CS3232 Fundamental of Deep learning

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Credits: Parts of the contents are based on material by Dr. Andrew Ng and other resources on the Internet

Classic networks (architectures)

- LeNet-5 (60K parameters)
- AlexNet (60M parameters)
- VGG (VGG 16 has ~ 138M parameters)

ResNet (152 layers) – some tricks on how to train them effectively

AlexNet: Introduced ReLU, dropout, and data augmentation; won ILSVRC 2012.

ZF-Net: Improved on AlexNet with better visualization techniques; won ILSVRC 2013.

VGGNet: Emphasized depth with small filters; influential design; top performance in ILSVRC 2014.

GoogLeNet: Introduced Inception modules; reduced parameters; won ILSVRC 2014.

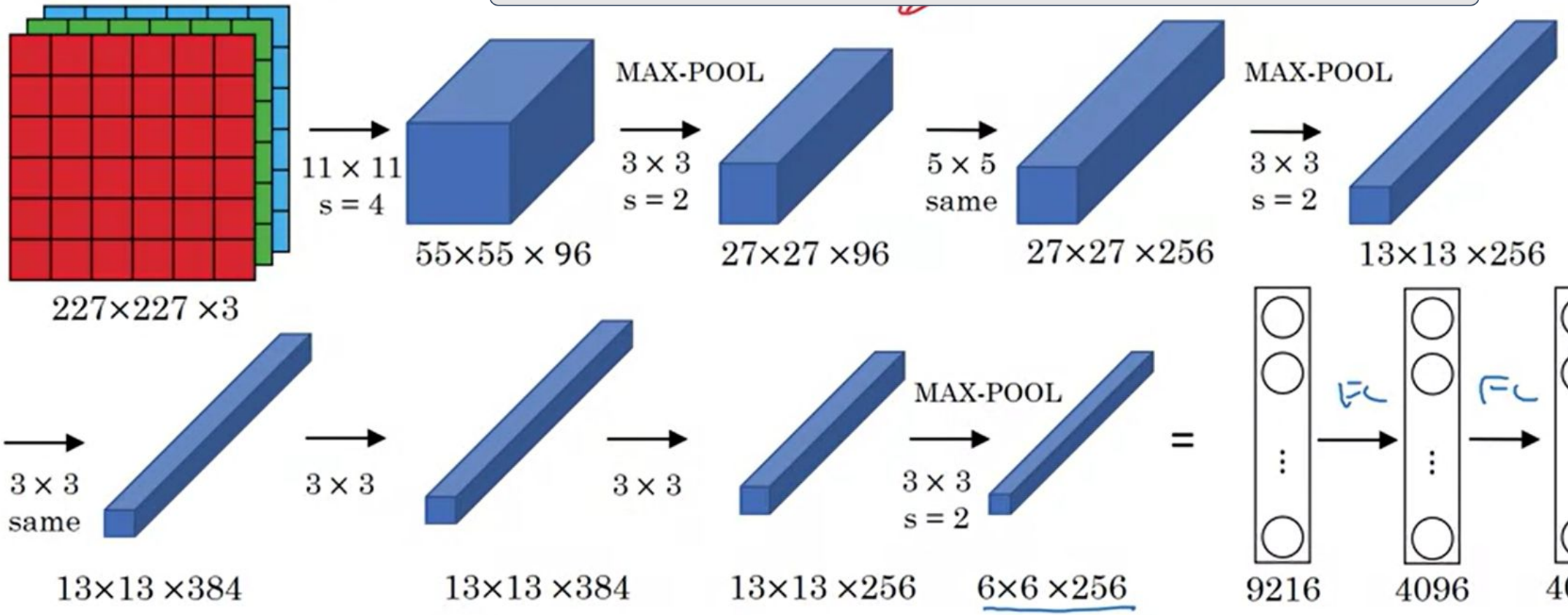
ResNet: Introduced residual connections; enabled very deep networks; won ILSVRC 2015.

Transfer Learning: Using pre-trained models to leverage learned features for new tasks.

Fine-Tuning: Adapting pre-trained models to specific tasks by continuing training on new data

AlexNet

Led to increased use of deep learning, even beyond computer vision



- Similar to LeNet, but much bigger.
- ReLU
- Multiple GPUs.
- Local Response Normalization.

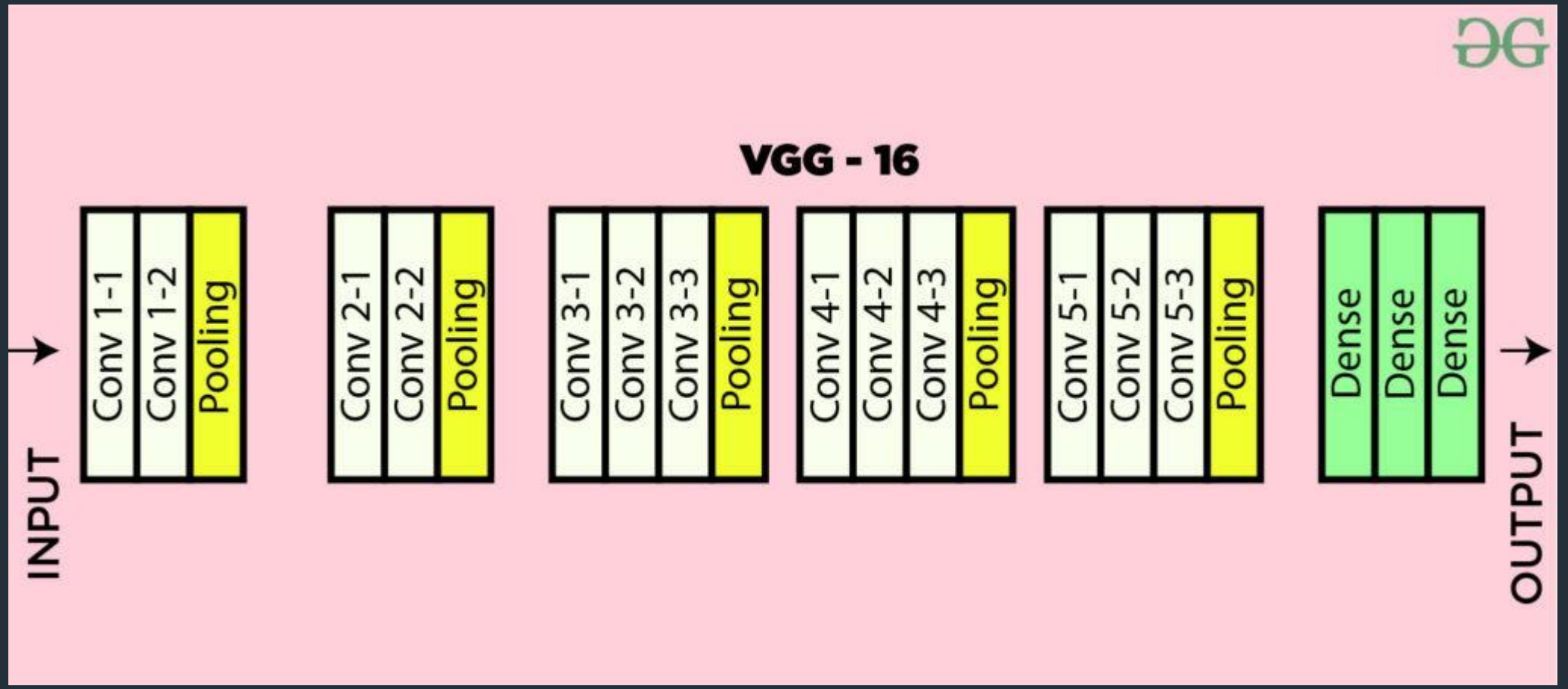
60M parameters

Not used any longer (as it does not help much)

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

Andrew Ng

VGG Net


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-16

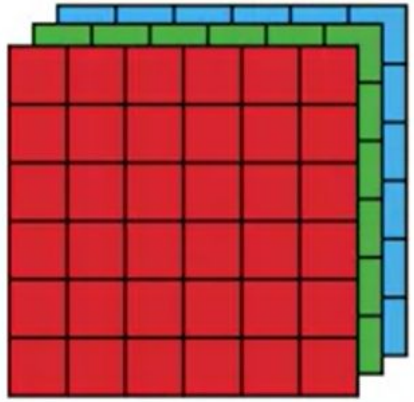
The VGG-16 architecture is a deep convolutional neural network (CNN) designed for image classification tasks. It was introduced by the Visual Geometry Group at the University of Oxford. VGG-16 is characterized by its simplicity and uniform architecture, making it easy to understand and implement.

The VGG-16 configuration typically consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. These layers are organized into blocks, with each block containing multiple convolutional layers followed by a max-pooling layer for downsampling.

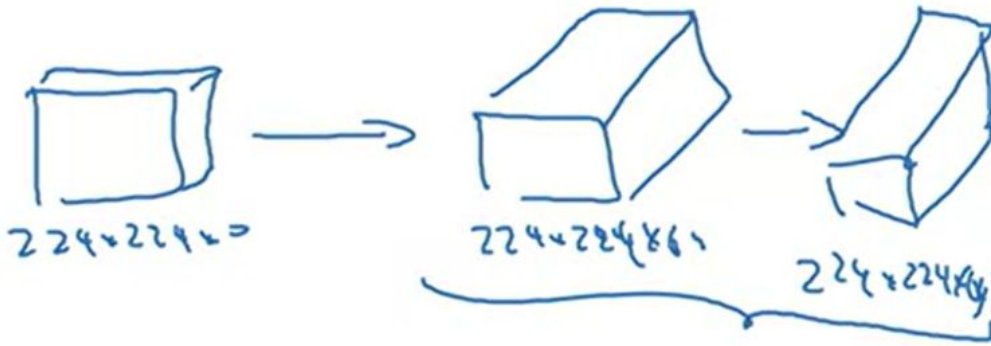
VGG - 16

16 layers with weights

CONV = 3×3 filter, s = 1, same MAX-POOL = 2×2 , s = 2



Simplified architecture (uniform)



224×224 ×3 → [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

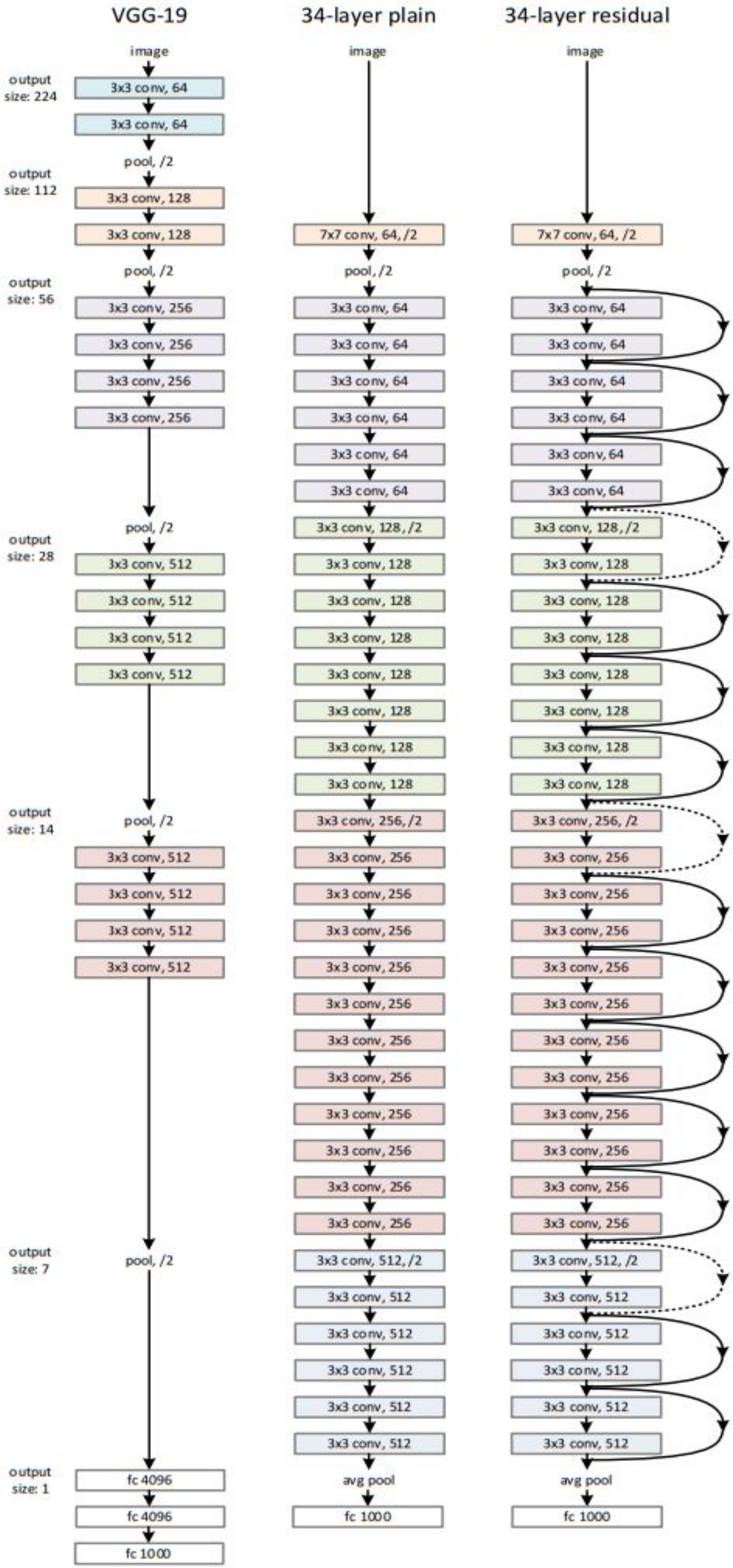
Doubled till 512 (while n_H, n_W are decreased)

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] × 3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

Andrew Ng


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

VGG-16 vs. VGG-19

- Since VGG-16 almost does as well as VGG-19, it is used by many

Model	Year	Architecture	Key Features	Impact
AlexNet	2012	8 layers (5 conv, 3 fully connected)	ReLU, dropout, data augmentation, won ILSVRC 2012	Popularized deep learning in computer vision
ZF-Net	2013	Similar to AlexNet with modified filter sizes	Deconvolutional layers for visualization, improved on AlexNet	Highlighted importance of feature visualization
VGGNet	2014	16-19 layers, 3x3 conv filters, 2x2 max pooling	Emphasized depth, consistent filter sizes, achieved top results	Influenced many future network designs with its simplicity
GoogLeNet (Inception)	2014	22 layers, Inception modules combining multiple filters	Reduced parameters significantly, auxiliary classifiers	Efficient network design, won ILSVRC 2014
ResNet	2015	50, 101, 152 layers with residual blocks	Residual connections, mitigated vanishing gradient problem	Enabled training of very deep networks, won ILSVRC 2015

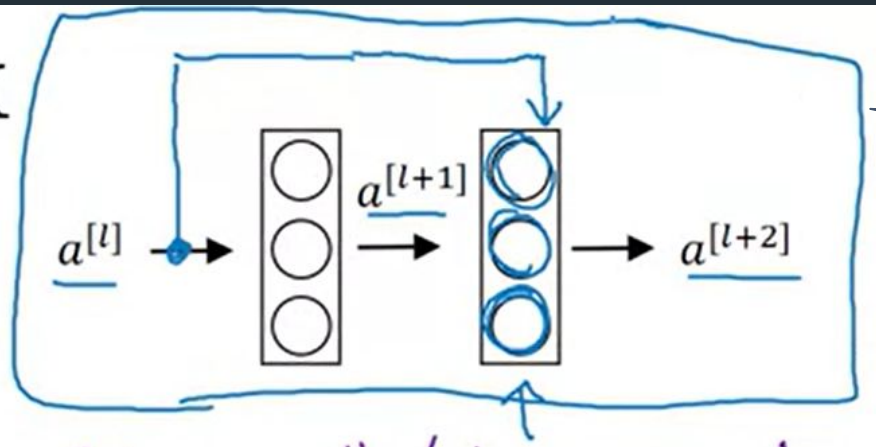
Modern networks

- ResNet
- Inception network (GoogLeNet)
- MobileNet

ResNet

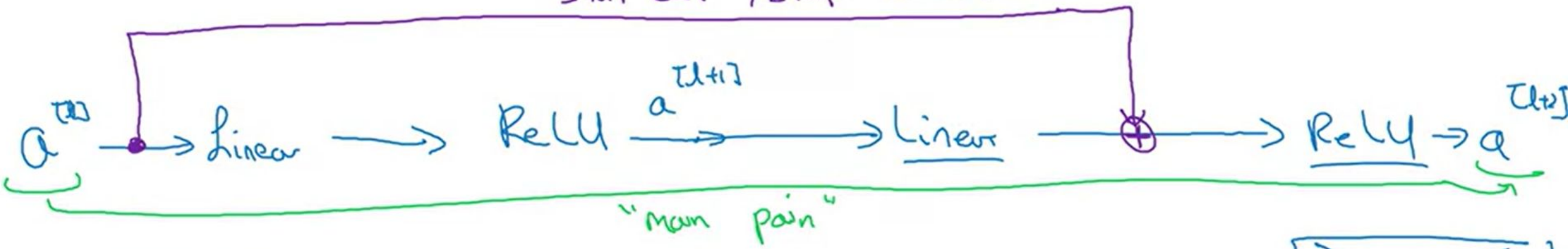
- Very deep networks (> 100 layers) are hard to train because of problems
  - Vanishing, exploding gradients etc.
- Skip connections
  - Take activation of one layer and feed it in another layer much deeper in the network
- Use of residual blocks help with training very deep networks

Residual block



Stacked together to form ResNet

"Short cut" / skip connection



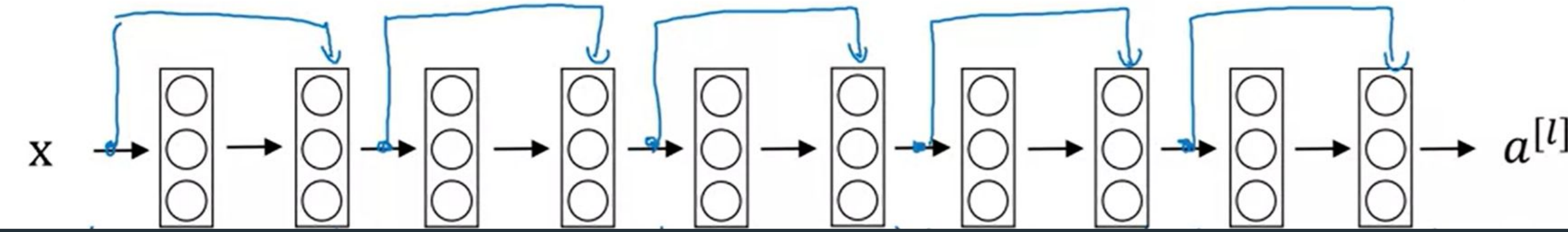
$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

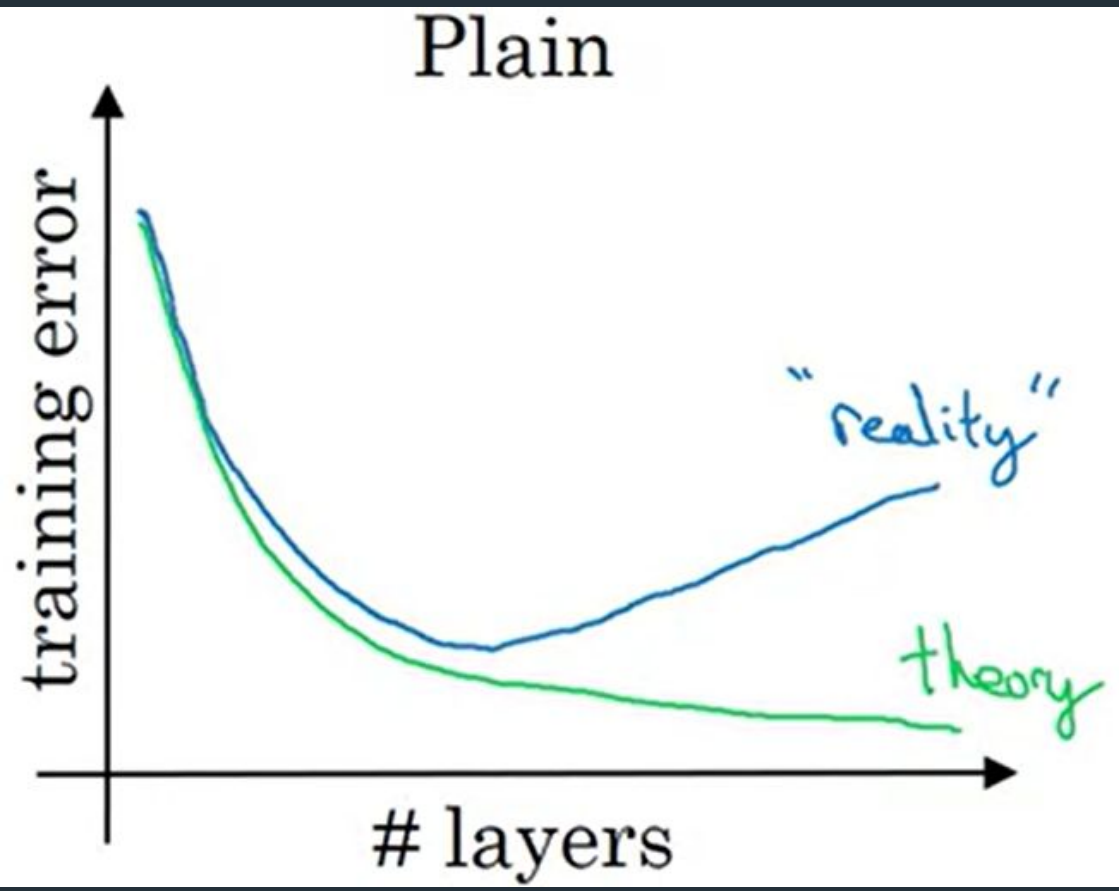
[He et al., 2015. Deep residual networks for image recognition]

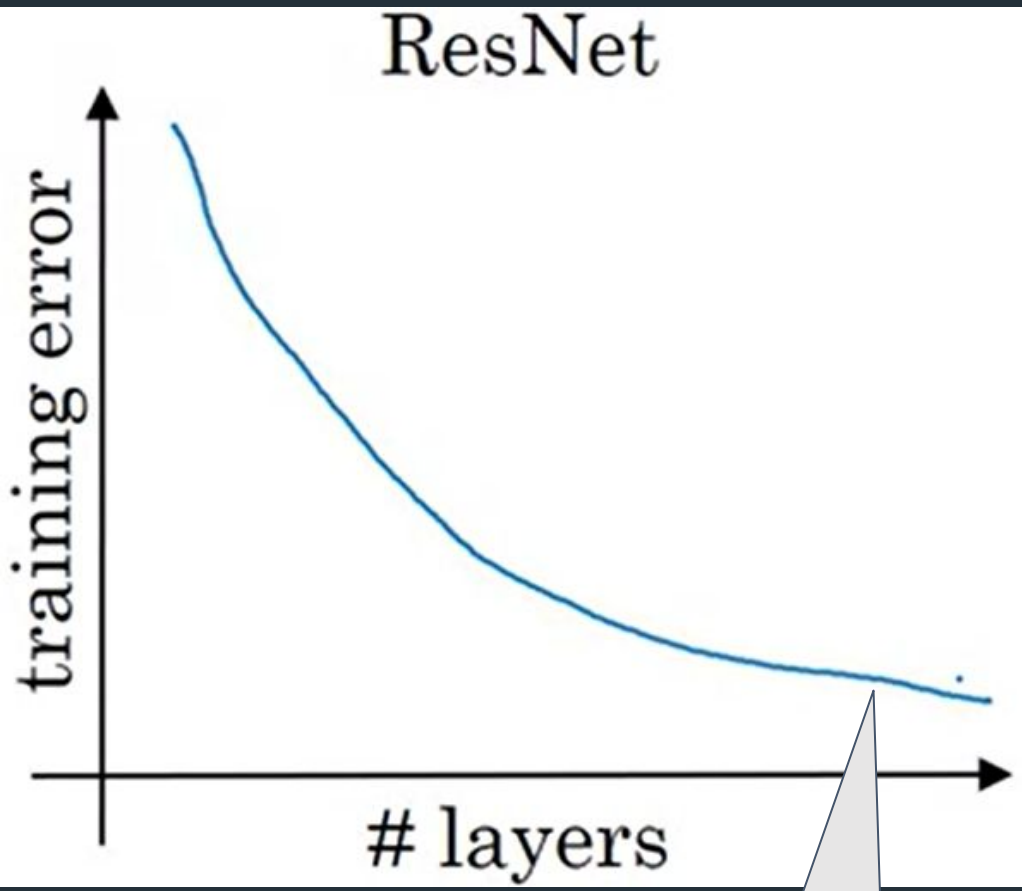
Andrew Ng

Residual Network

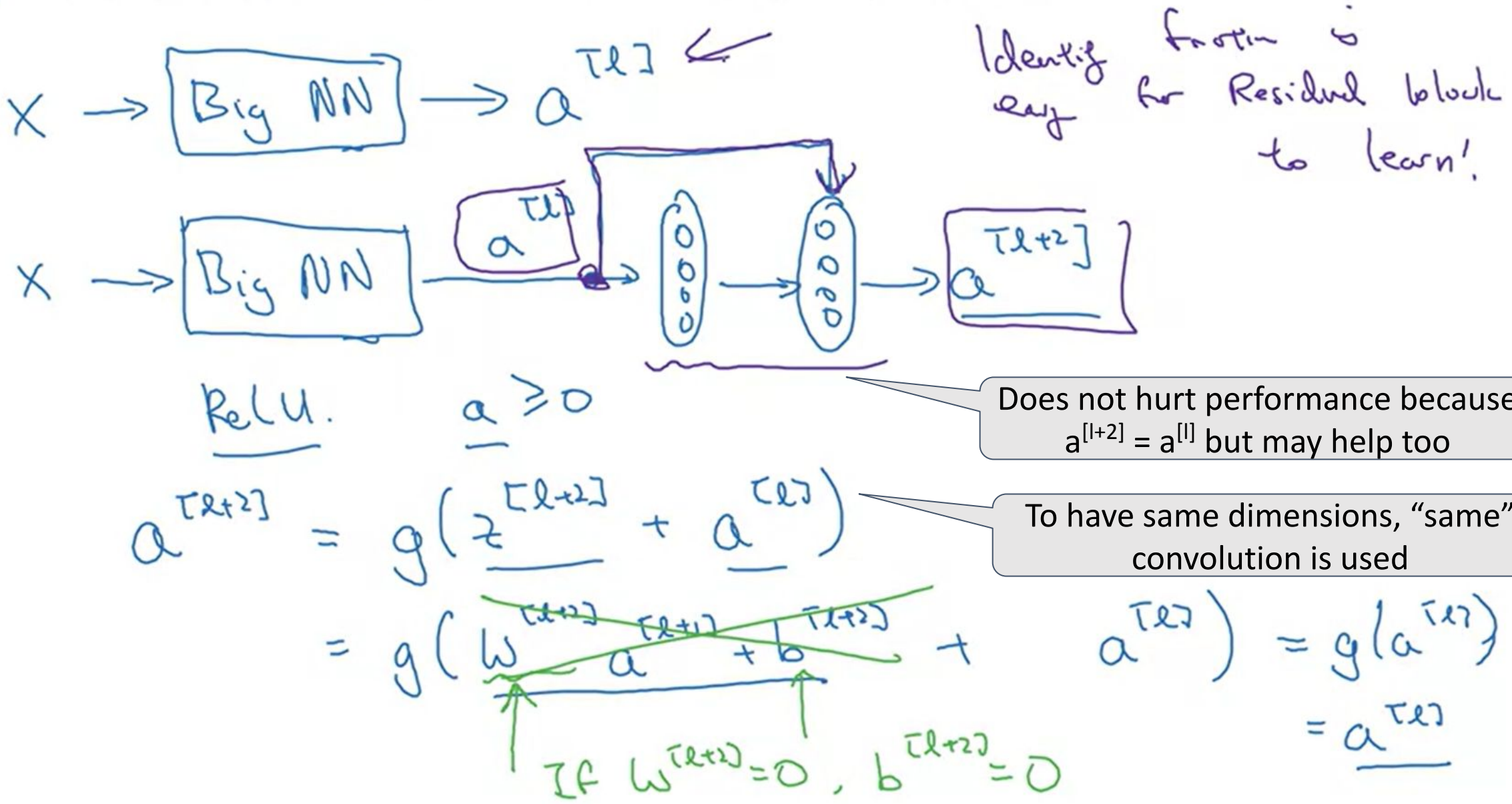


ResNet


Plain
training error
"reality"
theory
layers


ResNet
training error
layers
Plateau

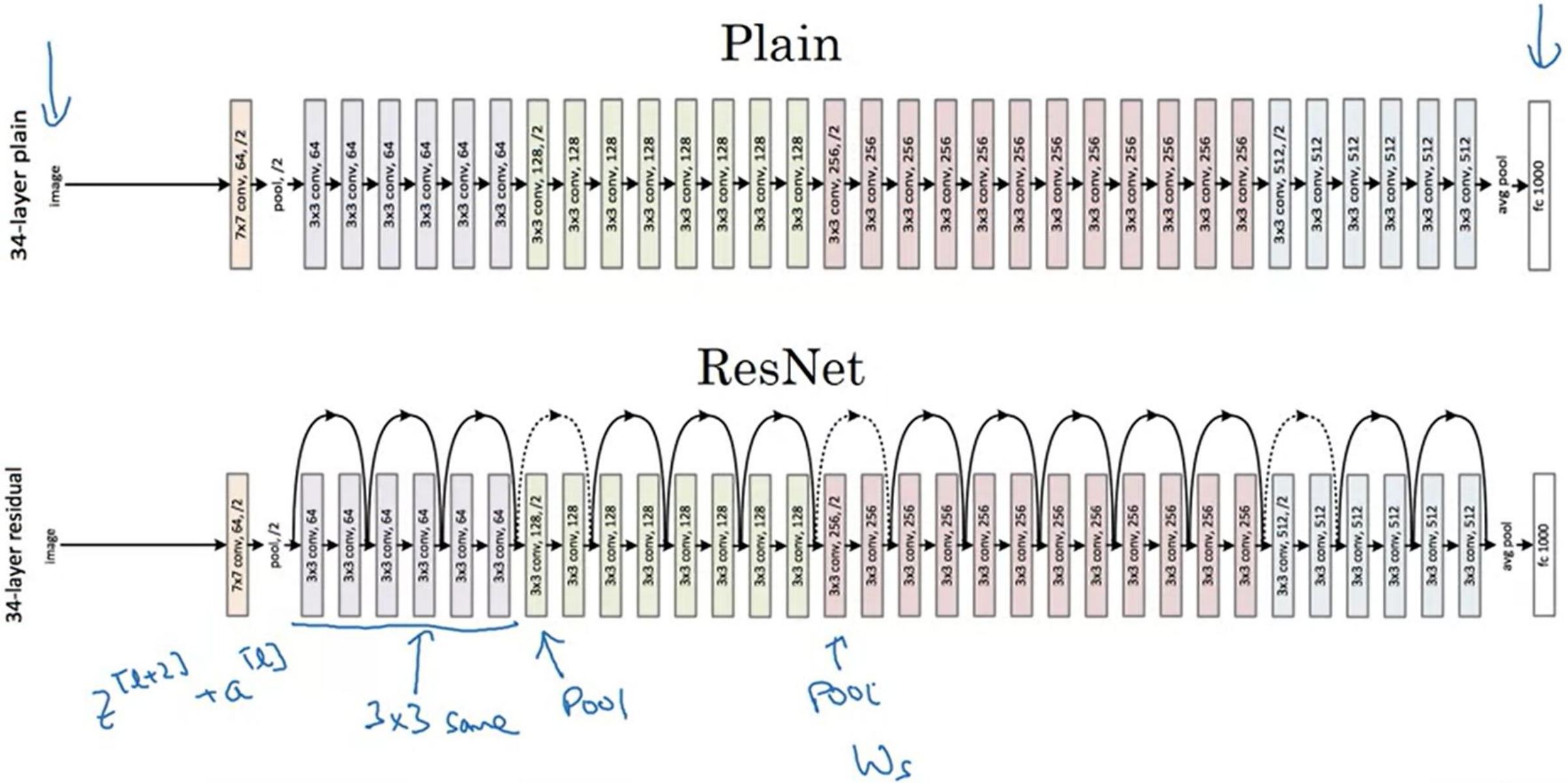
Why do residual networks work?



Does not hurt performance because $a^{[l+2]} = a^{[l]}$ but may help too

To have same dimensions, "same" convolution is used

ResNet



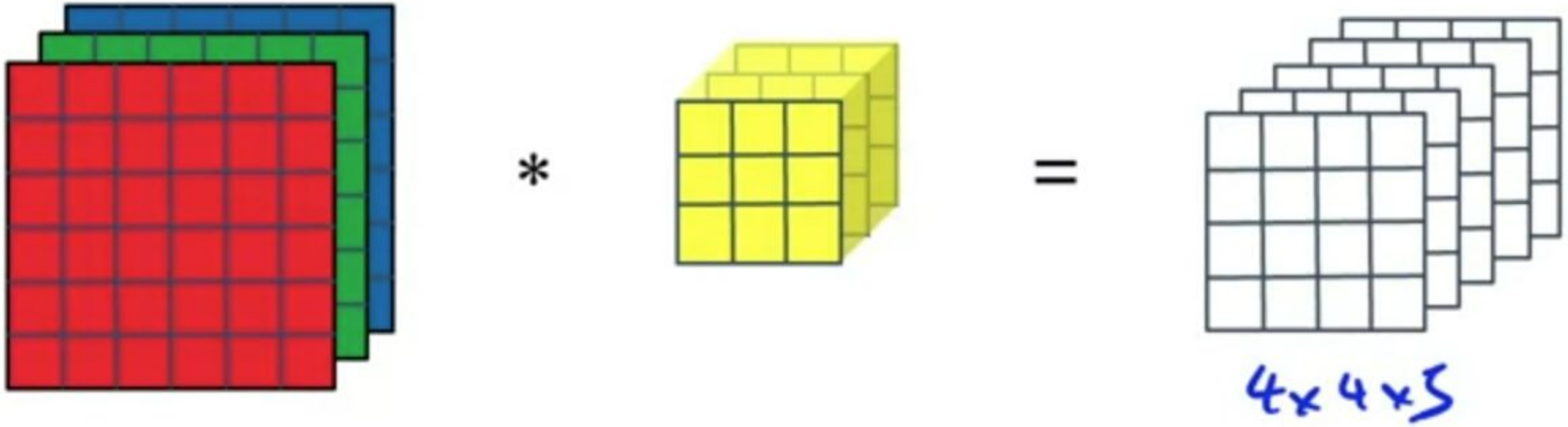
He et al., 2015. Deep residual networks for image recognition

Andrew Ng

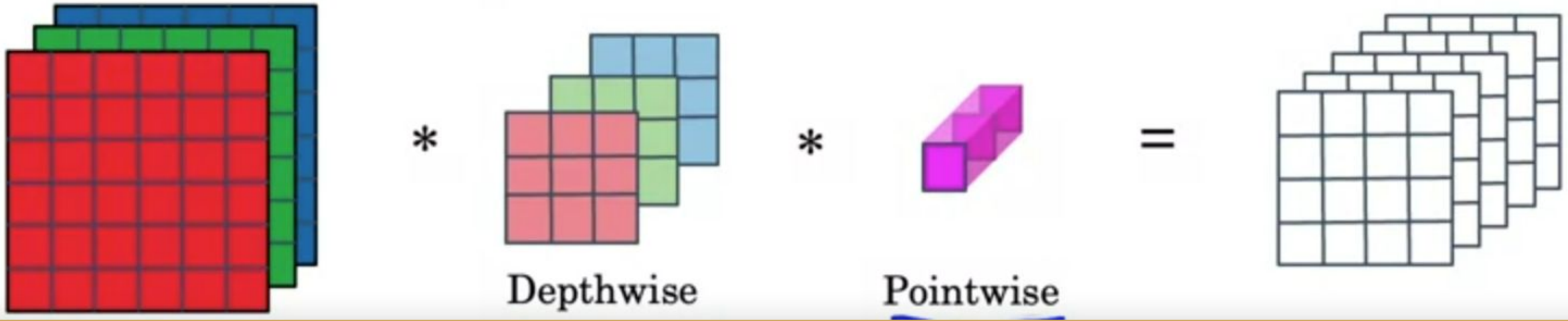
MobileNet

- Can be built and deployed even in low compute environments
  - Mobile phones, embedded vision applications
- Key idea
  - Normal vs. depth-wise separable convolutions

Normal Convolution

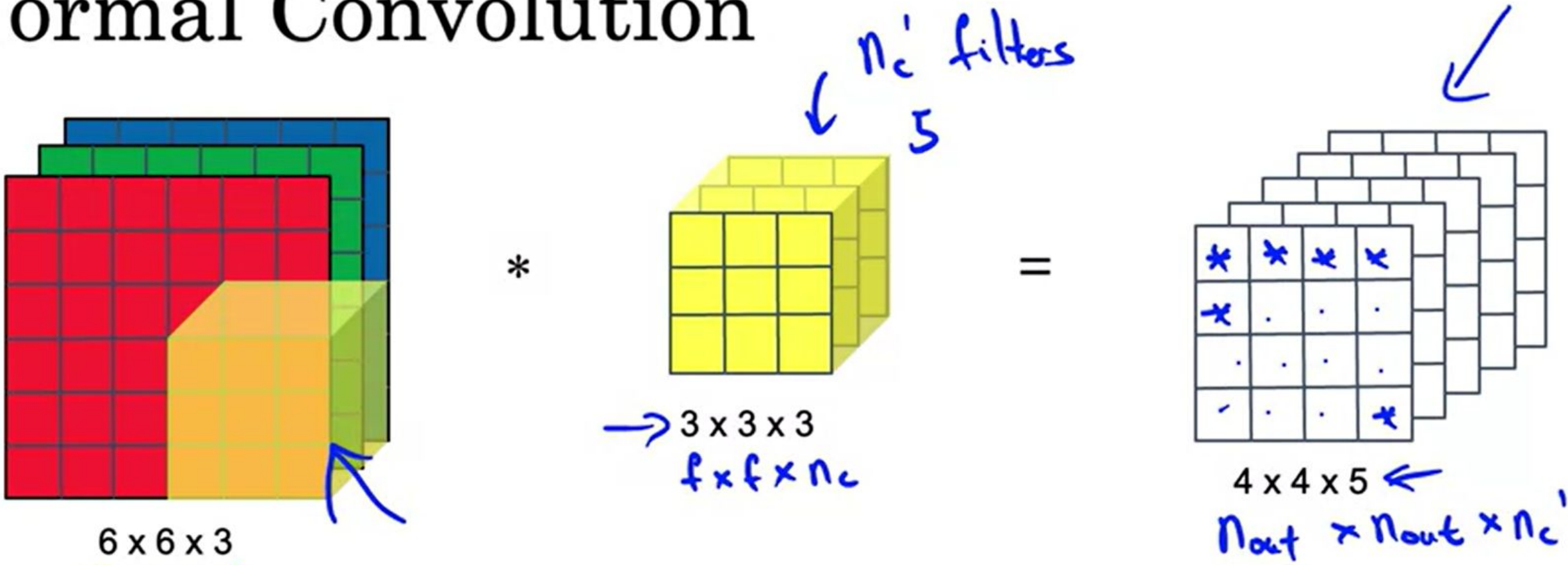


Depthwise Separable Convolution

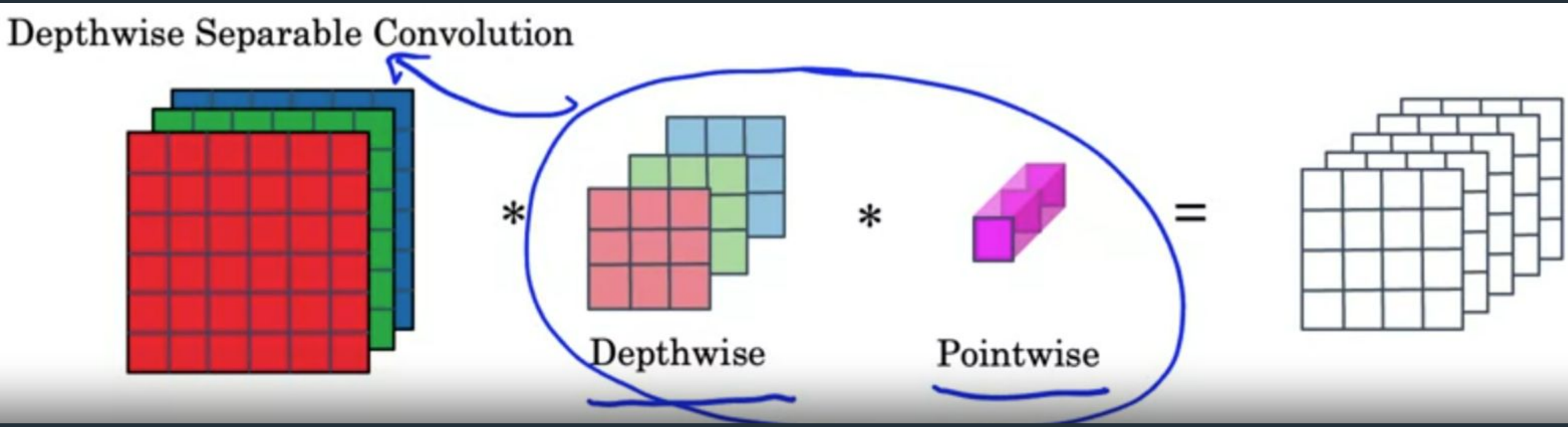


Computations

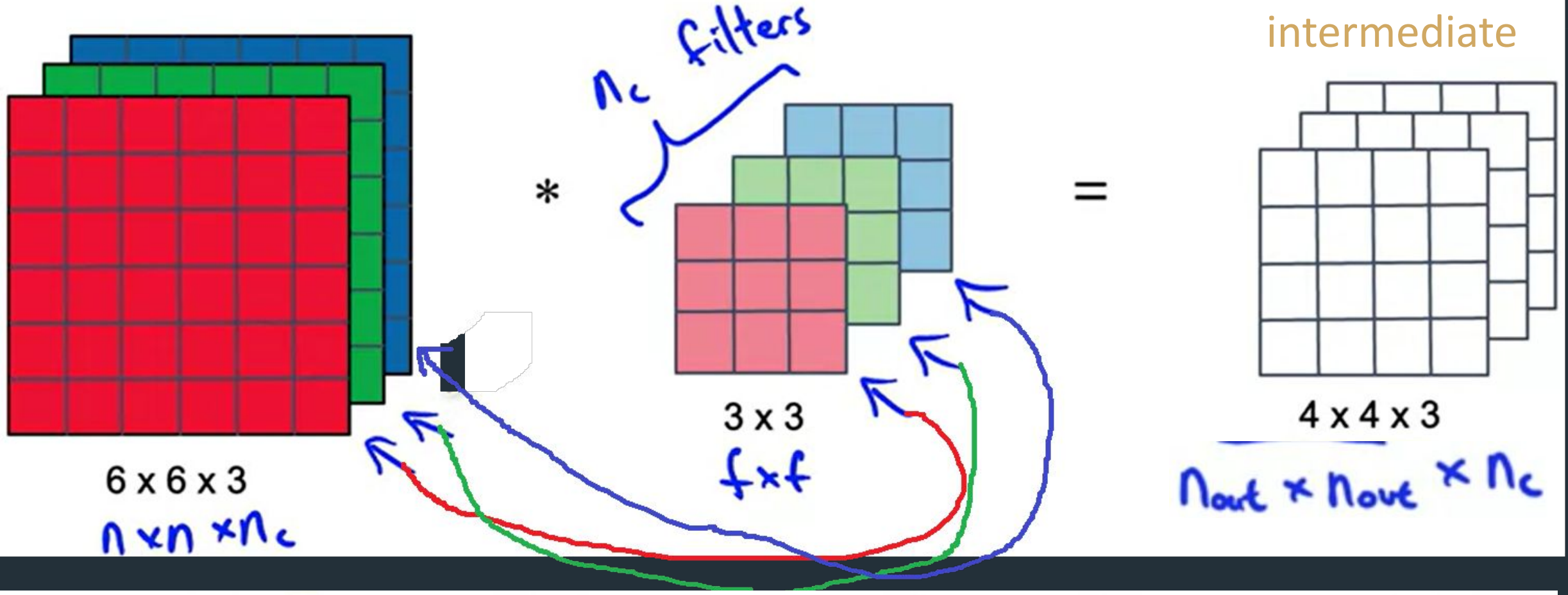
Normal Convolution



Computational cost	=	#filter params	x	# filter positions	x	# of filters
2160	=	3x3x3	x	4x4	x	5


Depthwise Separable Convolution
*
*
=
Depthwise
Pointwise

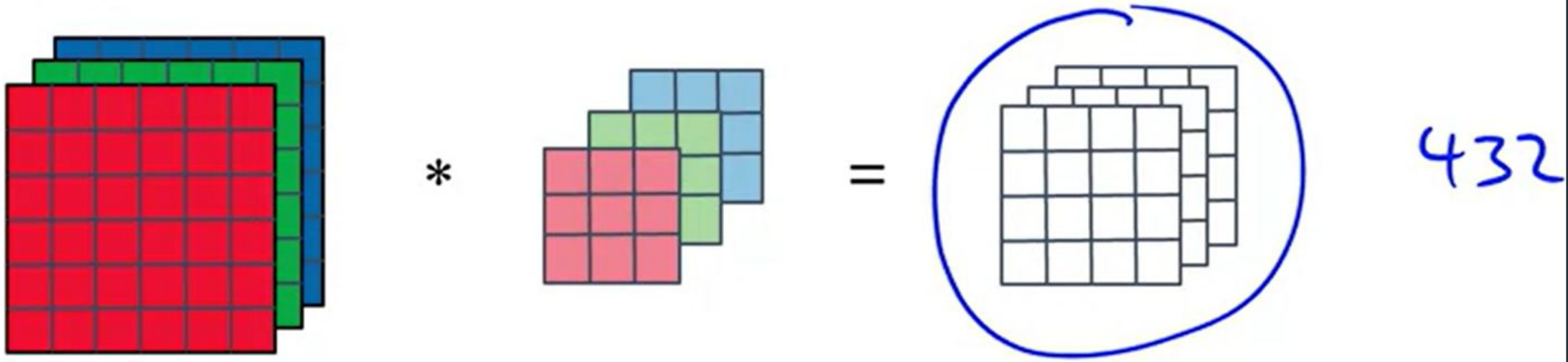
Depthwise Convolution



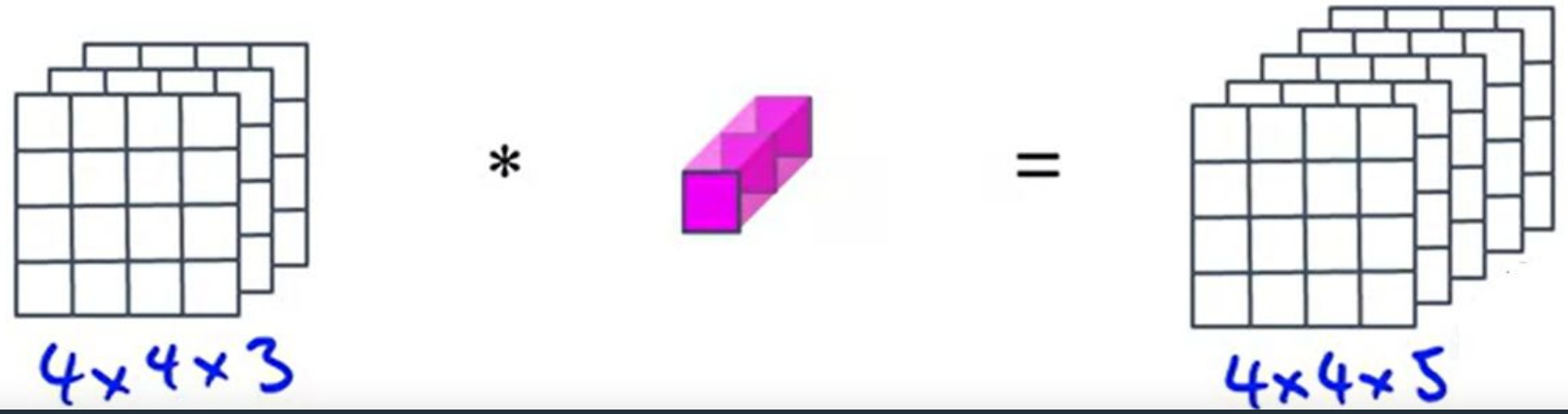
intermediate

Computational cost	=	#filter params	x	# filter positions	x	# of filters
432	=	3×3	×	4×4	×	3

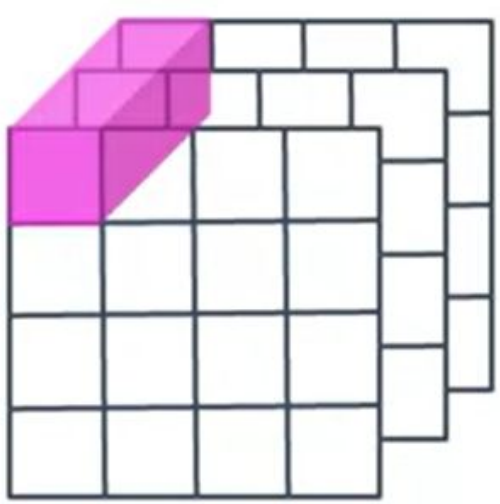
Depthwise Convolution



Pointwise Convolution



Pointwise Convolution



4 x 4 x 3

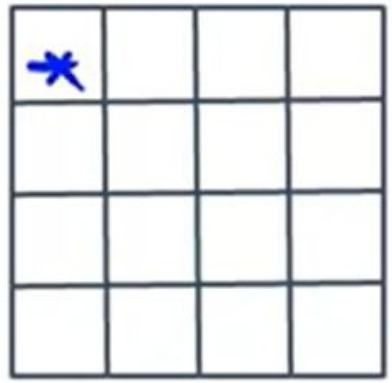
$n_{out} \times n_{out} \times n_c$

*

1 x 1 x 3

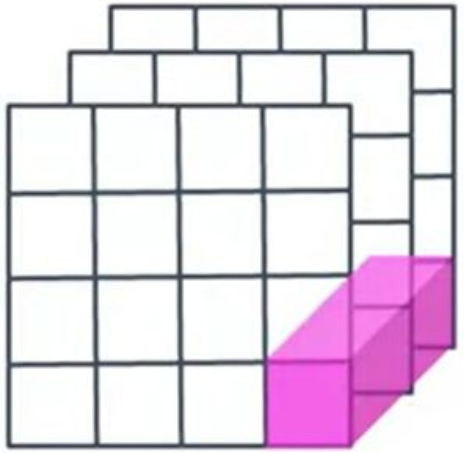
$1 \times 1 \times n_c$

=



4 x 4

Pointwise Convolution



4 x 4 x 3

$n_{out} \times n_{out} \times n_c$

*

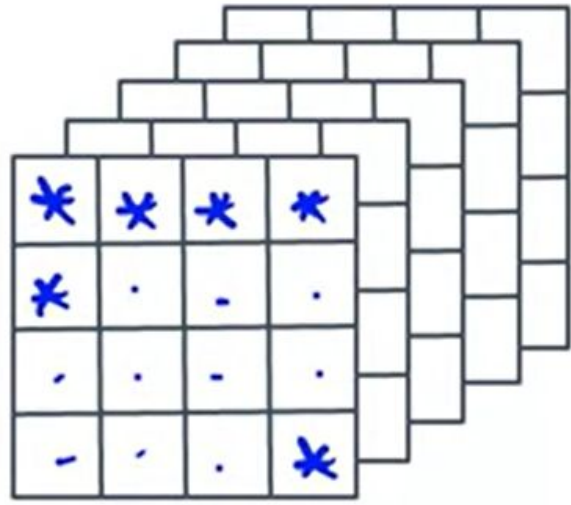
n_c' filters

5

1 x 1 x 3

$1 \times 1 \times n_c$

=



4 x 4 x 5

$n_{out} \times n_{out} \times n_c'$

Computational cost	=	#filter params	x	# filter positions	x	# of filters
240	=	$1 \times 1 \times 3$	x	4×4	x	5

Cost

Cost of normal convolution 2160

Cost of depthwise separable convolution

depthwise + pointwise

432 + 240 = 672

$$\frac{672}{2160} = 0.31$$

[Howard et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications]

Cost

Cost of normal convolution 2160

Cost of depthwise separable convolution

depthwise + pointwise

$432 + 240 = 672$

$$\frac{672}{2160} = 0.31$$

In general, ratio:

$$\frac{1}{n_c^1} + \frac{1}{f^2}$$

$$\frac{1}{5} + \frac{1}{9}$$

More generally:

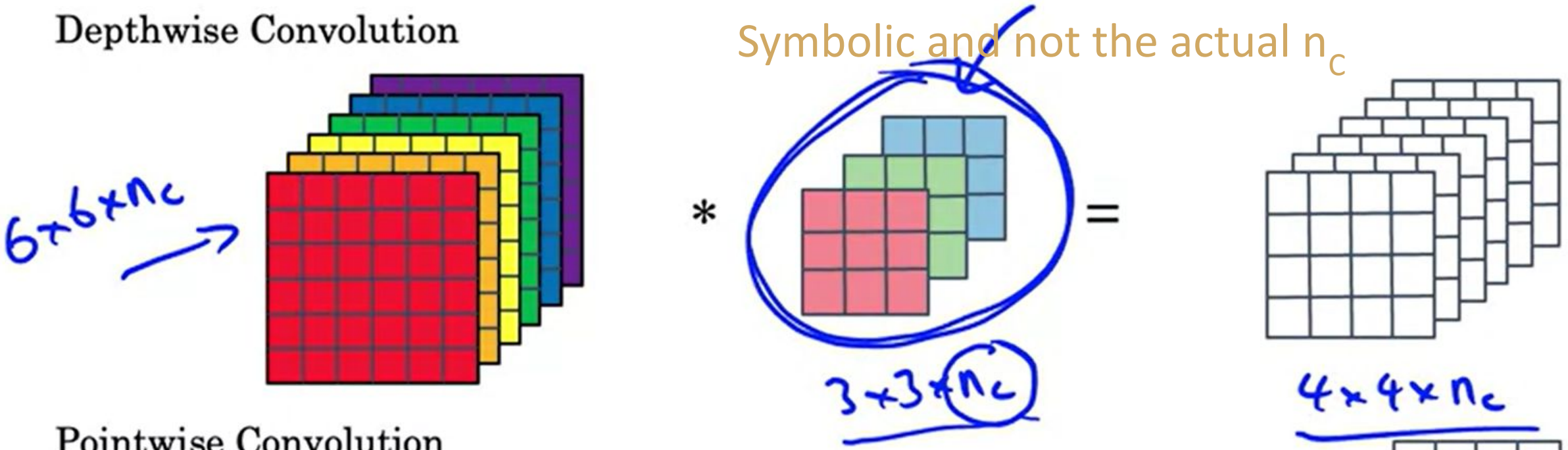
$$= \frac{1}{512} + \frac{1}{3^2}$$

~ 10 times cheaper

[Howard et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications]

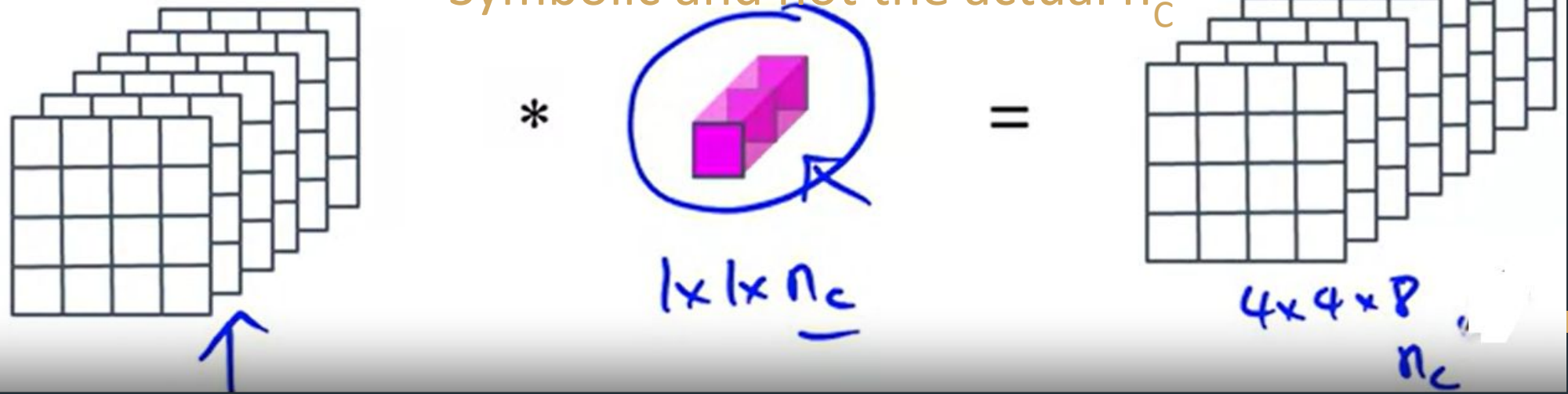
Depthwise Separable Convolution

Depthwise Convolution



Symbolic and not the actual n_c

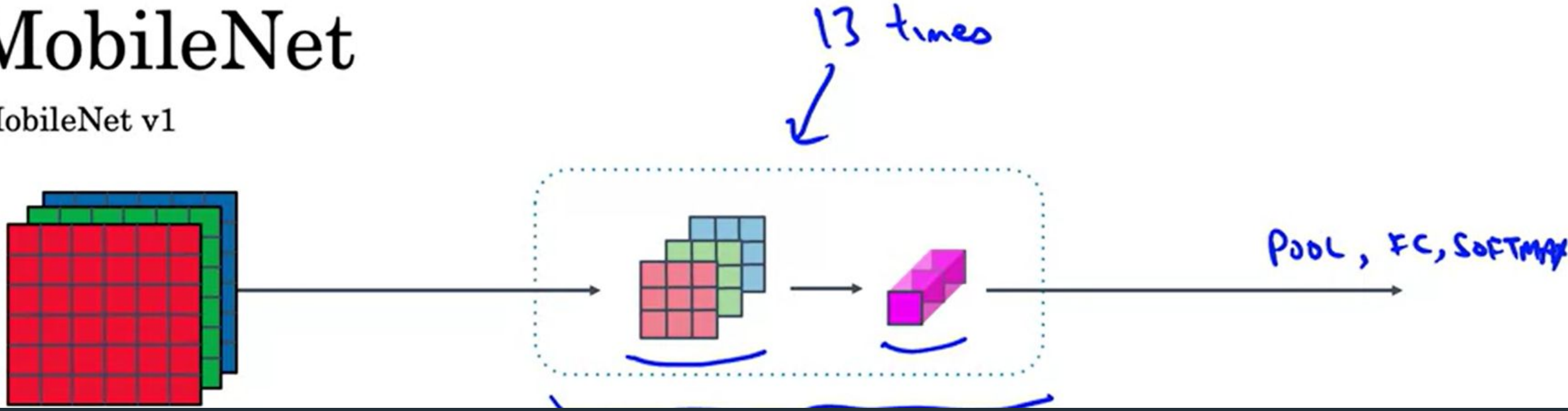
Pointwise Convolution



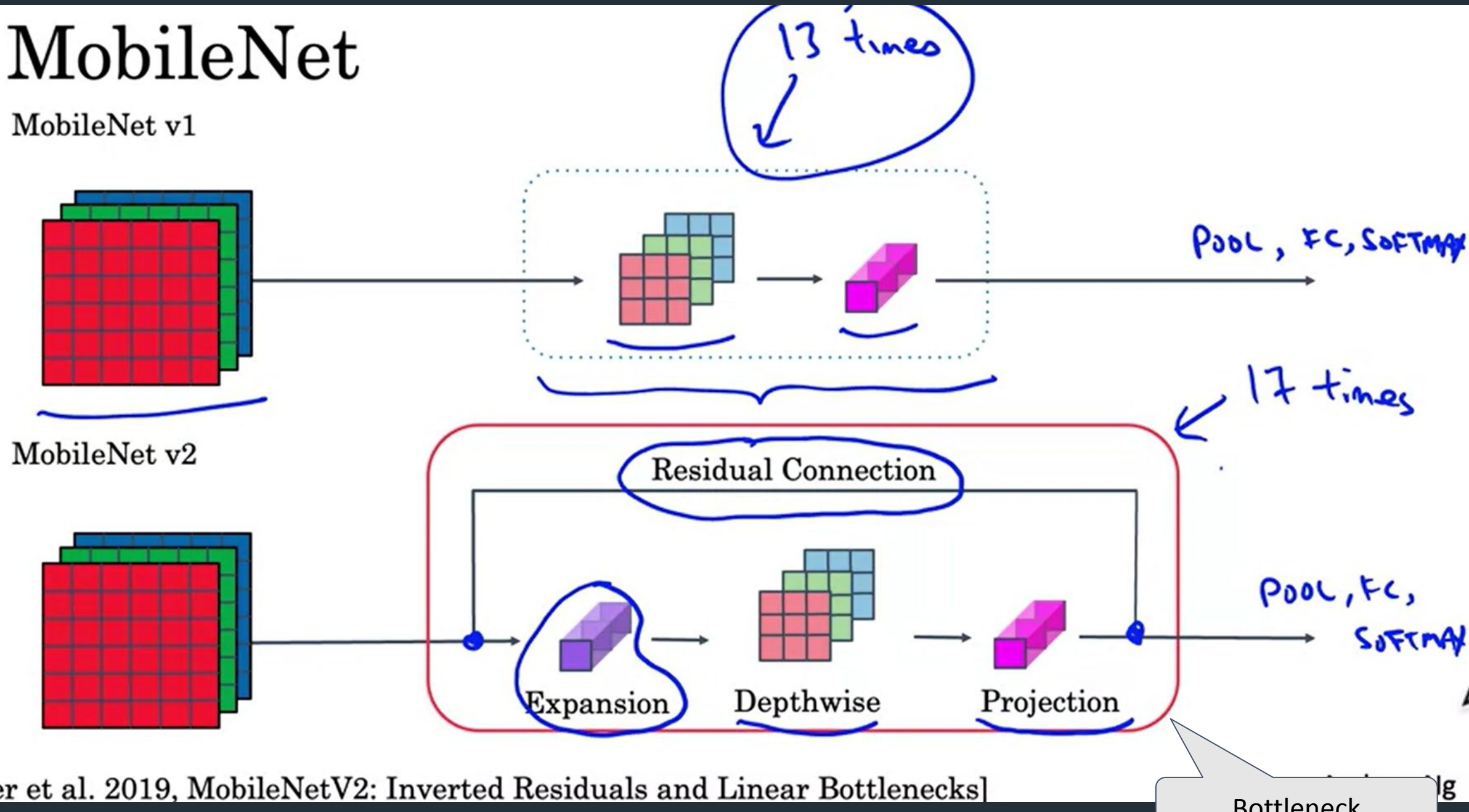
Symbolic and not the actual n_c

MobileNet

MobileNet v1



MobileNet



[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

Bottleneck