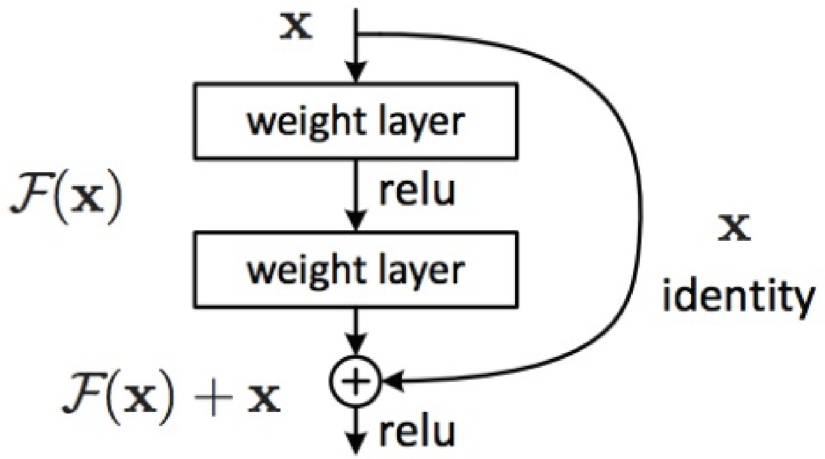
PyTorch

深度残差网络

主讲人：龙良曲

ResNet

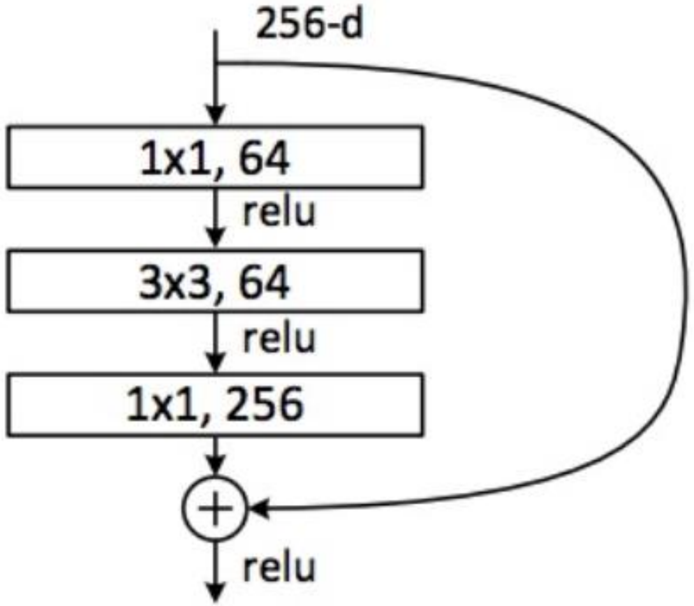
- The residual module
  - Introduce *skip* or *shortcut* connections (existing before in various forms in literature)
  - Make it easy for network layers to represent the identity mapping
  - For some reason, need to skip at least two layers



Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,
Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

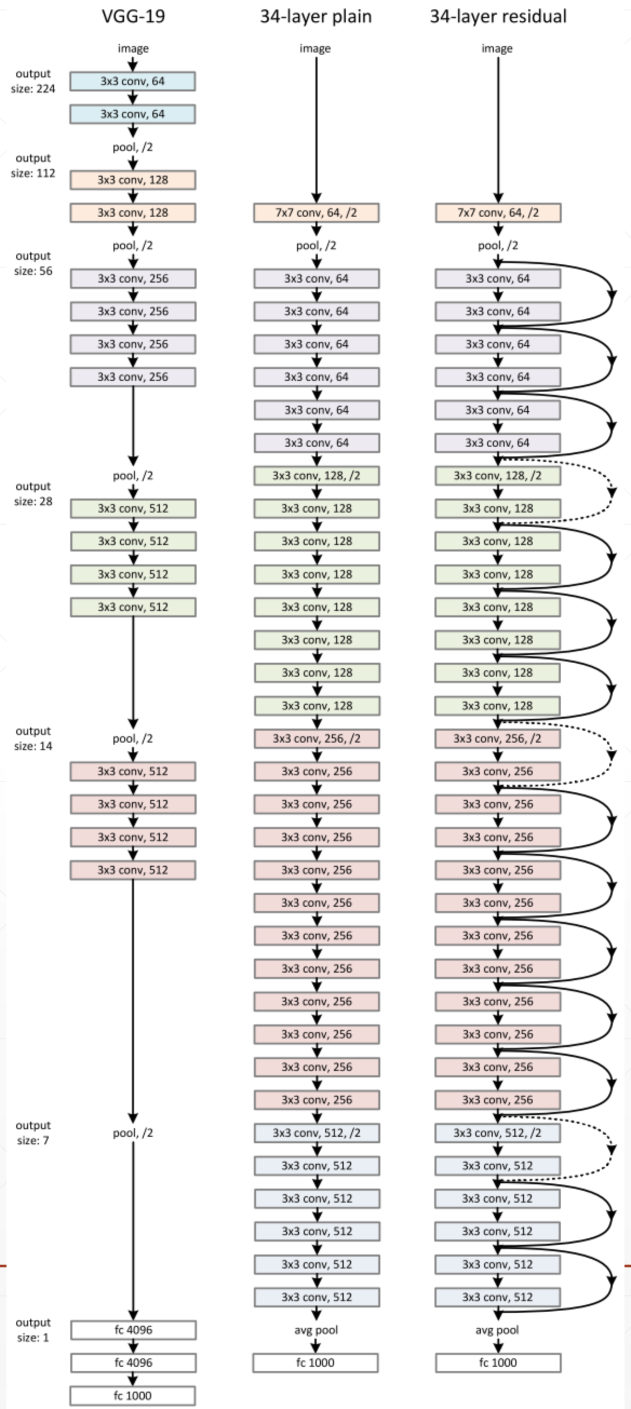
ResNet

Deeper residual module (bottleneck)



- Directly performing 3x3 convolutions with 256 feature maps at input and output: 256 x 256 x 3 x 3 ~ 600K operations
- Using 1x1 convolutions to reduce 256 to 64 feature maps, followed by 3x3 convolutions, followed by 1x1 convolutions to expand back to 256 maps:
  256 x 64 x 1 x 1 ~ 16K
  64 x 64 x 3 x 3 ~ 36K
  64 x 256 x 1 x 1 ~ 16K
  Total: ~70K

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)



VGG-19

image

output size: 224

3x3 conv, 64

3x3 conv, 64

pool, /2

output size: 112

3x3 conv, 128

3x3 conv, 128

pool, /2

output size: 56

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

pool, /2

output size: 28

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

pool, /2

output size: 14

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

pool, /2

output size: 7

output size: 1

fc 4096

fc 4096

fc 1000

34-layer plain

image

7x7 conv, 64, /2

pool, /2

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 128, /2

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 256, /2

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 512, /2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

avg pool

fc 1000

34-layer residual

image

7x7 conv, 64, /2

pool, /2

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 64

3x3 conv, 128, /2

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 128

3x3 conv, 256, /2

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 256

3x3 conv, 512, /2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

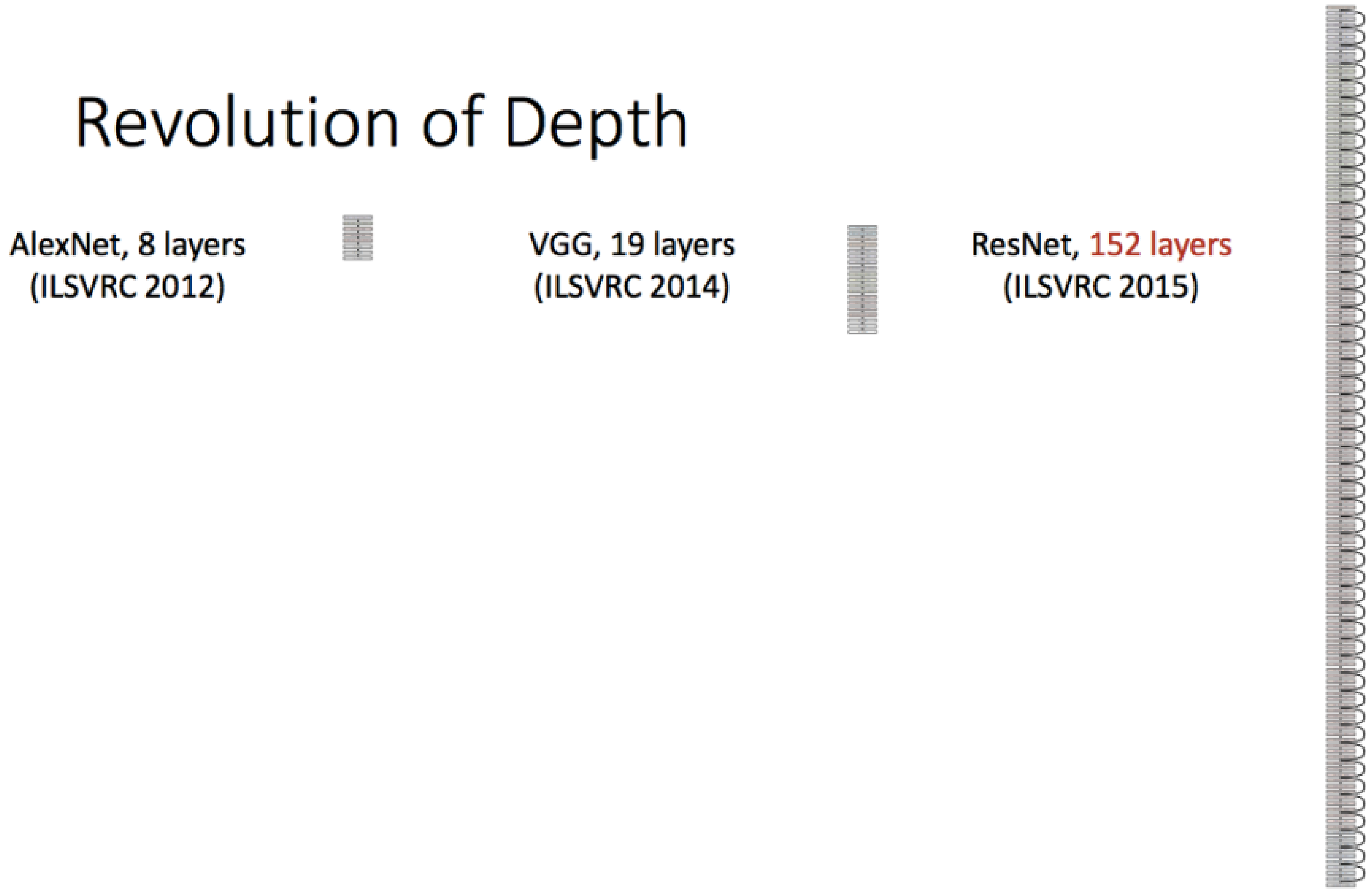
3x3 conv, 512

3x3 conv, 512

avg pool

fc 1000

ResNet: ILSVRC 2015 winner



Revolution of Depth

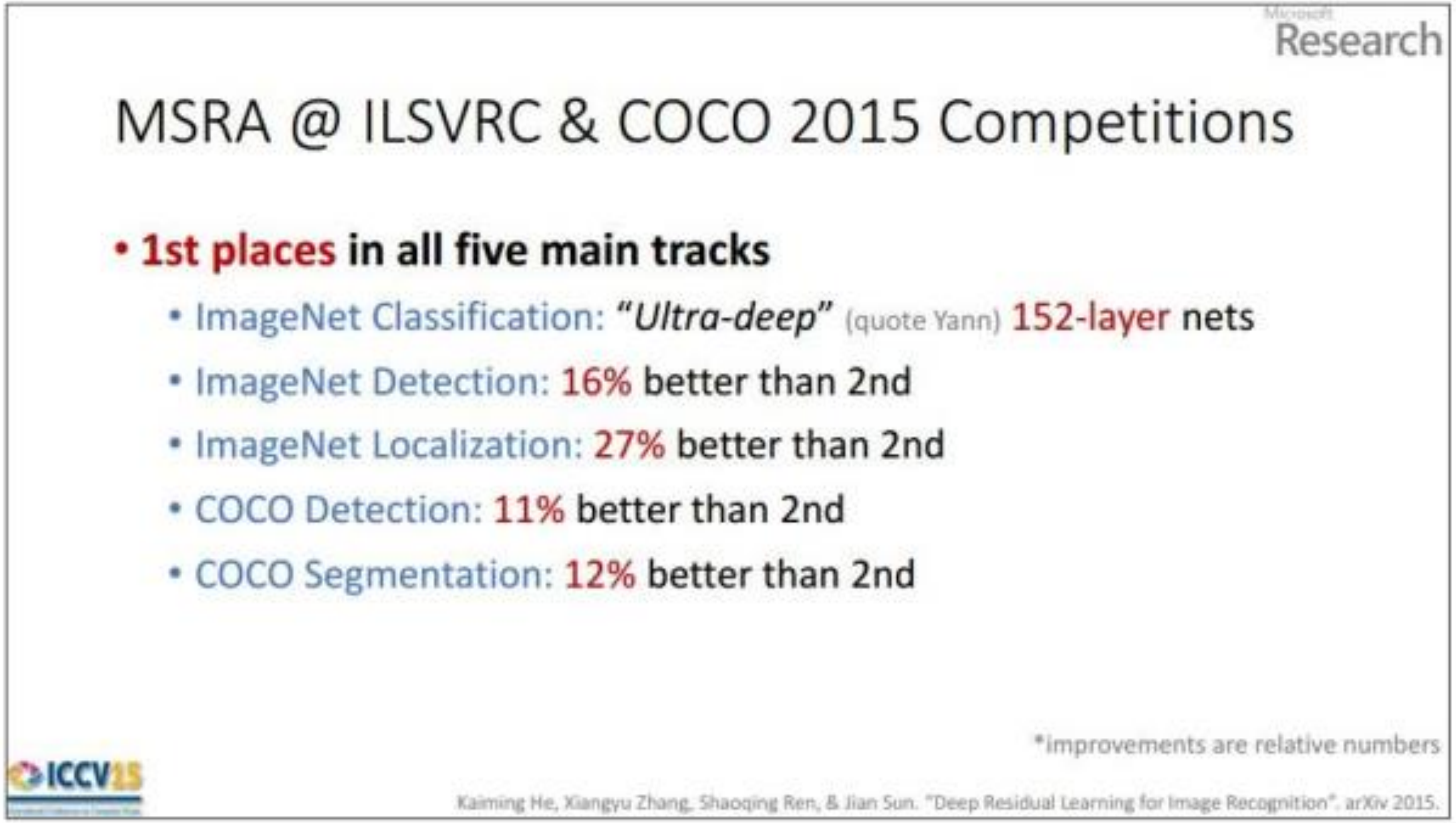
AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,
Deep Residual Learning for Image Recognition, CVPR 2016

BOOM!



Microsoft Research

MSRA @ ILSVRC & COCO 2015 Competitions

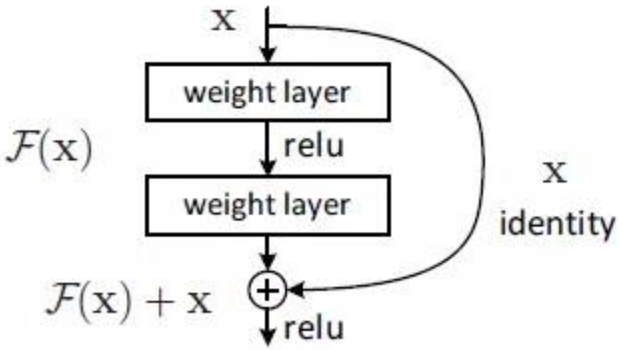
- **1st places in all five main tracks**
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

ICCV15

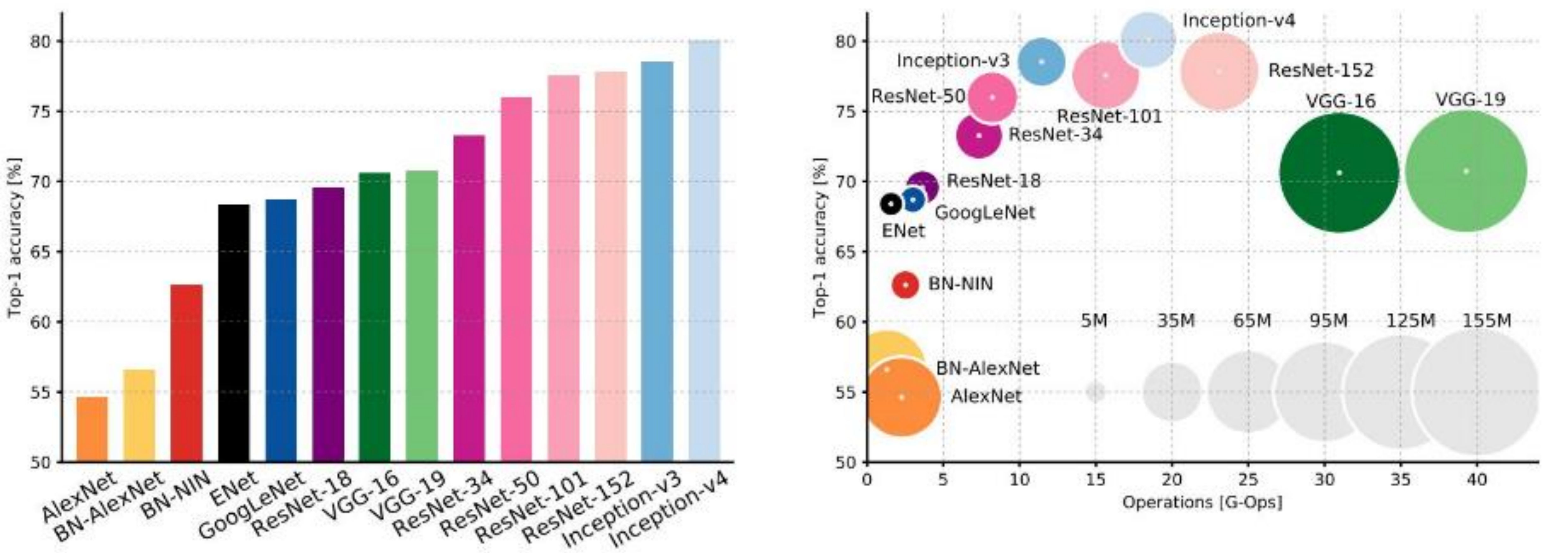
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

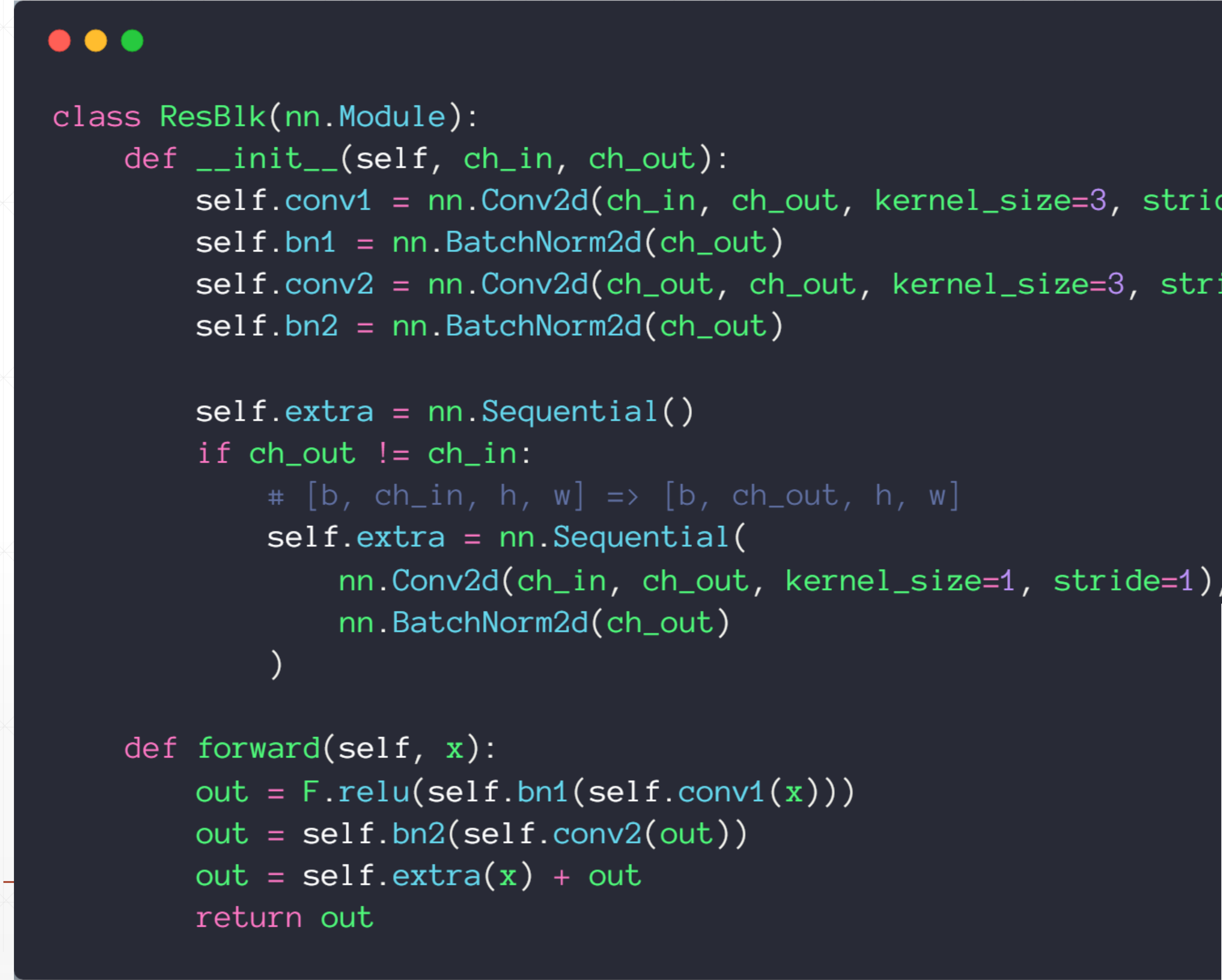
Why call Residual?


x
weight layer
$\mathcal{F}(\mathrm{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathrm{x}) + \mathrm{x}$
relu

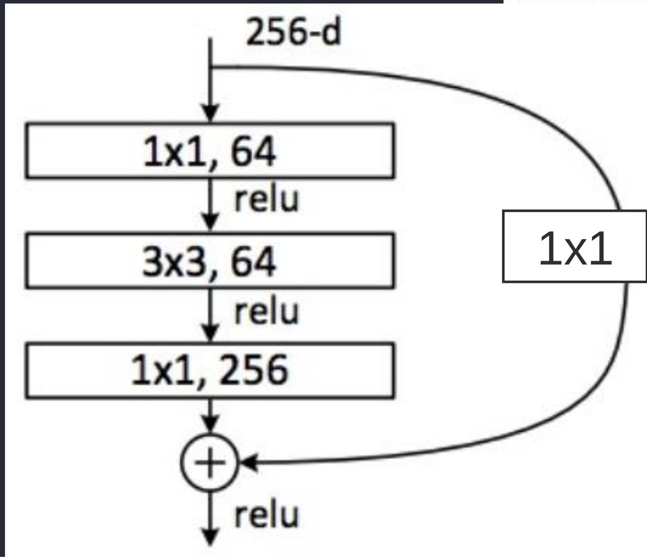
$$\mathcal{F}(\mathrm{x}) := \mathcal{H}(\mathrm{x}) - \mathrm{x}$$

https://www.jianshu.com/p/a71f0085157c

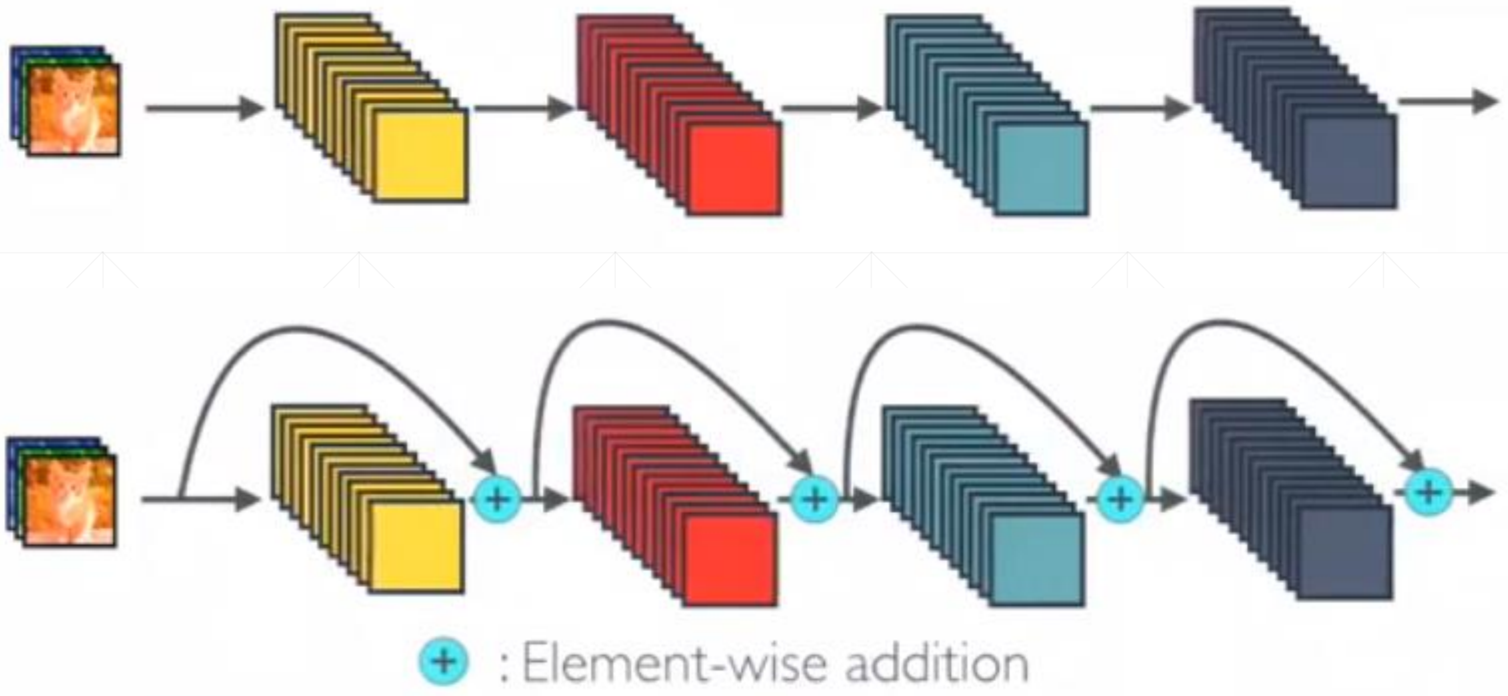

Top-1 accuracy [%]
80
75
70
65
60
55
50
AlexNet
BN-AlexNet
BN-NIN
ENet
GoogLeNet
ResNet-18
VGG-16
VGG-19
ResNet-34
ResNet-50
ResNet-101
ResNet-152
Inception-v3
Inception-v4
Top-1 accuracy [%]
Inception-v4
Inception-v3
ResNet-152
ResNet-50
ResNet-101
VGG-16
VGG-19
ResNet-34
ResNet-18
GoogLeNet
ENet
BN-NIN
5M
35M
65M
95M
125M
155M
BN-AlexNet
AlexNet
0
5
10
15
20
25
30
35
40
Operations [G-Ops]

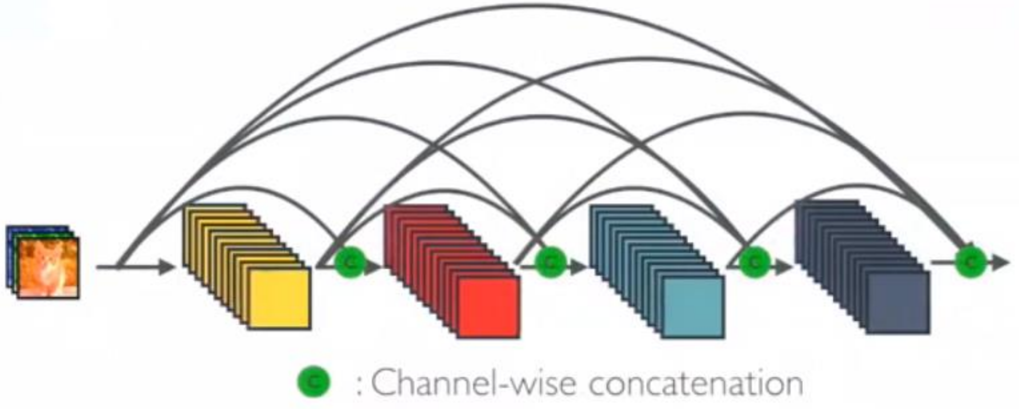


```
class ResBlk(nn.Module):
    def __init__(self, ch_in, ch_out):
        self.conv1 = nn.Conv2d(ch_in, ch_out, kernel_size=3, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(ch_out)
        self.conv2 = nn.Conv2d(ch_out, ch_out, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(ch_out)

        self.extra = nn.Sequential()
        if ch_out != ch_in:
            # [b, ch_in, h, w] => [b, ch_out, h, w]
            self.extra = nn.Sequential(
                nn.Conv2d(ch_in, ch_out, kernel_size=1, stride=1),
                nn.BatchNorm2d(ch_out)
            )

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out = self.extra(x) + out
        return out
```



DenseNet





下一课时

nn.Module

Thank You.