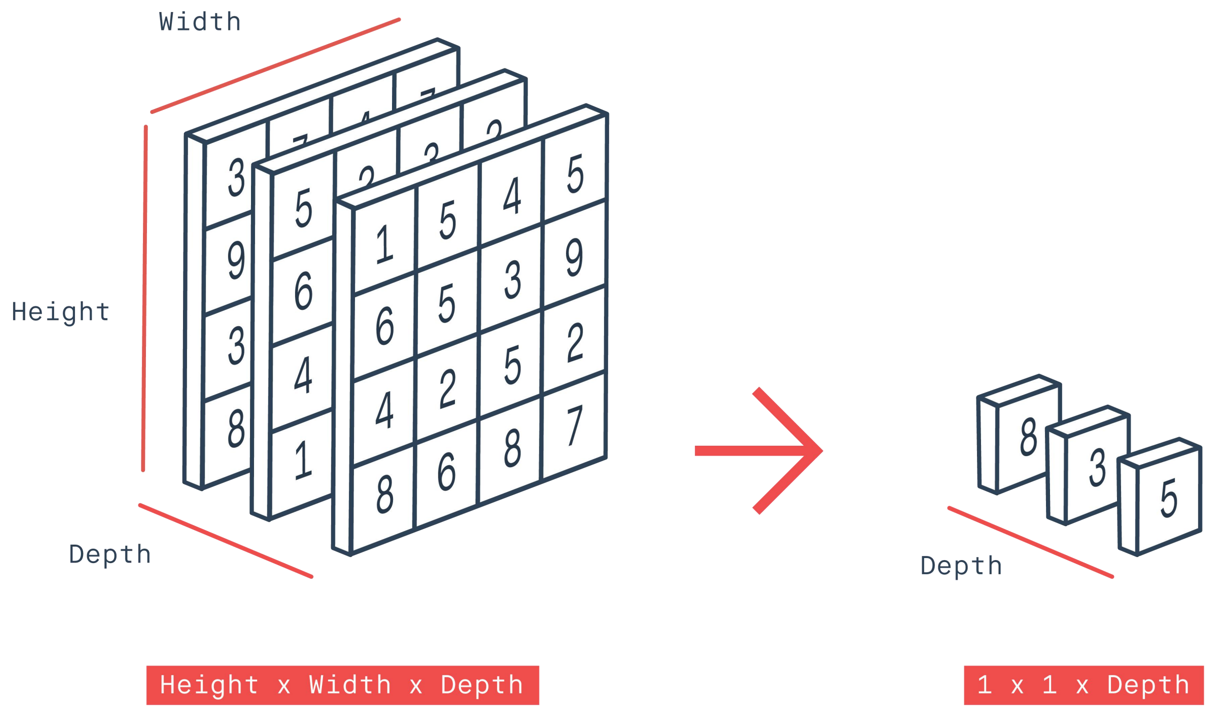
ResNet


34-layer residual
34-layer plain
VGG-19
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000
fc 4096
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

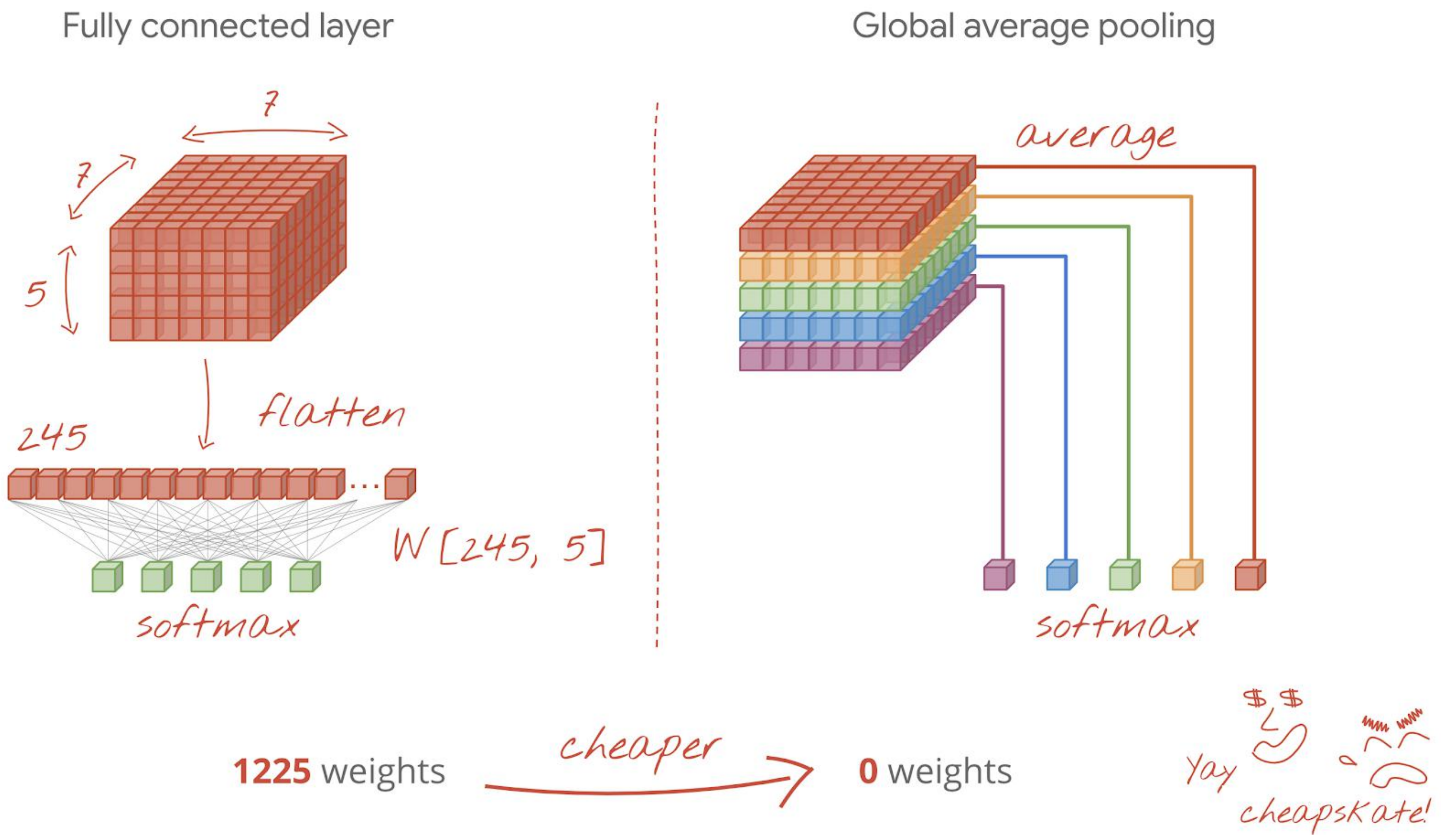
ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Global Average Pooling

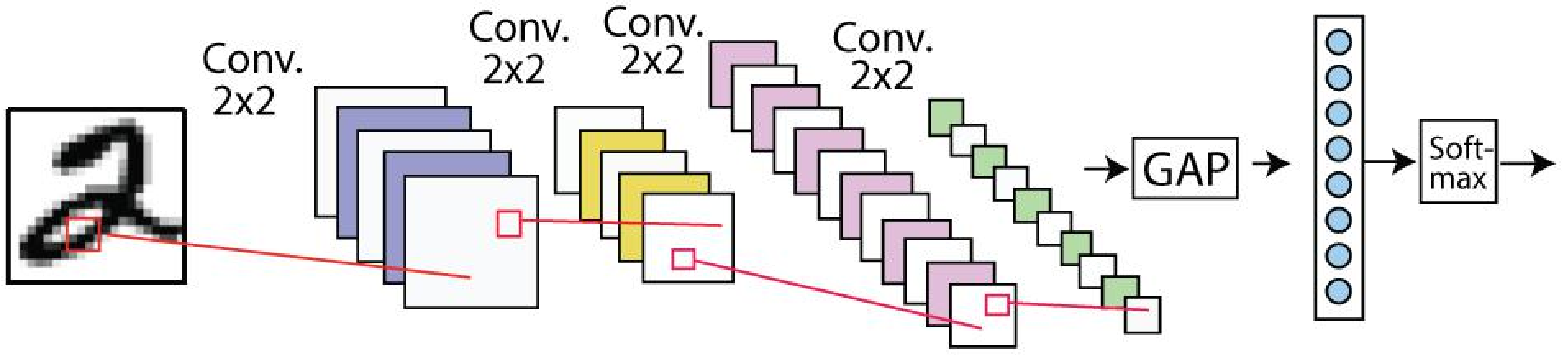


Global Average Pooling

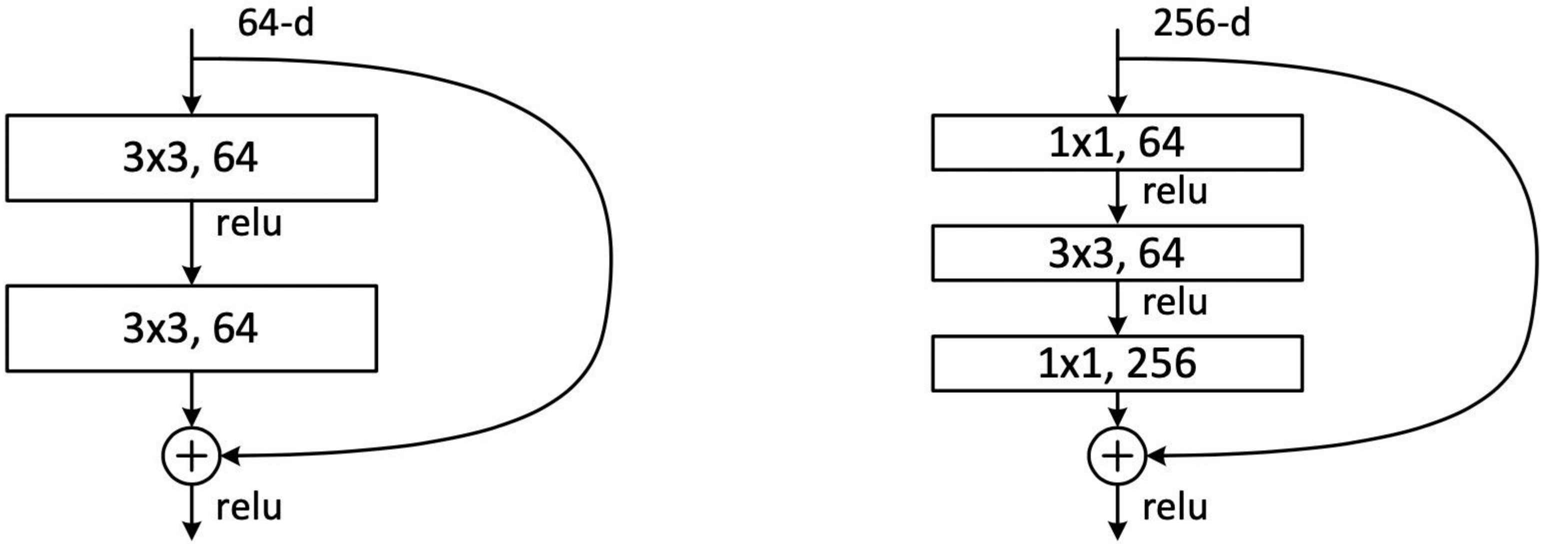


Global Average Pooling

- torch.nn.AdaptiveAvgPool2d(output_size)
- tf.keras.layers.GlobalAvgPool2D



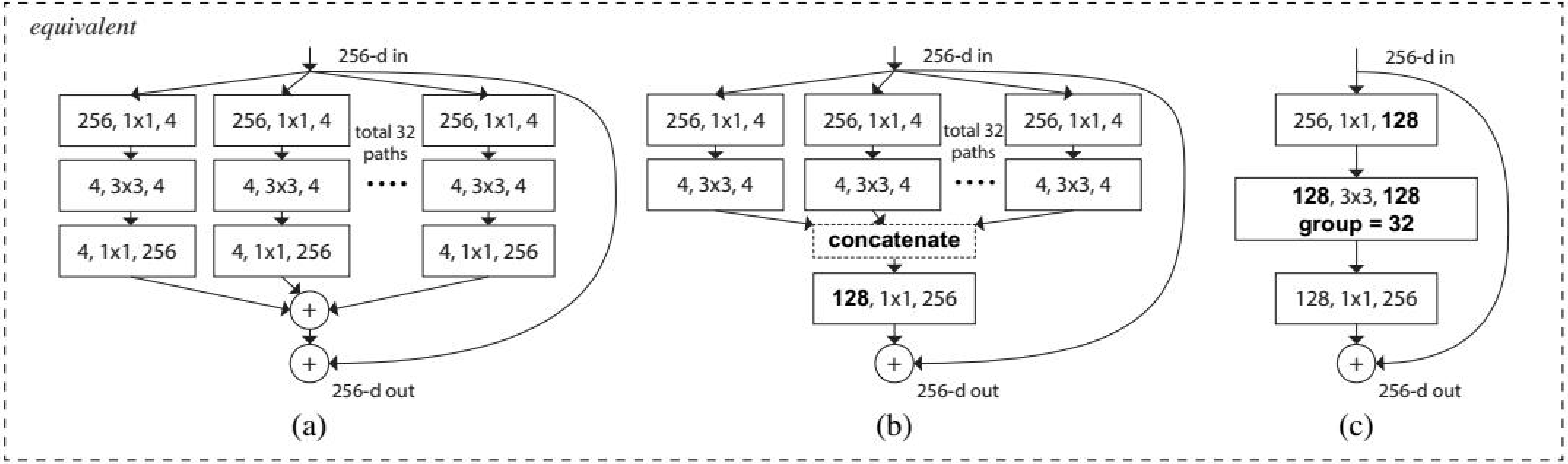
BottleNeck


64-d
3x3, 64
relu
3x3, 64
relu
256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu

ResNet新发展

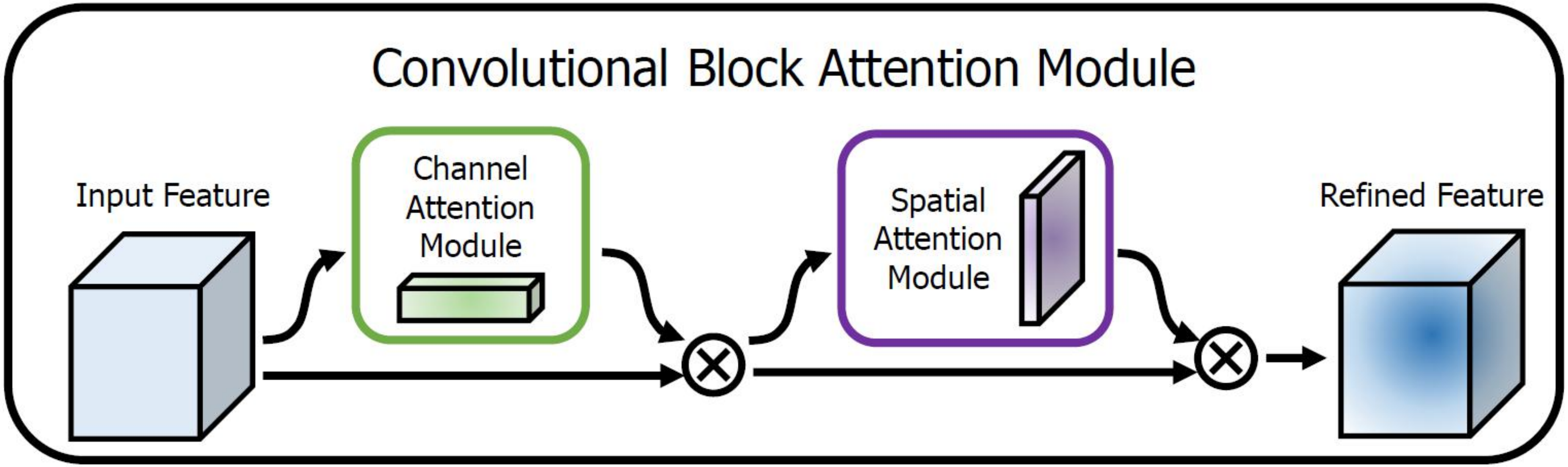
- ResNeXt
- ResNeXt-Attention
- ResNeXt WSL
- ResNeSt

ResNeXt



equivalent
256-d in
256, 1x1, 4
4, 3x3, 4
4, 1x1, 256
total 32 paths
256-d out
(a)
concatenate
128, 1x1, 256
(b)
128, 3x3, 128
group = 32
128, 1x1, 256
(c)

ResNeXt-Attention



SENet

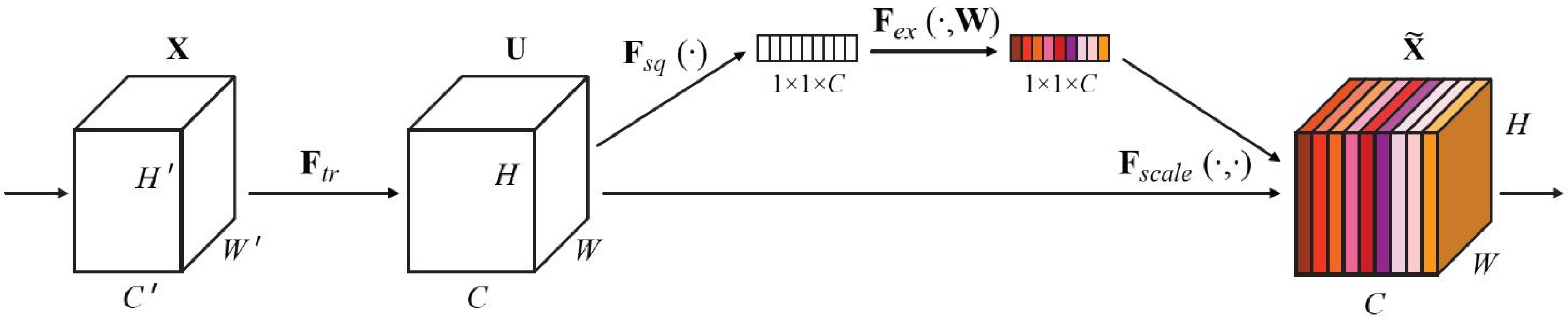


Figure 1: A Squeeze-and-Excitation block.

ResNeXt WSL

PyTorch

Get Started Ecosystem Mobile Blog Tutorials Docs Resources GitHub

View on Github

Open on Google Colab

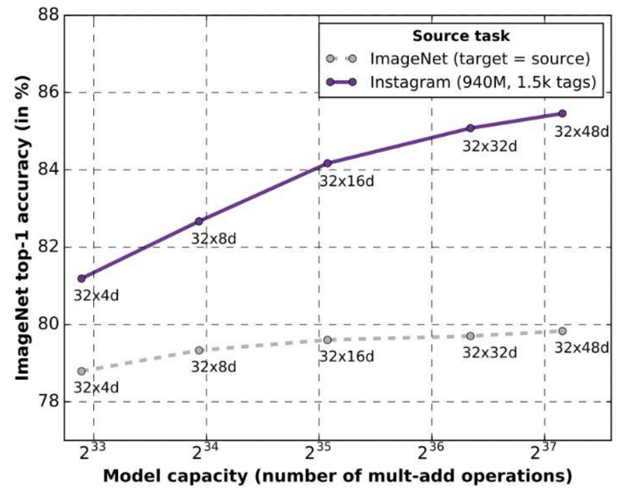


Fig. 5: Classification accuracy on val-IN-1k using ResNeXt-101 32×{4, 8 16, 32, 48}d with and without pretraining on the IG-940M-1.5k dataset.

```
import torch
model = torch.hub.load('facebookresearch/WSL-Images', 'resnext101_32x8d_wsl')
# or
# model = torch.hub.load('facebookresearch/WSL-Images', 'resnext101_32x16d_wsl')
# or
# model = torch.hub.load('facebookresearch/WSL-Images', 'resnext101_32x32d_wsl')
# or
#model = torch.hub.load('facebookresearch/WSL-Images', 'resnext101_32x48d_wsl')
model.eval()
```

All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape `(3 x H x W)`, where `H` and `W` are expected to be at least `224`. The images have to be loaded in to a range of `[0, 1]` and then normalized using `mean = [0.485, 0.456, 0.406]` and `std = [0.229, 0.224, 0.225]`.

ResNeSt

ResNeSt: Split-Attention Networks

Hang Zhang, Chongruo Wu*, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola

Amazon, University of California, Davis*

{hzaws,chongrwu,zhongyue,yzaws,haibilin,zhiz,ysunmzn, htong,jonasmue,manmatha,mli,smola}@amazon.com

Abstract. While image classification models have recently continued to advance, most downstream applications such as object detection and semantic segmentation still employ ResNet variants as the backbone net-

Notes

课程总结

- 计算机视觉学习提速绝招
- 项目概述
- ResNet
- ResNeXt

重难点

- ResNet

参考资料

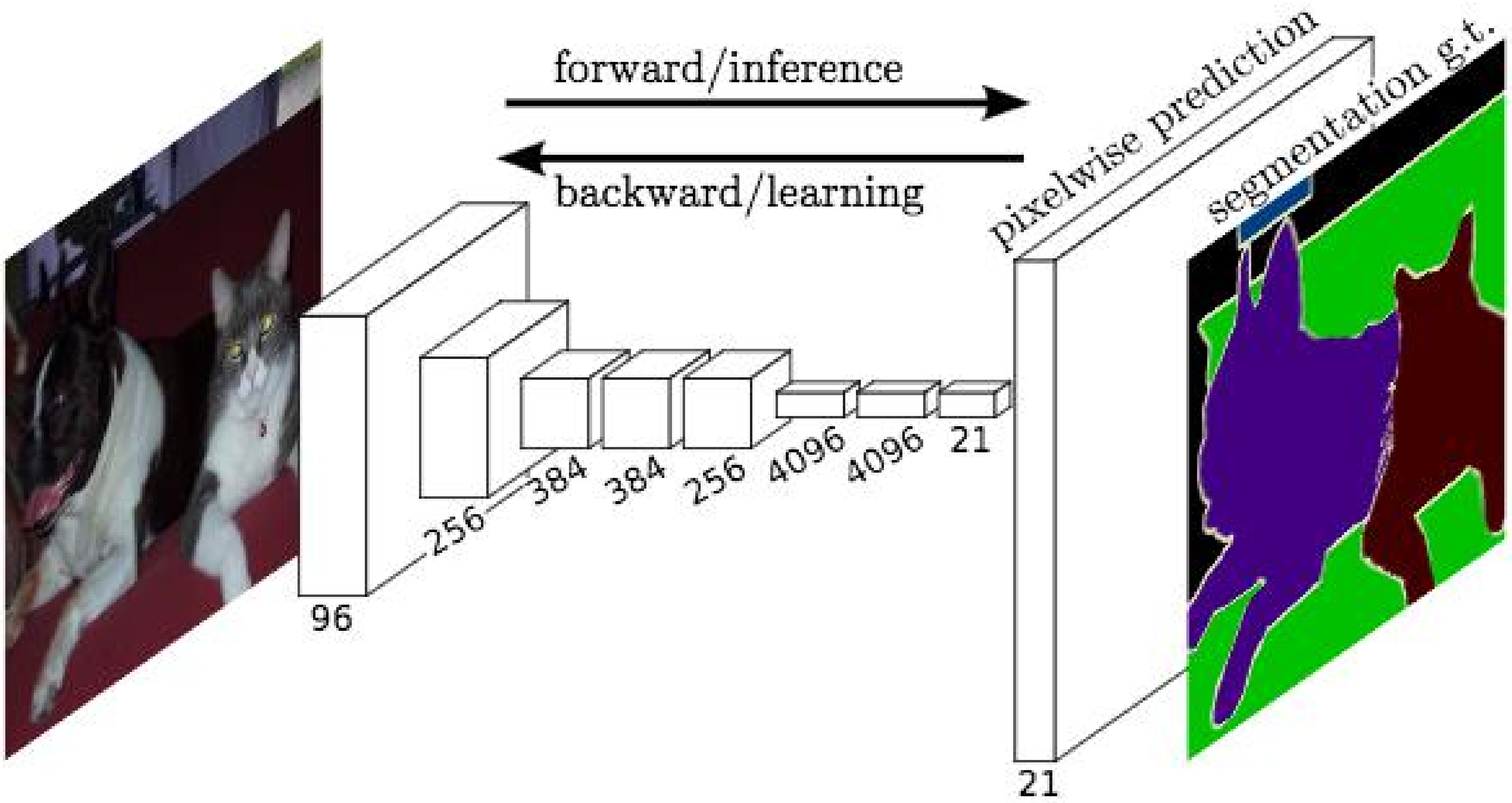
- 动手学深度学习
- Very deep convolutional networks for large-scale image recognition
- Deep Residual Learning for Image Recognition
- Aggregated Residual Transformations for Deep Neural Networks

ResNeSt: Split-Attention Networks

课程作业

- 使用Pytorch逐行实现ResNet-152
- 加载Pytorch官方预训练参数(ImageNet-1000)
- 实现图像识别功能

Week 2： FCN



forward/inference
backward/learning
pixelwise prediction
segmentation g.t.
96
256
384
384
256
4096
4096
21
21