计算机视觉

张健

数字媒体研究中心

信息工程学院

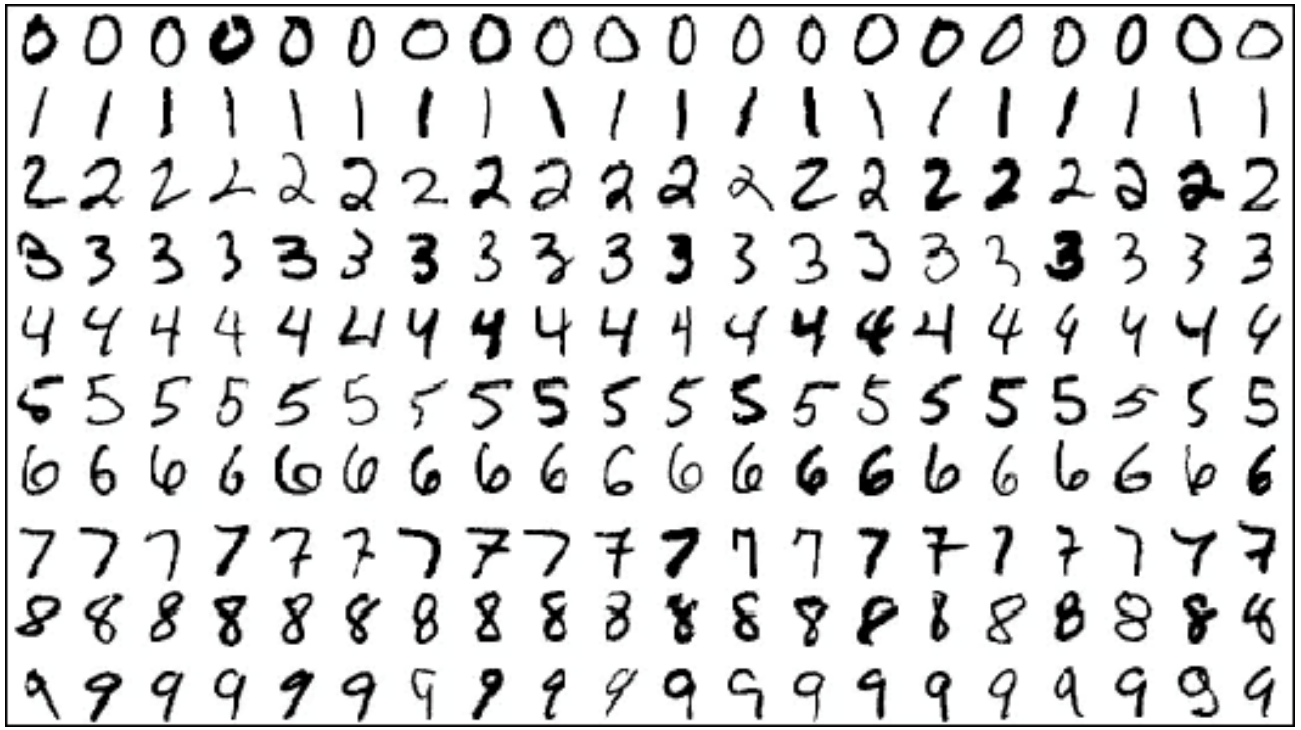
北京大学深圳研究生院

2021.11.03

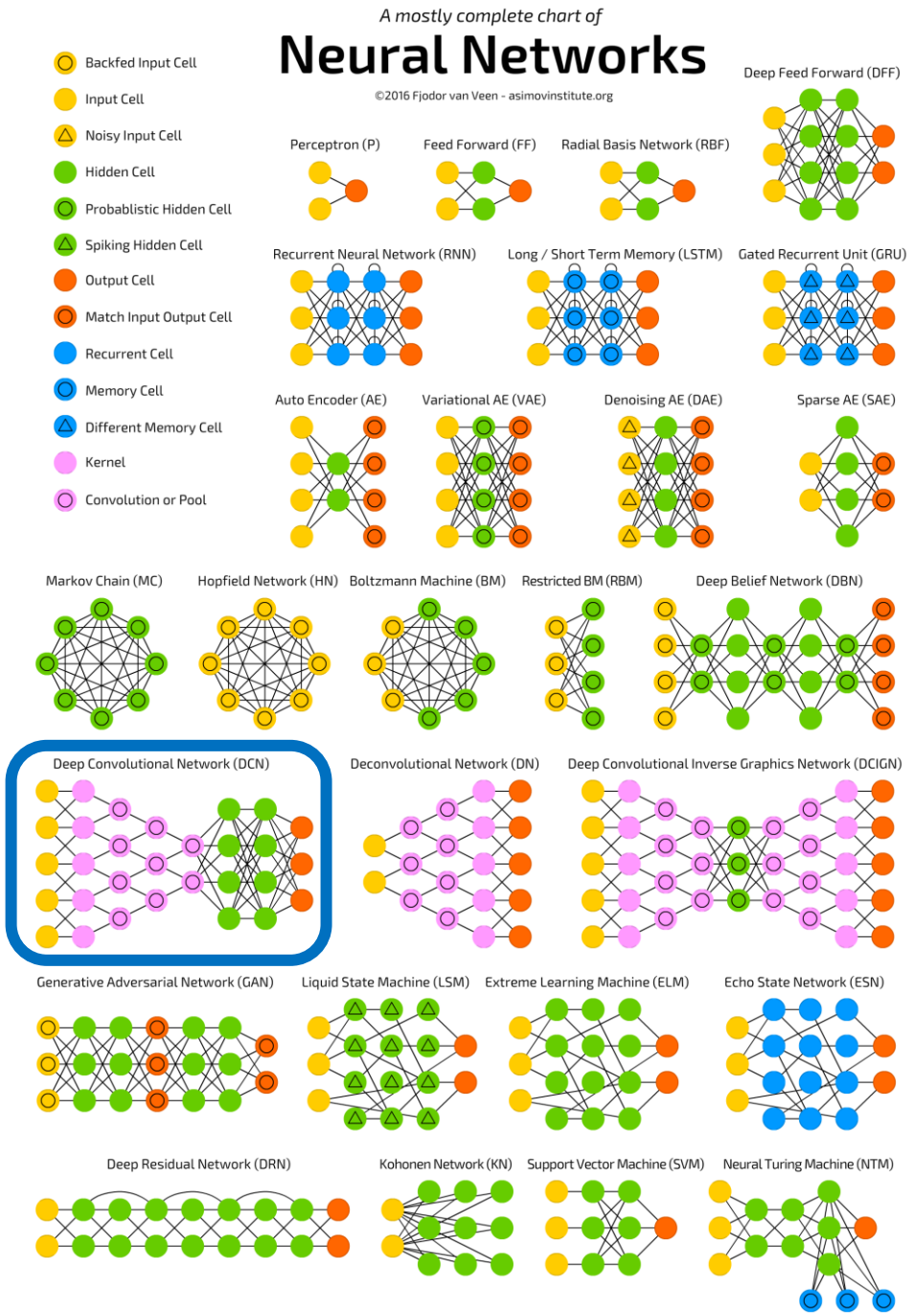
作业情况

北京大学 PEKING UNIVERSITY

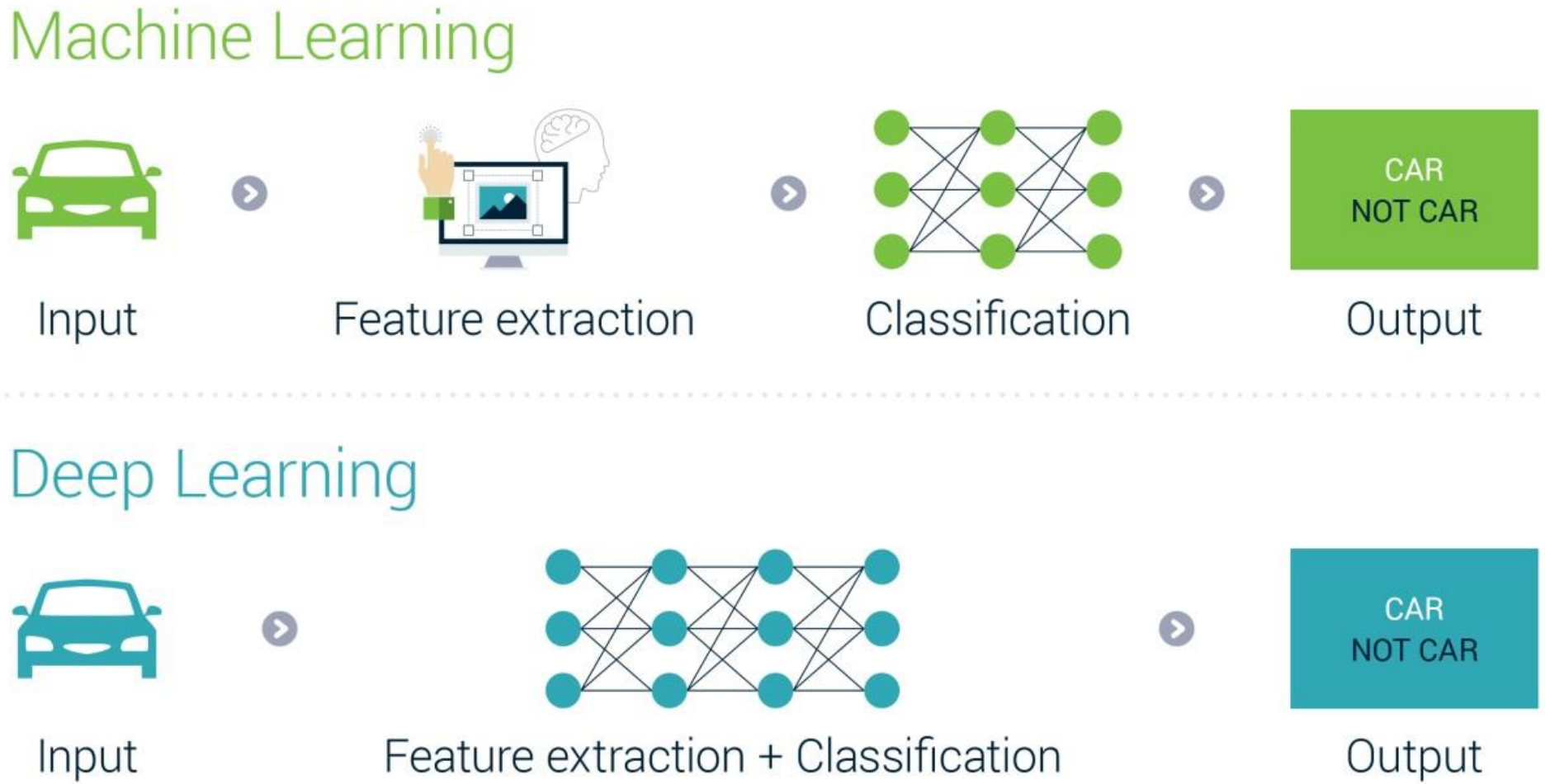
- 在W6_MNIST_FC.ipynb基础上，增加卷积层结构/增加dropout或者BN技术等，训练出尽可能高的MNIST分类效果。



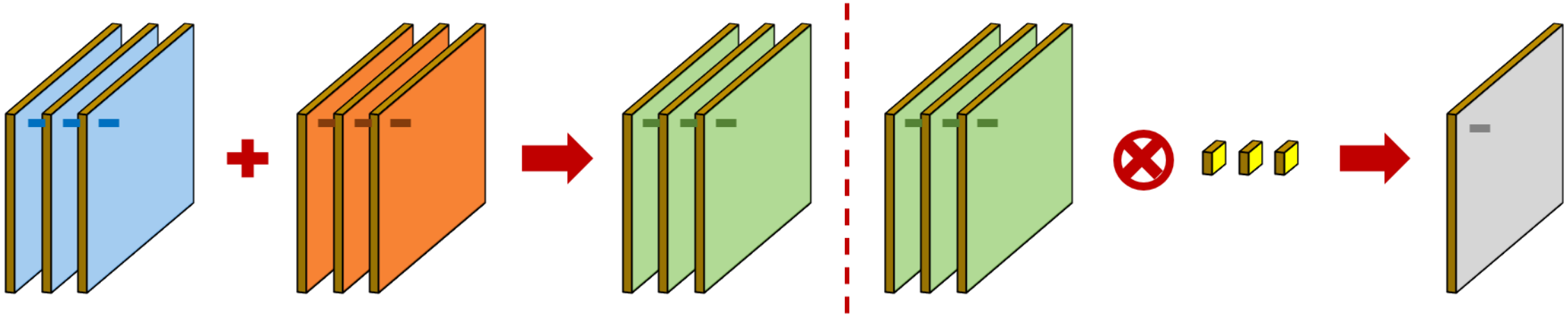
典型卷积神经网络


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

典型卷积神经网络

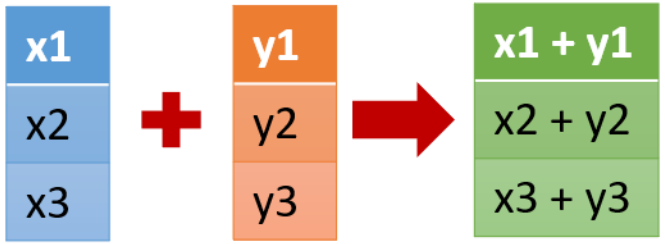

Machine Learning
CAR
NOT CAR
Input
Feature extraction
Classification
Output
Deep Learning
CAR
NOT CAR
Input
Feature extraction + Classification
Output

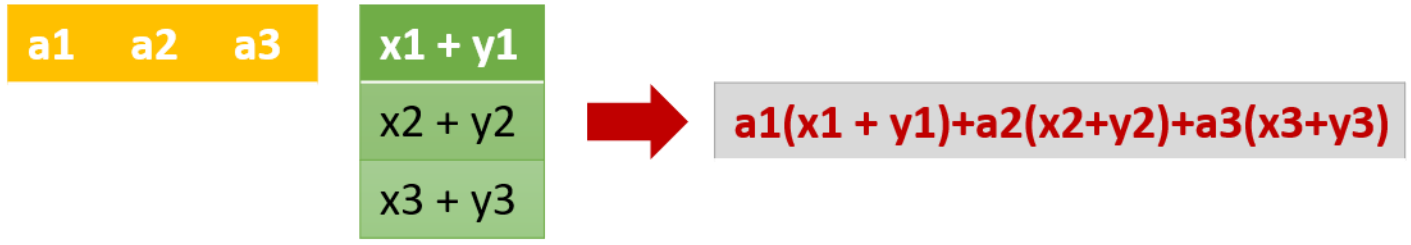
Concatenation V.S. Summation



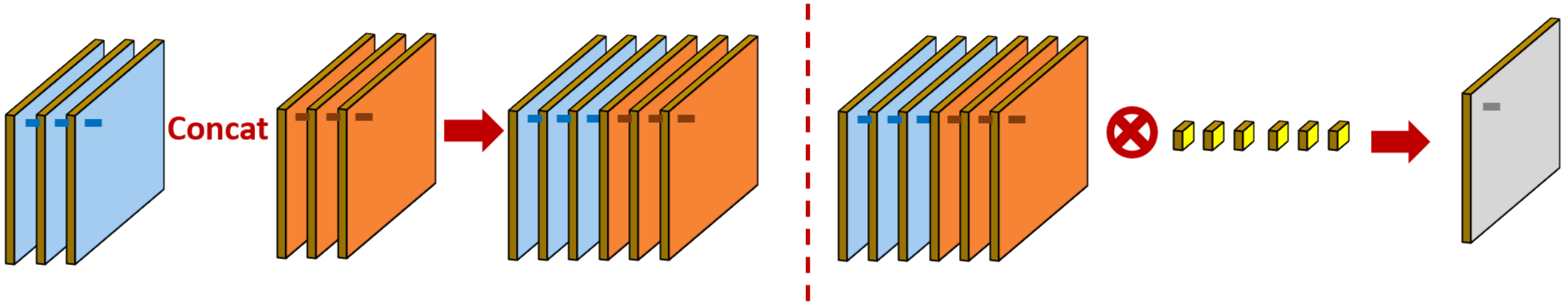
Summation

1*1 convolution

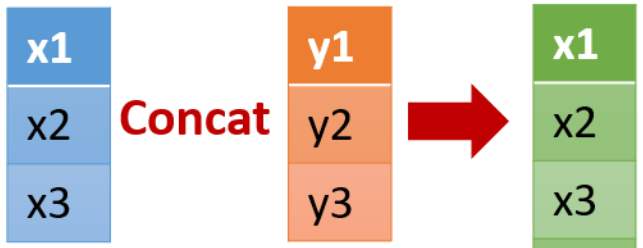




Concatenation V.S. Summation



Concatenation

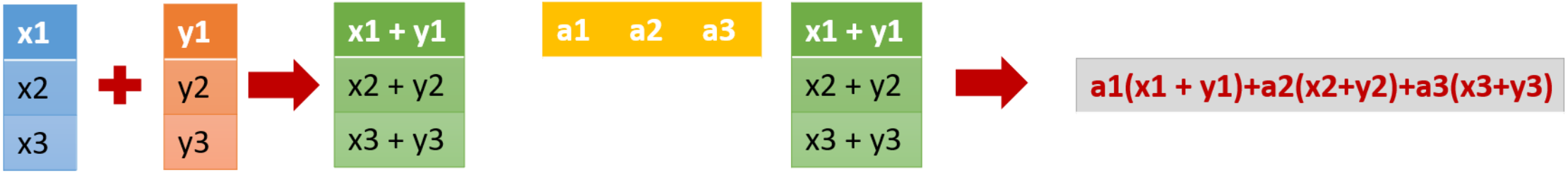


1*1 convolution

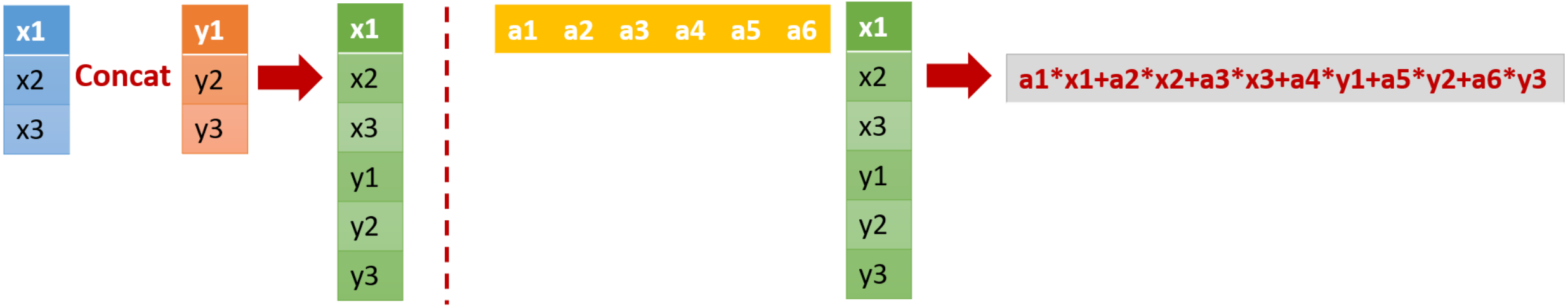
$a1*x1+a2*x2+a3*x3+a4*y1+a5*y2+a6*y3$

Concatenation V.S. Summation

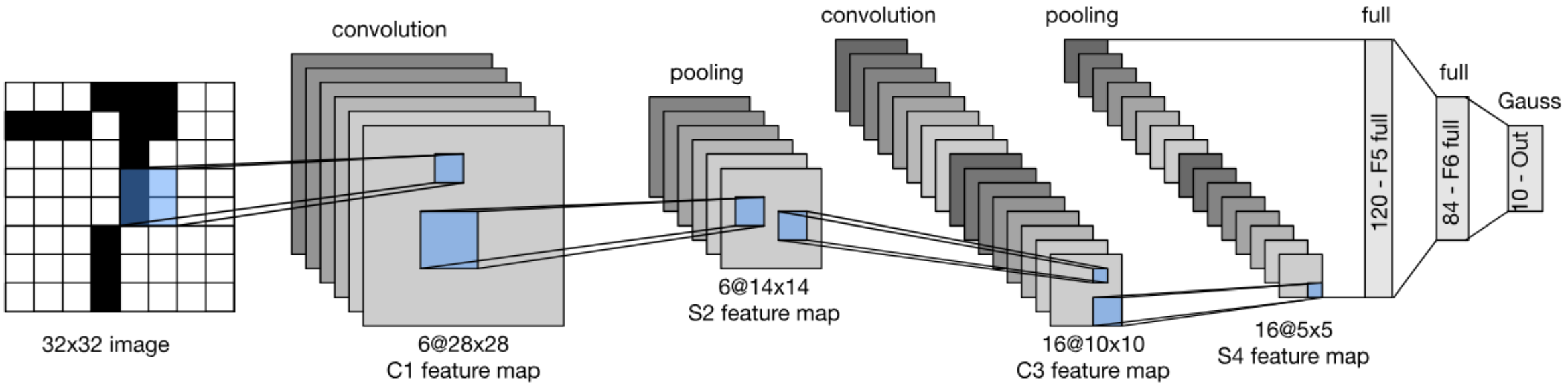
Summation; 1*1 Convolution



Concatenation; 1*1 Convolution



LeNet



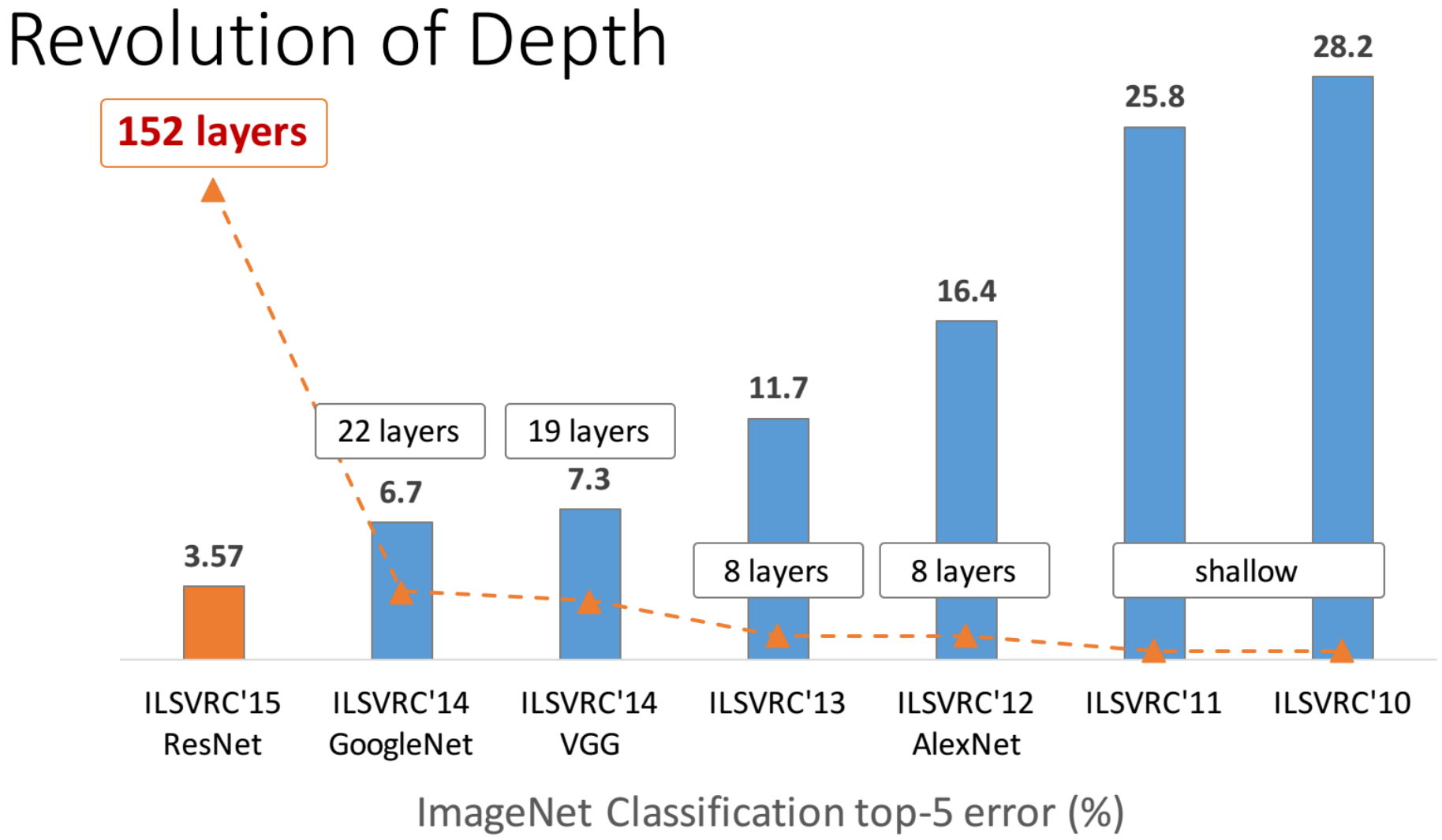

```
Net(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html#sphx-glr-beginner-blitz-neural-networks-tutorial-py

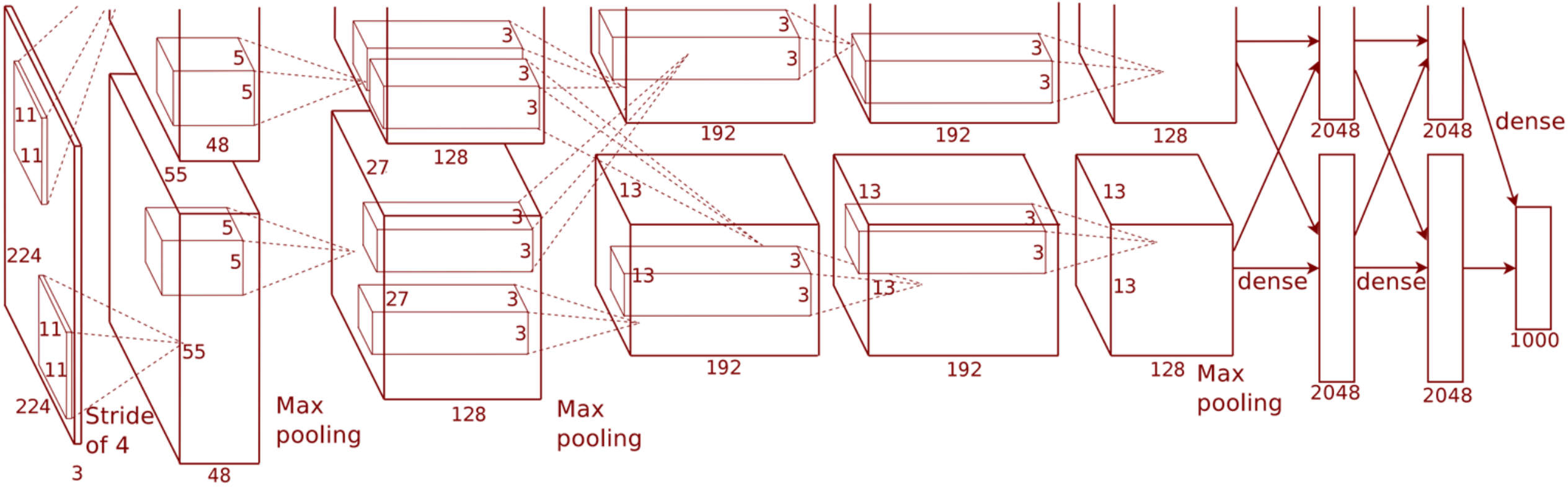
ImageNet 数据集 ——2010

图像	自然物体的彩色图像	手写数字的灰色图像
尺寸	469 x 387	28 x 28
# 例子	1.2 M	60 K
# 类	1,000	10

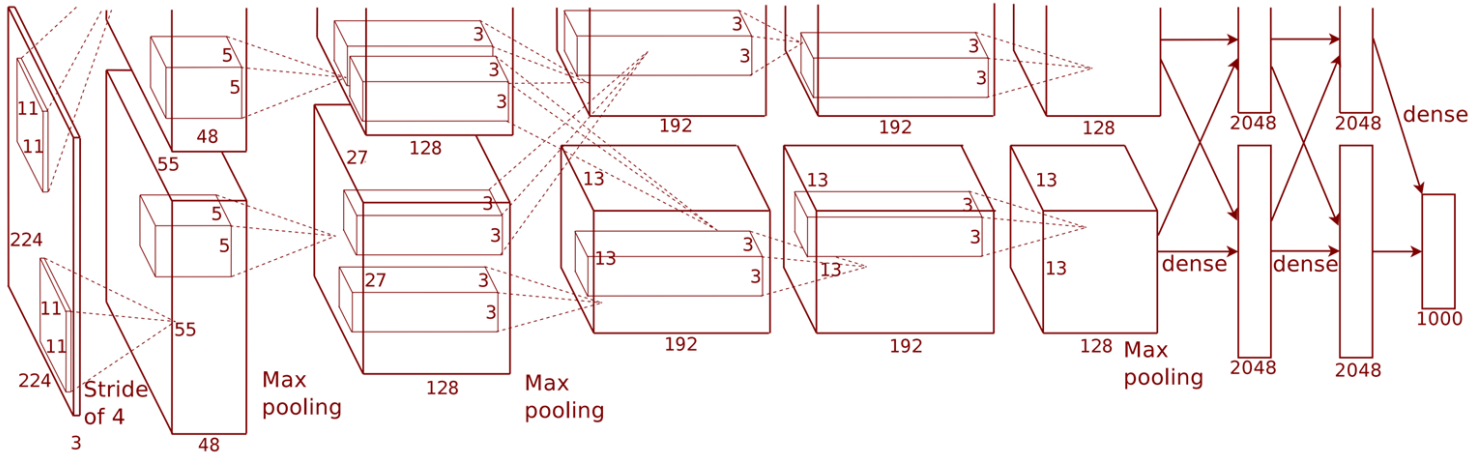
深度革命

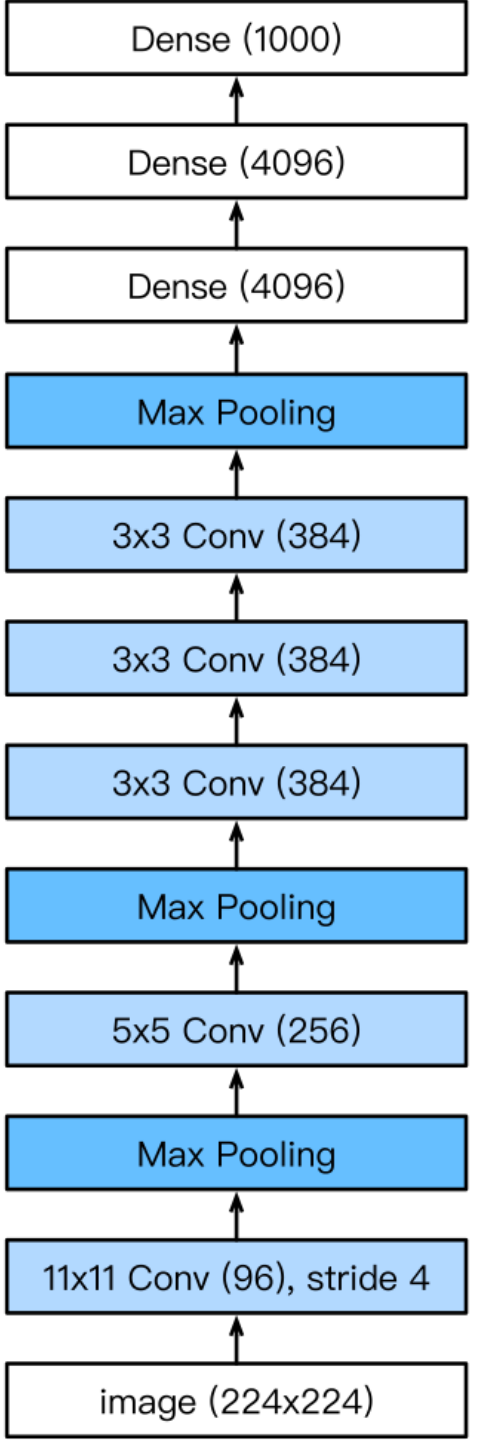

Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

AlexNet

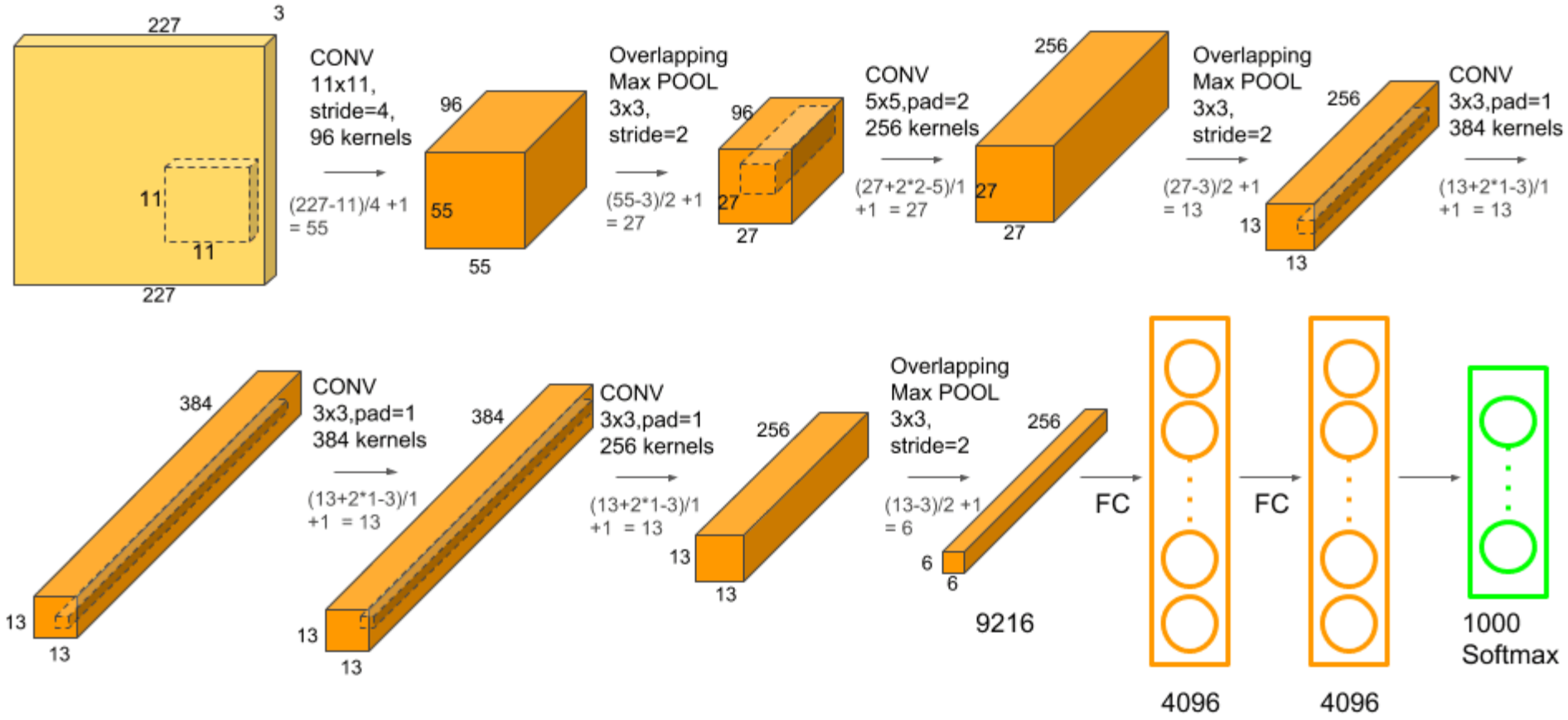

224
224
3
11
11
Stride
of 4
55
55
48
48
5
5
27
27
128
128
Max
pooling
Max
pooling
3
3
3
192
192
13
13
13
13
13
128
128
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000

AlexNet


224
224
3
11
11
Stride
of 4
48
55
5
5
48
Max
pooling
27
128
3
3
3
27
128
Max
pooling
13
192
3
3
3
13
192
3
3
3
13
192
128
Max
pooling
2048
2048
dense
dense
dense
2048
2048
1000

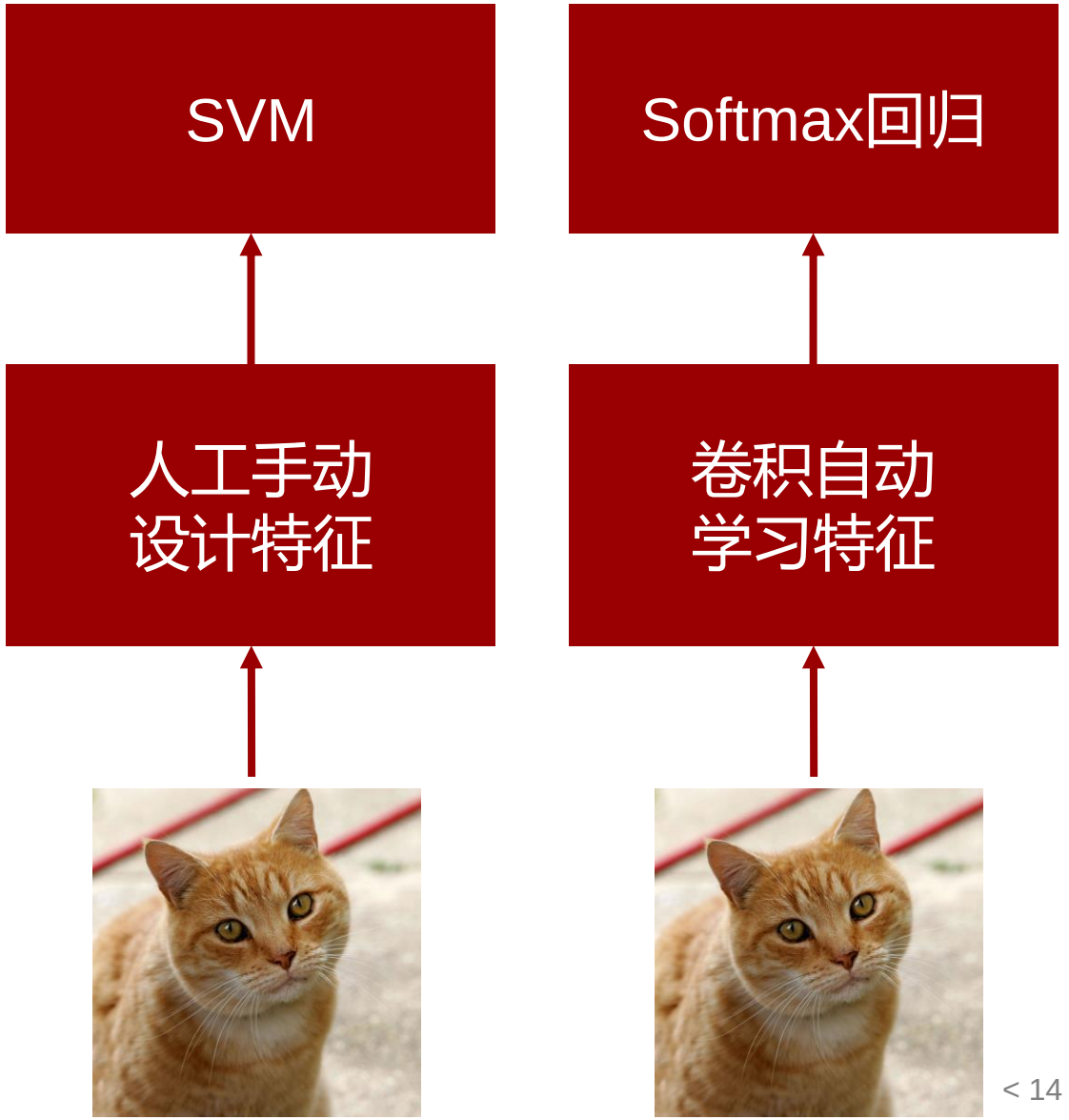

Dense (1000)
Dense (4096)
Dense (4096)
Max Pooling
3x3 Conv (384)
3x3 Conv (384)
3x3 Conv (384)
Max Pooling
5x5 Conv (256)
Max Pooling
11x11 Conv (96), stride 4
image (224x224)

AlexNet


227
3
11
11
227
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

AlexNet

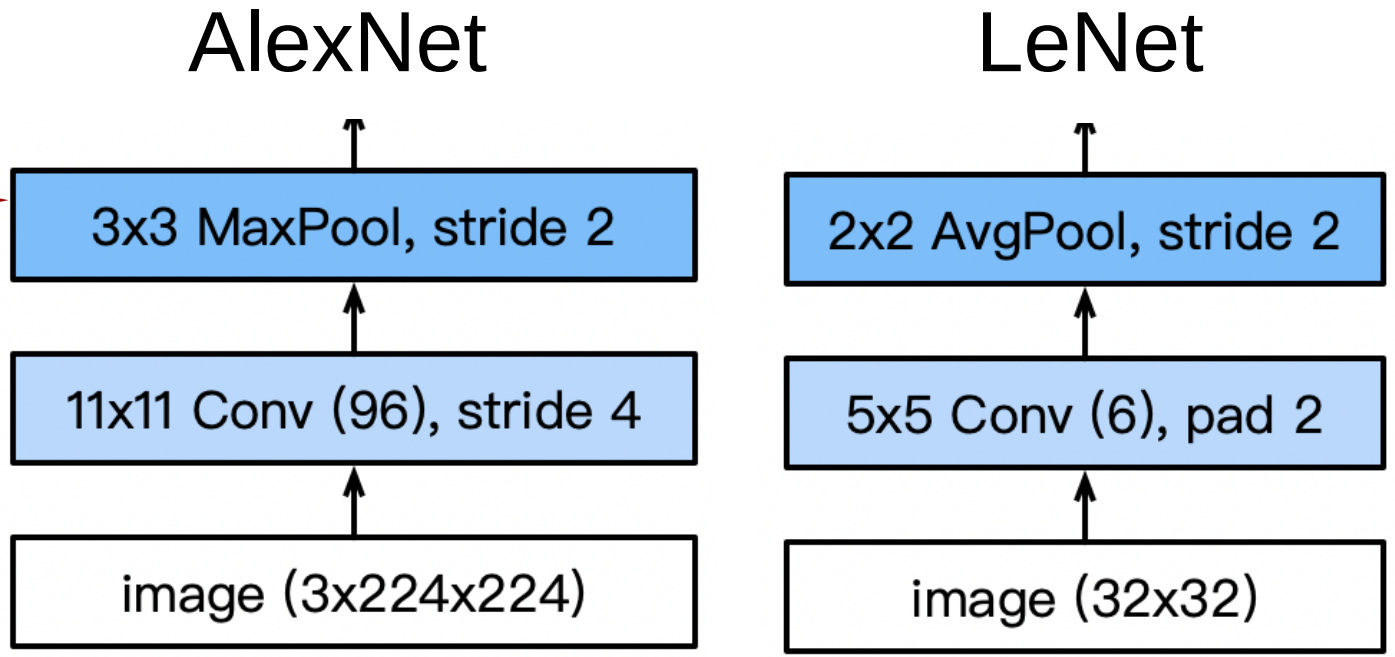
- AlexNet 在 2012 年赢得了ImageNet 竞赛
- 更深更大的 LeNet
- 主要修改
  - 丢弃法
  - ReLU 激活函数（训练）
  - 最大池化
- 计算机视觉的范式转变



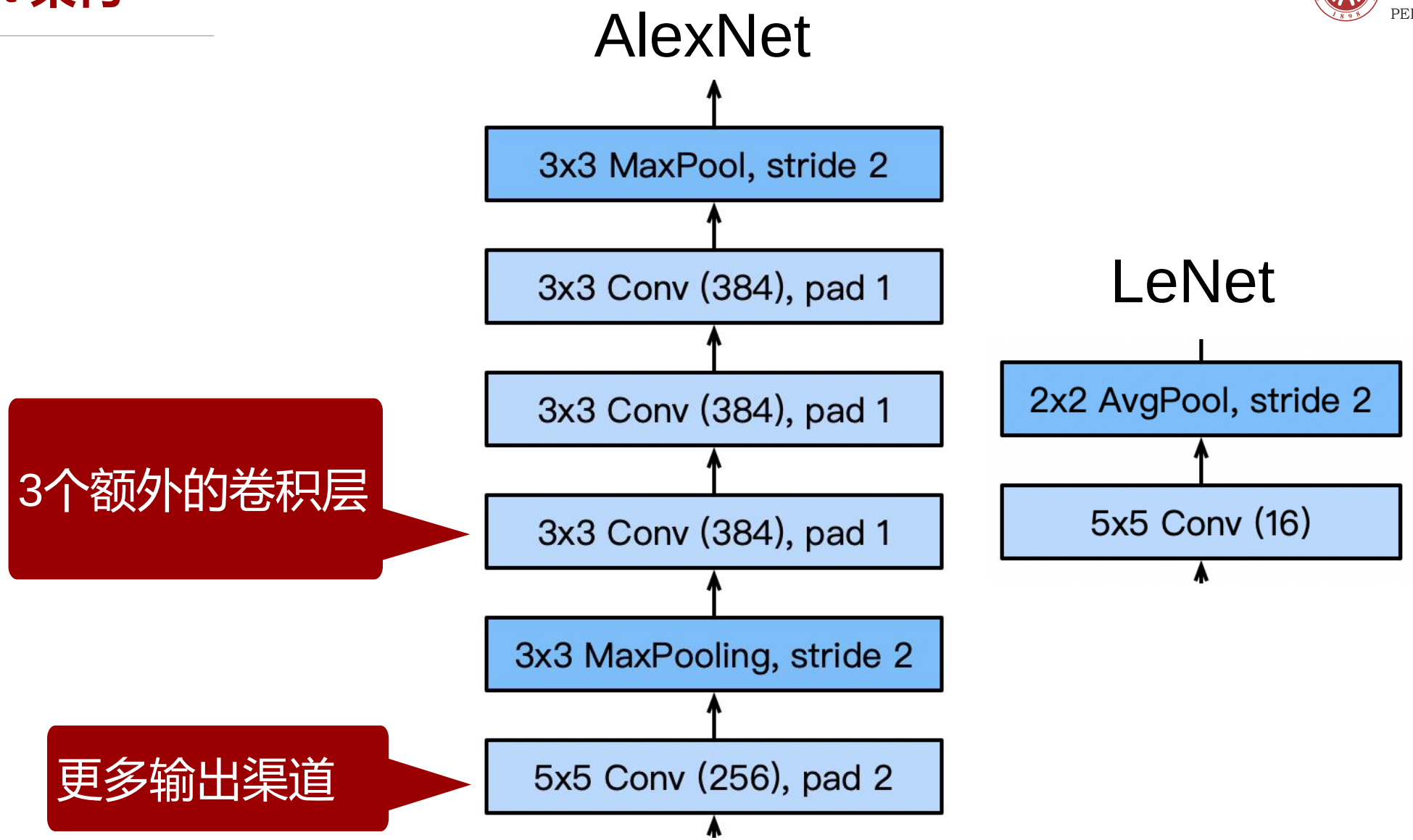
AlexNet 架构

池化窗口更大，更改为最大池

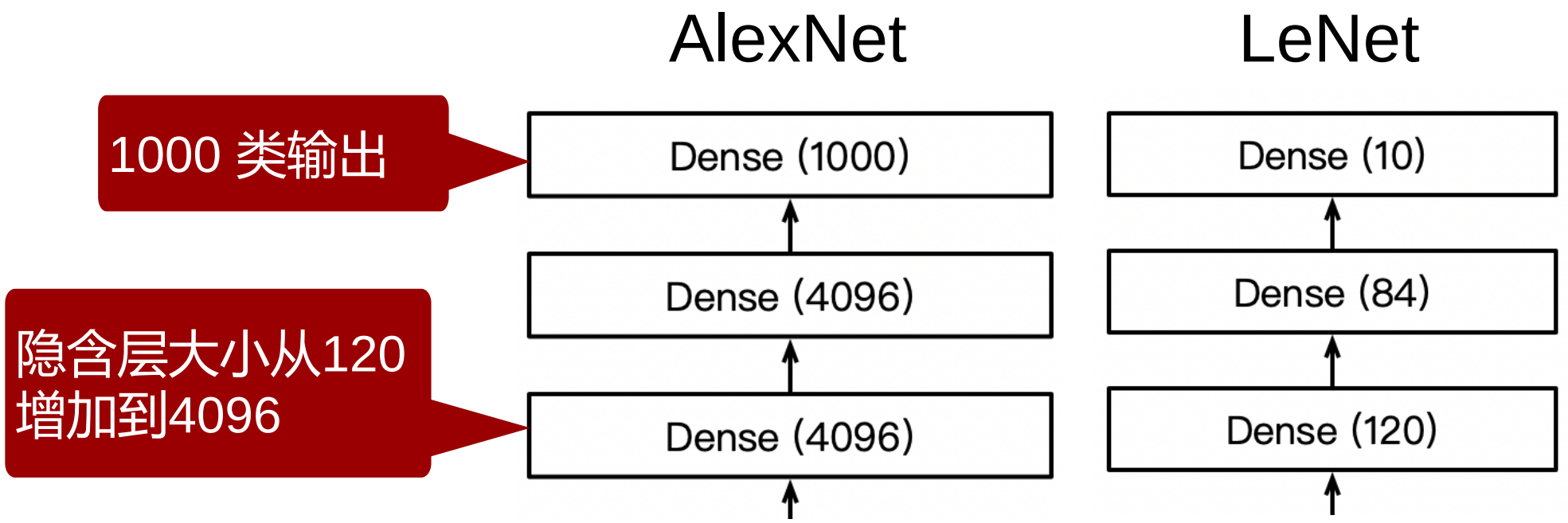
内核尺寸更大，由于图像尺寸增大，输出通道更多。



AlexNet 架构


AlexNet
3x3 MaxPool, stride 2
3x3 Conv (384), pad 1
3x3 Conv (384), pad 1
3x3 Conv (384), pad 1
3x3 MaxPooling, stride 2
5x5 Conv (256), pad 2
3个额外的卷积层
更多输出渠道
LeNet
2x2 AvgPool, stride 2
5x5 Conv (16)

AlexNet 架构

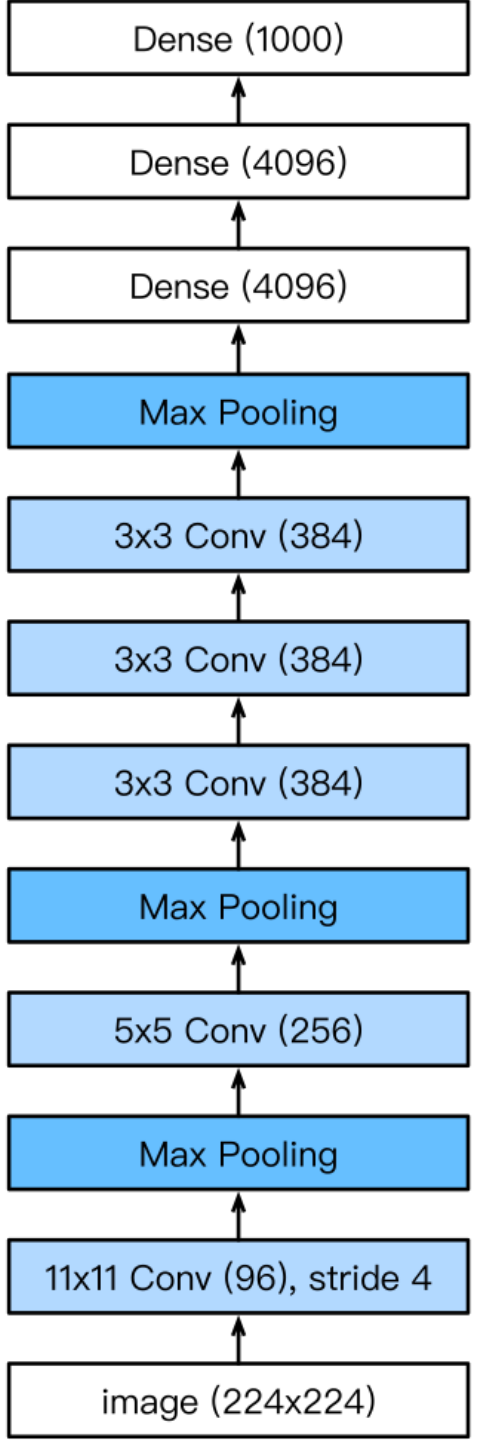

AlexNet
LeNet
1000 类输出
隐含层大小从120
增加到4096
Dense (1000)
Dense (4096)
Dense (4096)
Dense (10)
Dense (84)
Dense (120)

AlexNet 架构

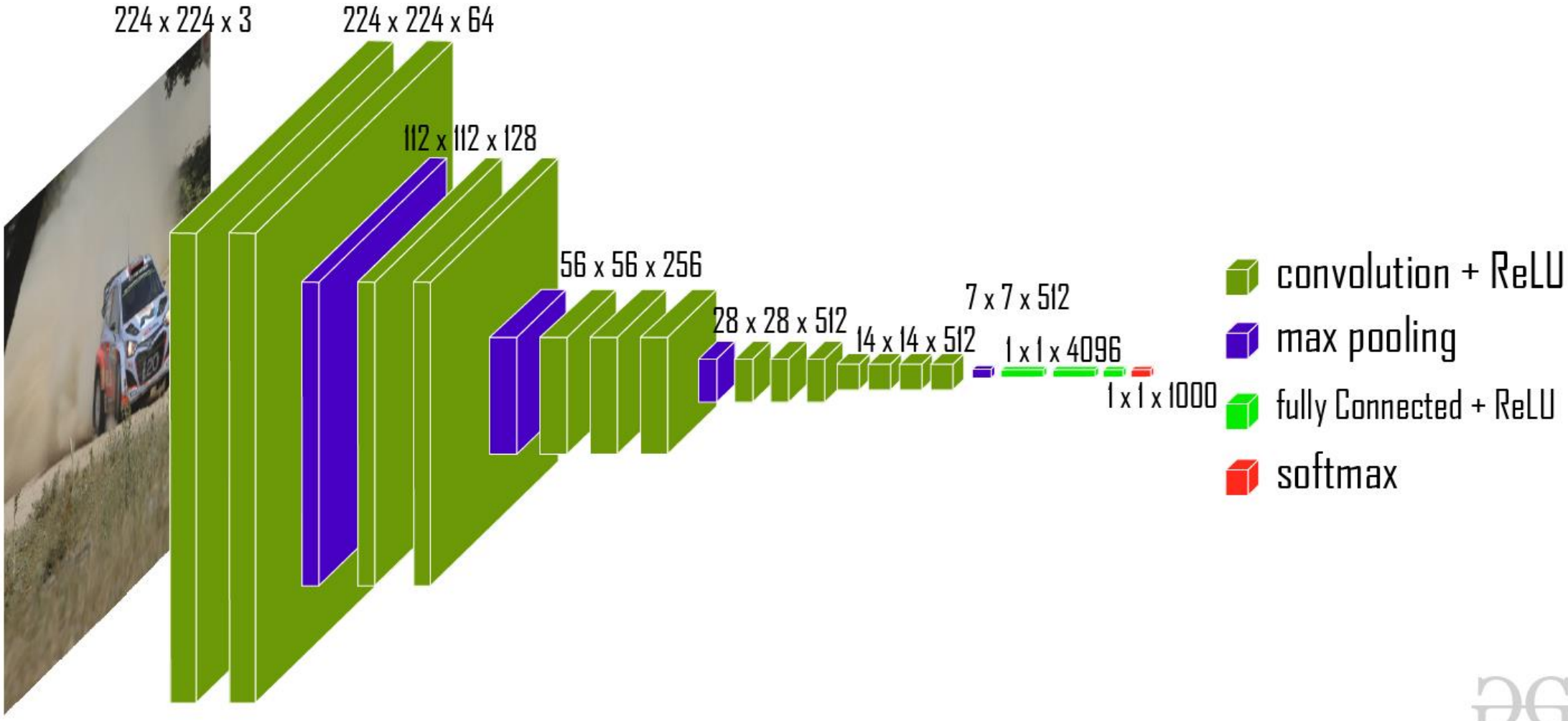
- 将激活函数从 sigmoid 更改为 ReLU（不再梯度消失）
- 在两个隐含层之后应用丢弃法（更好的稳定性 / 正则化）
- 人工增加数据集

AlexNet 复杂度

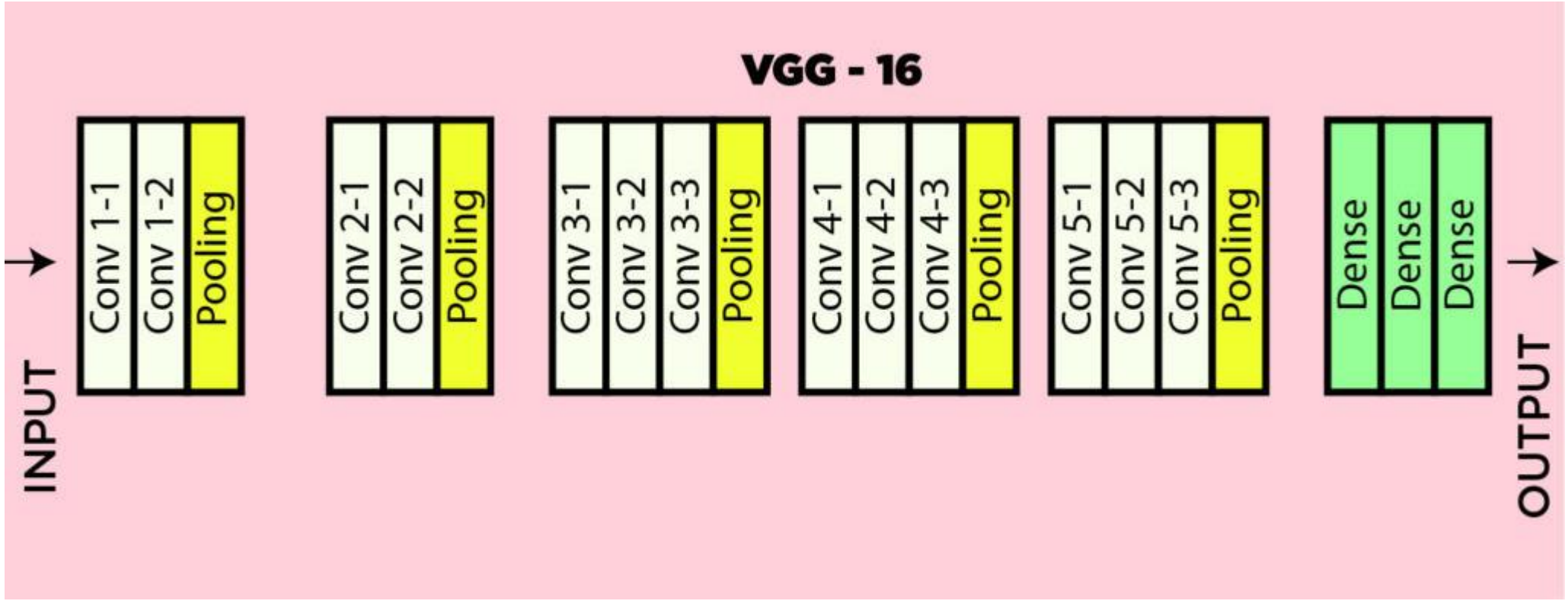
	# 参数		浮点运算（FLOP）	
	AlexNet	LeNet	AlexNet	LeNet
卷积层1	35K	150	101M	1.2M
卷积层2	614K	2.4K	415M	2.4M
卷积层3-5	3M		445M	
稠密层1	26M	0.48M	26M	0.48M
稠密层2	16M	0.1M	16M	0.1M
总共	46M	0.6M	1G	4M
倍数增加	11x	1x	250x	1x



VGG


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution + ReLU
max pooling
fully Connected + ReLU
softmax

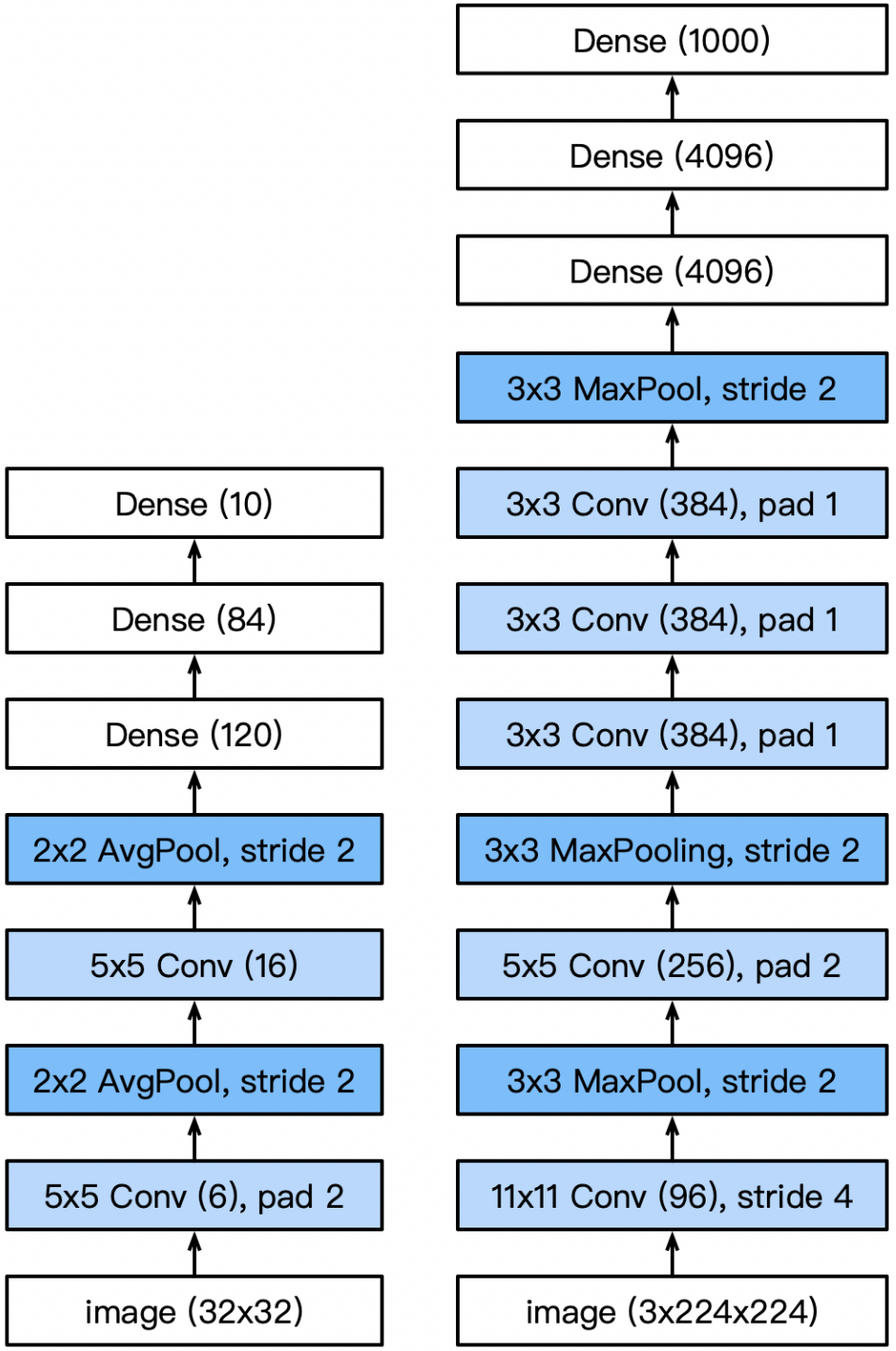
VGG


VGG - 16
INPUT
Conv 1-1
Conv 1-2
Pooling
Conv 2-1
Conv 2-2
Pooling
Conv 3-1
Conv 3-2
Conv 3-3
Pooling
Conv 4-1
Conv 4-2
Conv 4-3
Pooling
Conv 5-1
Conv 5-2
Conv 5-3
Pooling
Dense
Dense
Dense
OUTPUT

VGG

AlexNet 比 LeNet 更深入更大，以获得更加性能，怎么更大更深?

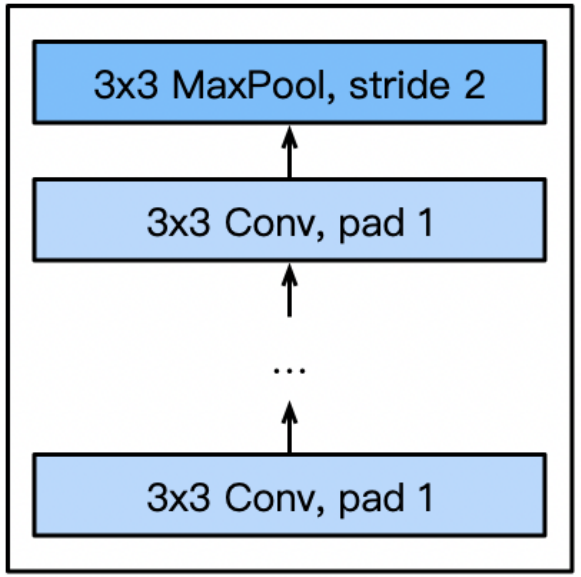
- 更密集的层（太贵）
- 更多的卷积
- 分组成块



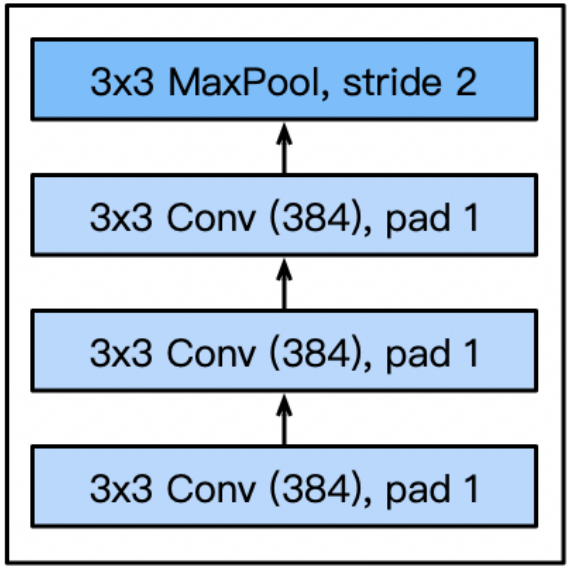
VGG 块

- 更深还是更宽?
  - 5x5 卷积
  - 3x3 卷积（更多）
  - 更深和更窄更好
- VGG 块
  - 3x3 卷积（填充=1）（n层，m个通道）
  - 2x2 最大池化层（步幅=2）

VGG 块

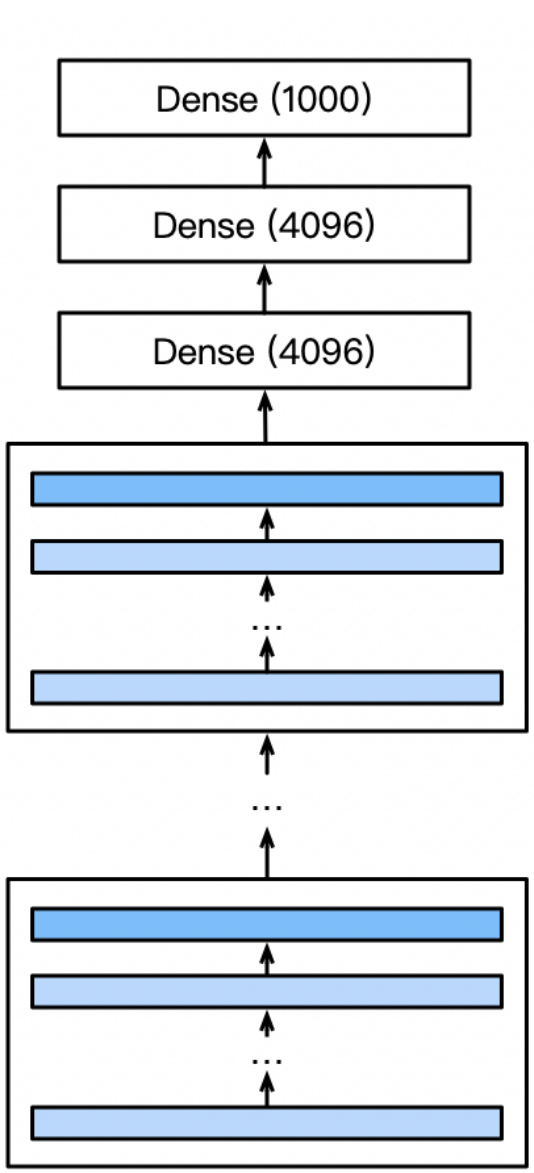


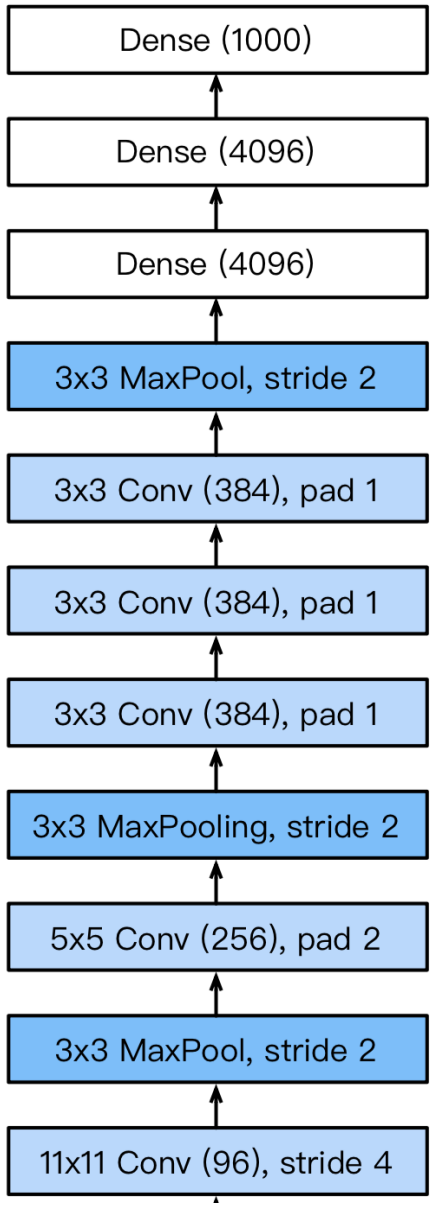
AlexNet 一部分



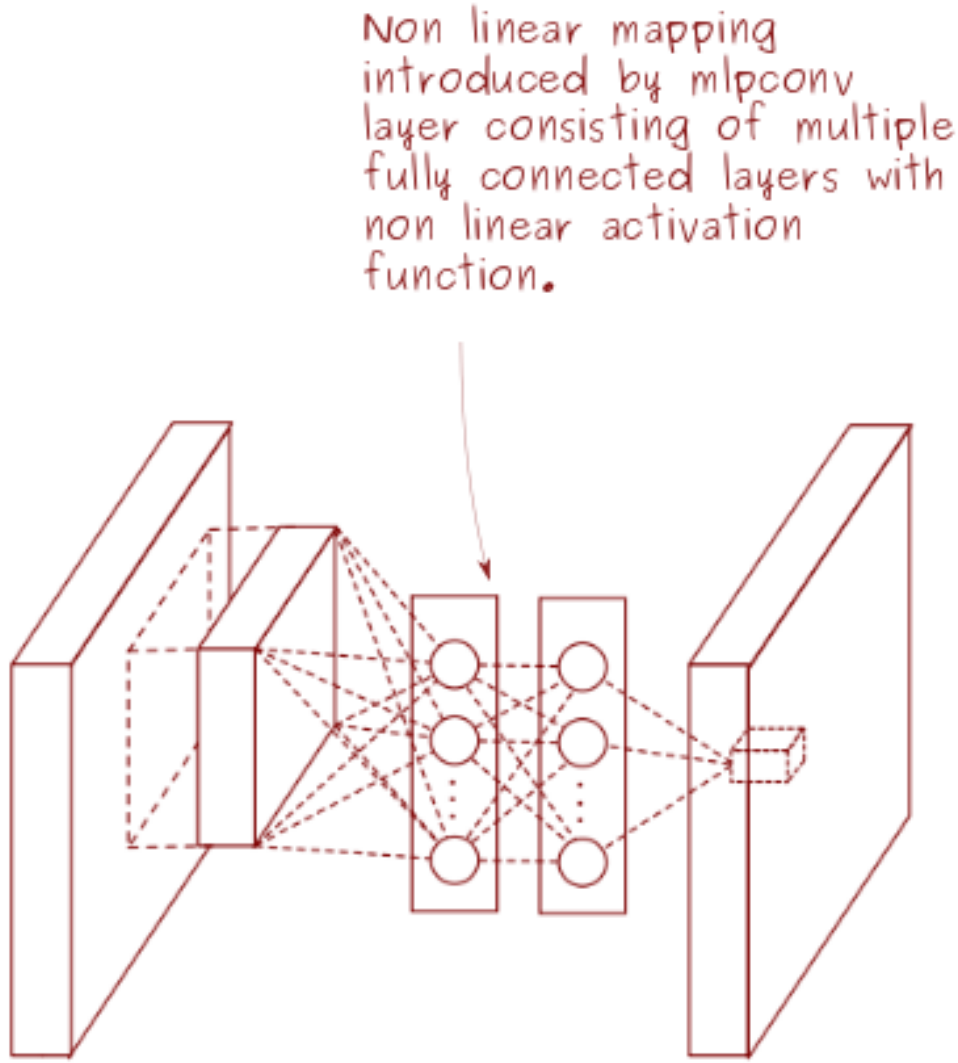
VGG 结构

- 多个VGG块后加稠密层
- 不同数目的重复VGG 块，可获得不同的架构，例如VGG-16，VGG-19，……





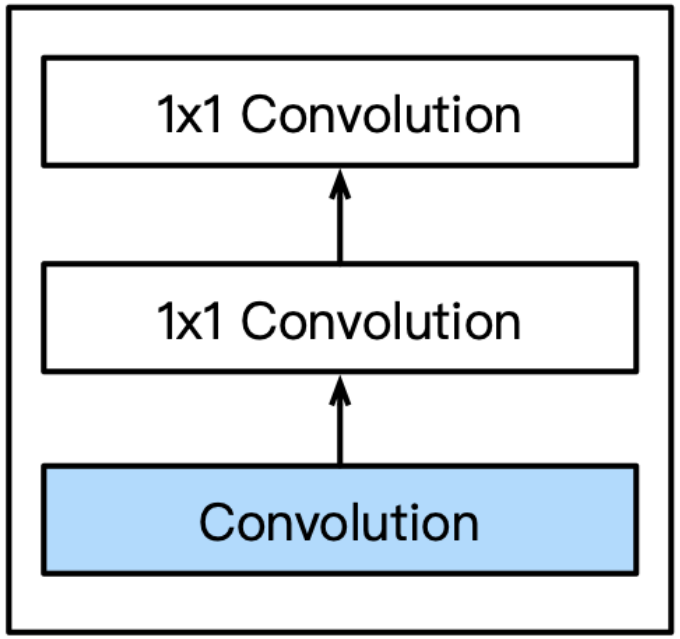
Network in Network (NiN)



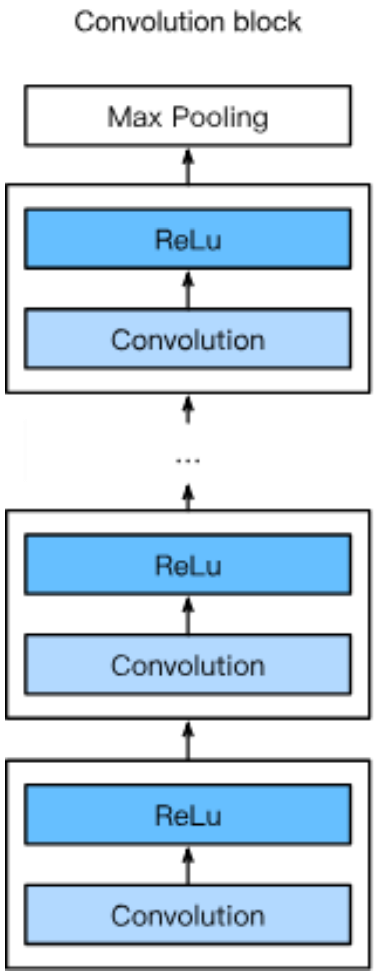
1x1 卷积

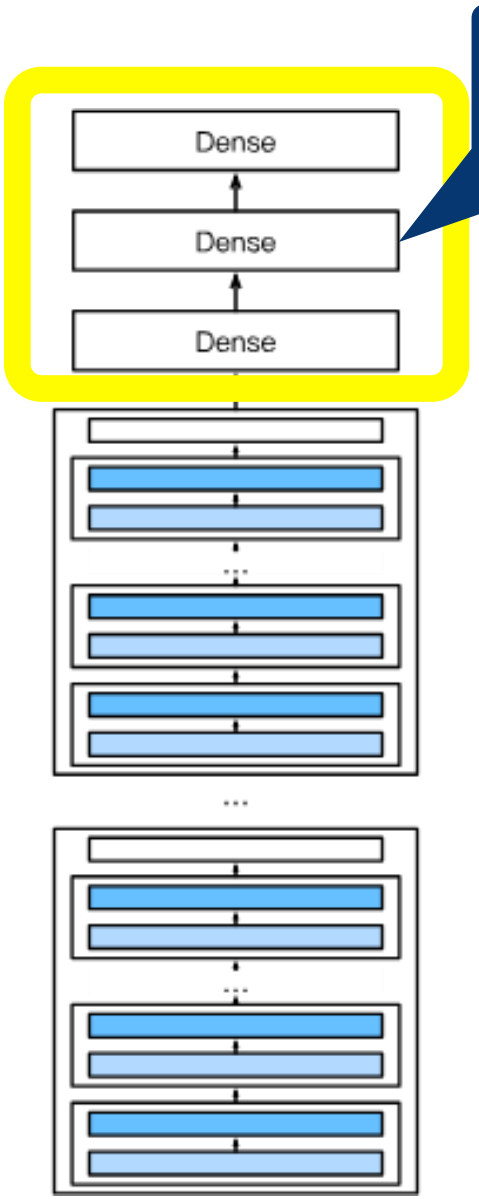
NiN 块

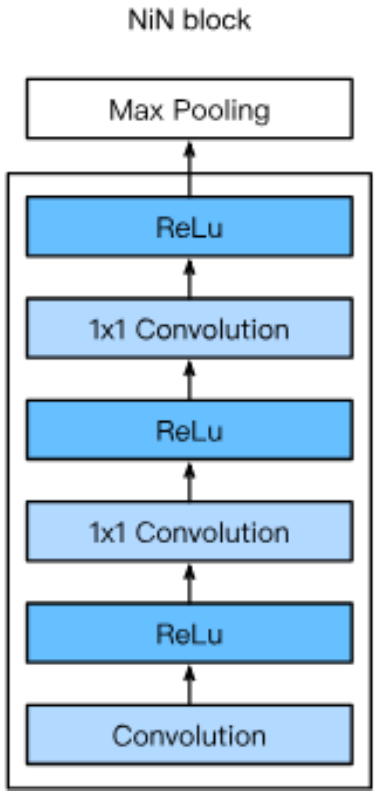
- 卷积层
  - 超参数：内核大小，步幅和填充
- 接下来是两个 1x1 卷积层
  - 步幅1
  - 无填充
  - 与第一层输出通道相同
  - 充当稠密层

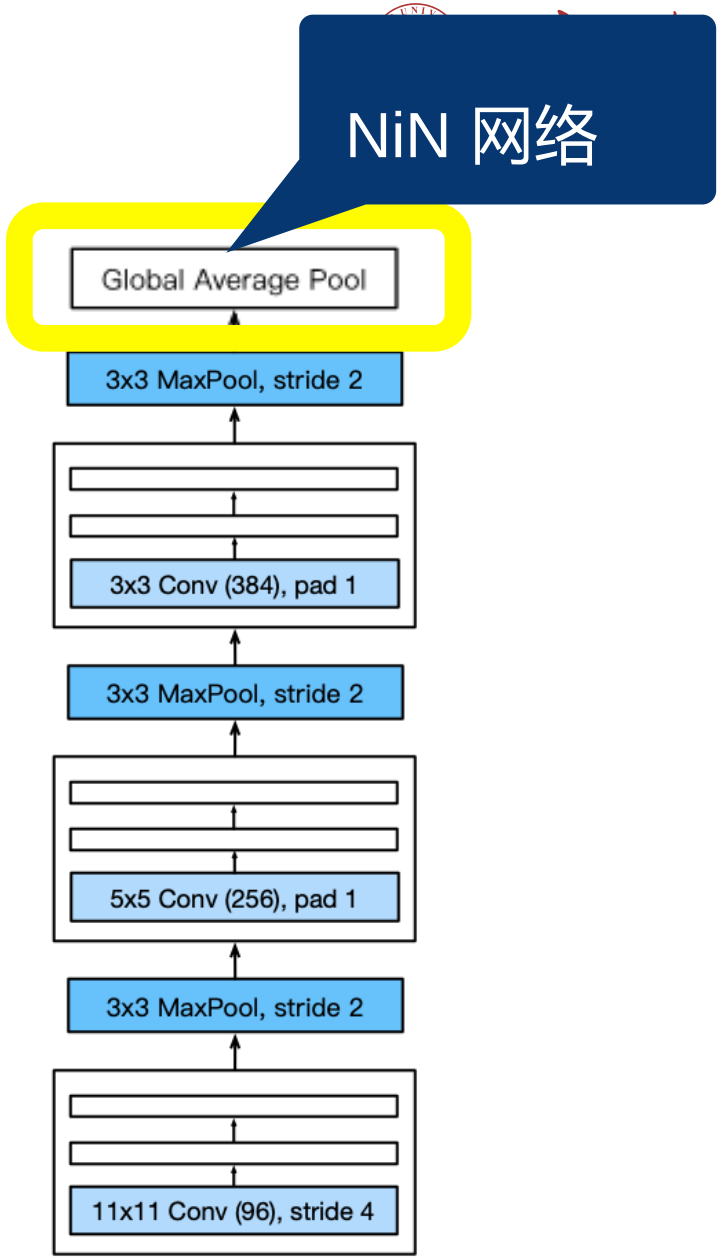


NiN 网络


Convolution block
Max Pooling
ReLu
Convolution
...
ReLu
Convolution
ReLu
Convolution

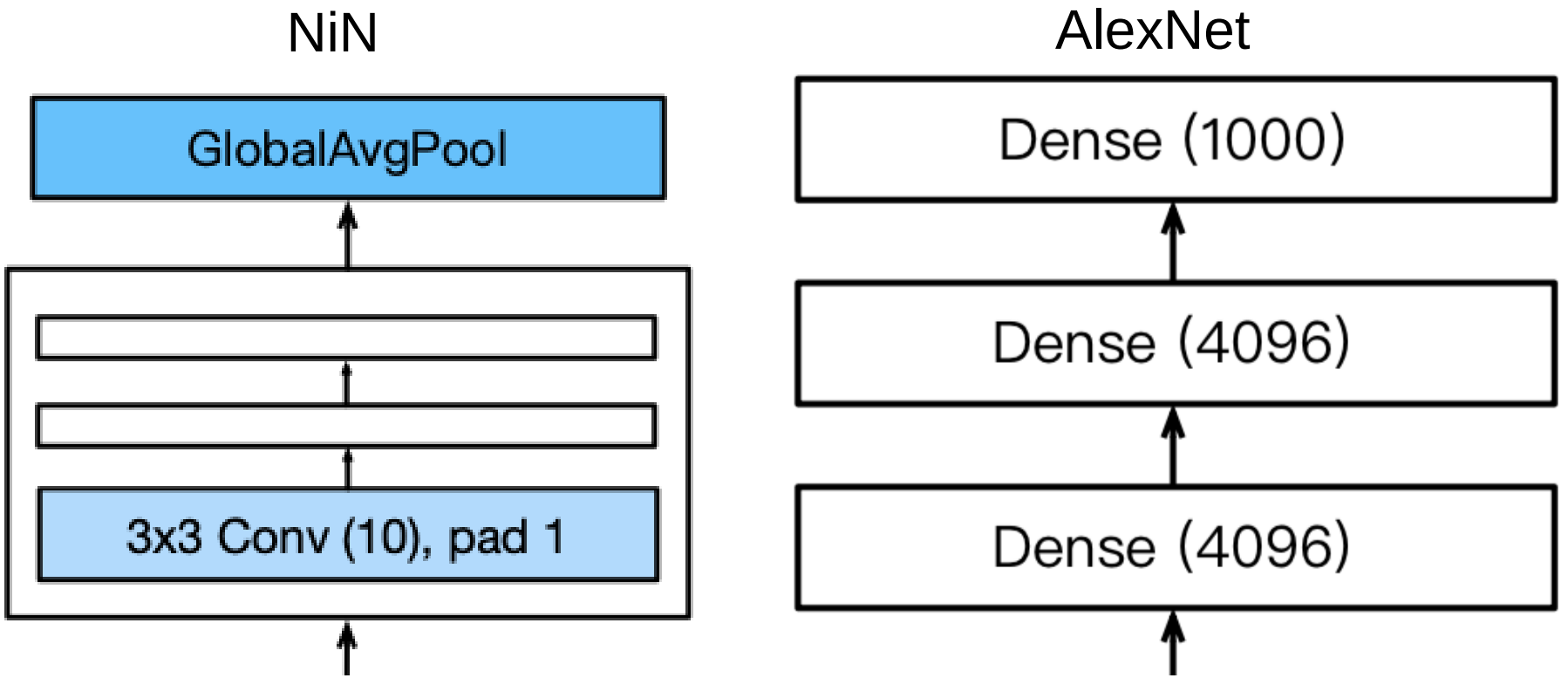

VGG 网络
Dense
Dense
Dense
...
...


NiN block
Max Pooling
ReLu
1x1 Convolution
ReLu
1x1 Convolution
ReLu
Convolution


NiN 网络
Global Average Pool
3x3 MaxPool, stride 2
3x3 Conv (384), pad 1
3x3 MaxPool, stride 2
5x5 Conv (256), pad 1
3x3 MaxPool, stride 2
11x11 Conv (96), stride 4

NiN最后一层

- 用 NiN 块替换了 AlexNet 的稠密层
- 输出：全局平均池化层



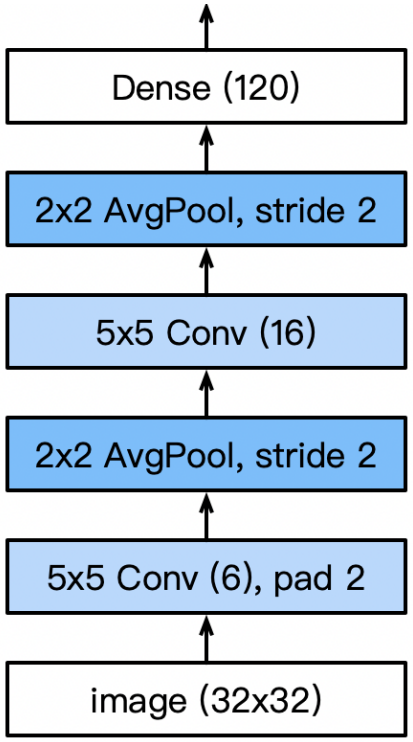
小结

- LeNet（第一个卷积层）
- AlexNet
  - 升华版的 LeNet
  - ReLU 激活，丢弃法，平移稳定性
- VGG
  - 升华版的 AlexNet
  - 重复的 VGG 块
- NiN
  - 1x1 卷积 + 全局池化（而不是稠密层）

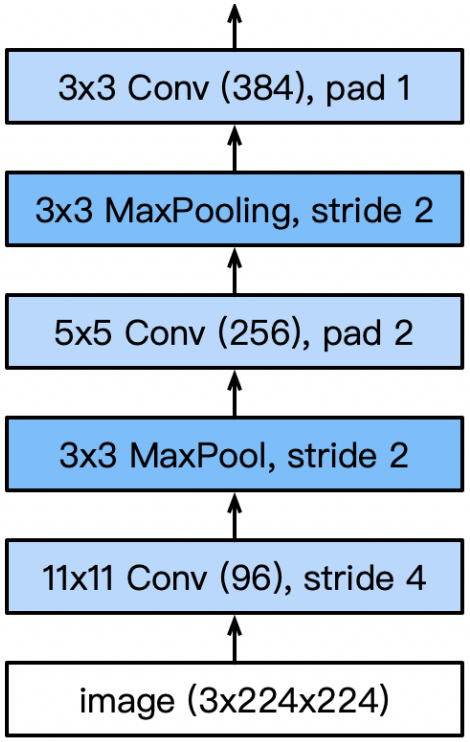
选择合适的卷积

1x1　3x3　5x5　最大池化　许多 1x1

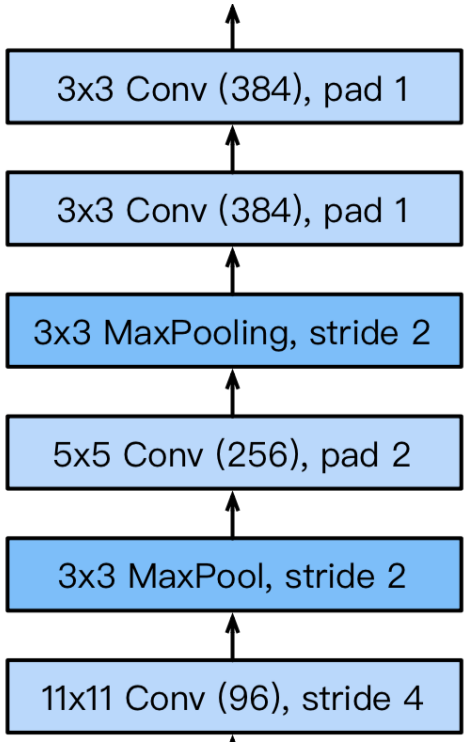
LeNet



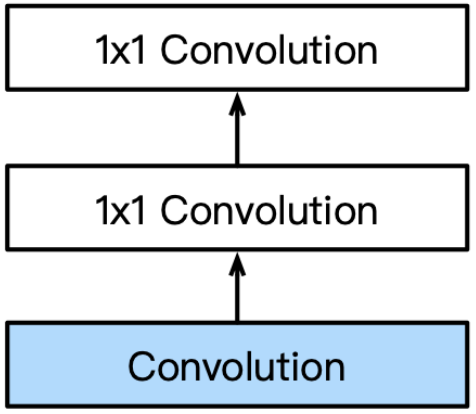
AlexNet



VGG

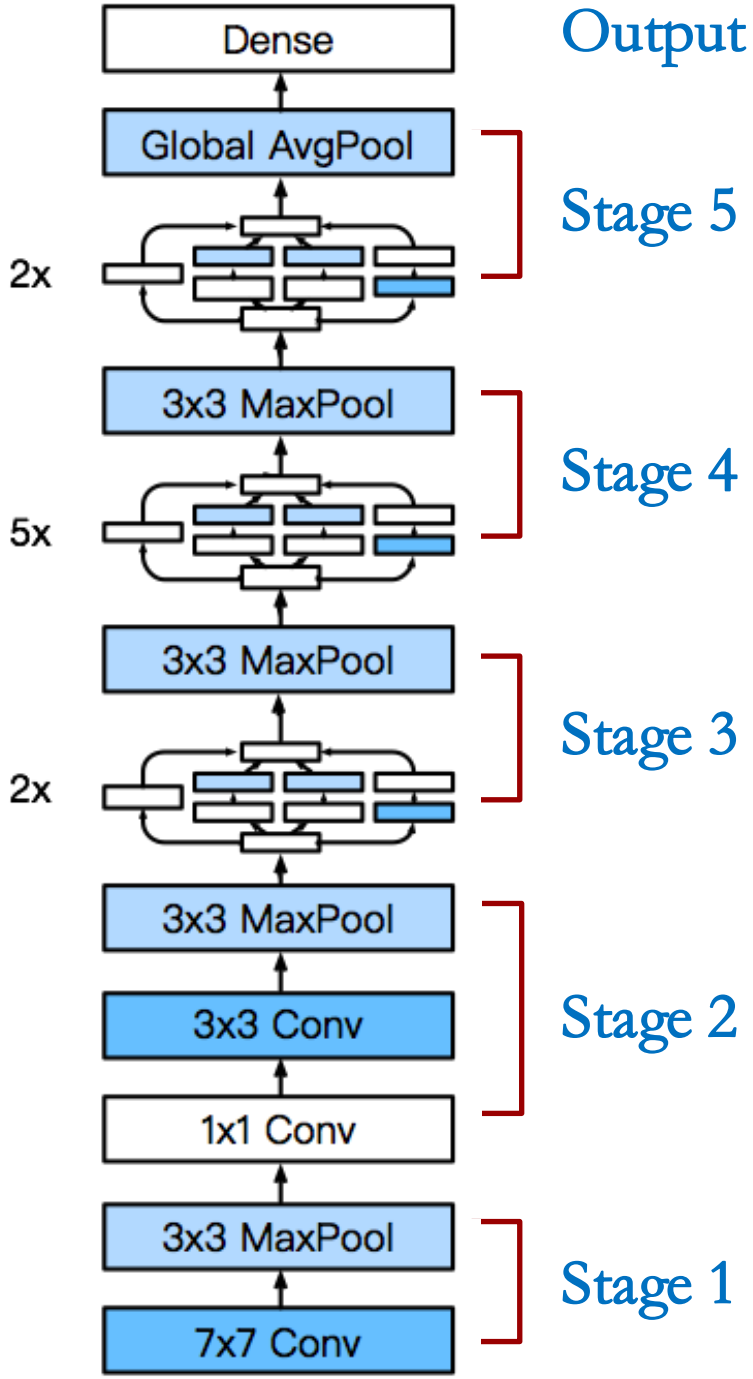


NiN



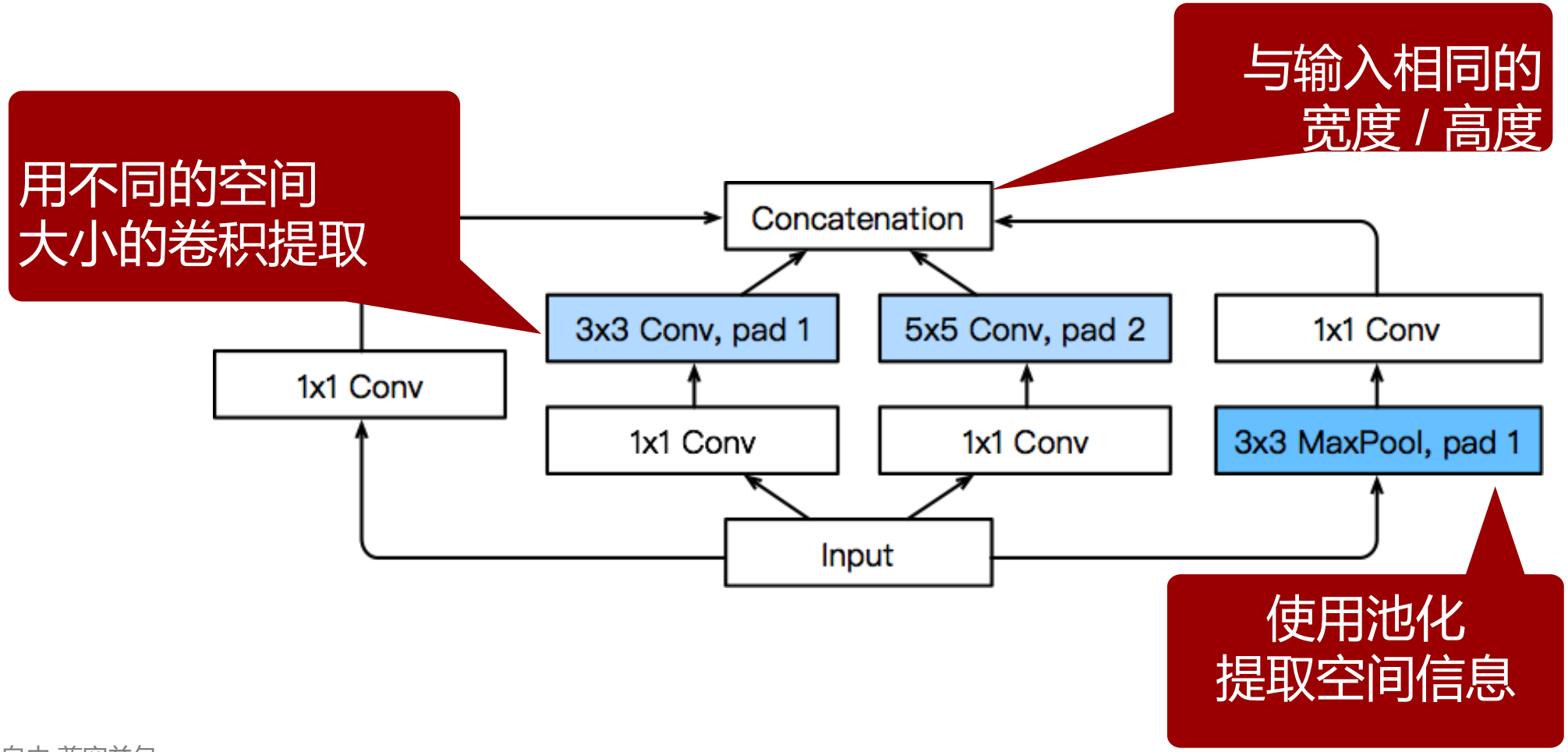
GoogLeNet

- 5 个阶段
- 9 个 Inception 块

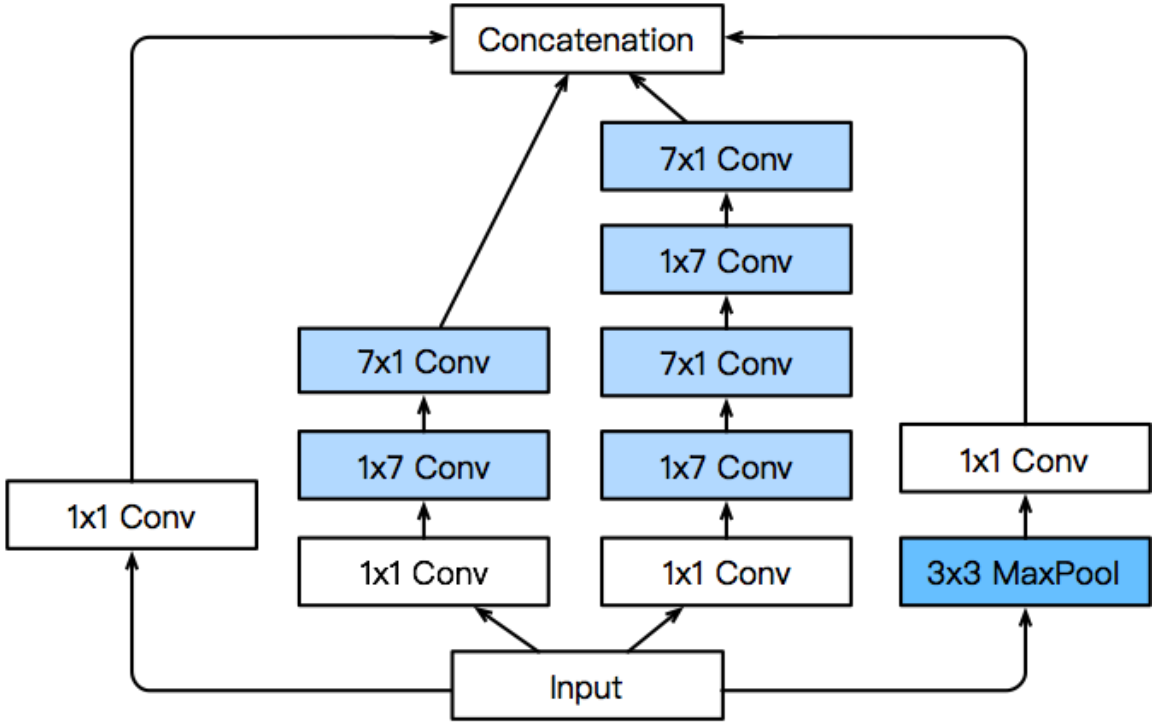


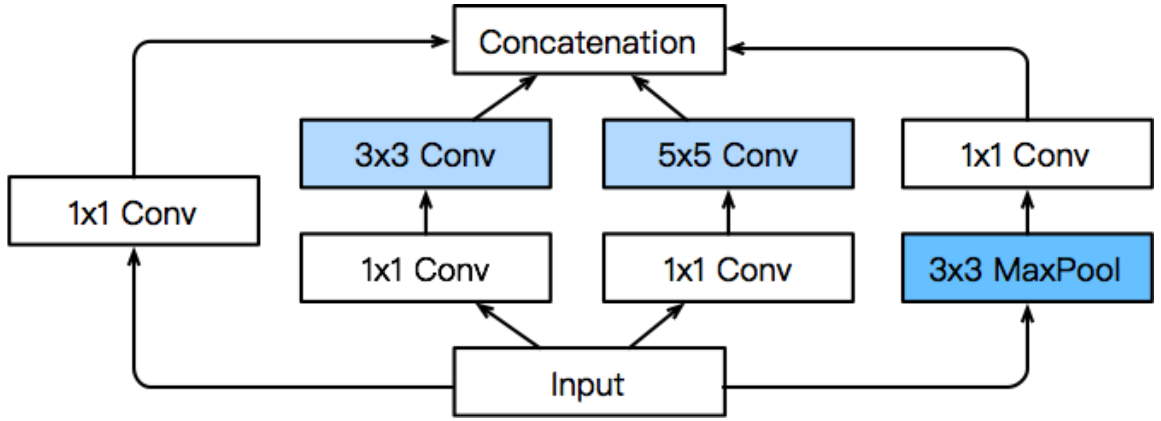
Inception 块

4个路径从不同方面提取信息，然后连接输出通道

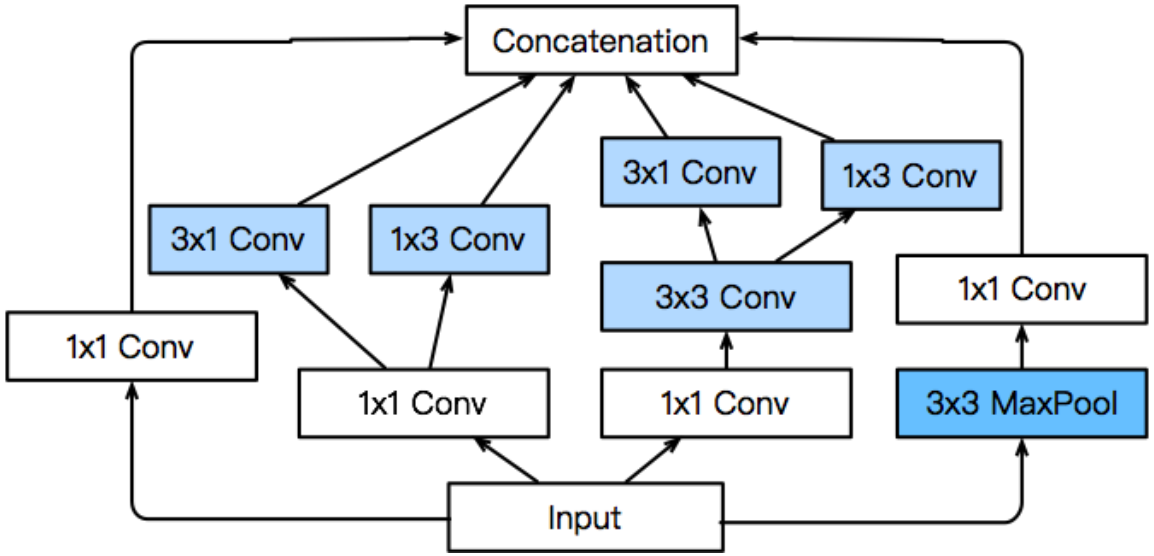


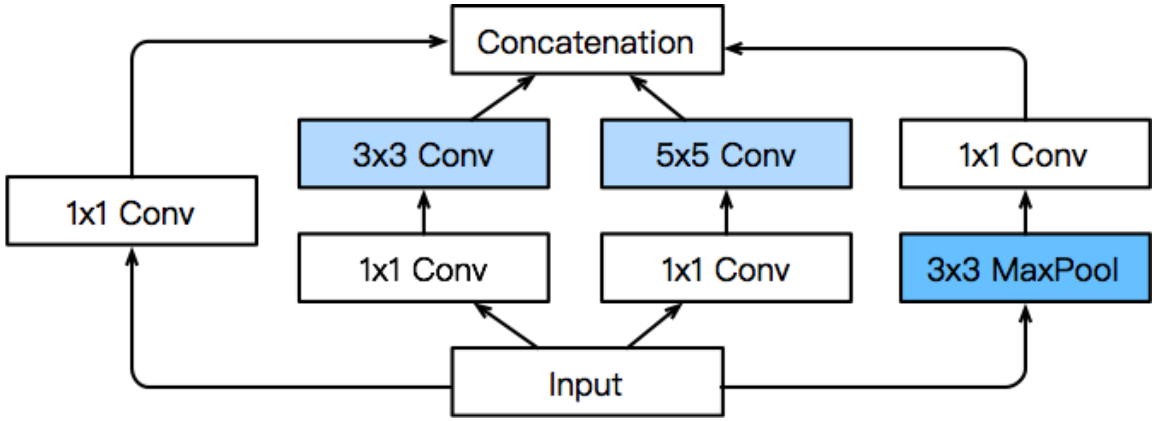
Inception 块


Concatenation
7x1 Conv
1x7 Conv
7x1 Conv
7x1 Conv
1x7 Conv
1x7 Conv
1x1 Conv
1x1 Conv
1x1 Conv
1x1 Conv
3x3 MaxPool
Input


Concatenation
1x1 Conv
3x3 Conv
5x5 Conv
1x1 Conv
1x1 Conv
1x1 Conv
3x3 MaxPool
Input

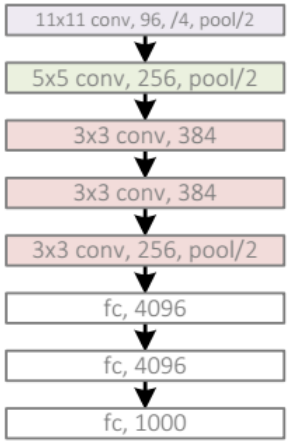
Inception 块


Concatenation
3x1 Conv
1x3 Conv
3x1 Conv
1x3 Conv
1x1 Conv
3x3 Conv
1x1 Conv
1x1 Conv
1x1 Conv
1x1 Conv
3x3 MaxPool
Input

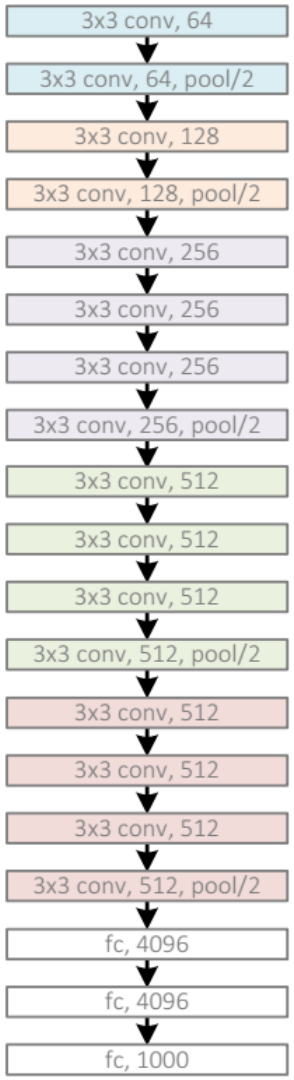

Concatenation
1x1 Conv
3x3 Conv
5x5 Conv
1x1 Conv
1x1 Conv
1x1 Conv
3x3 MaxPool
Input

网络发展

AlexNet, 8 layers
(ILSVRC 2012)

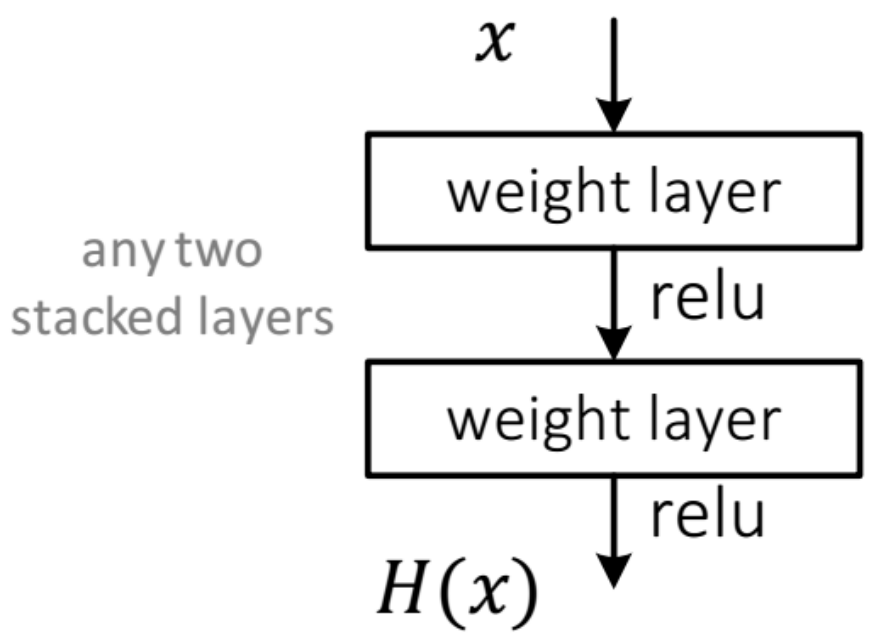


VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

- Plain net



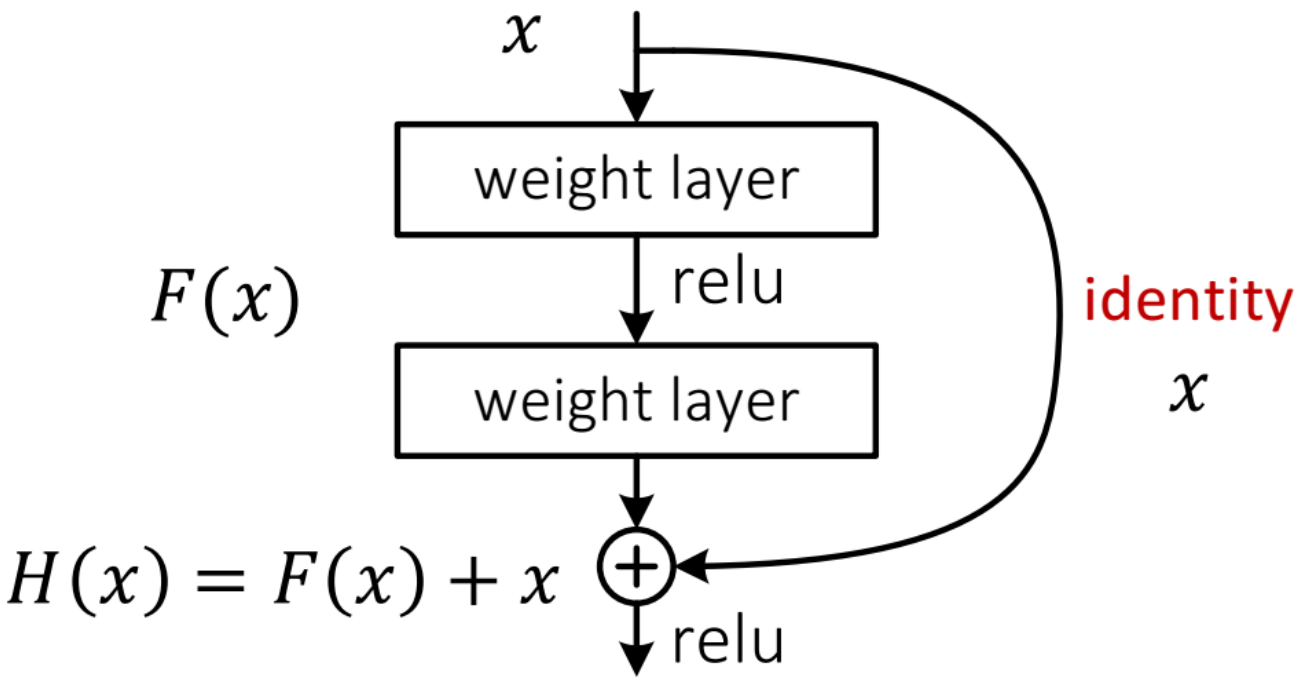
$H(x)$ is any desired mapping,

hope the 2 weight layers fit $H(x)$

Deep Residual Learning for Image Recognition, CVPR 2016.

ResNet

- Residual net



$H(x)$ is any desired mapping,

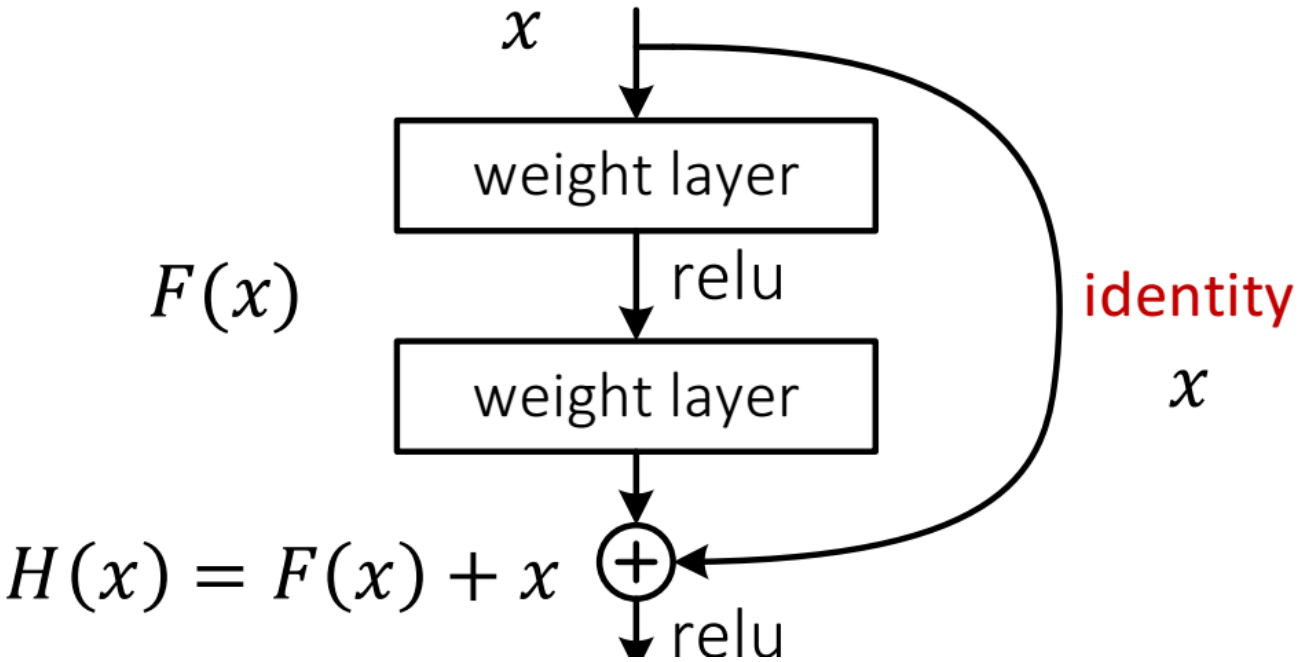
~~hope the 2 weight layers fit $H(x)$~~

hope the 2 weight layers fit $F(x)$

let $H(x) = F(x) + x$

ResNet

- $F(x)$ is a residual mapping w.r.t. identity



- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

在残差块中，有

$h(x_n) = F(x_n, w_n) + x_n$

$x_{n+1} = re(h(x_n))$

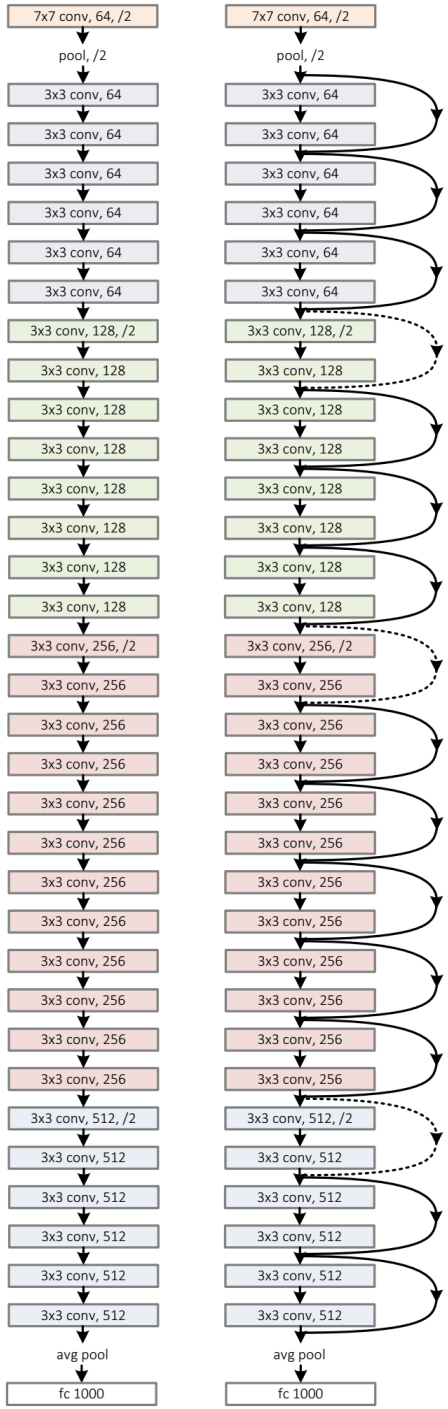
后向传播求x_n的梯度为:

$$\frac{\partial x_{n+1}}{\partial x_n} = \frac{\partial re}{\partial h} * \frac{\partial F(x_n, w_n) + x_n}{\partial x_n}$$
$$= \frac{\partial F(x_n, w_n)}{\partial x_n} + 1$$

$F(x_n, w_n)$是x_n的一个残差，当残差变化大时，可以利用梯度进行更新参数，当残差变化小的时候，也能够保证梯度不会消失。

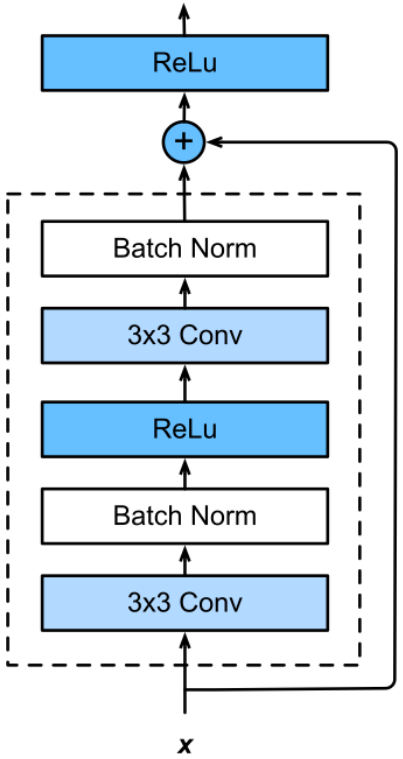
ResNet

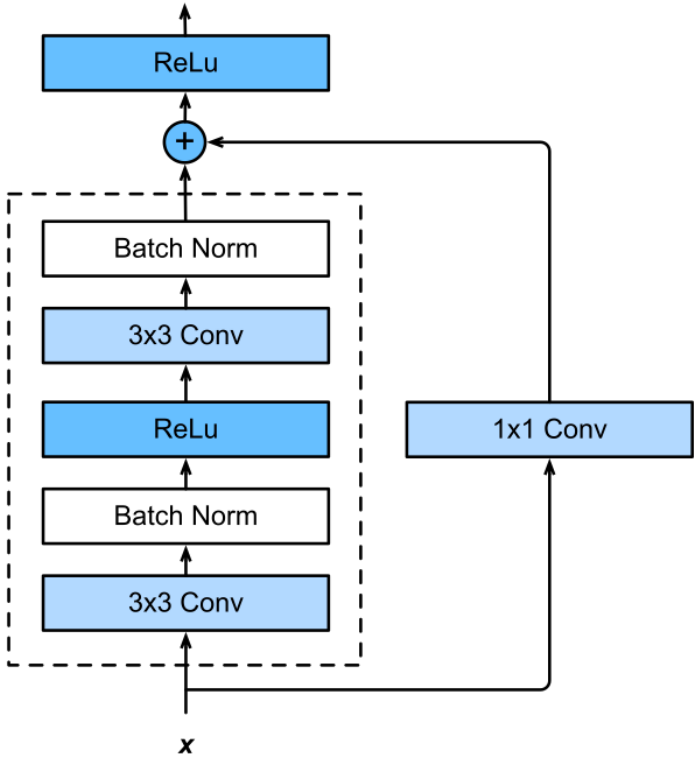
Plain net

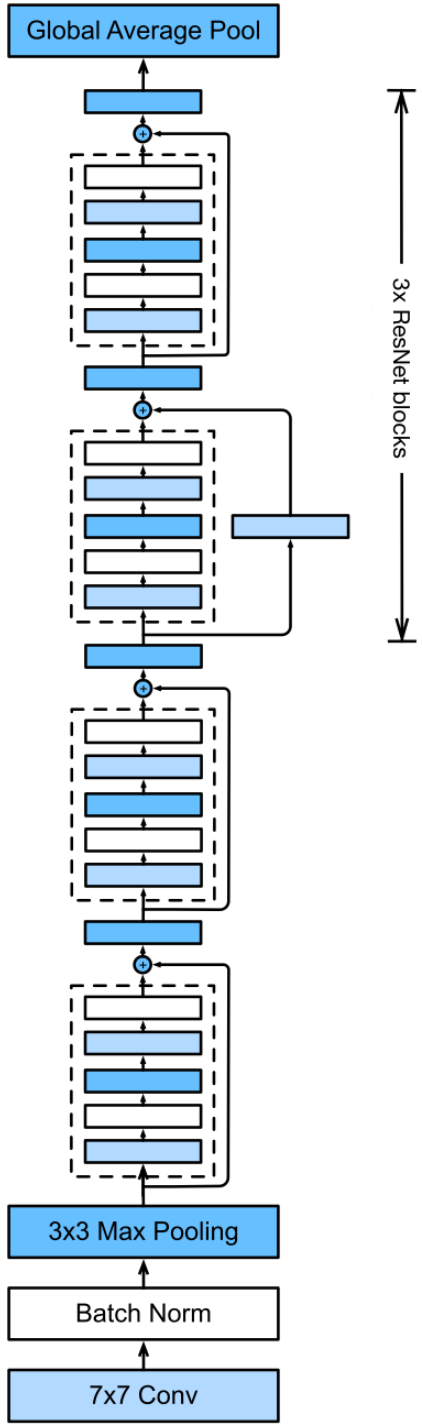


Residual net

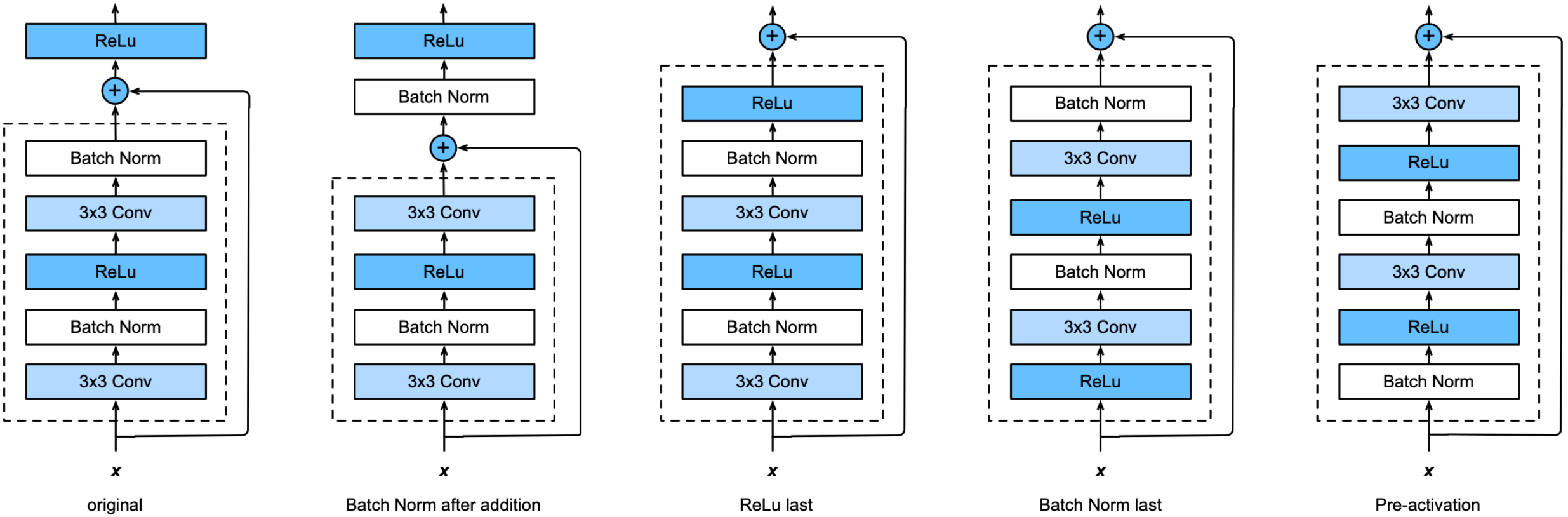
ResNet


ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
x

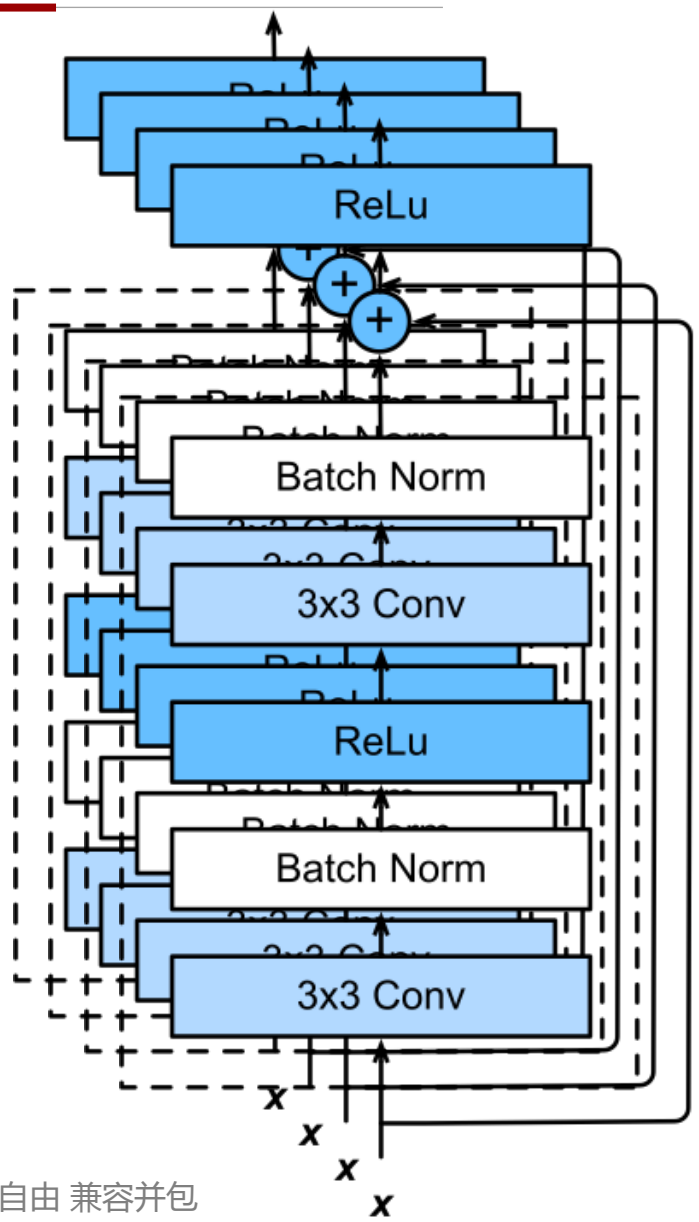

ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
1x1 Conv
x


Global Average Pool
3x ResNet blocks
3x3 Max Pooling
Batch Norm
7x7 Conv

ResNet


ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
x
original
ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
x
Batch Norm after addition
ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
x
ReLu last
Batch Norm
3x3 Conv
ReLu
Batch Norm
3x3 Conv
ReLu
x
Batch Norm last
3x3 Conv
ReLu
Batch Norm
3x3 Conv
ReLu
Batch Norm
x
Pre-activation

降低卷积成本



- 参数 $k_h \cdot k_w \cdot c_i \cdot c_o$
- 计算 $m_h \cdot m_w \cdot k_h \cdot k_w \cdot c_i \cdot c_o$
- 切割卷积（Inception v4），例如 3x3 vs. 1x5 和5x1
- 分解通道（仅在内部）

$$m_h \cdot m_w \cdot k_h \cdot k_w \cdot \frac{c_i}{b} \cdot \frac{c_o}{b} \cdot b$$

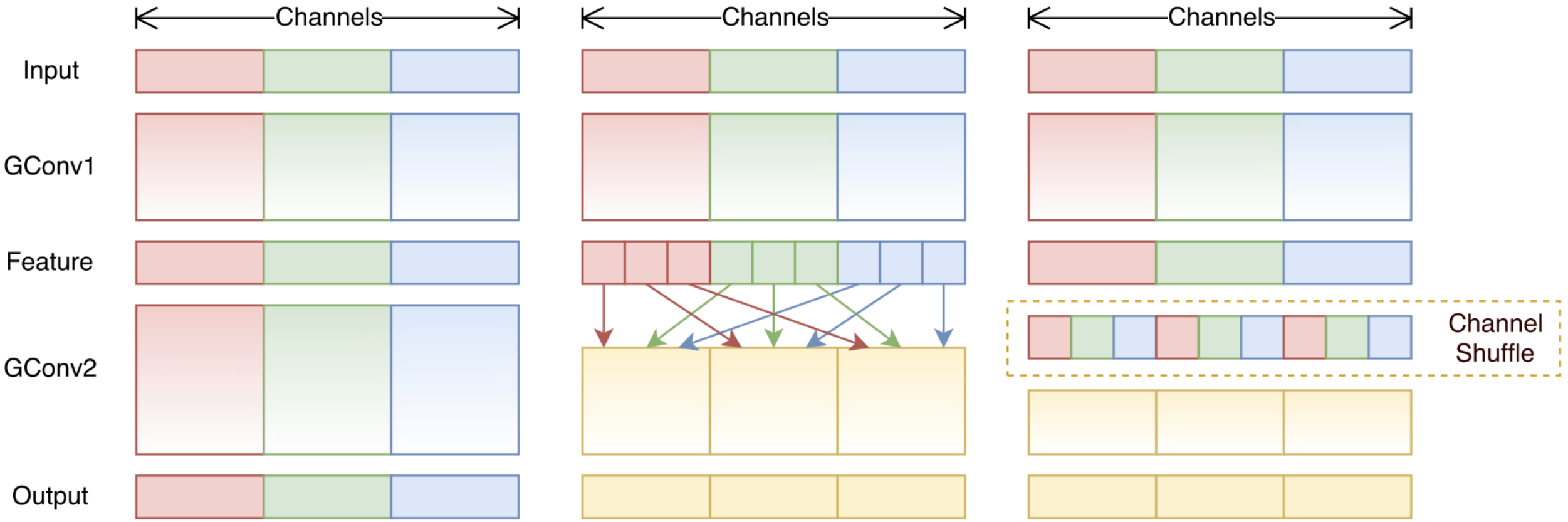
ResNeXt

- 切割成 32 个子块
- 可以使用更多尺寸
- 精度更高

nn.Conv2D调整groups参数

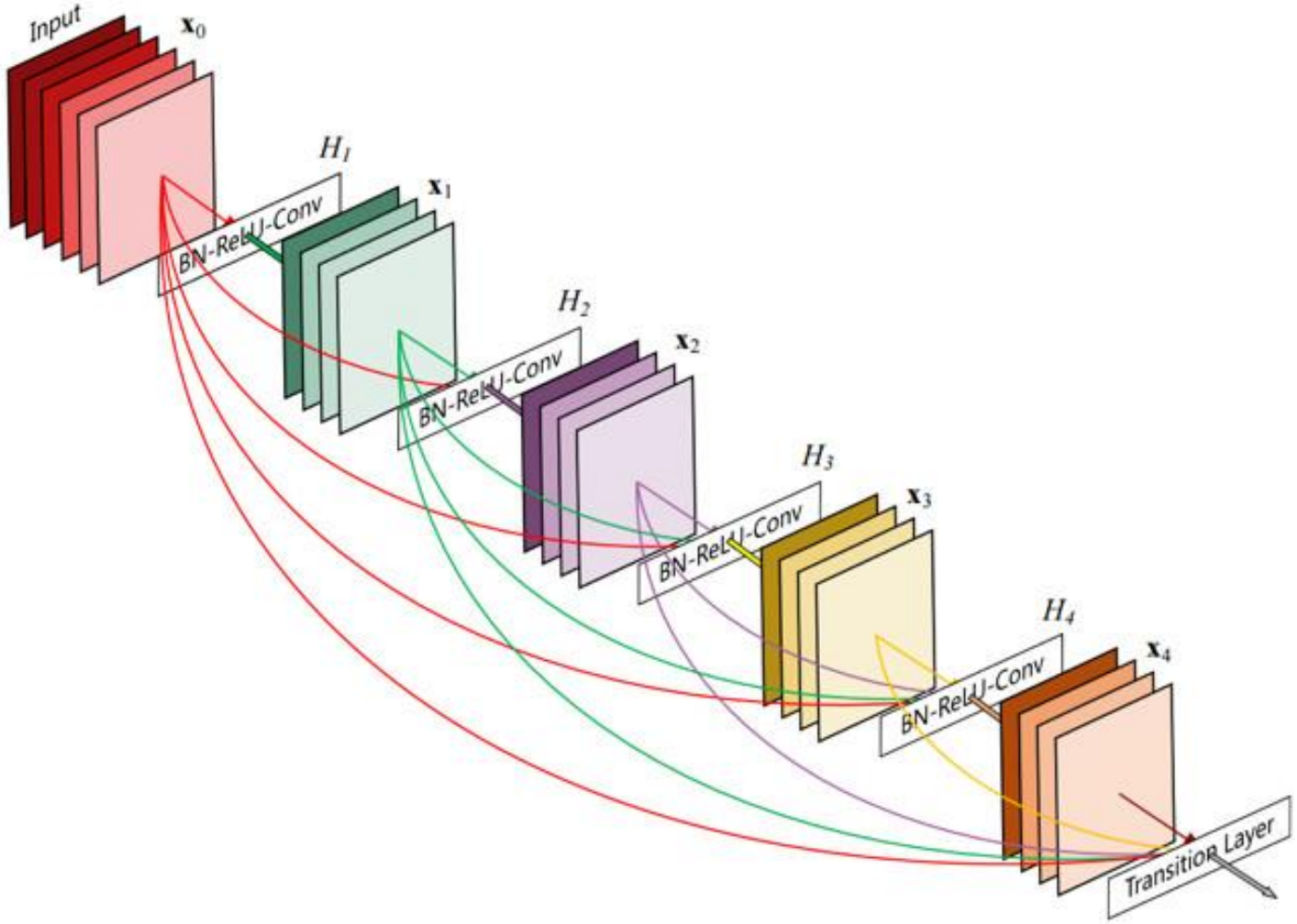
stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C=32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

ShuffleNet



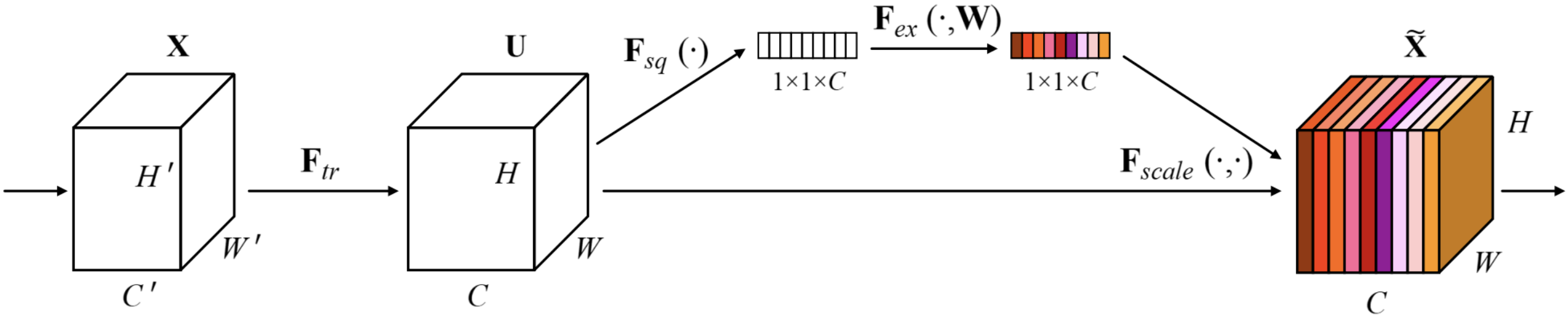
- ResNeXt 将卷积层分成不同通道
- ShuffleNet 通过分组混合不同通道（对移动设备非常有效）

DenseNet

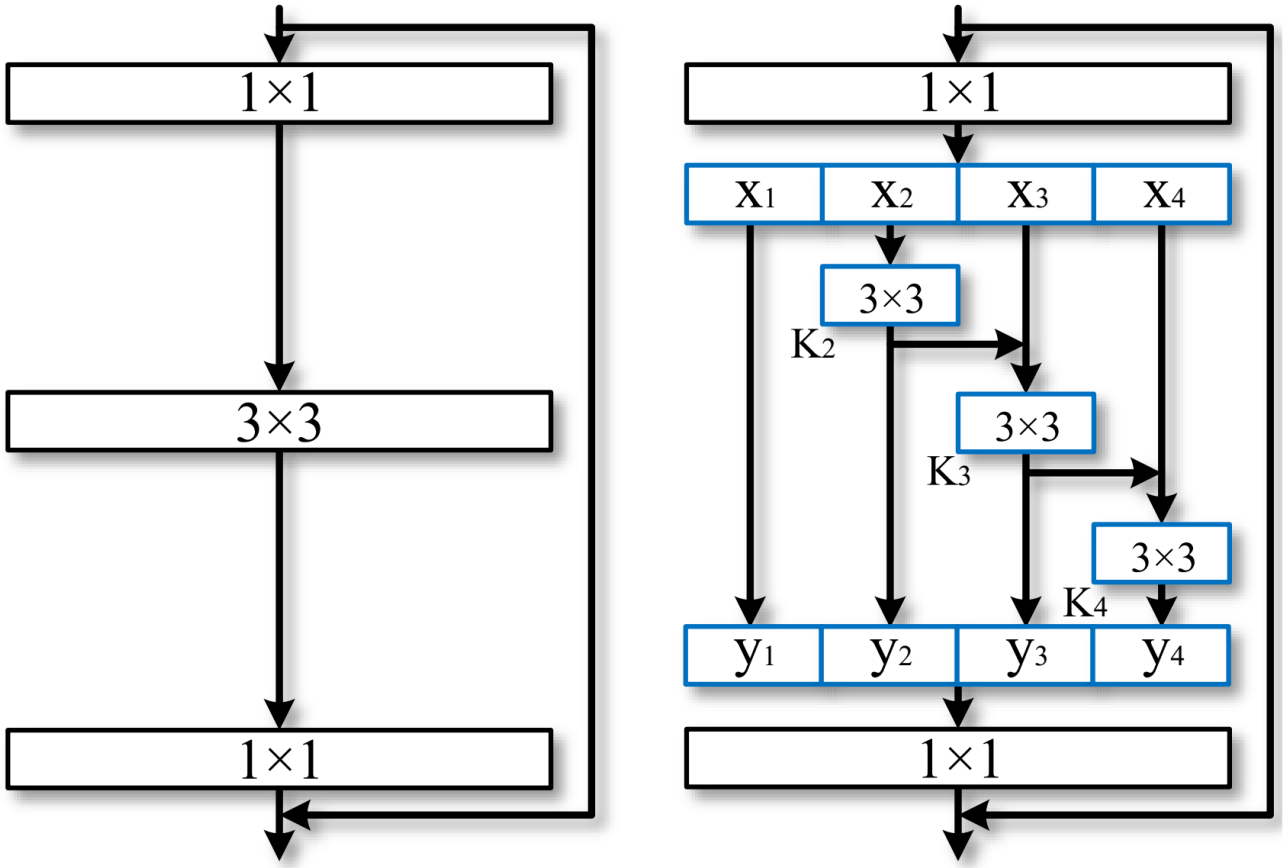


Densely Connected Convolutional Networks, CVPR2017

Squeeze-Excite Net



Res2Net

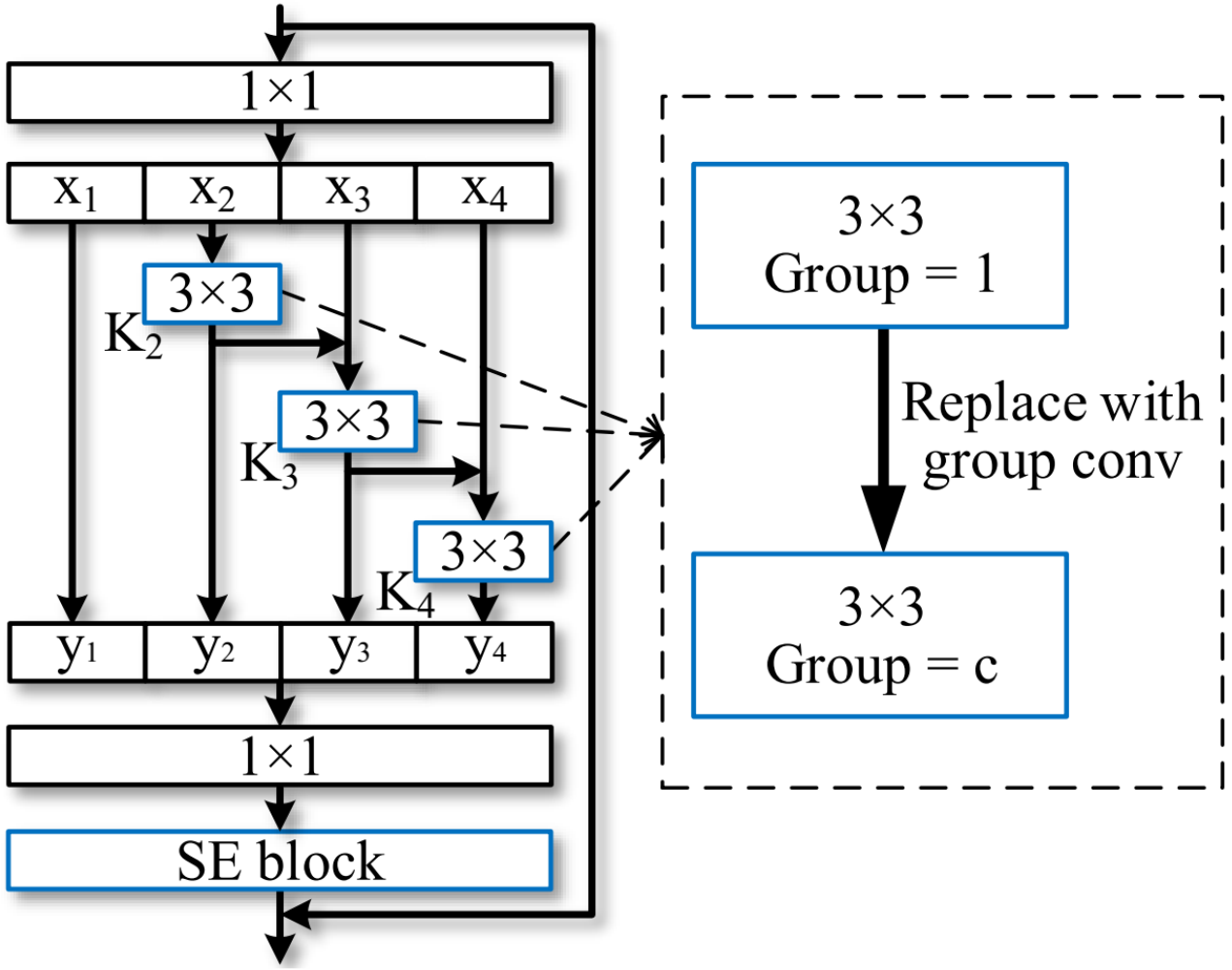


(a) Bottleneck block (b) Res2Net module

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i & i = 1; \\ \mathbf{K}_i(\mathbf{x}_i) & i = 2; \\ \mathbf{K}_i(\mathbf{x}_i + \mathbf{y}_{i-1}) & 2 < i \leqslant s. \end{cases}$$

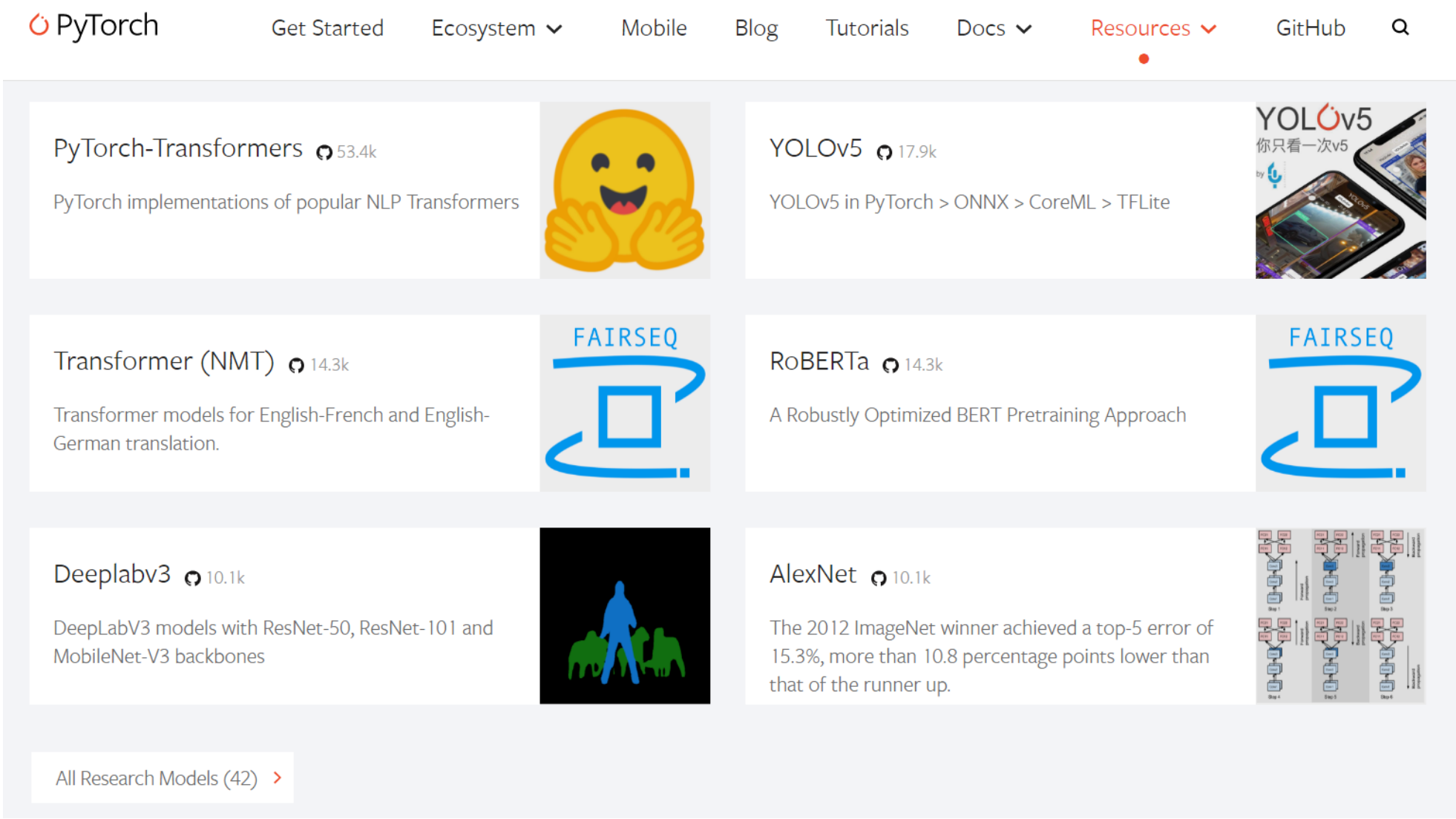
Res2Net: A New Multi-scale Backbone Architecture (TPAMI 2020)

Res2Net


1×1
x1
x2
x3
x4
3×3
K2
3×3
K3
3×3
K4
y1
y2
y3
y4
1×1
SE block
3×3
Group = 1
Replace with
group conv
3×3
Group = c

Hub 预训练模型

https://pytorch.org/hub/



Hub 预训练模型

北京大学 PEKING UNIVERSITY

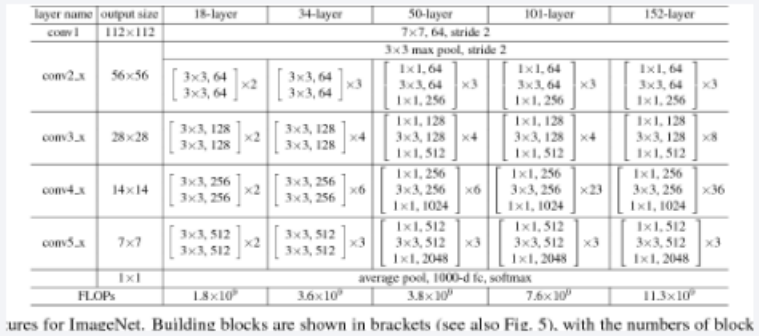
https://pytorch.org/hub/pytorch_vision_resnet/

PyTorch Get Started Ecosystem Mobile Blog Tutorials Docs Resources GitHub

By Pytorch Team

Deep residual networks pre-trained on ImageNet

View on Github Open on Google Colab



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	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

ures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of block

```python
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', pretrained=True)
# or any of these variants
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet34', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet101', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet152', pretrained=True)
model.eval()
```

Hub 预训练模型

北京大学 PEKING UNIVERSITY

https://www.paddlepaddle.org.cn/hub

飞桨 安装 教程 文档 模型库 产品全景 应用案例 直播日历 开发者社区 请输入搜索关键词 论坛 GitHub

PaddleHub

便捷地获取PaddlePaddle生态下的预训练模型，完成模型的管理和一键预测。配合使用Fine-tune API，可以基于大规模预训练模型快速完成迁移学习，让预训练模型能更好地服务于用户特定场景的应用。

- 无需数据和训练，一键模型应用
- 一键模型转服务
- 易用的迁移学习
- 丰富的预训练模型

去Github关注 关注量 7.1k

您只需几行代码即可重复使用经过训练的模型，例如

```
!pip install --upgrade paddlepaddle -i https://mirror.baidu.com/pypi/s
!pip install --upgrade paddlehub -i https://mirror.baidu.com/pypi/simp

import paddlehub as hub

lac = hub.Module(name="lac")
test_text = ["今天是个好天气。"]

results = lac.cut(text=test_text, use_gpu=False, batch_size=1, return_
print(results)
#{'word': ['今天', '是', '个', '好天气', '。'], 'tag': ['TIME', 'v', 'q
```

快速上手

了解如何使用PaddleHub，几行代码完成深度学习任务

官方教程

官方提供精选项目，涵盖ocr、口罩检测、文本生成、视频分类等应用

创意项目

开发者基于PaddleHub实现的众多创意项目，快去关注吧

Hub 预训练模型

PaddleHub一键动物识别

https://aistudio.baidu.com/aistudio/projectdetail/437648

PaddleHub一键动物识别

图像分类已经应用于各个领域当中，如ImageNet中花类识别。现PaddleHub已将定制打造的动物识别模型resnet50_vd_animals和mobilenet_v2_animals开源， 并且支持一键完成动物识别，适用于拍照识图类APP中。

一、定义待预测数据

以本示例中文件夹下图片为待预测图片。

In [1]
```
!pip install paddlehub==1.6.2 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

In [10]
```
# 待预测图片
test_img_path = ["./1.jpg", "./2.jpg", "./3.jpg", "./4.jpg", "./5.jpg"]


import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

Hub 预训练模型

AI抠图及图片合成

https://aistudio.baidu.com/aistudio/projectdetail/341116

PaddleHub抠图比赛

本示例用DeepLabv3+模型完成一键抠图。DeepLabv3+ 是Google DeepLab语义分割系列网络的最新作，其前作有 DeepLabv1，DeepLabv2, DeepLabv3。在最新作中，作者通过encoder-decoder进行多尺度信息的融合，同时保留了原来的空洞卷积和ASSP层， 其骨干网络使用了Xception模型，提高了语义分割的健壮性和运行速率，在 PASCAL VOC 2012 dataset取得新的state-of-art performance。在完成一键抠图之后，通过图像合成，实现扣图比赛任务

NOTE: 如果您在本地运行该项目示例，需要首先安装PaddleHub。如果您在线运行，需要首先fork该项目示例。之后按照该示例操作即可。

一、定义待抠图照片

以本示例中文件夹下meditation.jpg为待预测图片

In [ ]

```
!pip install paddlehub==1.7.1 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

Hub 预训练模型

北京大学 PEKING UNIVERSITY

一张黑白老照片的奇妙之旅　　https://aistudio.baidu.com/aistudio/projectdetail/1264565

飞桨 AI Studio　项目　数据集　课程　比赛　认证　更多　论坛　访问飞桨官网　中 | En

精 PaddleHub：一张黑白老照片的奇妙之旅

Fork 608　喜欢 47　分享

通过一张老照片的视角，体验PaddleHub中各种图像处理的预训练模型，如图片上色、超分辨率、风格转换等等

寂寞你快进去　AI Studio 经典版　2.0.2　Python3　初级　计算机视觉　深度学习　2020-11-24 20:27:48

版本内容　Fork记录　评论(19)　运行一下

初稿 2020-11-25 00:50:34

新版Notebook- BML CodeLab上线，fork后可修改项目版本进行体验

请选择预览文件

- 序
- 准备
- 启程
  - 第一站：换身漂亮衣服
  - 第二站：画个精细的妆容

序

- 嗨，你好！我是一张有年代感的黑白老照片
- 今天我将乘着PaddleHub这艘飞船，体验一次奇妙的旅程

图像超分辨率（super resolution）

图像超分辨率（super resolution）

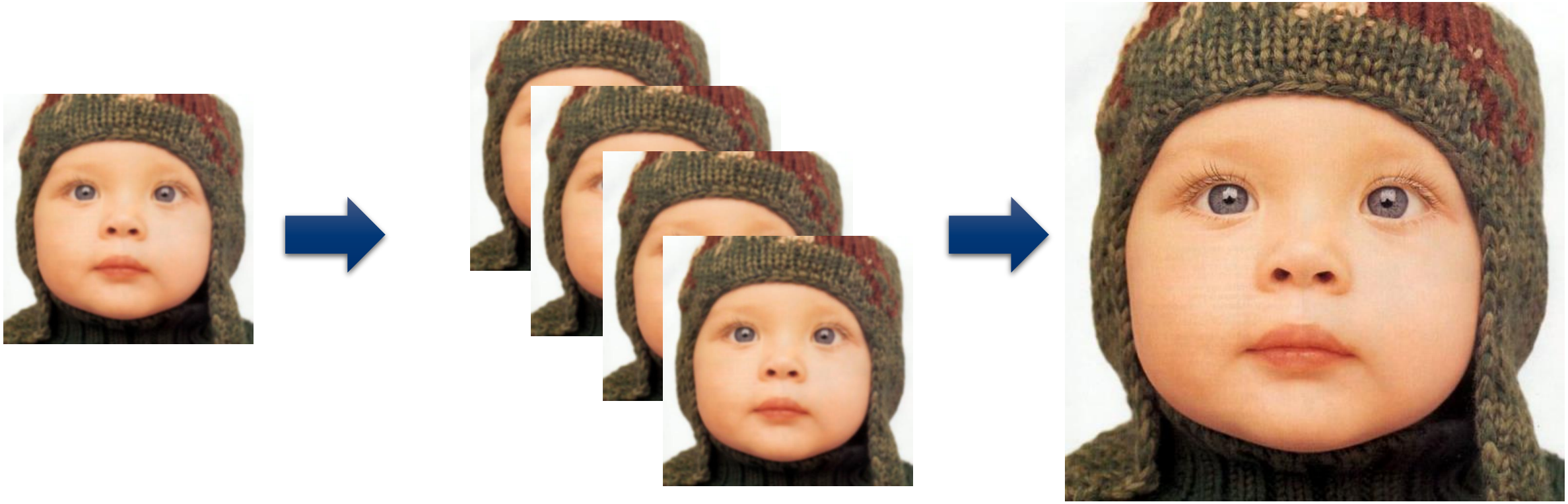
北京大学 PEKING UNIVERSITY

第一种方式：

图像超分辨率（super resolution）

北京大学 PEKING UNIVERSITY

第二种方式：

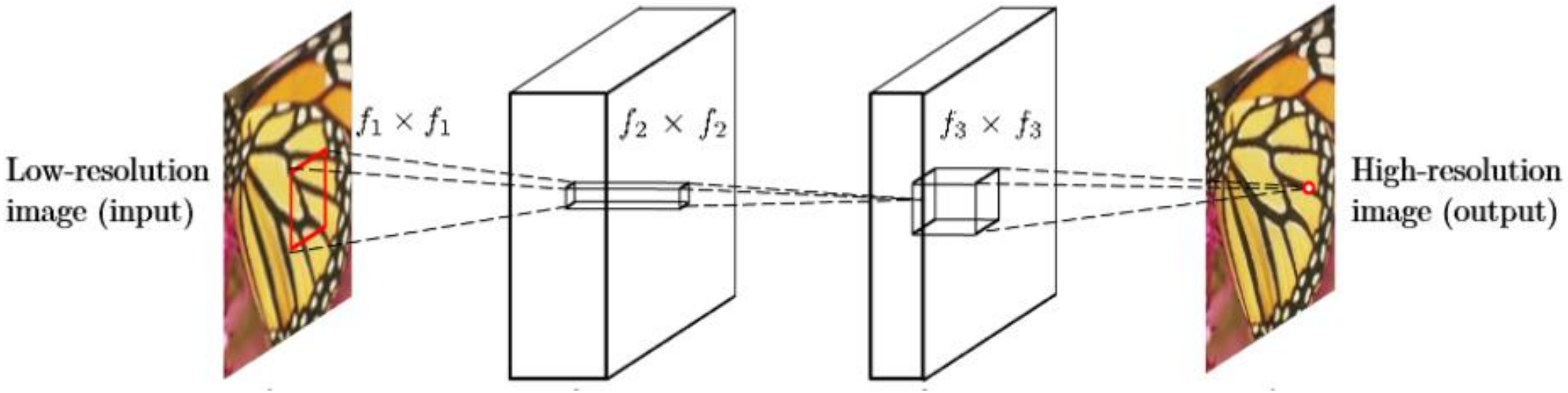


图像超分辨率（super resolution）

回归问题

图像超分辨率（super resolution）

- 滤波器模板固定 V.S. 利用神经网络学习滤波器系数
- 需要对图像有先验认知 V.S. 不需要任何图像先验模型
- 一般为浅层操作 V.S. 建立深层神经网络



SRCNN

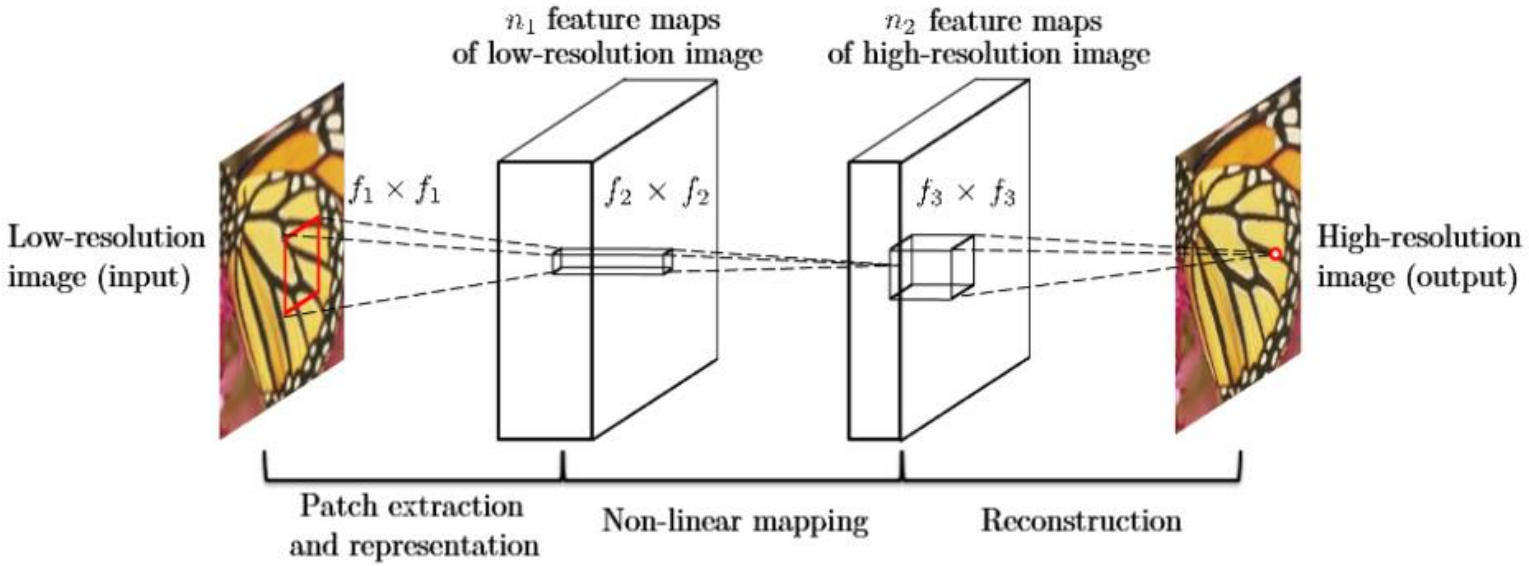
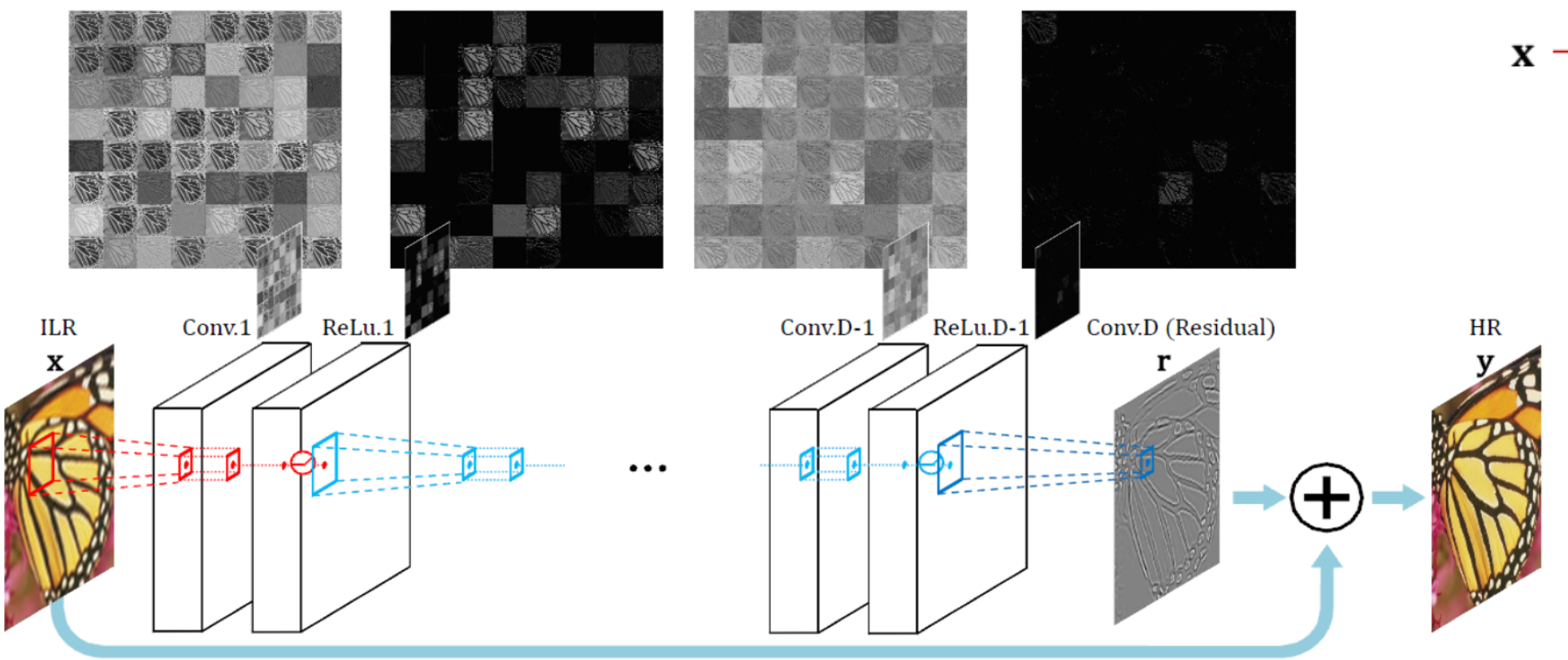


Fig. 2. Given a low-resolution image $\mathbf{Y}$, the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps nonlinearly to high-resolution patch representations. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image $F(\mathbf{Y})$.

SRCNN/IRCNN/DnCNN

256×256 (input, bicubic interpolation) → 256 × 256 × 64 (feature map of Conv1) → 256 × 256 × 32 (feature map of Conv2) → 256 × 256 (output)

VDSR



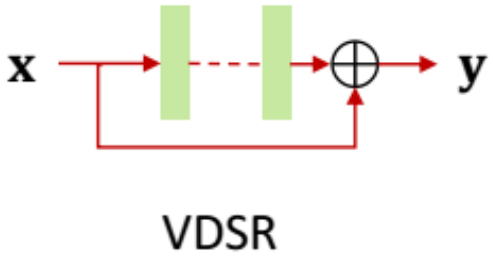
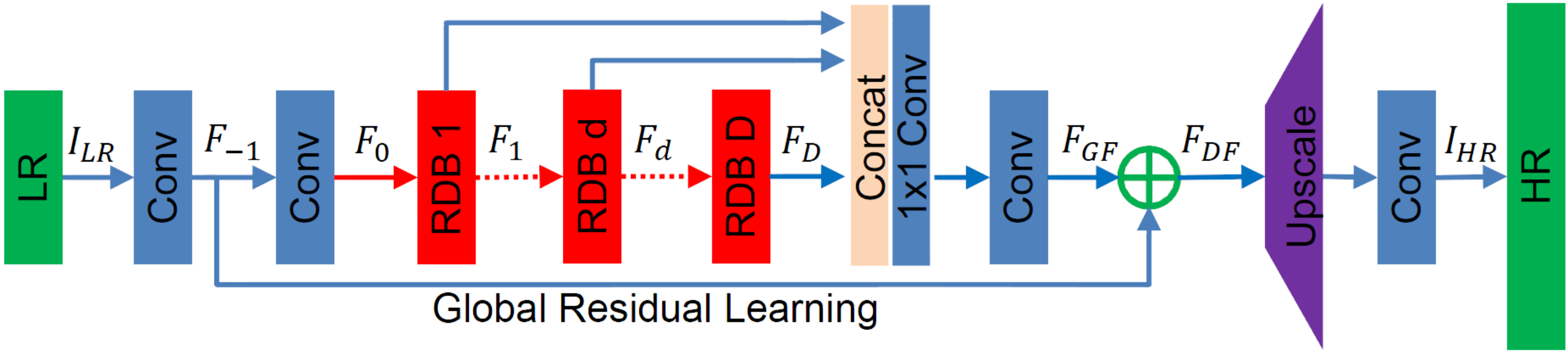
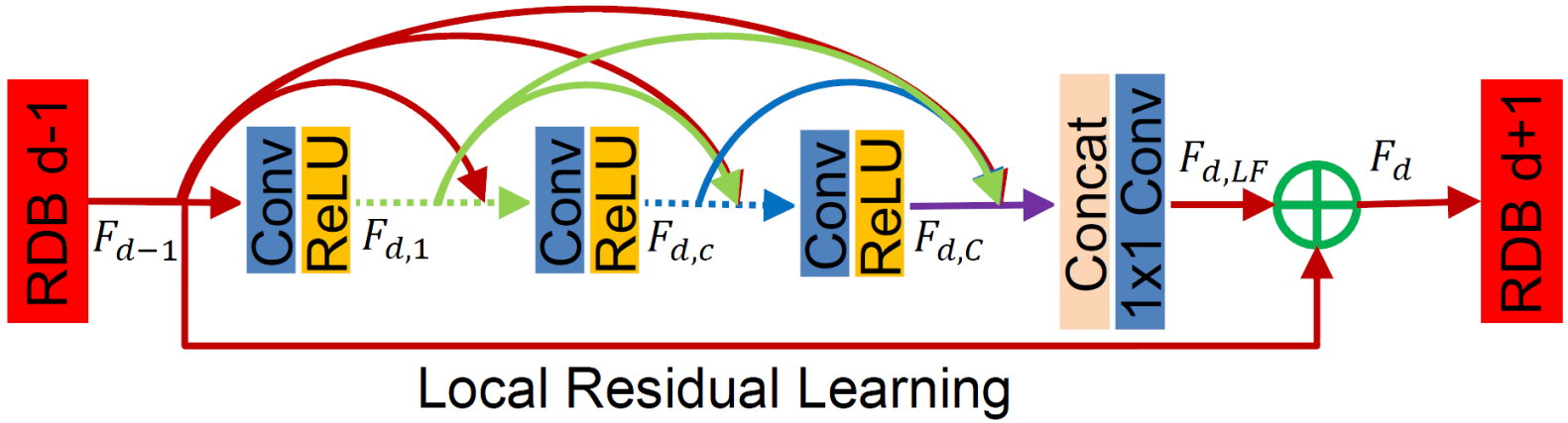


Figure 2: Our Network Structure. We cascade a pair of layers (convolutional and nonlinear) repeatedly. An interpolated low-resolution (ILR) image goes through layers and transforms into a high-resolution (HR) image. The network predicts a residual image and the addition of ILR and the residual gives the desired output. We use 64 filters for each convolutional layer and some sample feature maps are drawn for visualization. Most features after applying rectified linear units (ReLu) are zero.

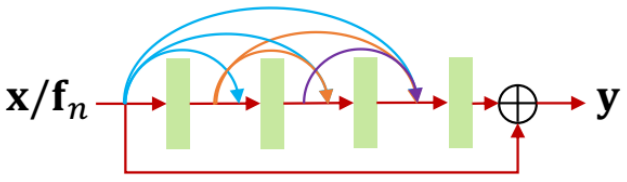
RDN



The architecture of the proposed residual dense network (RDN).

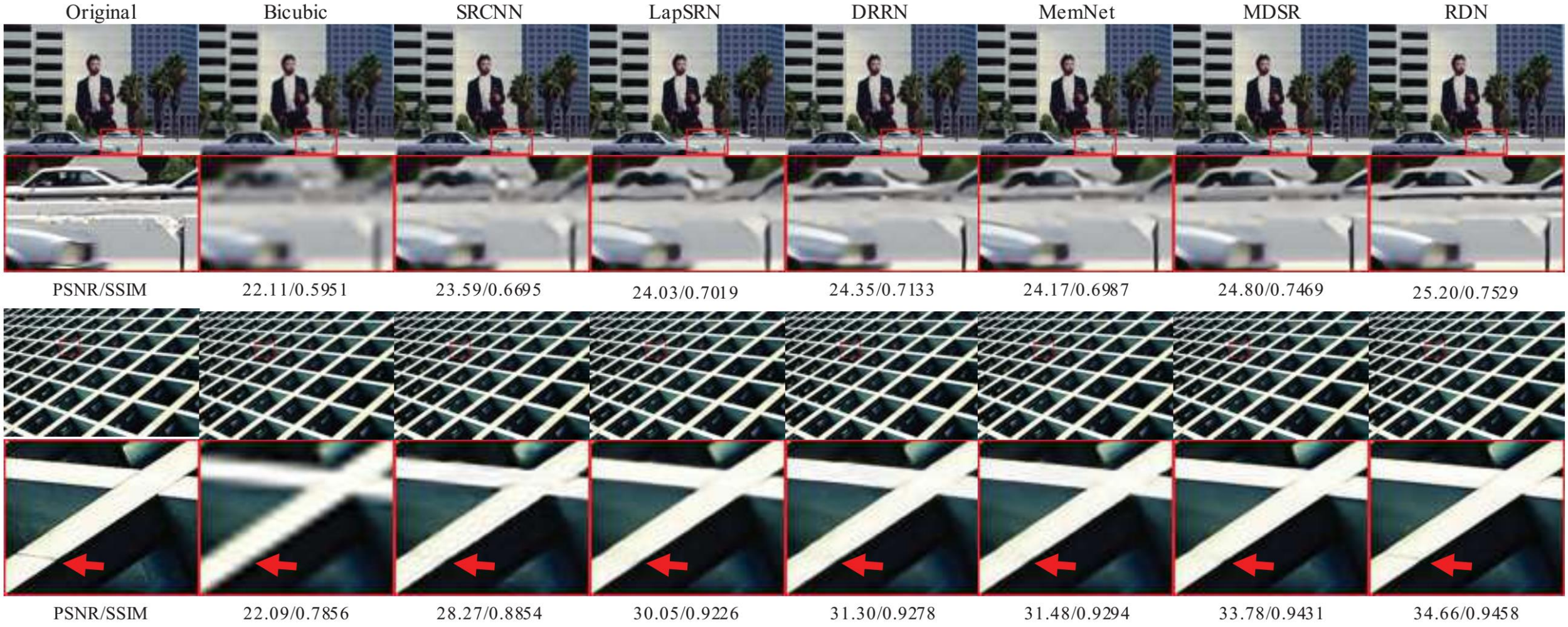


Residual dense block (RDB) architecture.



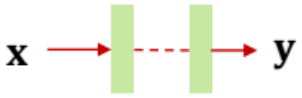
Residual Dense Block (RDN)

RDN

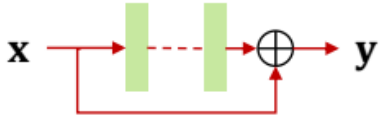


Residual Dense Network for Image Super-Resolution, CVPR 2018.

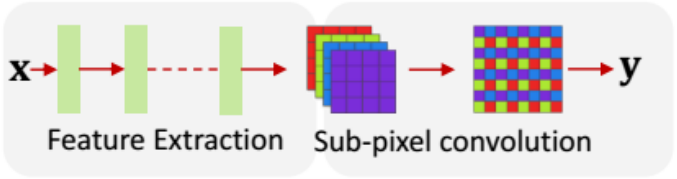
超分辨率网络概览-1



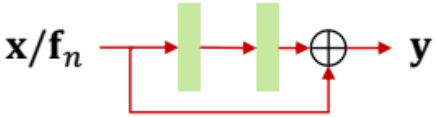
SRCNN/IRCNN/DnCNN



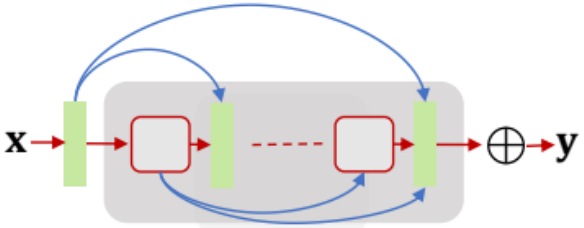
VDSR



ESPCN

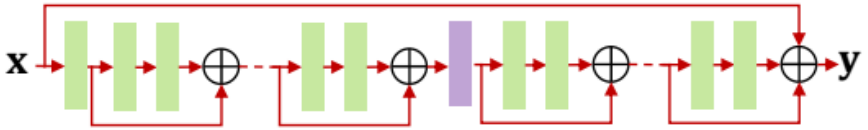


Residual block
(EDSR/MDSR/SR-ResNet)

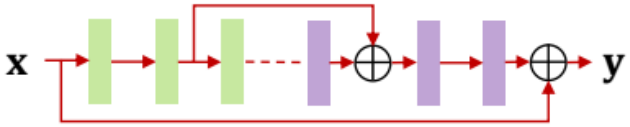


CARN

CARN Block



BTSRN

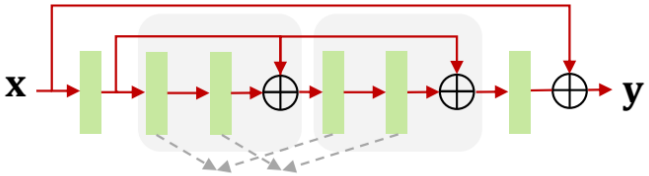


RED-Net

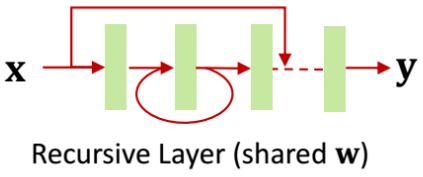
Convolutional Layer splitting
Convolution Layer (generally followed by ReLU)
Convolution Transpose Layer (generally followed by ReLU)
Concatenation of Layers
Unfolded recursive unit
Element-wise addition
Unfolded Block or unit
Sigmoid Function
Element-wise multiplication
Group Convolution Layer
Global Feature Pooling Layer
Element-wise subtraction

https://arxiv.org/pdf/1904.07523.pdf

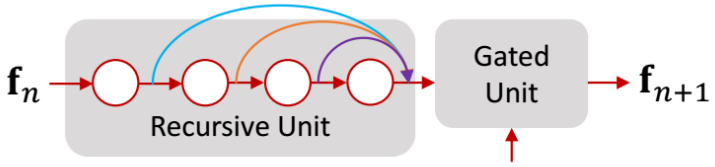
超分辨率网络概览-2



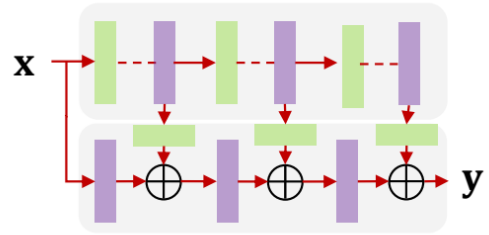
DRRN



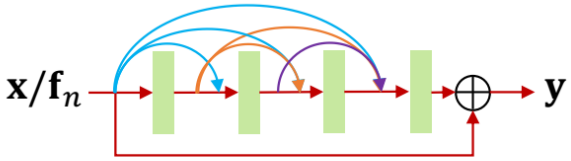
DRCN



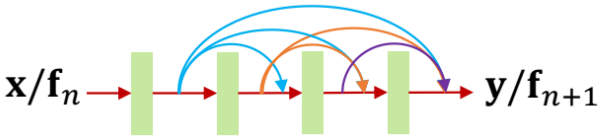
Memory block (MemNet)



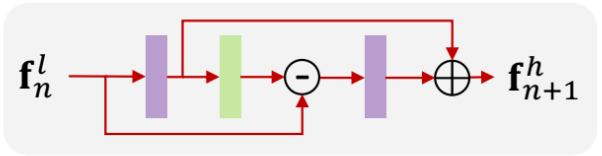
LapSRN



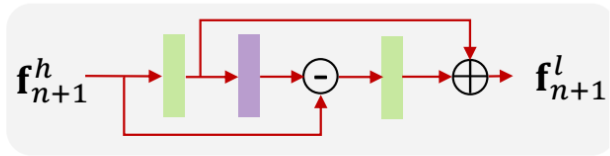
Residual Dense Block (RDN)



Dense Block (SRDenseNet)

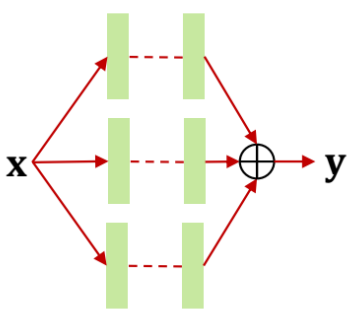


DBPN

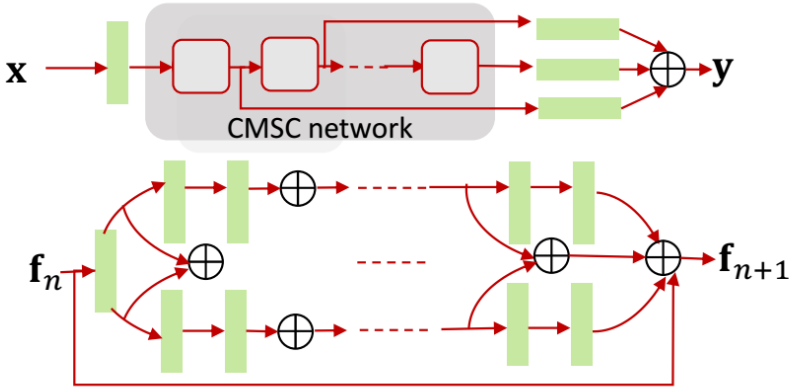


Convolutional Layer splitting
Convolution Layer (generally followed by ReLU)
Convolution Transpose Layer (generally followed by ReLU)
Concatenation of Layers
Unfolded recursive unit
Element-wise addition
Unfolded Block or unit
Sigmoid Function
Element-wise multiplication
Group Convolution Layer
Global Feature Pooling Layer
Element-wise subtraction

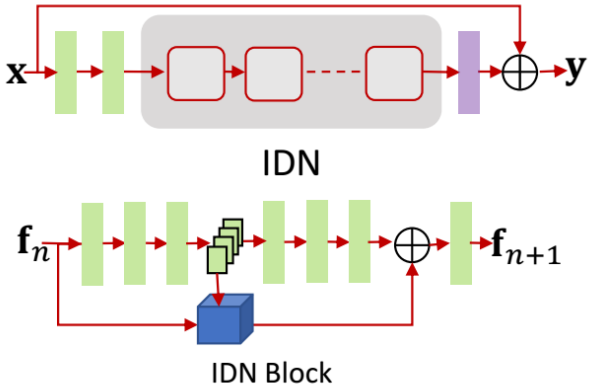
超分辨率网络概览-3

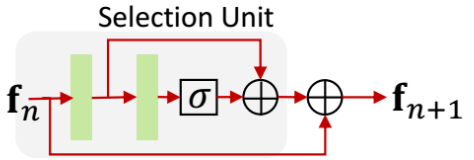


CNF

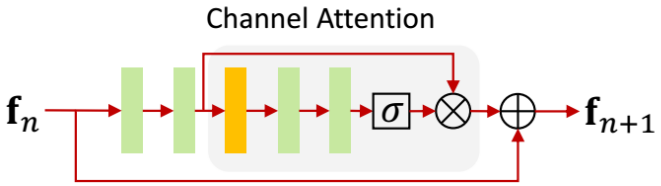


CMSC Block

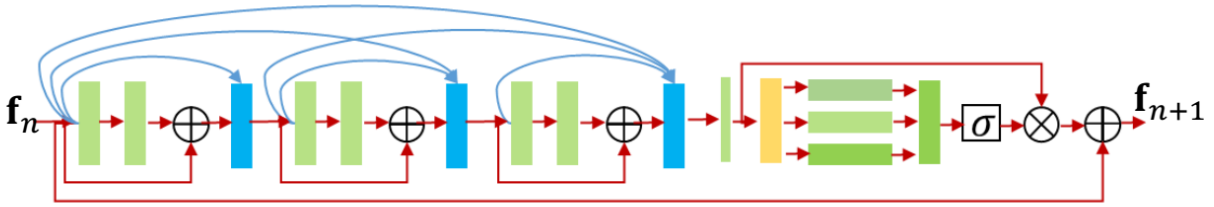




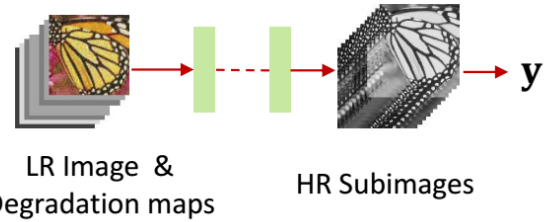
SelNet Block



RCAN/RAM Block



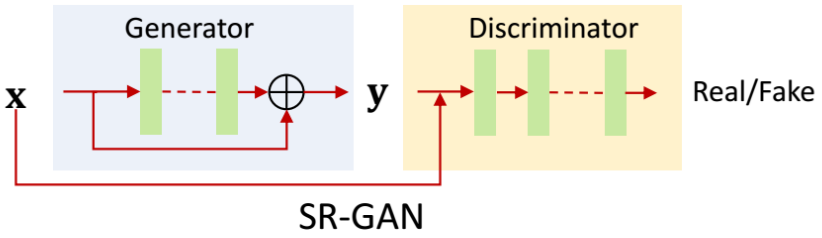
DRLN Block

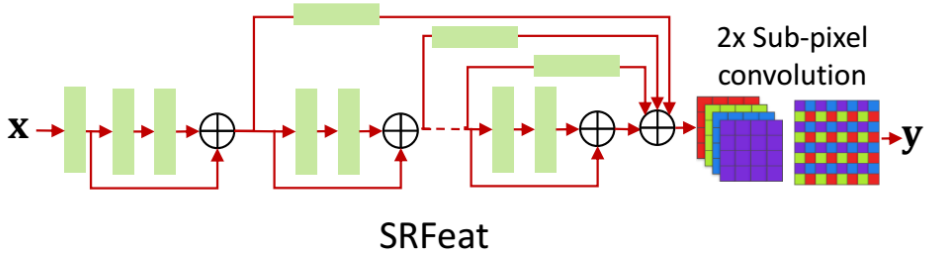


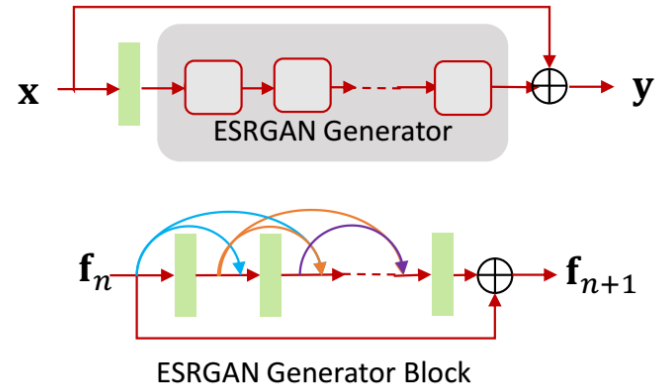
SRMD

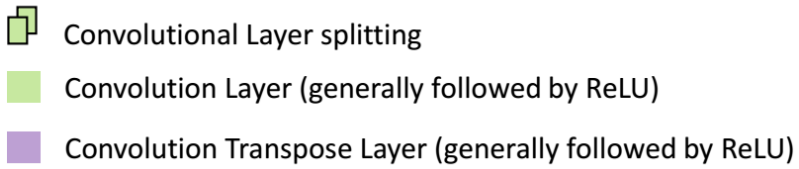
Convolutional Layer splitting
Convolution Layer (generally followed by ReLU)
Convolution Transpose Layer (generally followed by ReLU)
Concatenation of Layers
Unfolded recursive unit
Element-wise addition
Unfolded Block or unit
Sigmoid Function
Element-wise multiplication
Group Convolution Layer
Global Feature Pooling Layer
Element-wise subtraction

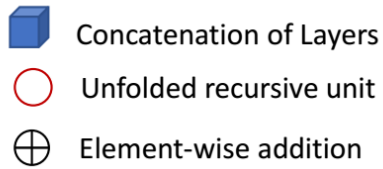
超分辨率网络概览-4

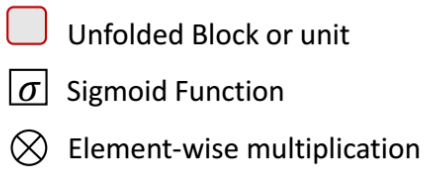

Generator
Discriminator
x
y
Real/Fake
SR-GAN

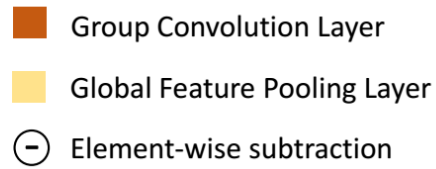

2x Sub-pixel convolution
x
y
SRFeat


x
y
ESRGAN Generator
f_n
f_{n+1}
ESRGAN Generator Block


Convolutional Layer splitting
Convolution Layer (generally followed by ReLU)
Convolution Transpose Layer (generally followed by ReLU)


Concatenation of Layers
Unfolded recursive unit
Element-wise addition


Unfolded Block or unit
Sigmoid Function
Element-wise multiplication


Group Convolution Layer
Global Feature Pooling Layer
Element-wise subtraction

本次作业

北京大学 PEKING UNIVERSITY

Github或者主页下载运行一个超分算法，获得结果试着训练一两个Epoch，给出超分结果

交流&问题?