CNN Case Study

이영기
서울대학교 컴퓨터공학부

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

References

- Stanford CS231n 2019 Lecture Notes
  - ✓Lecture 9 “CNN Architectures”
  - ✓Lecture 12 “Detection and Segmentation”
- Papers and blogs from the Internet

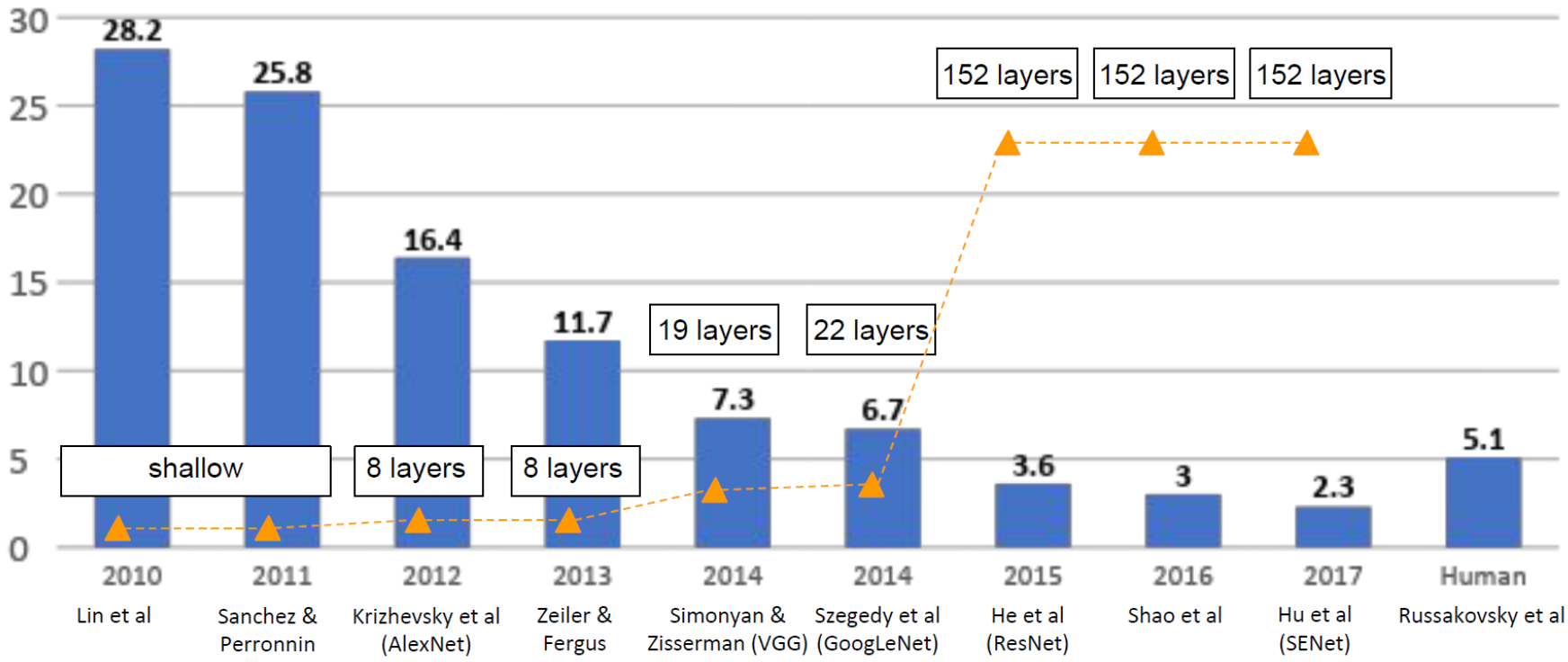
Content Overview

- Image classification networks (with relation to ILSVRC)
  - ✓ AlexNet
  - ✓ VGG
  - ✓ Inception v1, v2, v3, v4 – Xception
  - ✓ ResNet – ResNeXt – SENet
- Efficient CNNs
  - ✓ SqueezeNet
  - ✓ MobileNet v1, v2, v3
  - ✓ ShuffleNet
  - ✓ EfficientNet
- Object detection networks
  - ✓ Two stage detection (R-CNN, Fast R-CNN, Faster R-CNN)
  - ✓ Single stage detection (YOLO v1, v2, SSD)
  - ✓ Small object detection

Content Overview

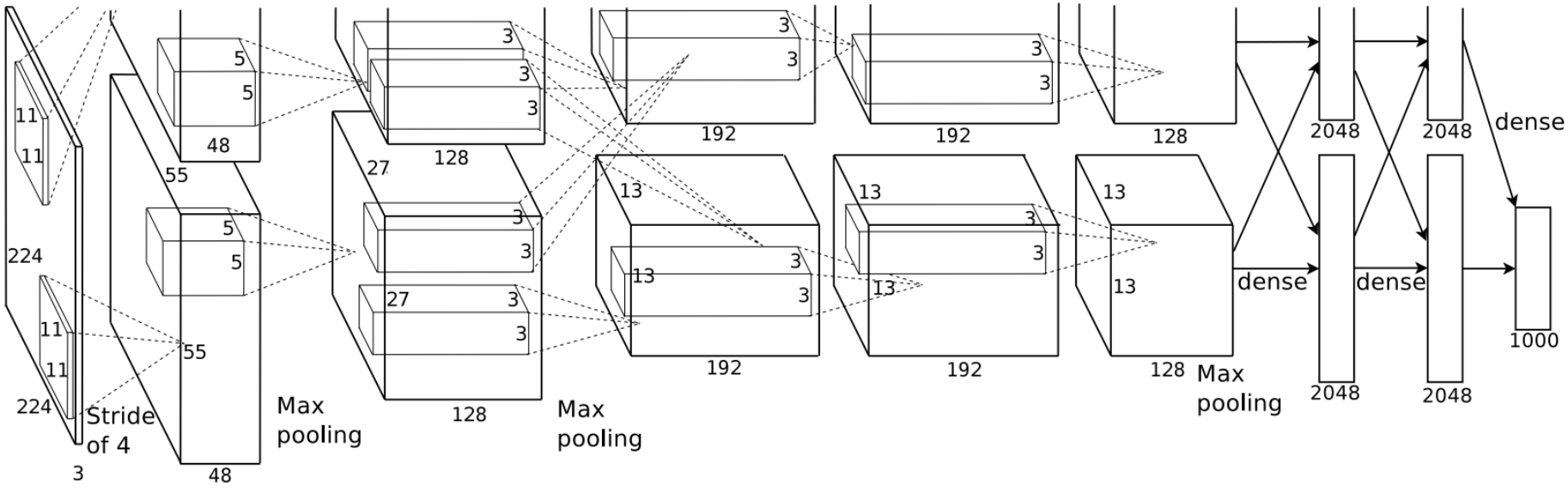
- **Image classification networks (with relation to ILSVRC)**
  - ✓ AlexNet
  - ✓ VGG
  - ✓ Inception v1, v2, v3, v4 – Xception
  - ✓ ResNet – ResNeXt – SENet
- Efficient CNNs
  - ✓ SqueezeNet
  - ✓ MobileNet v1, v2, v3
  - ✓ ShuffleNet
  - ✓ EfficientNet
- Object detection networks
  - ✓ Two stage detection (R-CNN, Fast R-CNN, Faster R-CNN)
  - ✓ Single stage detection (YOLO v1, v2, SSD)
  - ✓ Small object detection

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



AlexNet

- First CNN-based winner of ImageNet challenge
- First to use ReLU (LeNet-5 used Tanh)
- Network spread across 2 GPUs (∵ GTX 580 had only 3 GB RAM)



A. Krizhevsky et al. "ImageNet Classification with Deep Convolutional Neural Networks," NIPS 2014.

VGG

- Deeper networks
  - ✓8 layers (AlexNet) → 16/19 layers (VGG)
- Smaller filters (3x3)
  - ✓Stack of 3 3x3 conv. layers have effective receptive field of 7x7 conv., with fewer parameters
    - ➢ $3*(3^2C^2)$ vs. 7^2C^2
- Still heavy used (∵ simplicity)

K. Simonyan et al. "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR 2015.

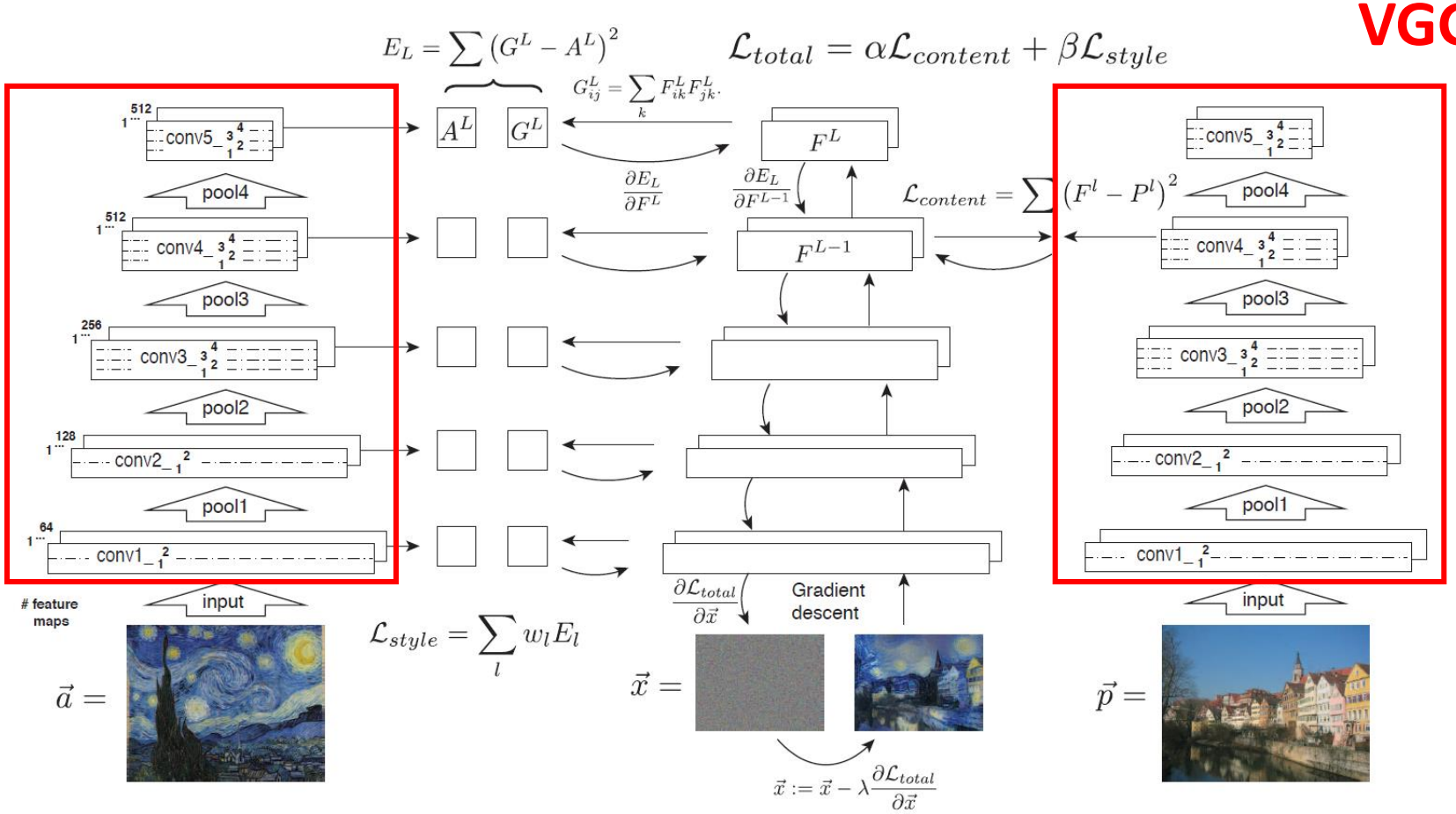
VGG

- Application – image feature extractor

L. Gats et al. “Image Style Transfer Using Convolutional Neural Networks,” CVPR 2016.

VGG

- Application – image feature extractor



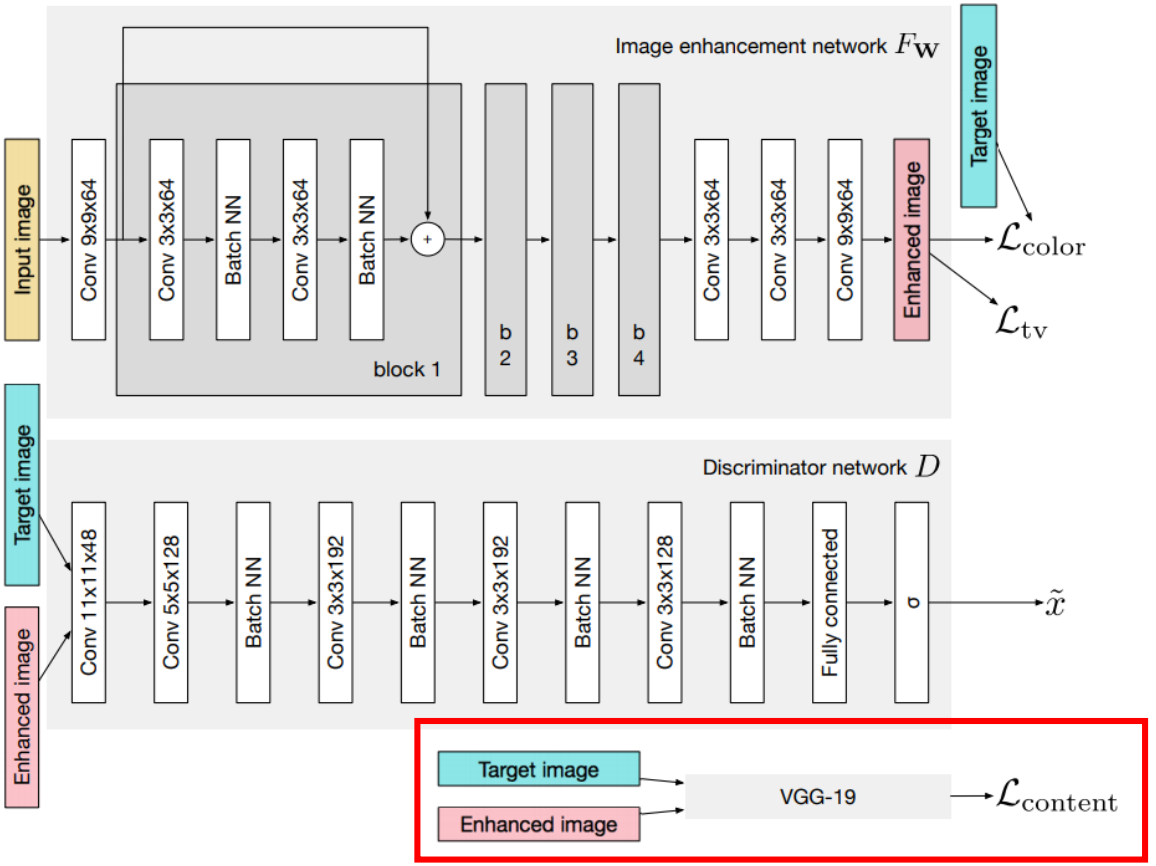
L. Gats et al. “Image Style Transfer Using Convolutional Neural Networks,” CVPR 2016.

VGG

- Application - training loss

Input (mobile camera)

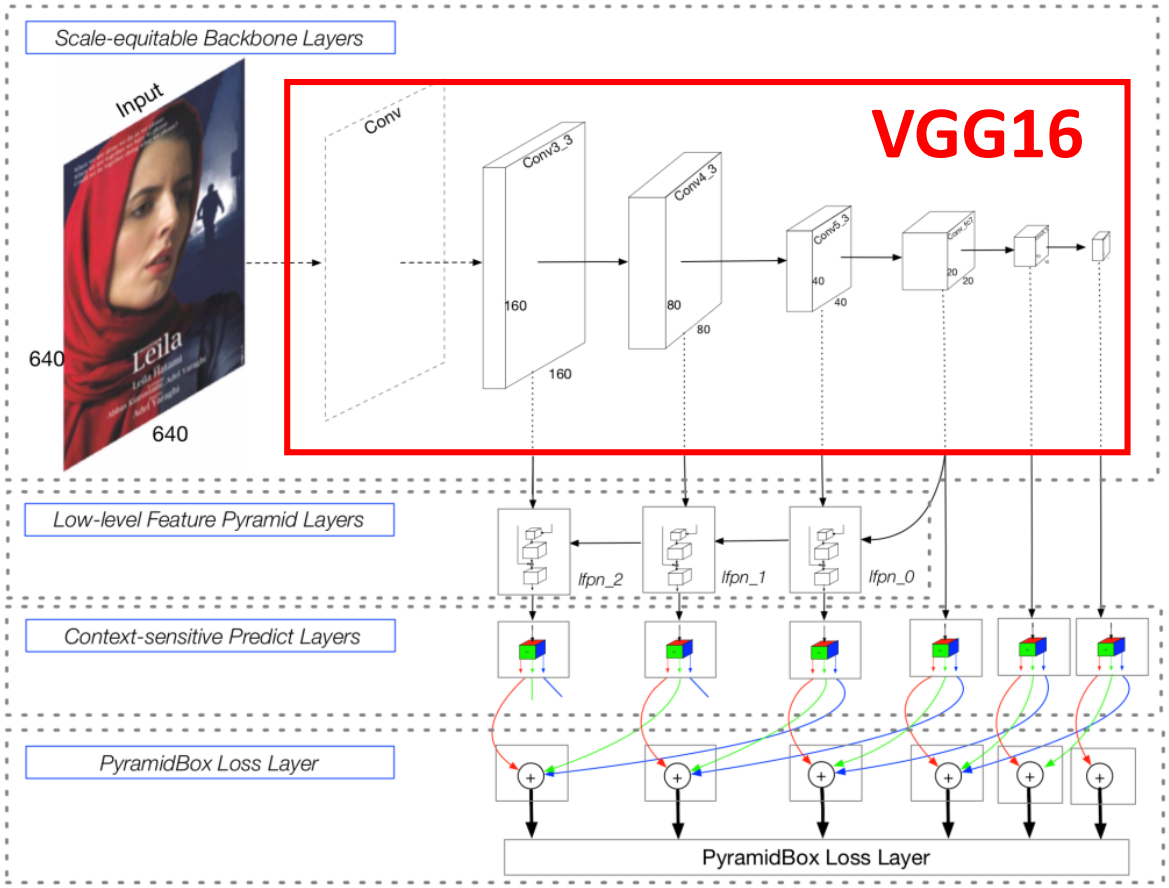
Target (DSLR camera)



A. Ignatov et al. "DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks," ICCV 2017.

VGG

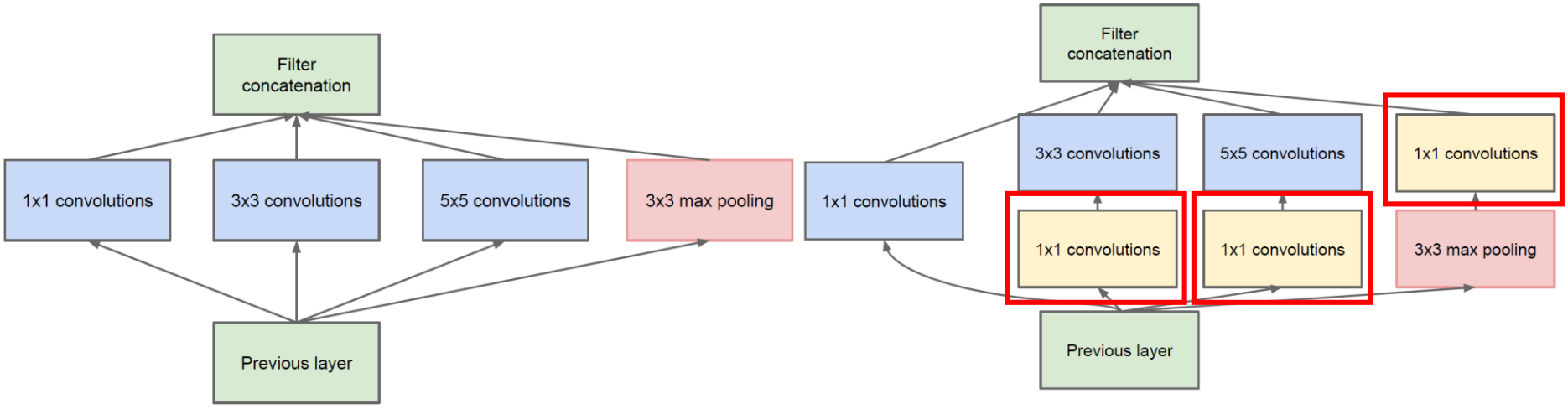
- Application – backbone network



X. Tang et al. “PyramidBox: A Context-assisted Single Shot Face Detector.” ECCV 2018.

Inception v1

- Deeper network (22 layers)
- Parallel filter operations for diverse feature extraction
- Computational efficiency via 1x1 conv. layers



(a) Inception module, naïve version

(b) Inception module with dimensionality reduction

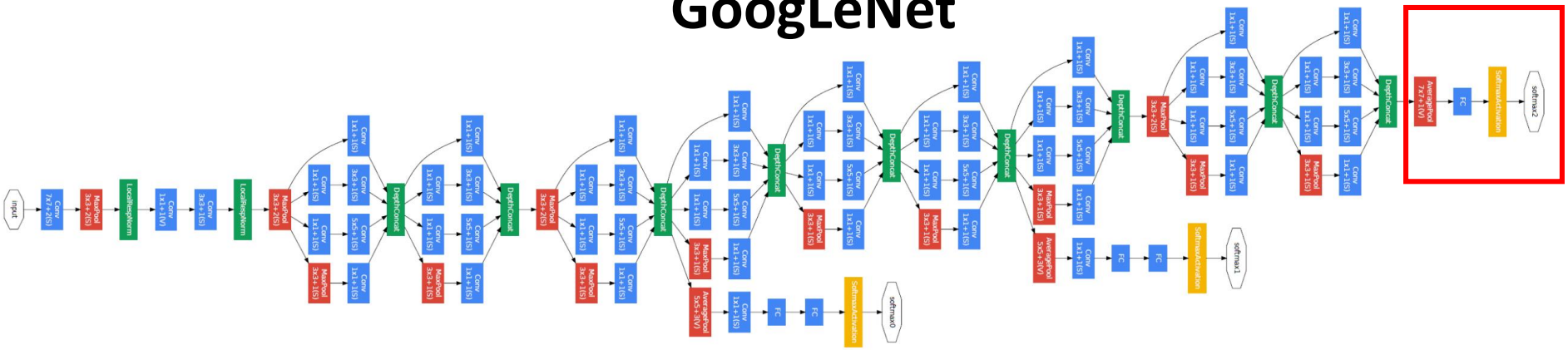
C. Szegedy et al. "Going Deeper with Convolutions," CVPR 2015.

Inception v1

- Use average pooling instead of multiple FC layers
  (i.e., only one FC layer for final classification output)
  → Smaller number of parameters (12x less than AlexNet)

AlexNet

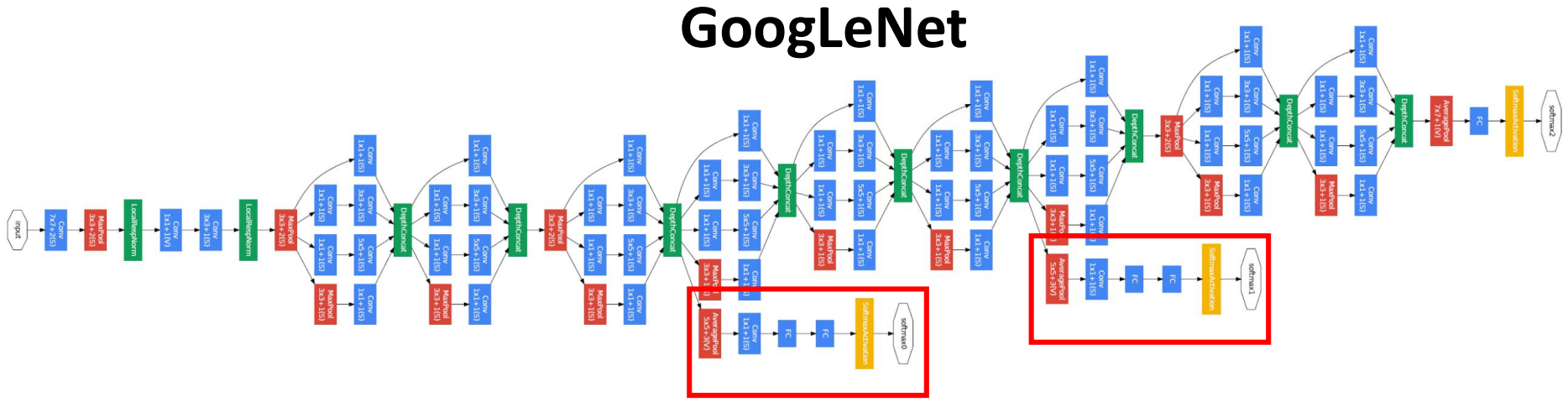
GoogLeNet



C. Szegedy et al. "Going Deeper with Convolutions," CVPR 2015.

Inception v1

- Auxiliary classification outputs to inject gradient at lower layers



C. Szegedy et al. “Going Deeper with Convolutions,” CVPR 2015.

Inception v2, v3

- For computational efficiency, replace 5x5 conv. with 2 3x3 conv. layers
  - ✓5x5 vs. 2x3x3 → 28% reduced computation

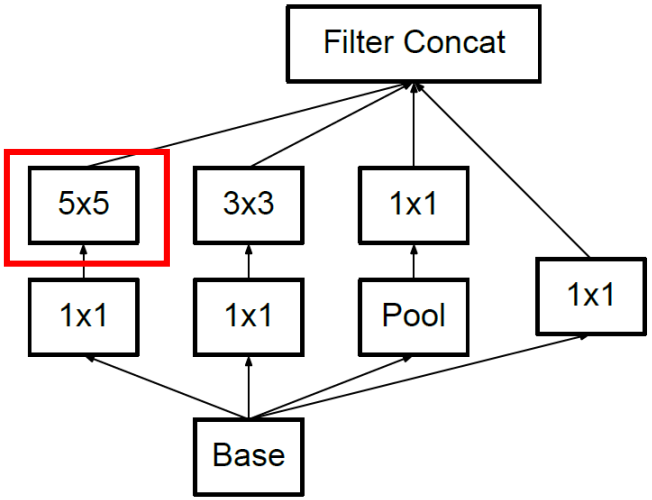


Figure 4. Original Inception module as described in [20].

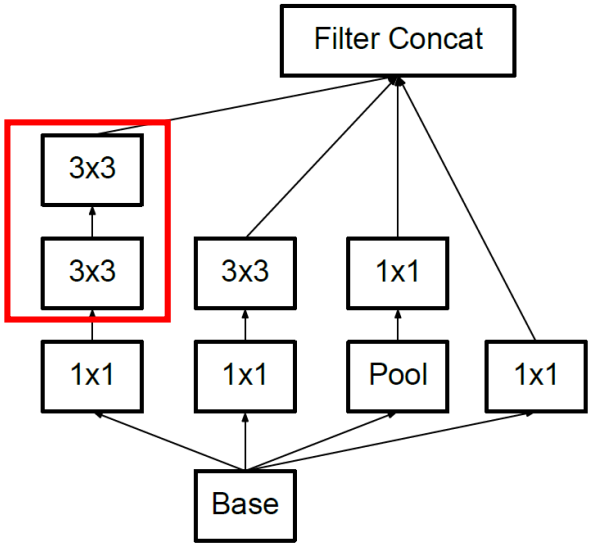


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

C. Szegedy et al. "Rethinking the Inception Architecture for Computer Vision," CVPR 2016.

Inception v2, v3

- For further computation efficiency, replace NxN conv. with Nx1 + 1xN conv. Layers (asymmetric conv.)

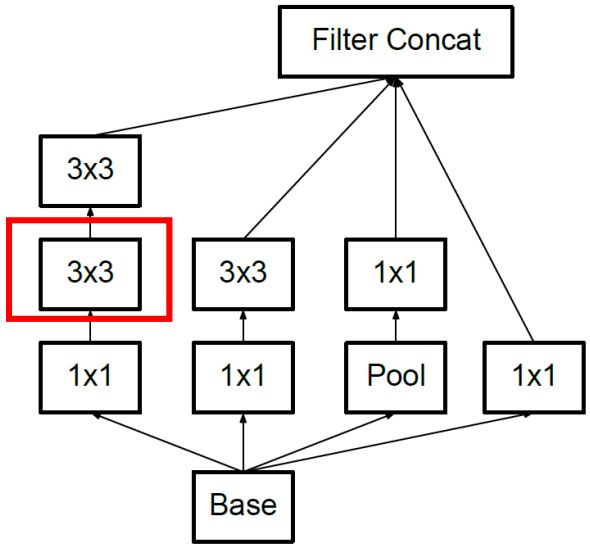


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

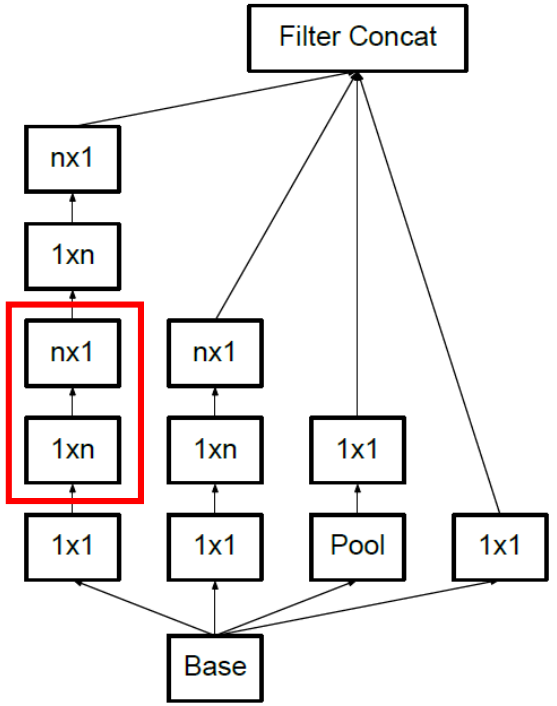
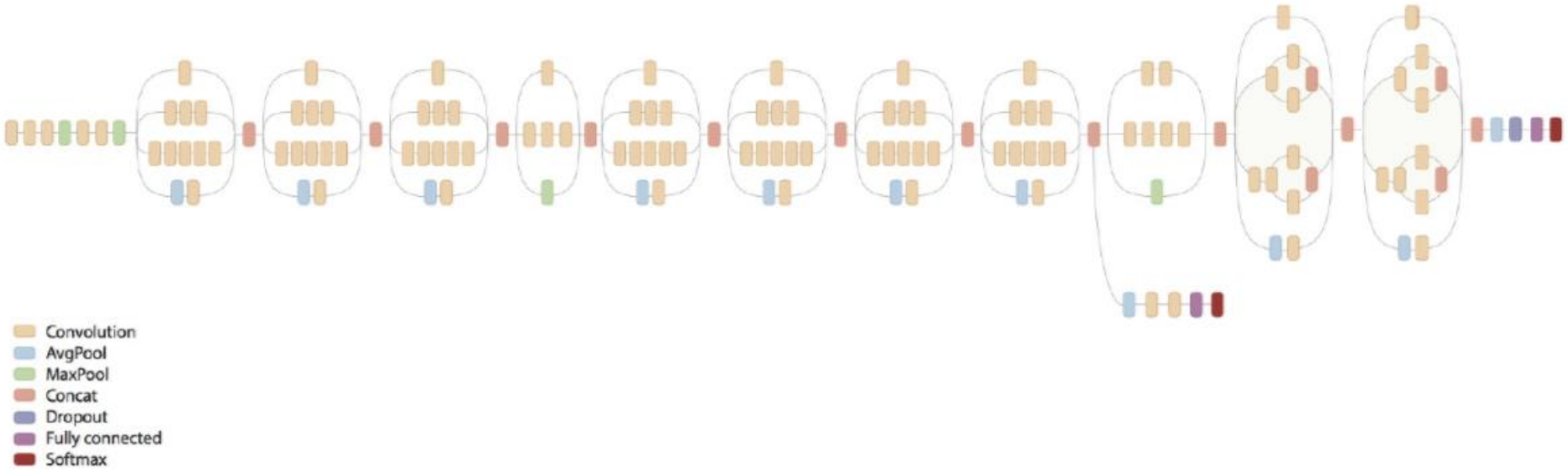


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

C. Szegedy et al. "Rethinking the Inception Architecture for Computer Vision," CVPR 2016.

Inception v2, v3

- Inception v2 (overall architecture is same as v1)

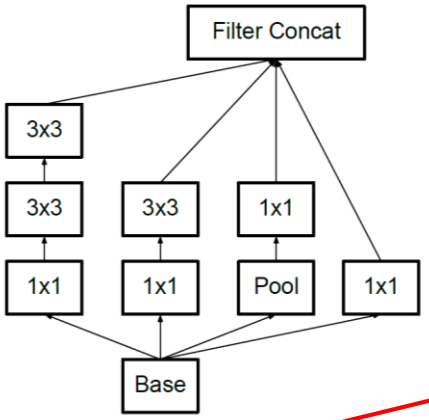


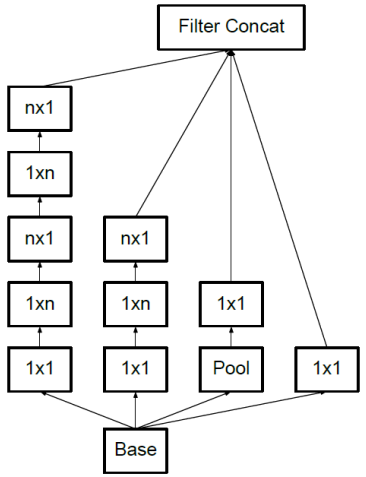
C. Szegedy et al. “Rethinking the Inception Architecture for Computer Vision,” CVPR 2016.

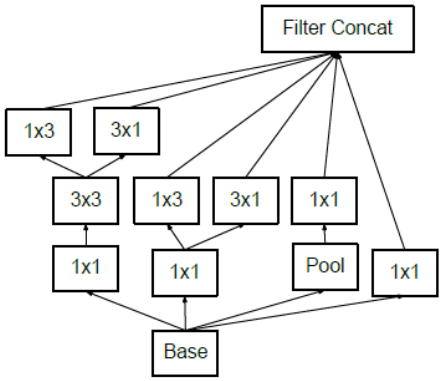
Inception v2, v3

- Inception v2 (overall architecture is same as v1)

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000







C. Szegedy et al. "Rethinking the Inception Architecture for Computer Vision," CVPR 2016.

Inception v2, v3

- Inception v3: some additional optimizations
  - ✓RMSProp optimizer
  - ✓Label smoothing
  - ✓Factorize 7x7 conv. Layer at the initial stage
  - ✓Batch normalization

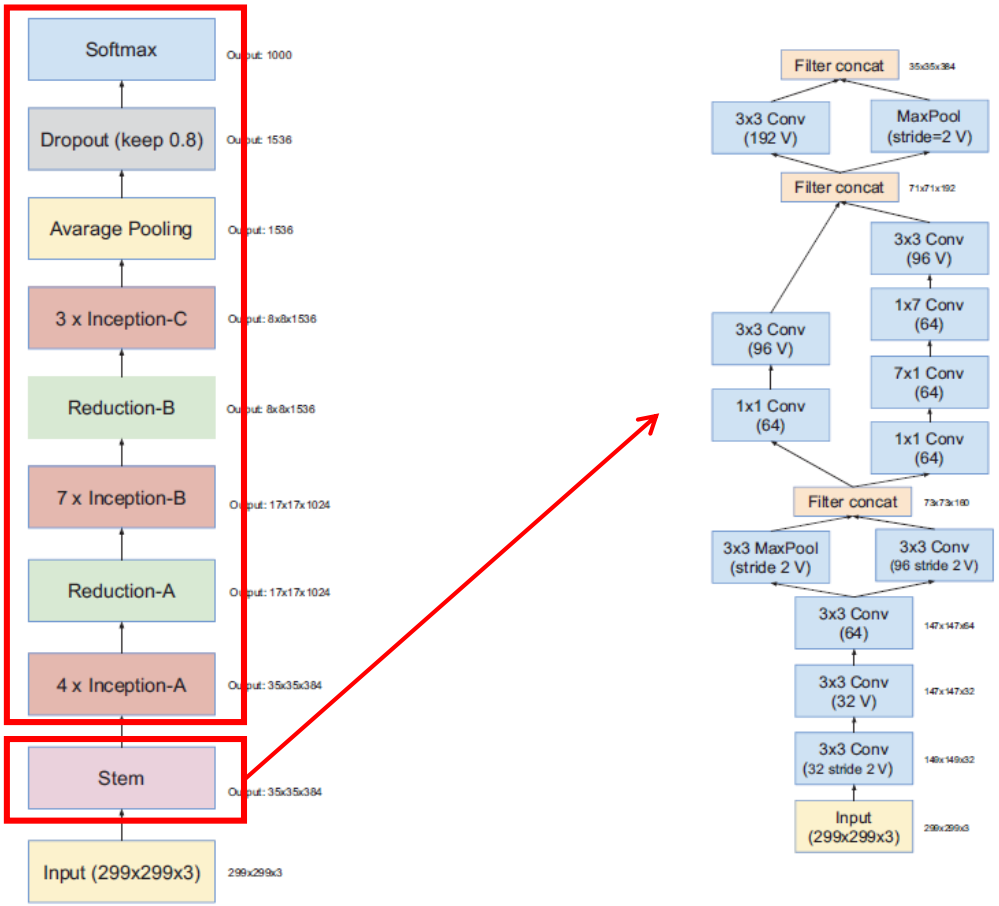
Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v3-basic	23.4%	-	3.8
Inception-v3-rmsprop RMSProp	23.1%	6.3	3.8
Inception-v3-smooth Label Smoothing	22.8%	6.1	3.8
Inception-v3-fact Factorized 7×7	21.6%	5.8	4.8
Inception-v3 BN-auxiliary	**21.2%**	**5.6%**	4.8

C. Szegedy et al. “Rethinking the Inception Architecture for Computer Vision,” CVPR 2016.

Inception v4 & Inception-ResNet

- Inception v4: modifies initial layers (Stem net.) of Inception v3

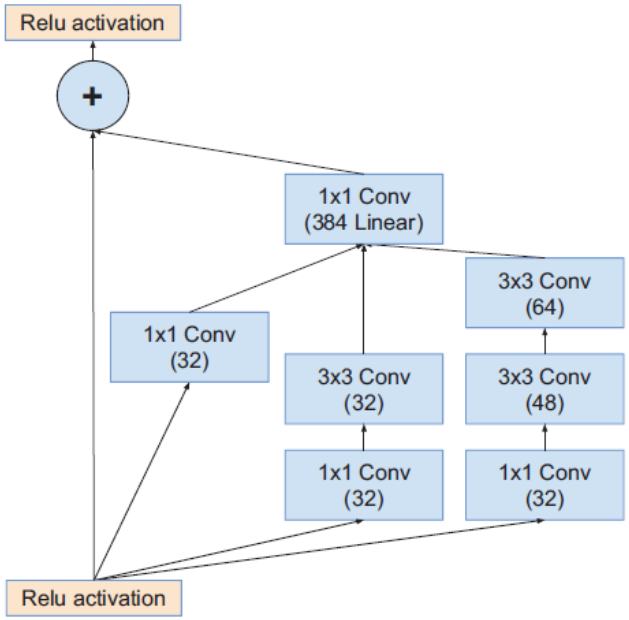
Inception-v3 with very few modifications

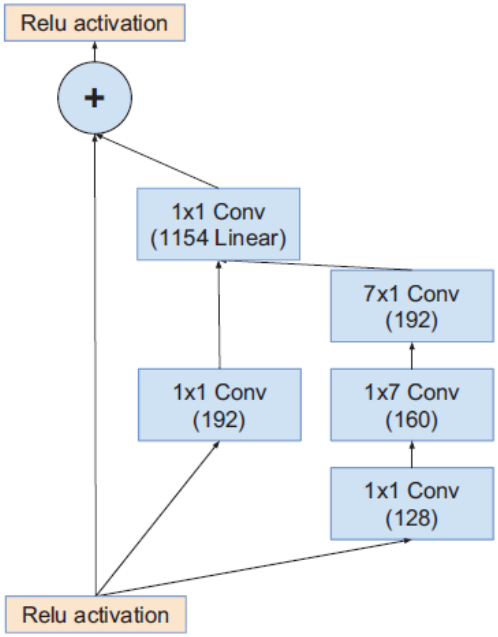


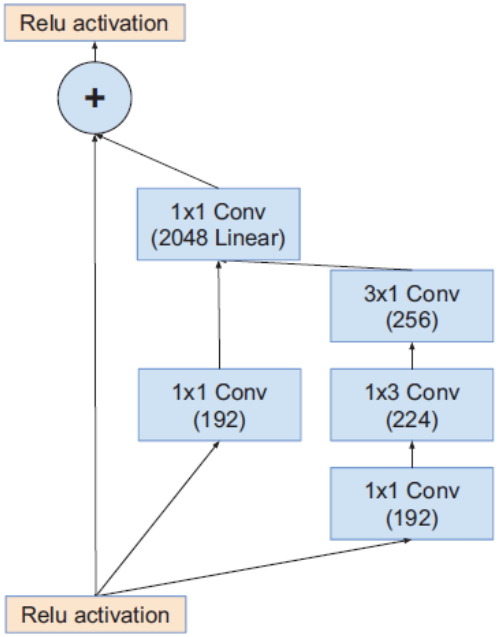
C. Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," AAAI 2017.

Inception v4 & Inception-ResNet

- Inception-ResNet: add Inception module on ResNet



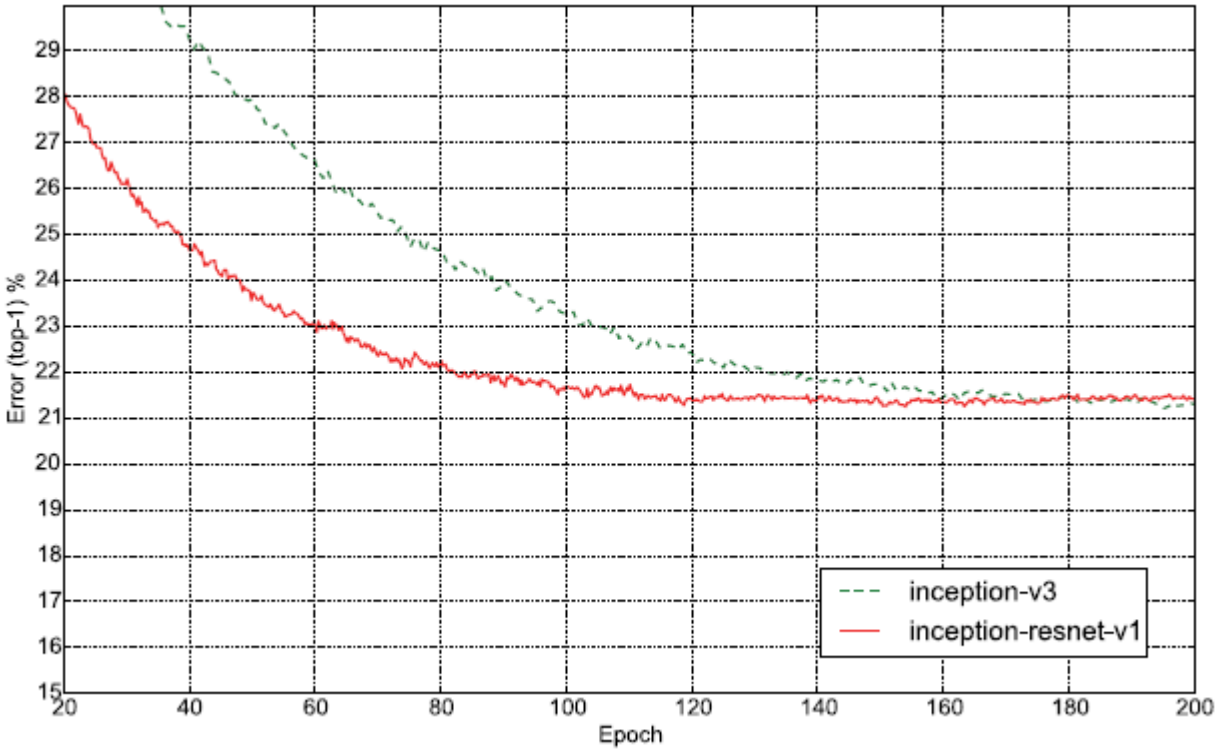




C. Szegedy et al. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” AAAI 2017.

Inception v4 & Inception-ResNet

- Inception-ResNet: add Inception module on ResNet
  → improved training speed



C. Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," AAAI 2017.

Inception v4 & Inception-ResNet

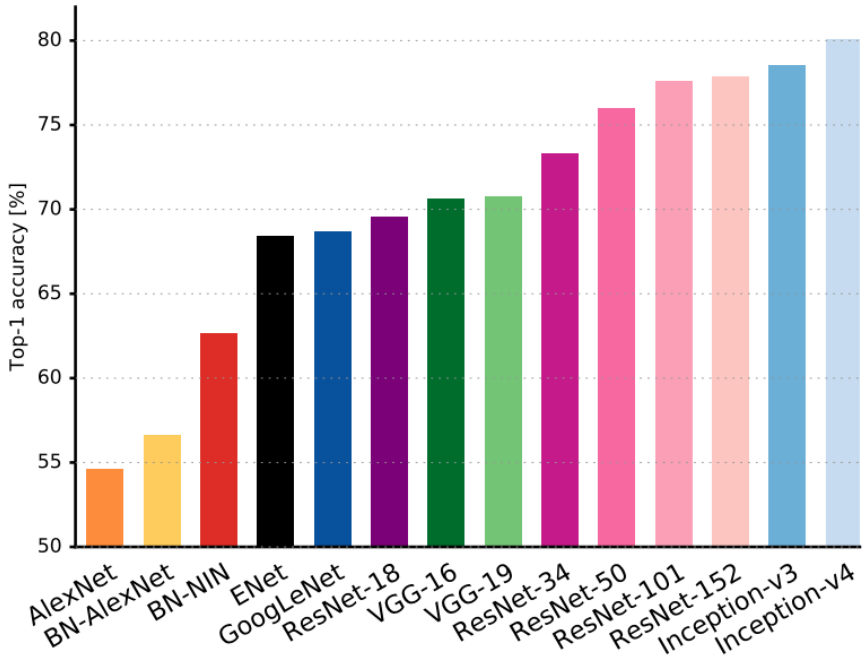
- Inception-ResNet: add Inception module on ResNet
- Performance improvement is marginal

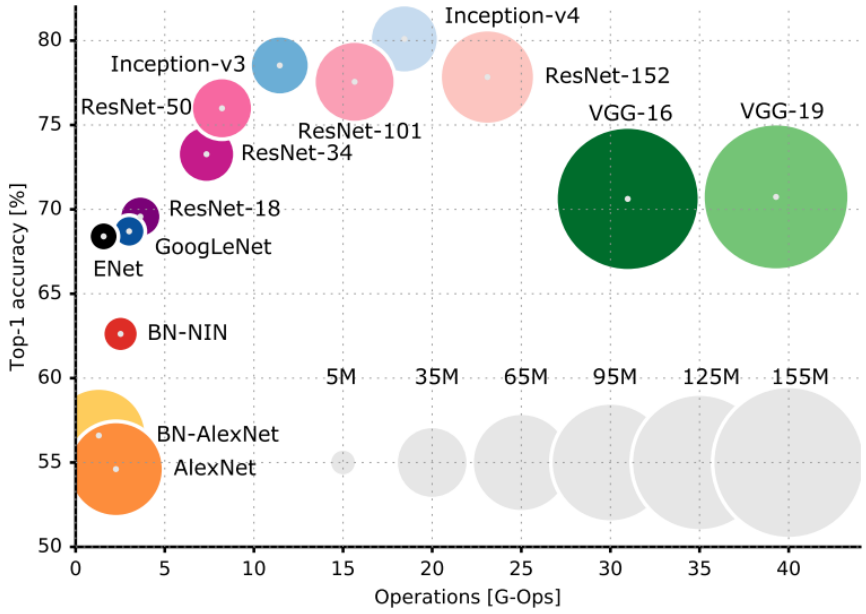
Network	Crops	**Top-1 Error**	**Top-5 Error**
ResNet (He et al. 2015)	10	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

C. Szegedy et al. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” AAAI 2017.

Inception v4 & Inception-ResNet

- Note: Inception is not often used due to …
  - ✓Compared to VGG, model architecture is too complex
  - ✓Compared to ResNet, small merits on performance-complexity tradeoff





A. Canaziani et al., "An Analysis of Deep Neural Network Models for Practical Applications," ICLR 2017.

Xception

- Inception + residual connection + depthwise separable conv.

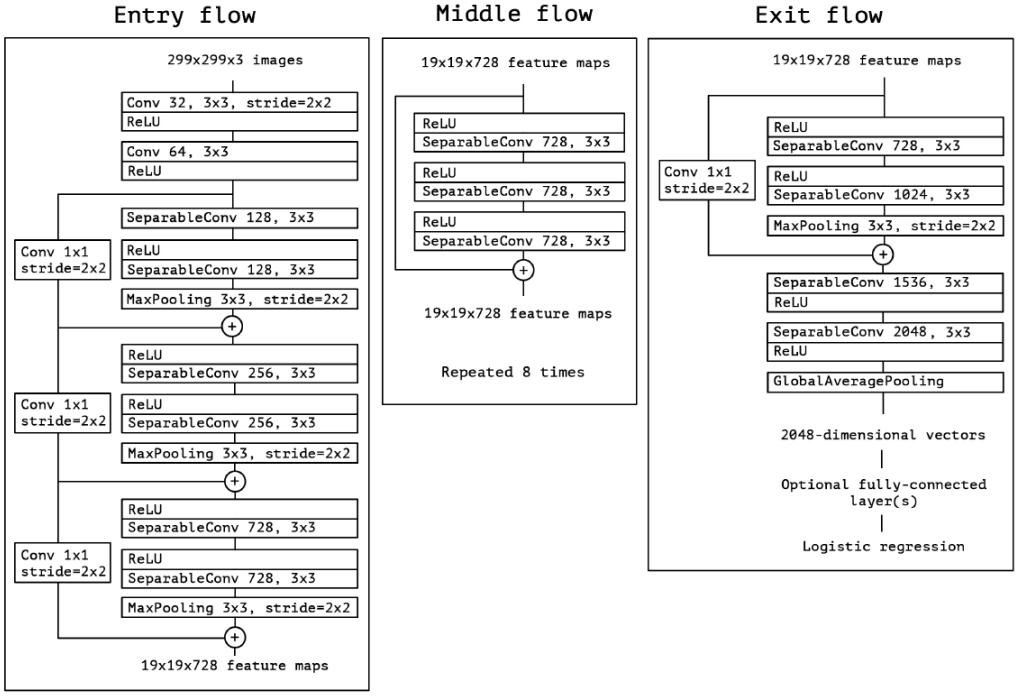


Table 1. Classification performance comparison on ImageNet (single crop, single model). VGG-16 and ResNet-152 numbers are only included as a reminder. The version of Inception V3 being benchmarked does not include the auxiliary tower.

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” CVPR 2017.

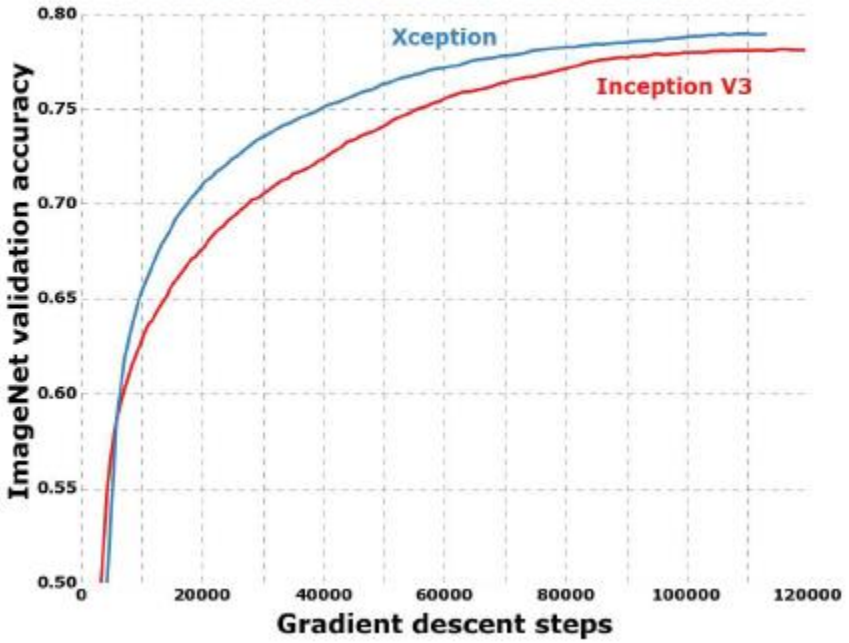
Xception

- Inception + residual connection + depthwise separable conv.

Table 3. Size and training speed comparison.

	Parameter count	Steps/second
Inception V3	23,626,728	31
Xception	22,855,952	28

Figure 6. Training profile on ImageNet



F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," CVPR 2017.

ResNet

- "Revolution of depth"
  - ✓GoogLeNet (22 layers) vs. ResNet (152 layers)
- "The absolute winner" of the year 2015
  - ✓3.6% top 5 error → better than human

K. He et al. "Deep Residual Learning for Image Recognition," CVPR 2016.

ResNet

- How can we train extremely deep models?

  → Residual learning

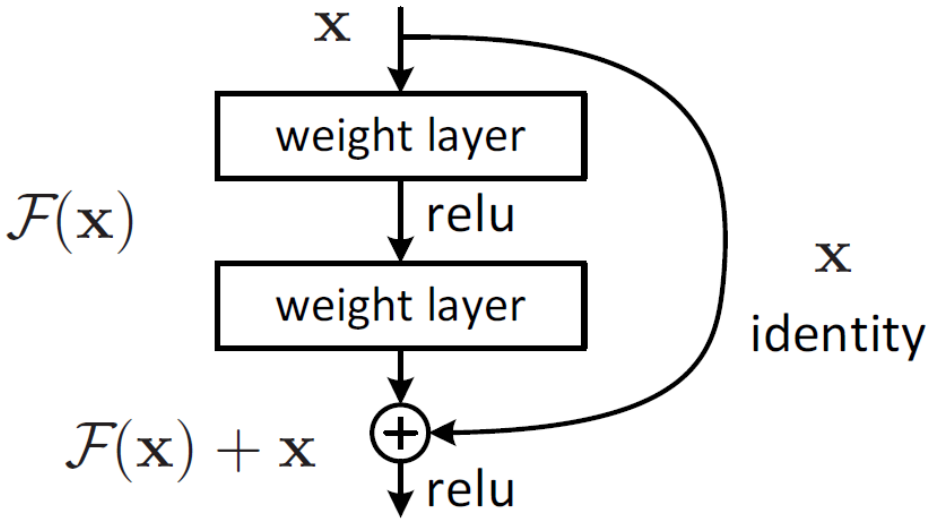


Figure 2. Residual learning: a building block.

K. He et al. "Deep Residual Learning for Image Recognition," CVPR 2016.

ResNet

- Different layer configurations
  - ✓For deeper models, use 1x1 conv. layer for efficiency

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

K. He et al. “Deep Residual Learning for Image Recognition,” CVPR 2016.

ResNet

- ResNet is a “de-facto standard” backbone network for computer vision tasks
  - ✓Face detection (e.g., [CVPR ‘17] TinyFace)
  - ✓Face recognition (e.g., [CVPR ‘19] ArcFace)
  - ✓Image super-resolution (e.g., [CVPR ‘17] EDSR)

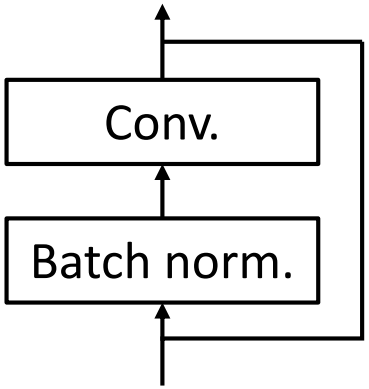
    … and many more

P. Hu et al. “Finding Tiny Faces,” CVPR 2017.
J. Deng et al., “ArcFace: Additive Angular Margin Loss for Deep Face Recognition,” CVPR 2019.
B. Lim et al., “Enhanced Deep Residual Networks for Single Image Super-Resolution,” CVPR 2017.

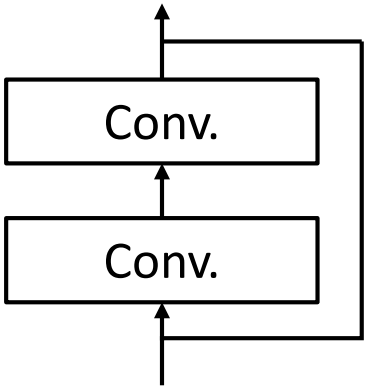
ResNet

- Residual blocks are flexibly adapted depending on target tasks

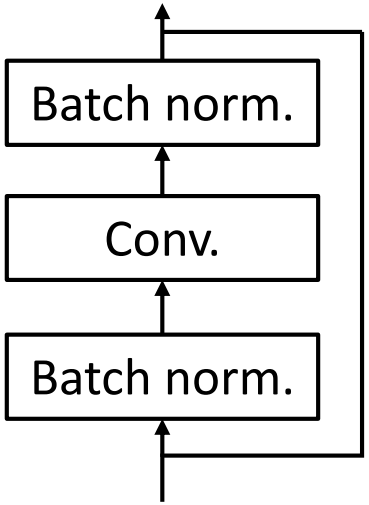
Plain residual block



[CVPR '17] EDSR (super-resolution)



[CVPR '19] ArcFace (face recognition)



P. Hu et al. "Finding Tiny Faces," CVPR 2017.
J. Deng et al., "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," CVPR 2019.
B. Lim et al., "Enhanced Deep Residual Networks for Single Image Super-Resolution," CVPR 2017.

ResNeXt

- 2nd place in ImageNet challenge 2016
- Split input channels into multiple parallel paths i.e., cardinality (similar to group convolution)

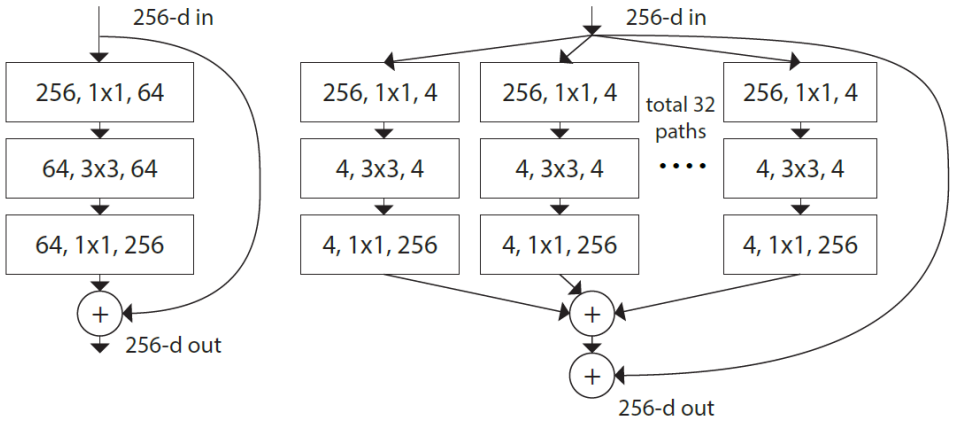


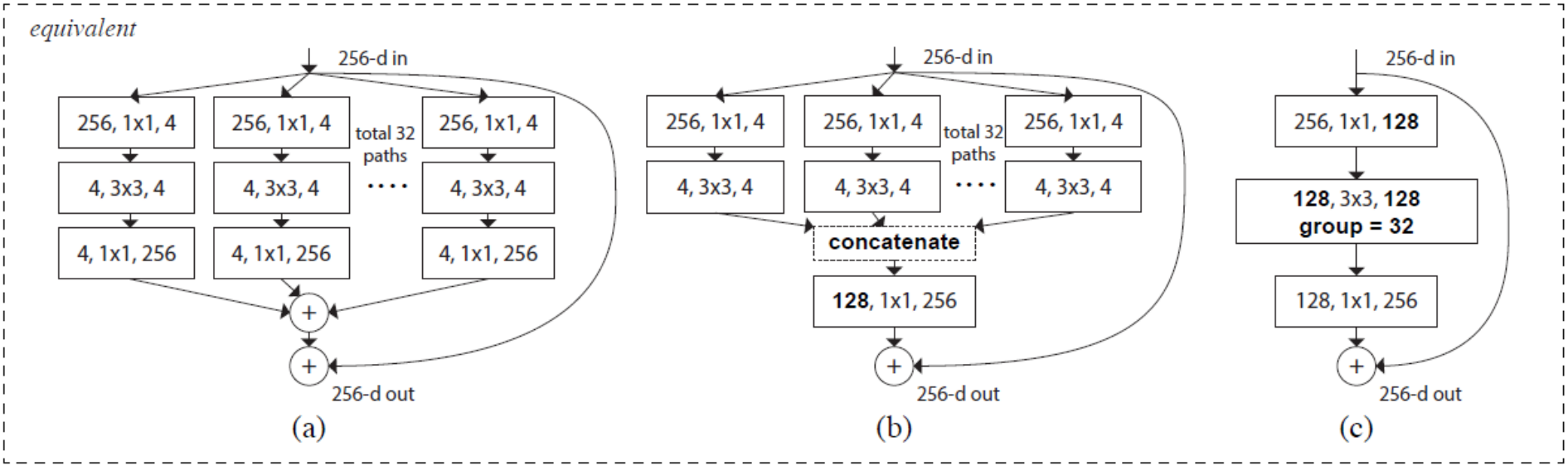
Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

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		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

S. Xie et al. "Aggregated Residual Transformations for Deep Neural Networks," CVPR 2017.

ResNeXt

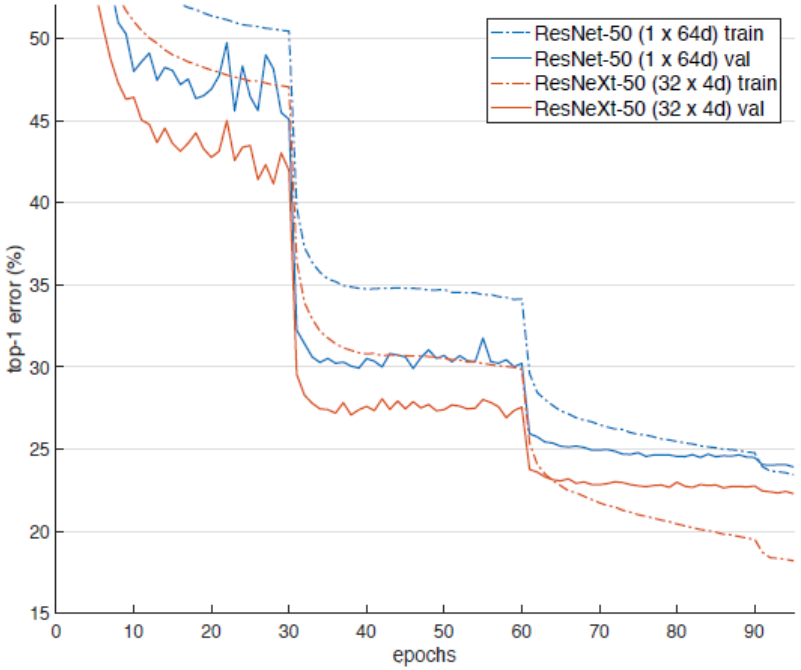
- 2nd place in ImageNet challenge 2016
- Split input channels into multiple parallel paths i.e., cardinality (similar to group convolution)



S. Xie et al. "Aggregated Residual Transformations for Deep Neural Networks," CVPR 2017.

ResNeXt

- Performance



	224×224		320×320 / 299×299	
	top-1 err	top-5 err	top-1 err	top-5 err
ResNet-101 [14]	22.0	6.0	-	-
ResNet-200 [15]	21.7	5.8	20.1	4.8
Inception-v3 [39]	-	-	21.2	5.6
Inception-v4 [37]	-	-	20.0	5.0
Inception-ResNet-v2 [37]	-	-	19.9	4.9
ResNeXt-101 (**64** × **4d**)	20.4	5.3	**19.1**	**4.4**

S. Xie et al. "Aggregated Residual Transformations for Deep Neural Networks," CVPR 2017.

SENet

- Learn channel interdependencies and adaptively recalibrate channel-wise feature responses

  → Similar to attention mechanism

  ✓Squeeze: global information embedding (via global average pooling)

  ✓Excitation: learn channel dependencies and channel-wise scale factors

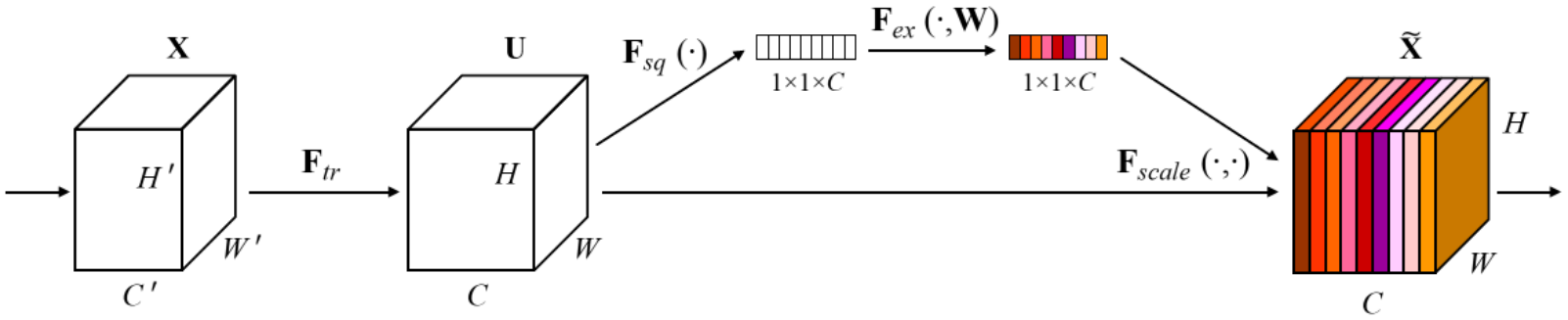


Figure 1: A Squeeze-and-Excitation block.

J. Hu et al. "Squeeze-and-Excitation Networks," CVPR 2018.

SENet

- Can be applied flexibly to any existing network

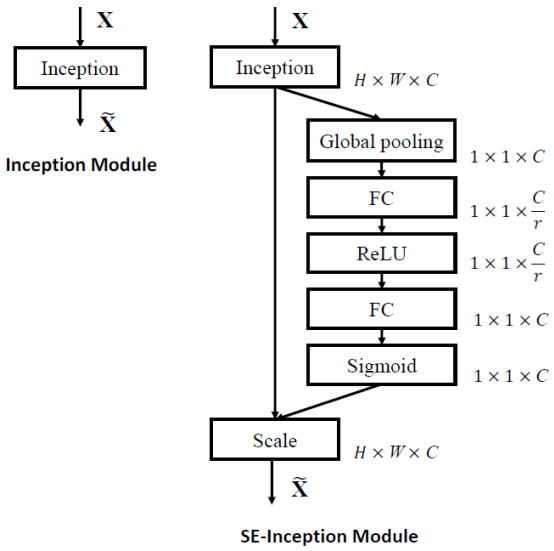


Figure 2: The schema of the original Inception module (left) and the SE-Inception module (right).

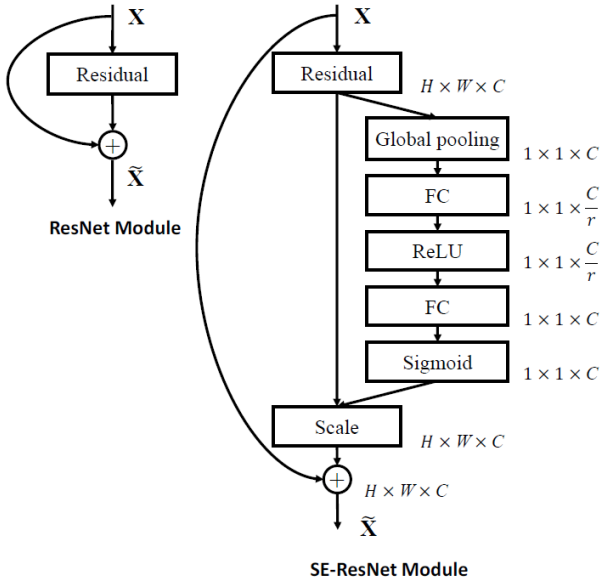


Figure 3: The schema of the original Residual module (left) and the SE-ResNet module (right).

J. Hu et al. "Squeeze-and-Excitation Networks," CVPR 2018.

SENet

- 1-2% accuracy gain with minimal increase in model size & complexity

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [10]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [47]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [47]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [39]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [16]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [42]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

	original		re-implementation				SENet			
	top-1 err.	top-5 err.	top-1 err.	top-5 err.	MFLOPs	Million Parameters	top-1 err.	top-5 err.	MFLOPs	Million Parameters
MobileNet [13]	29.4	-	29.1	10.1	569	4.2	$25.3_{(3.8)}$	$7.9_{(2.2)}$	572	4.7
ShuffleNet [52]	34.1	-	33.9	13.6	140	1.8	$31.7_{(2.2)}$	$11.7_{(1.9)}$	142	2.4

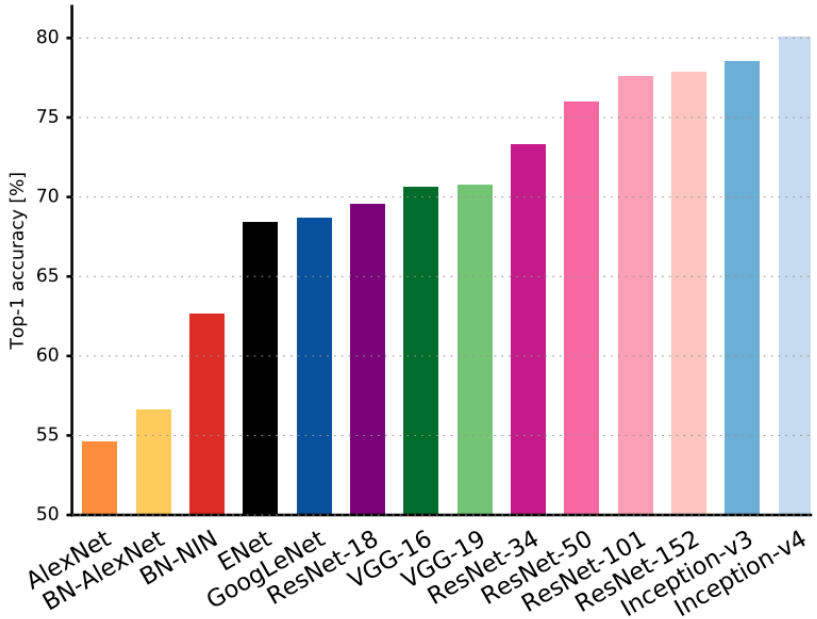
J. Hu et al. "Squeeze-and-Excitation Networks," CVPR 2018.

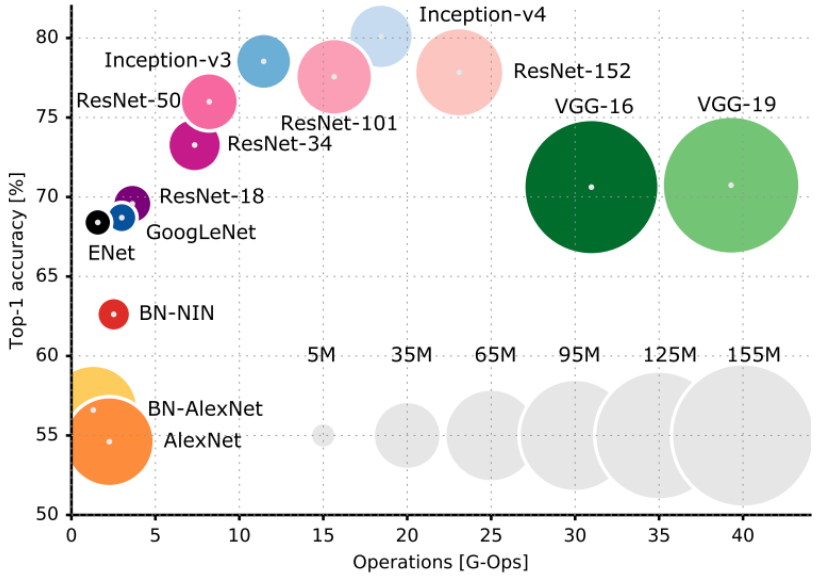
Content Overview

- Image classification networks (with relation to ILSVRC)
  - ✓ AlexNet
  - ✓ VGG
  - ✓ Inception v1, v2, v3, v4 – Xception
  - ✓ ResNet – ResNeXt – SENet
- **Efficient CNNs**
  - ✓ SqueezeNet
  - ✓ MobileNet v1, v2, v3
  - ✓ ShuffleNet
  - ✓ EfficientNet
- Object detection networks
  - ✓ Two stage detection (R-CNN, Fast R-CNN, Faster R-CNN)
  - ✓ Single stage detection (YOLO v1, v2, SSD)
  - ✓ Small object detection

Efficient CNNs

- State-of-the-art classification networks achieve high accuracy, but with 100s of MB model size and 10~100 GFLOPs…
- Can we design "efficient" models in the performance-complexity tradeoff?





A. Canaziani et al., "An Analysis of Deep Neural Network Models for Practical Applications," ICLR 2017.

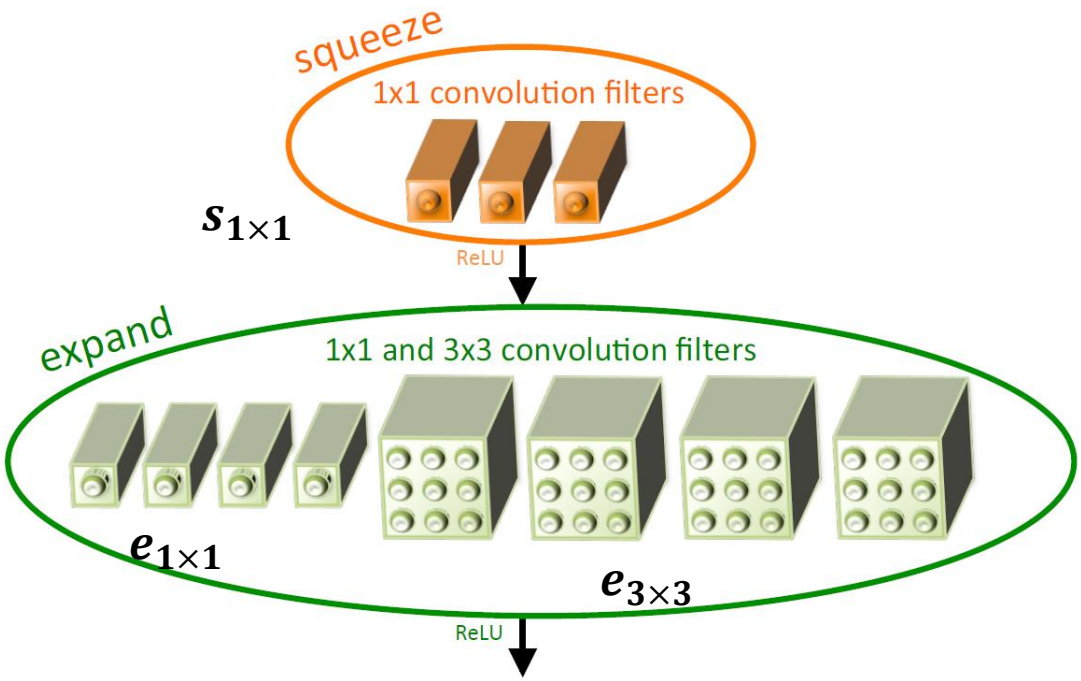
SqueezeNet

- How can we reduce model size?
  - ✓Replace 3x3 filters with 1x1 filters
  - ✓Decrease the number of input channels to 3x3 filters
  - ✓Downsample late in the network so that conv. layers have large activation

F. Iandola et al. “SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size,” ICLR 2017.

SqueezeNet

- Building block: Fire module
- Set $s_{1\times1} < e_{1\times1} + e_{3\times3}$ to reduce number of input channels to 3x3 filters



F. Iandola et al. "SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size," ICLR 2017.

SqueezeNet

- Overall architecture

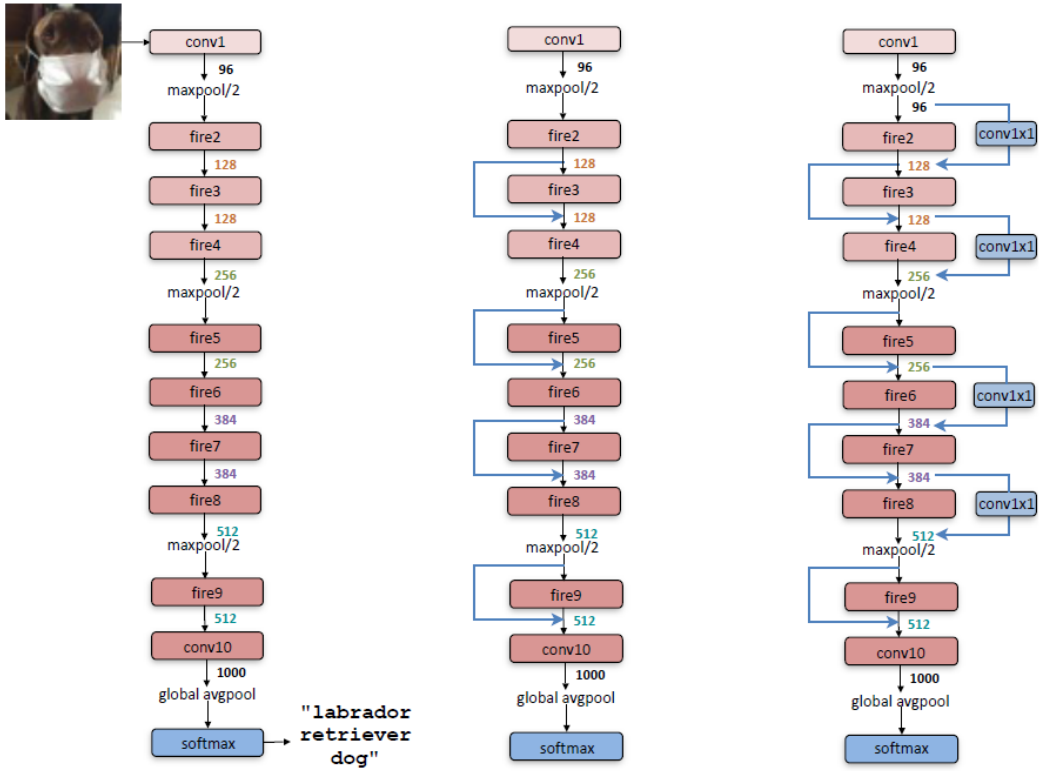


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

F. Iandola et al. "SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size," ICLR 2017.

SqueezeNet

- Model size
  - ✓< 5MB with plain SqueezeNet
  - ✓< 0.5 MB with SqueezeNet + compression

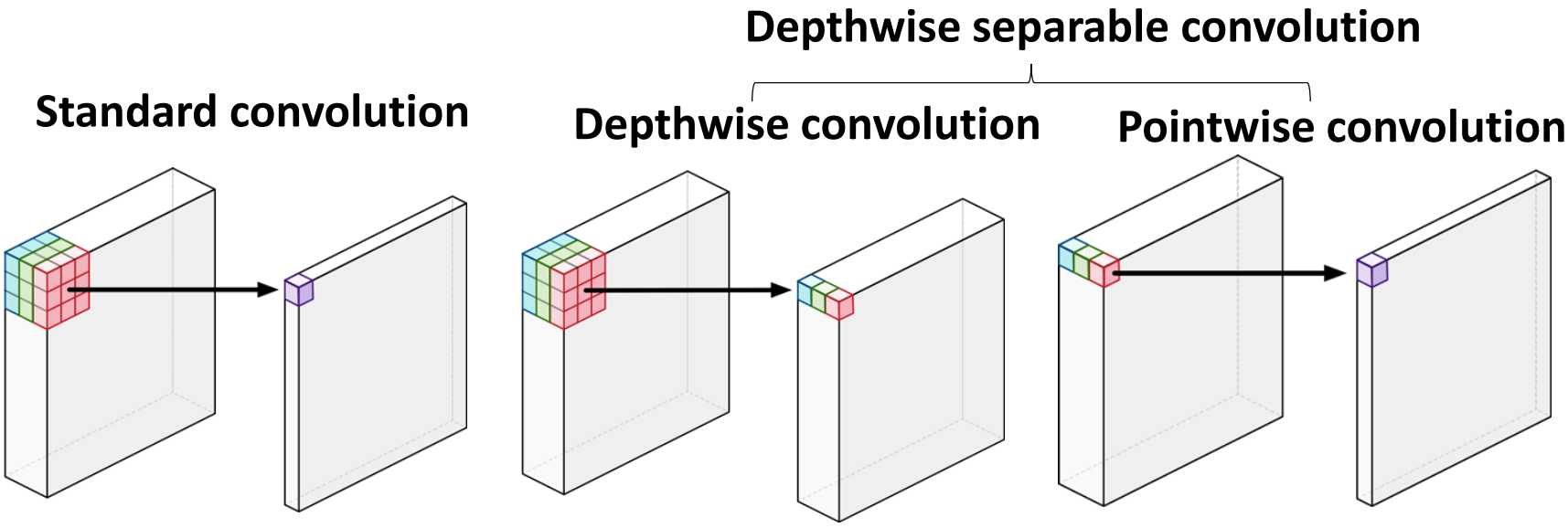
Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

F. Iandola et al. “SqueezeNet: AlexNet-level Accuracy With 50x Fewer Parameters and <0.5Mb Model Size,” ICLR 2017.

MobileNet v1

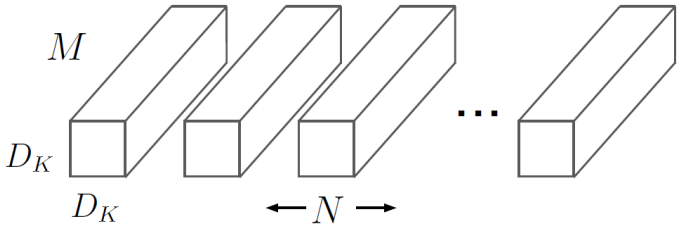
- Depthwise separable convolution



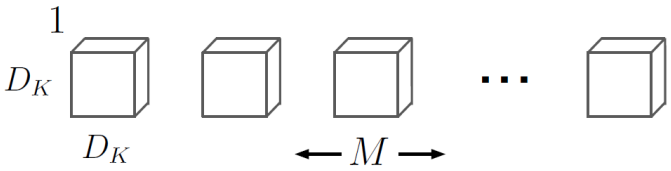
A. Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv 2017.

MobileNet v1

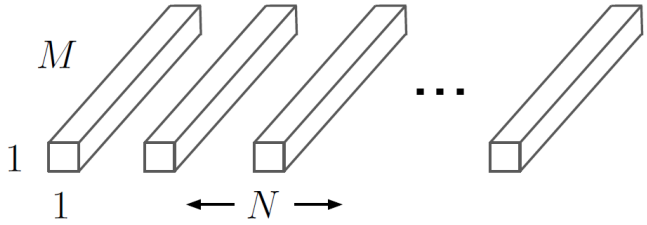
- Depthwise separable convolution



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Complexity of std. conv.

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$$

Complexity of depthwise separable conv.

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

Complexity reduction

$$\frac{1}{N} + \frac{1}{D_K^2}$$

A. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv 2017.

MobileNet v1

- Network architecture

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

A. Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv 2017.

MobileNet v1

- Shrinking hyperparameter
  - ✓Width multiplier → adjust number of input/output channels

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

  - ✓Resolution multiplier → adjust input image size

Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

A. Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv 2017.

MobileNet v1

- Performance

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

A. Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv 2017.

MobileNet v1

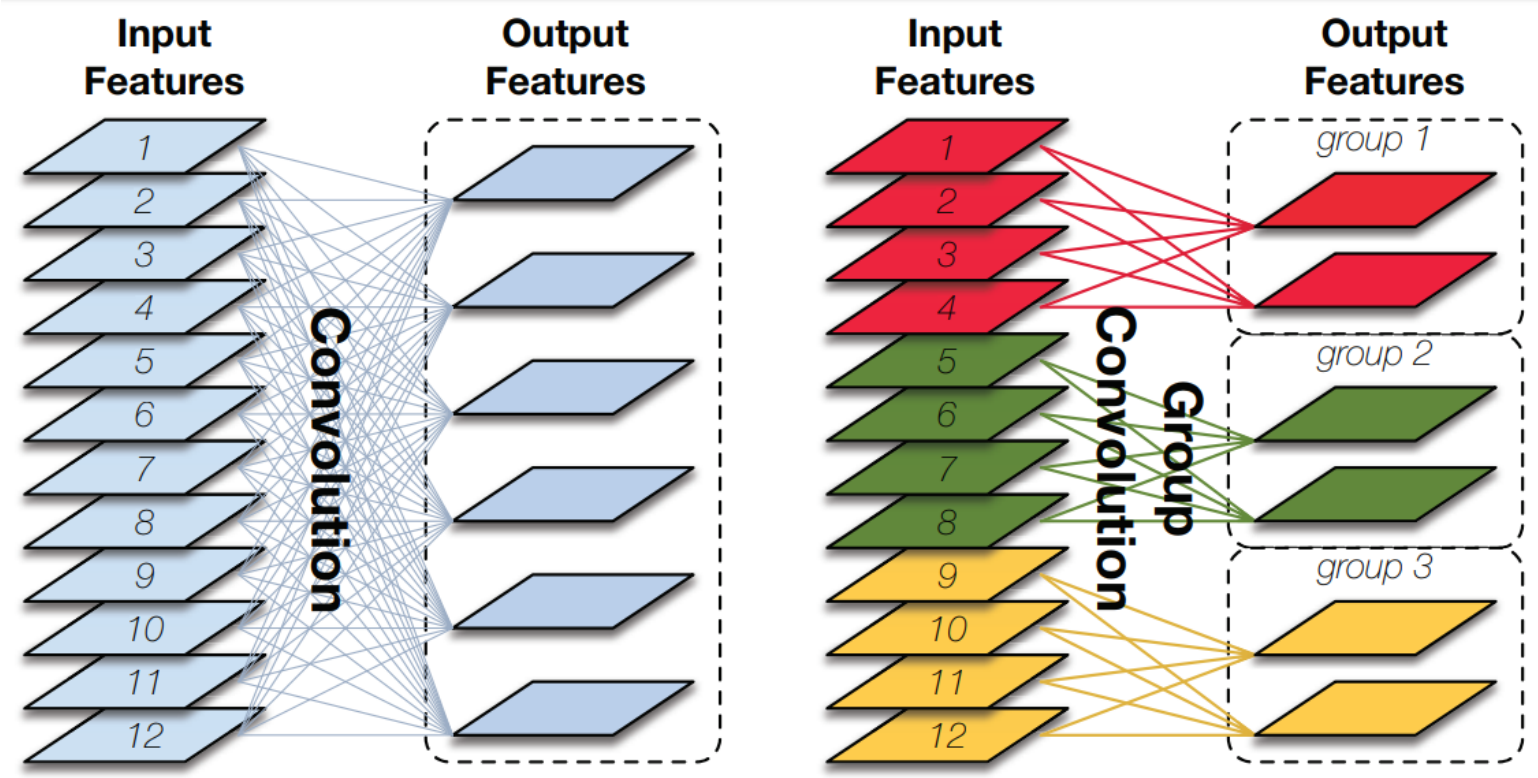
- MobileNet is “de-facto standard” lightweight backbone network for computer vision tasks
  - ✓Face detection (e.g., [arXiv ’19] RetinaFace)
  - ✓Face recognition (e.g., [arXiv ’18] MobileFaceNet)
  - ✓Object detection

    … and many more

J. Deng et al. “RetinaFace: Single-stage Dense Face Localisation in the Wild,” arXiv 2019.
S. Chen et al. “MobileFaceNets: Efficient CNNs for Accurate Real-Time Face Verification on Mobile Devices,” arXiv 2018.

ShuffleNet

- Group convolution reduces computation

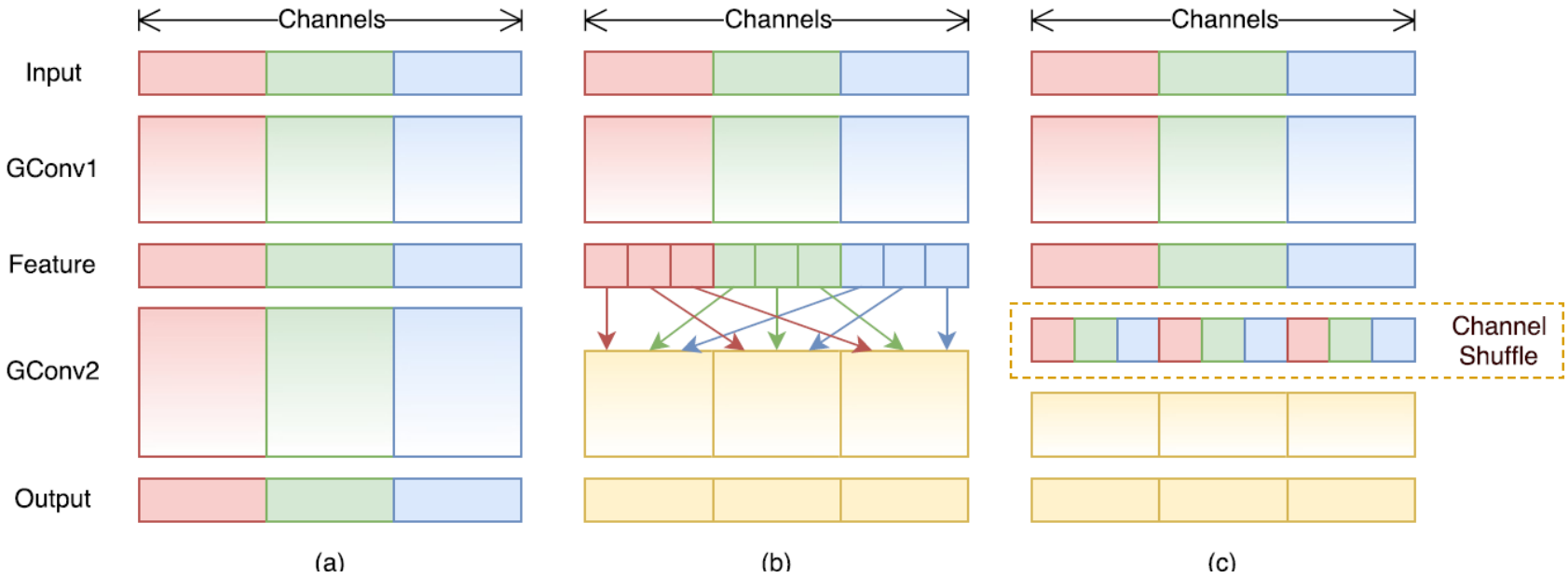


(figure from CondenseNet paper)

G. Huang et al. "CondenseNet: An Efficient DenseNet using Learned Group Convolutions," CVPR 2018.

ShuffleNet

- However, naïvely applying group convolution degrades performance (∵ no cross talk between groups)
  → Channel shuffle layer



X. Zhang et al. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," CVPR 2018.

ShuffleNet

- Higher accuracy than MobileNet v1 with less FLOPs

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**26.3**	3.1
ShuffleNet 2× (with SE[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× ($g = 4$)	38	**41.6**	7.8
ShuffleNet 0.5× (shallow, $g = 3$)	40	42.8	6.6

X. Zhang et al. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices,” CVPR 2018.

MobileNet v2

- Memory-efficient MobileNet
  - ✓ Conventional residual block
    - ➢ Decrease # of channels with 1x1. → 3x3 conv. → Increase # of channels with 1x1 conv.
  - ✓ Inverted residual block
    - ➢ Increase # of channels with 1x1. → 3x3 conv. → Decrease # of channels with 1x1 conv.
  - → Memory efficient (∵ input-output channels are smaller)

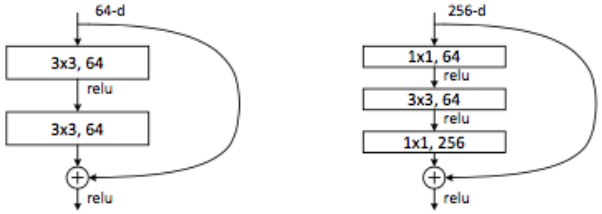
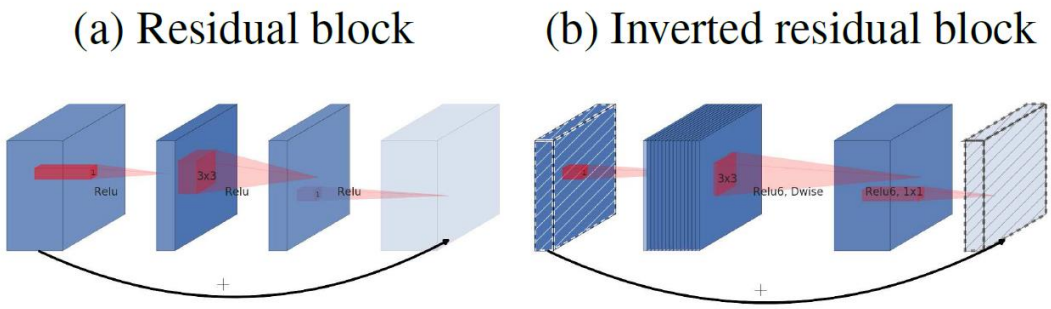


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.



M. Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks," CVPR 2018.

MobileNet v2

- Performance on various tasks

Classification

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Detection

Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

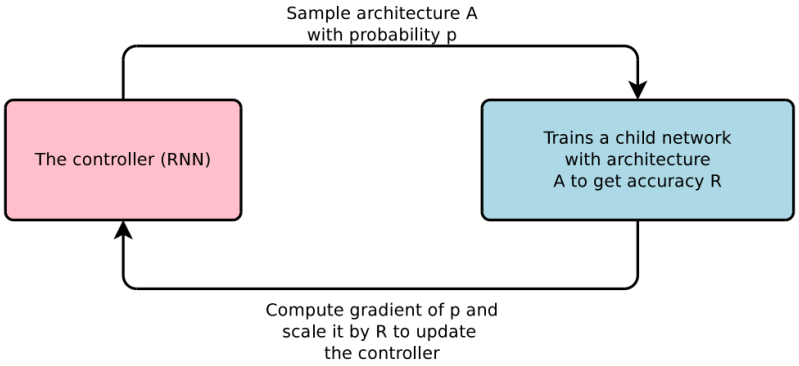
Segmentation

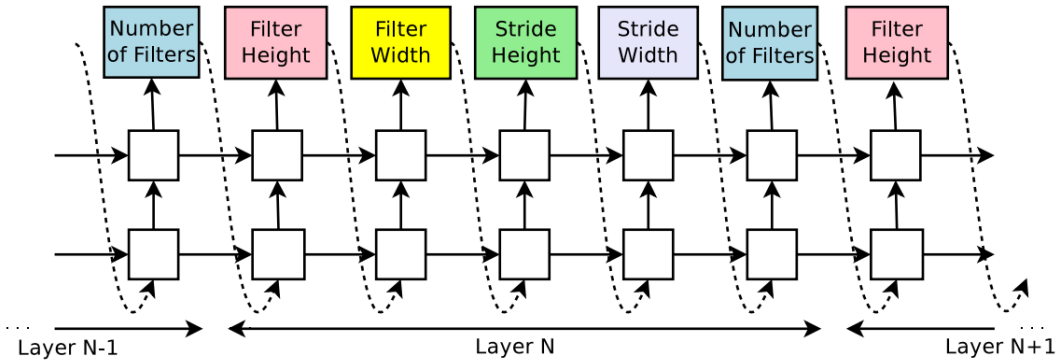
Network	OS	ASPP	MF	mIOU	Params	MAdds
MNet V1	16	✓		75.29	11.15M	14.25B
	8	✓	✓	78.56	11.15M	941.9B
MNet V2*	16	✓		75.70	4.52M	5.8B
	8	✓	✓	78.42	4.52M	387B
MNet V2*	16			**75.32**	**2.11M**	**2.75B**
	8		✓	77.33	2.11M	152.6B
ResNet-101	16	✓		80.49	58.16M	81.0B
	8	✓	✓	82.70	58.16M	4870.6B

M. Sandler et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” CVPR 2018.

Background: NAS

- Designing a state-of-the-art network architecture requires a lot of expert knowledge & efforts
  → Can we train AI to design the network architecture?
- Neural Architecture Search (NAS)

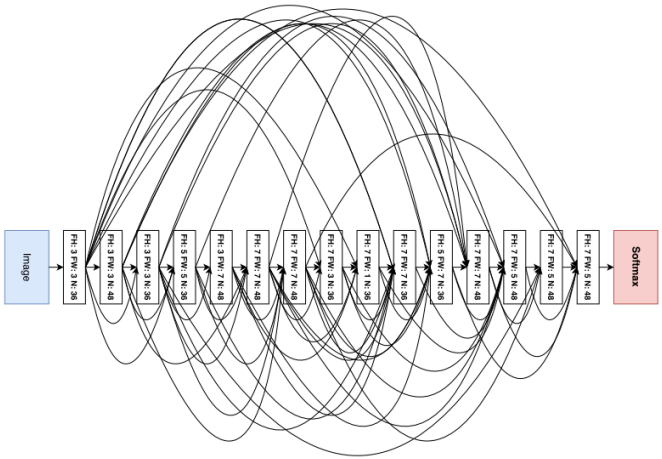




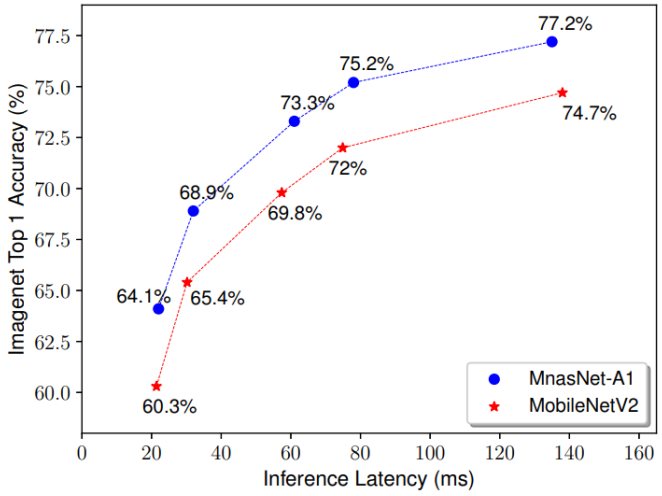
B. Zoph and Q. Le., "Neural Architecture Search with Reinforcement Learning," ICLR 2017.
B. Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition," CVPR 2018.
M. Tan et al., "MnasNet: Platform-Aware Neural Architecture Search for Mobile," CVPR 2019.

Background: NAS

- With lots of GPUs & search time…
  - ✓NAS: 800 GPUs, 28 days
  - ✓NASNet: 500 GPUs, 4 days
  - ✓MnasNet: 64 TPUs, 4.5 days



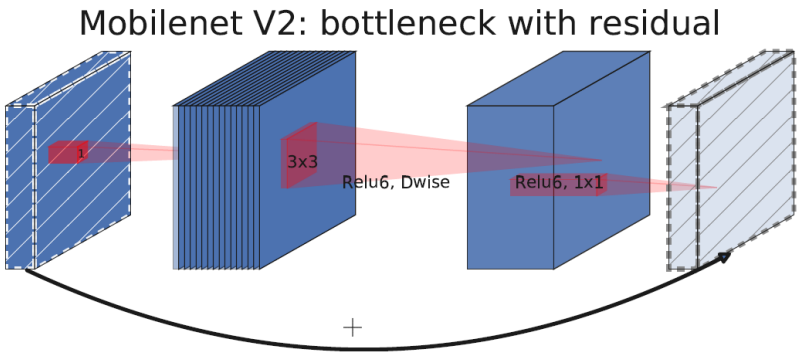
Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [60]	299×299	23.8 M	5.72 B	78.8	94.4
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [58]	299×299	55.8 M	13.2 B	80.1	95.1
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [68]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [69]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

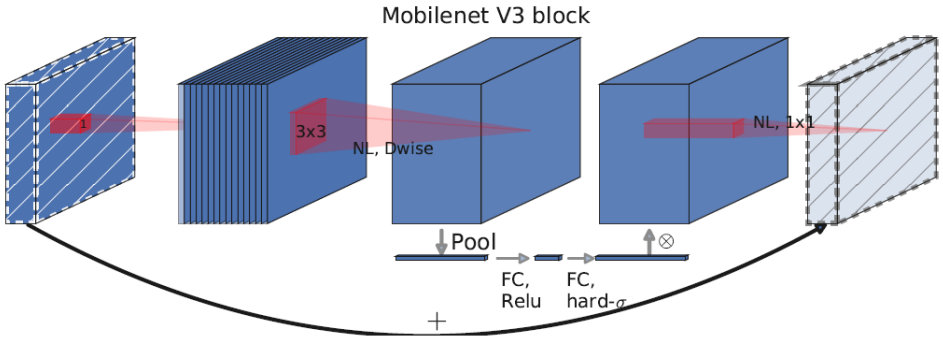


B. Zoph and Q. Le., "Neural Architecture Search with Reinforcement Learning," ICLR 2017.
B. Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition," CVPR 2018.
M. Tan et al., "MnasNet: Platform-Aware Neural Architecture Search for Mobile," CVPR 2019.

MobileNet v3

- MobileNet v2 + Squeeze-and-Excite block

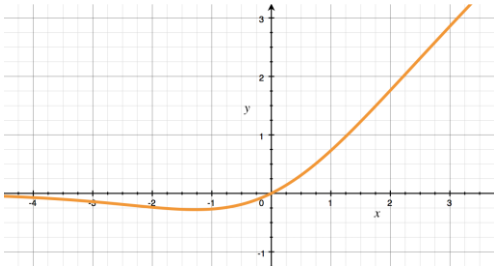




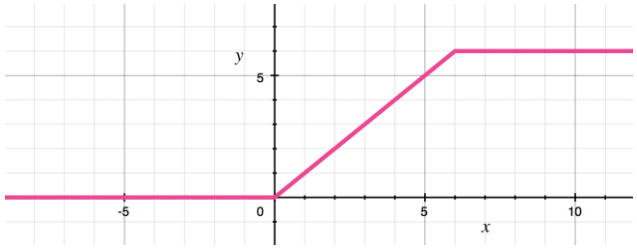
- NAS

- H-Swish nonlinearity

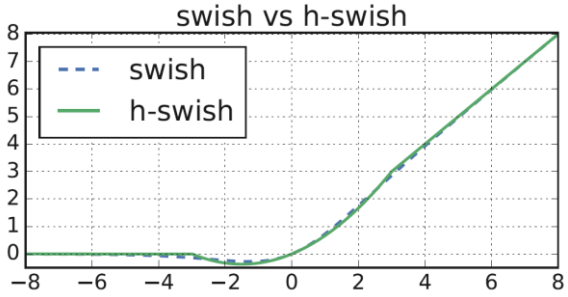
$$\text{swish}\, x = x \cdot \sigma(x)$$



$$ReLU6 = \min(\max(0, x), 6)$$



$$\text{h-swish}[x] = x\frac{\text{ReLU6}(x+3)}{6}$$



A. Howard et al. “Searching for MobileNetV3,” ICCV 2019.

MobileNet v3

- Architecture

MobileNetV3-Large

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

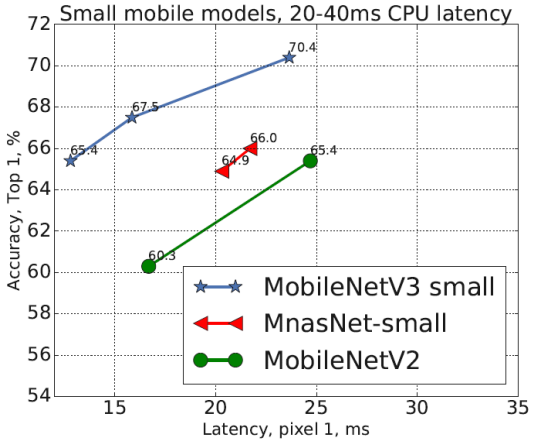
MobileNetV3-Small

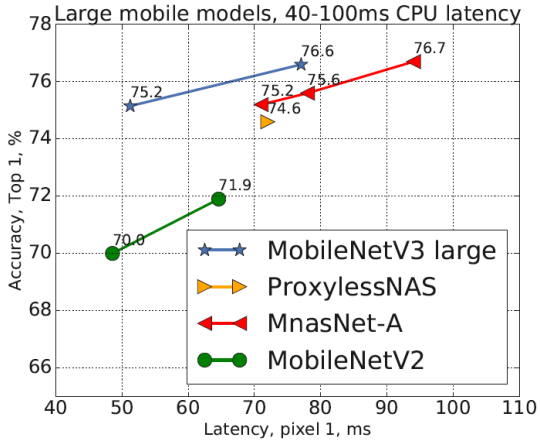
Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

A. Howard et al. “Searching for MobileNetV3,” ICCV 2019.

MobileNet v3

- Performance





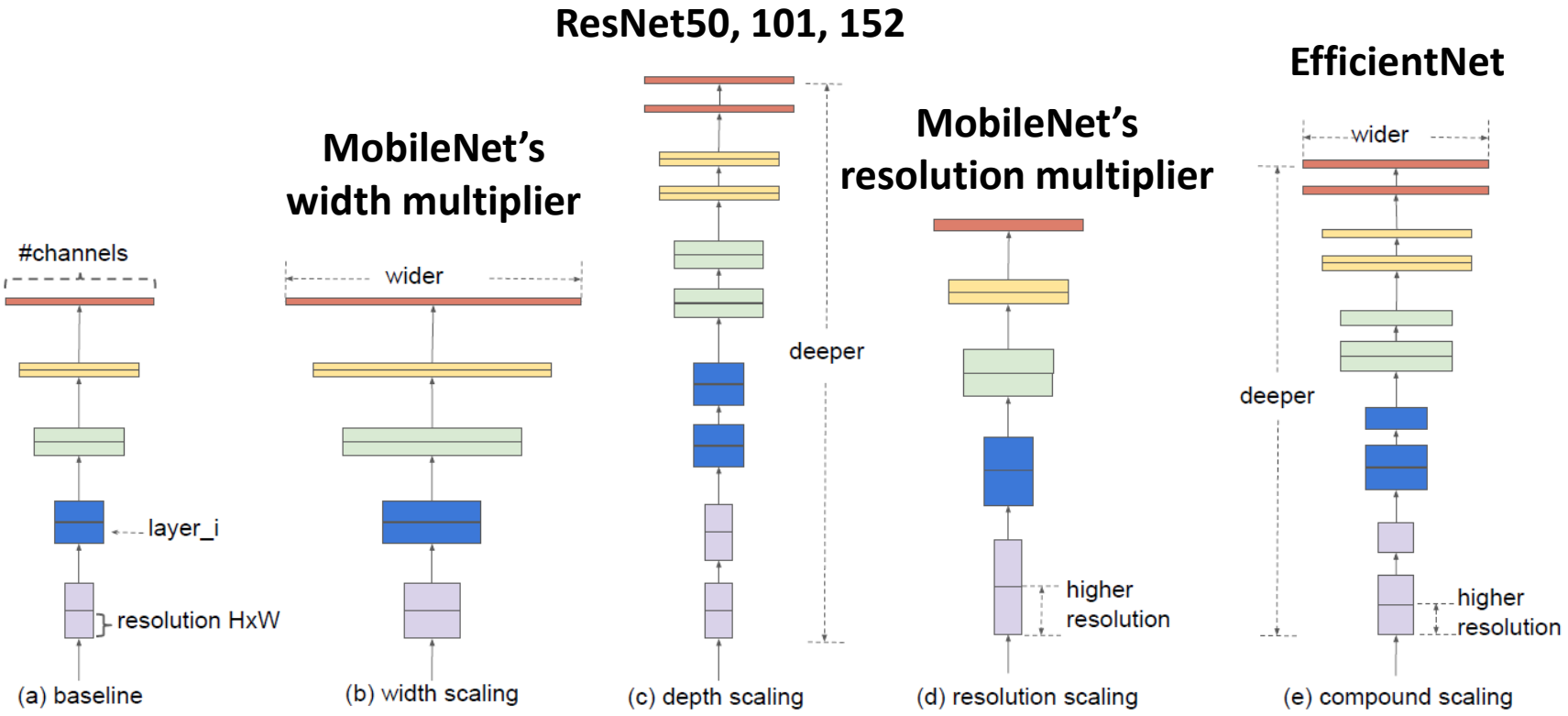
Network	Top-1	MAdds	Params	P-1	P-2	P-3
V3-Large 1.0	**75.2**	**219**	5.4M	**51**	**61**	**44**
V3-Large 0.75	73.3	155	4.0M	39	46	40
MnasNet-A1	75.2	315	3.9M	71	86	61
Proxyless[5]	74.6	320	4.0M	72	84	60
V2 1.0	72.0	300	3.4M	64	76	56
V3-Small 1.0	67.4	66	2.9M	15.8	19.4	14.4
V3-Small 0.75	65.4	44	2.4M	12.8	15.6	11.7
Mnas-small [43]	64.9	65.1	1.9M	20.3	24.2	17.2
V2 0.35	60.8	59.2	1.6M	16.6	19.6	13.9

Table 3. Floating point performance on the Pixel family of phones (P-n denotes a Pixel-n phone). All latencies are in ms and are measured using a single large core with a batch size of one. Top-1 accuracy is on ImageNet.

A. Howard et al. “Searching for MobileNetV3,” ICCV 2019.

EfficientNet

- Various methods to adjust complexity of models…
- Can we consider them all to maximize performance?

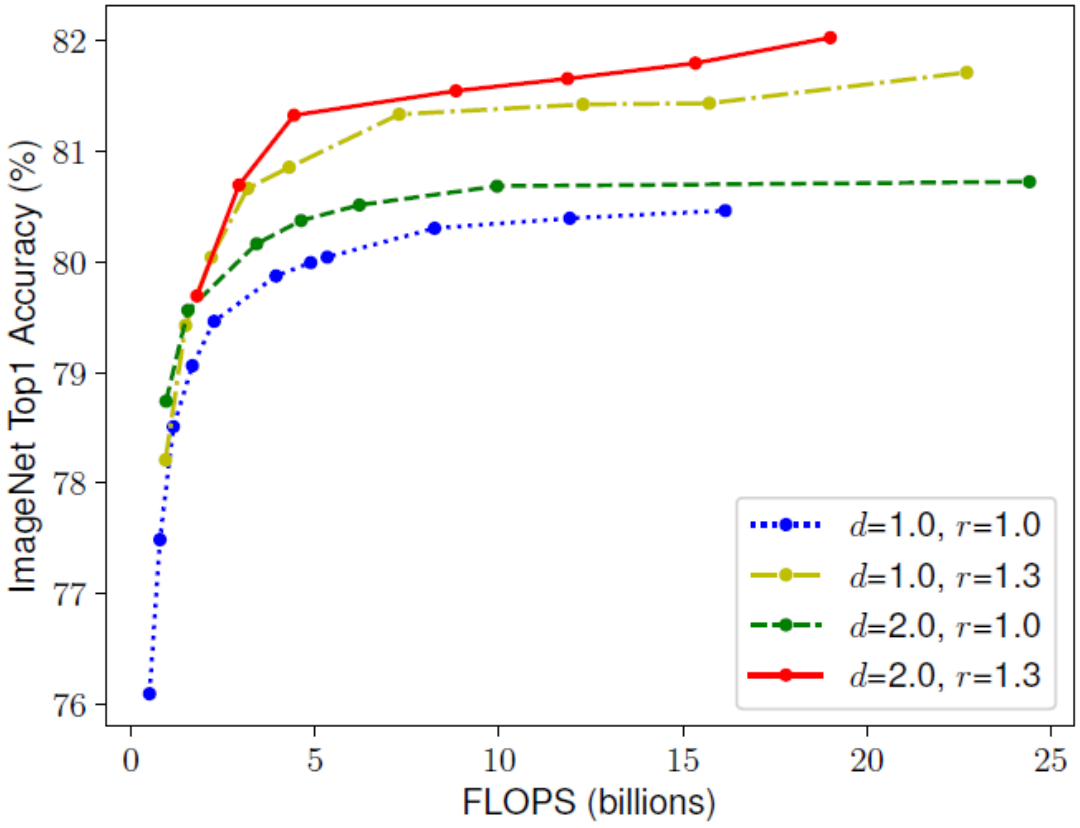


M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML 2019.

EfficientNet

- Is compound scaling feasible?

  → Yes!



M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML 2019.

EfficientNet

- Baseline EfficientNet-B0 is chosen by NAS, similar to MnasNet

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

- Compound scaling
  - ✓ Fix ϕ =1, and do grid search to find best (α, β, γ)
    → (1.2, 1.1, 1.15)
  - ✓ Uniformly adjust ϕ to get EfficientNet-B1~B7

$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML 2019.

EfficientNet

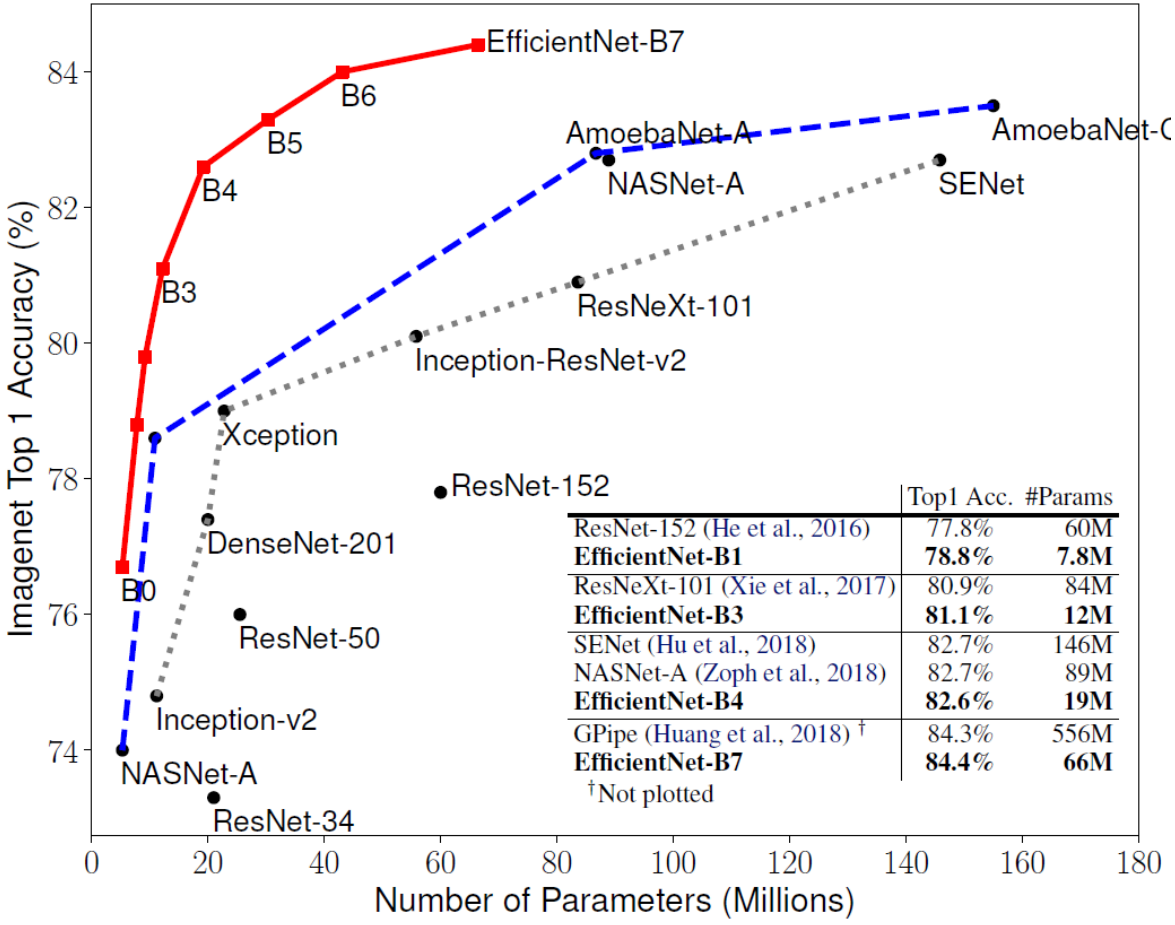
- Performance comparison

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**78.8%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**79.8%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.1%**	**95.5%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML 2019.

EfficientNet

- Performance comparison



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML 2019.

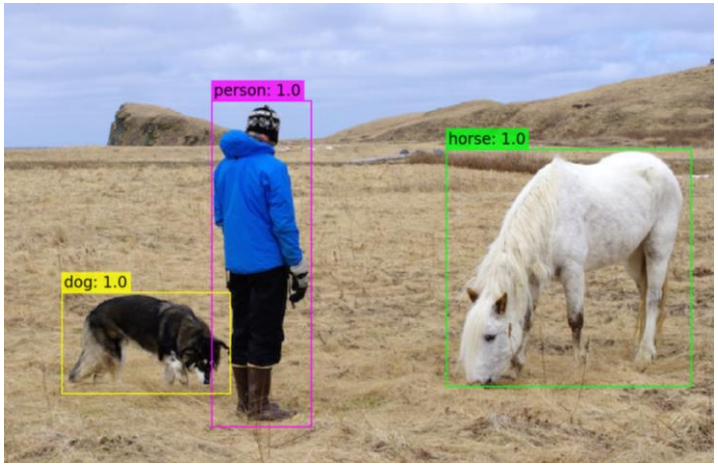
Content Overview

- Image classification networks (with relation to ILSVRC)
  - ✓ AlexNet
  - ✓ VGG
  - ✓ Inception v1, v2, v3, v4 – Xception
  - ✓ ResNet – ResNeXt – SENet
- Efficient CNNs
  - ✓ SqueezeNet
  - ✓ MobileNet v1, v2, v3
  - ✓ ShuffleNet
  - ✓ EfficientNet
- **Object detection networks**
  - ✓ Two stage detection (R-CNN, Fast R-CNN, Faster R-CNN)
  - ✓ Single stage detection (YOLO v1, v2, SSD)
  - ✓ Small object detection

Basics

- Task overview

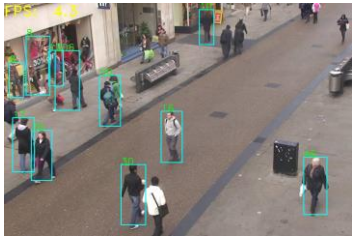
General object detection



Specific object detection

Face

Pedestrian



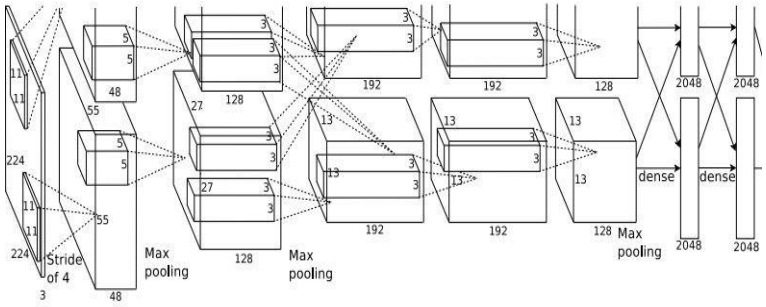
Road sign

Text

Basics

- How do we detect multiple objects in an image?

Apply CNN to region of an image



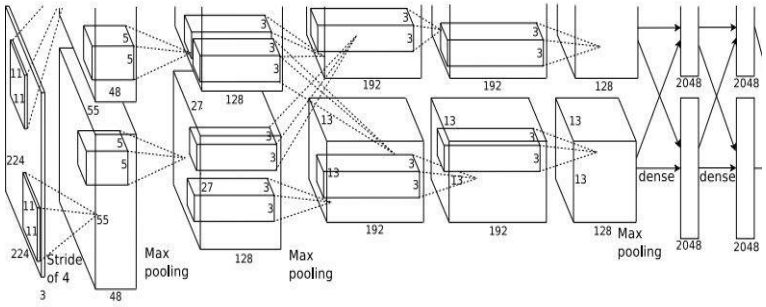
Dog → No
Cat → No
Background → Yes

Figures and text from CS231n 2019 Lecture 12. “Detection and Segmentation”

Basics

- How do we detect multiple objects in an image?

Apply CNN to region of an image

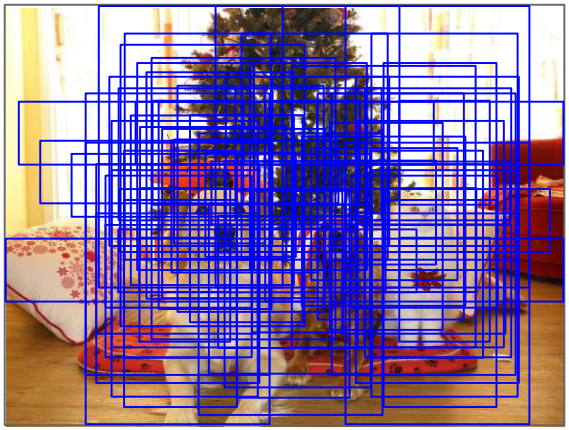


Dog → Yes
Cat → No
Background → No

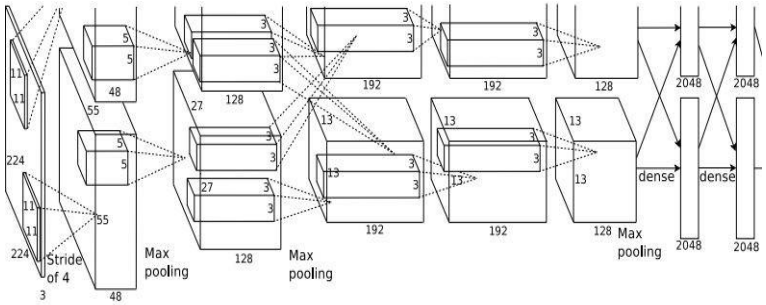
Figures and text from CS231n 2019 Lecture 12. “Detection and Segmentation”

Basics

- How do we detect multiple objects in an image?



Apply CNN to region of an image



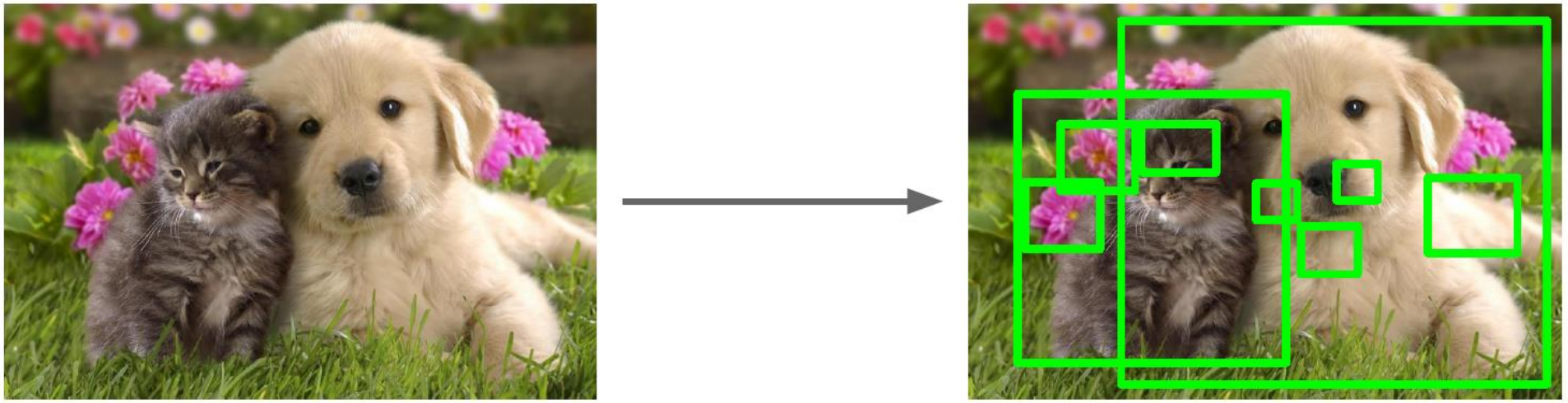
Dog → No
Cat → No
Background → Yes

Question: how do we choose regions?

Figures and text from CS231n 2019 Lecture 12. “Detection and Segmentation”

Basics

- Selective search
  - ✓Find “blobby” regions
  - ✓Runs relatively fast (e.g., 2000 proposals in few seconds in CPU)

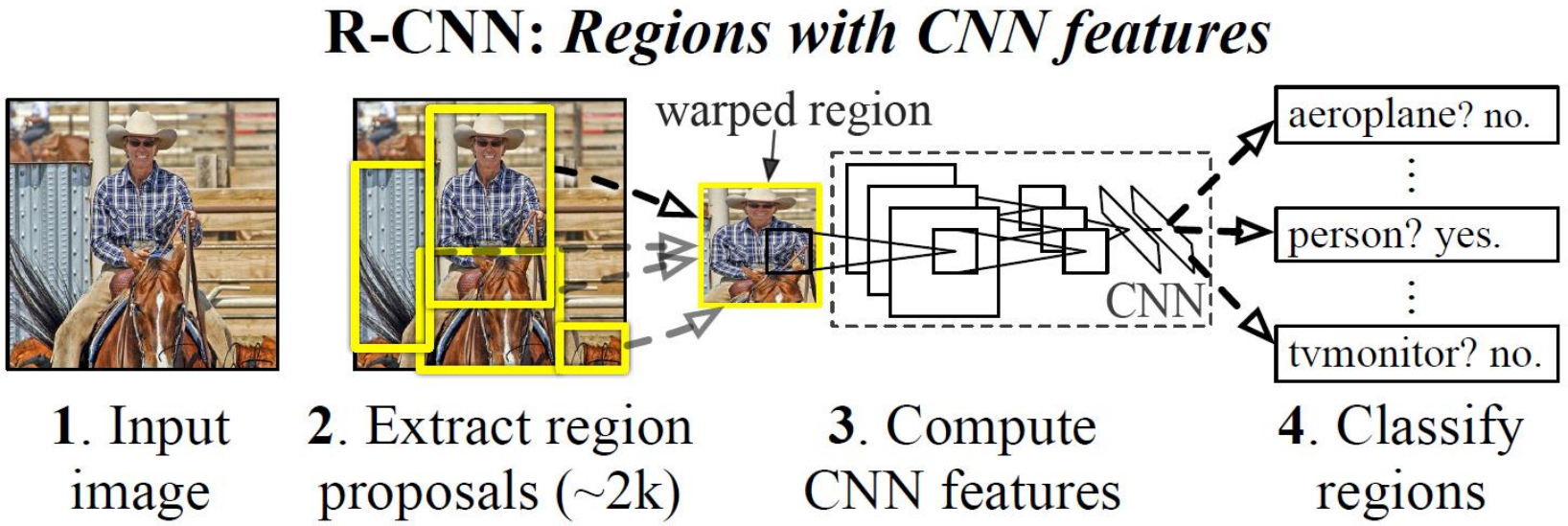


Uijilings et al., “Selective Search for Object Recognition,” IJCV 2013.

Figures text from CS231n 2019 Lecture 12. “Detection and Segmentation”

R-CNN

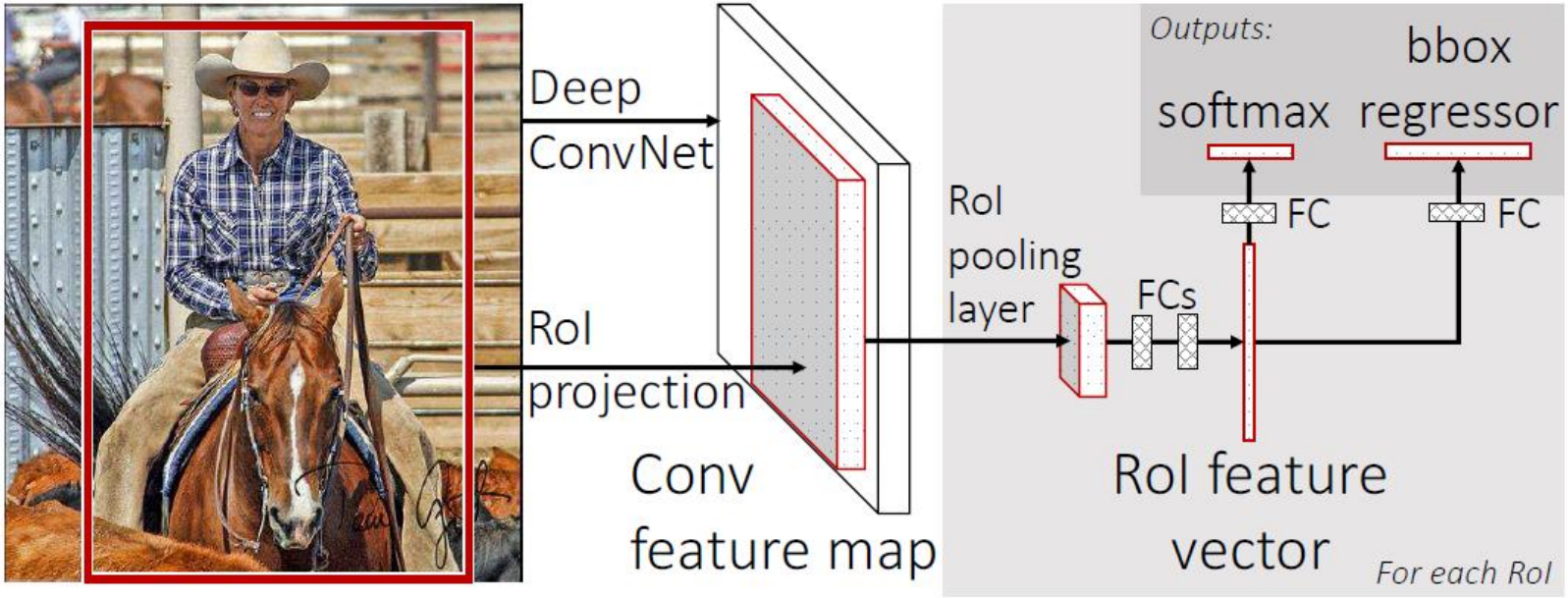
- CNN-based feature extraction + SVM-based region classification
- Apply convolution on 2,000 region proposals from selective search



R. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation," CVPR 2014.

Fast R-CNN

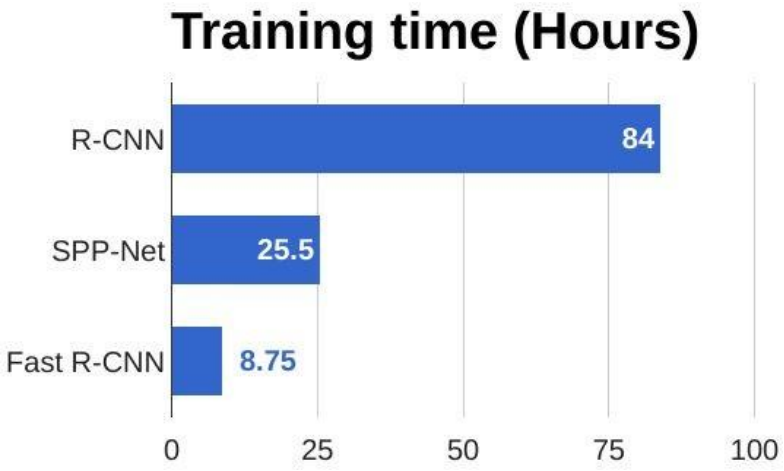
- Problem: R-CNN is super slow! (has to run CNN 2000 times)
- Enhancements
  - ✓ Apply CNN on an whole image, pool ROI regions for each proposal
  - ✓ Replace SVM to fully connected + softmax layers

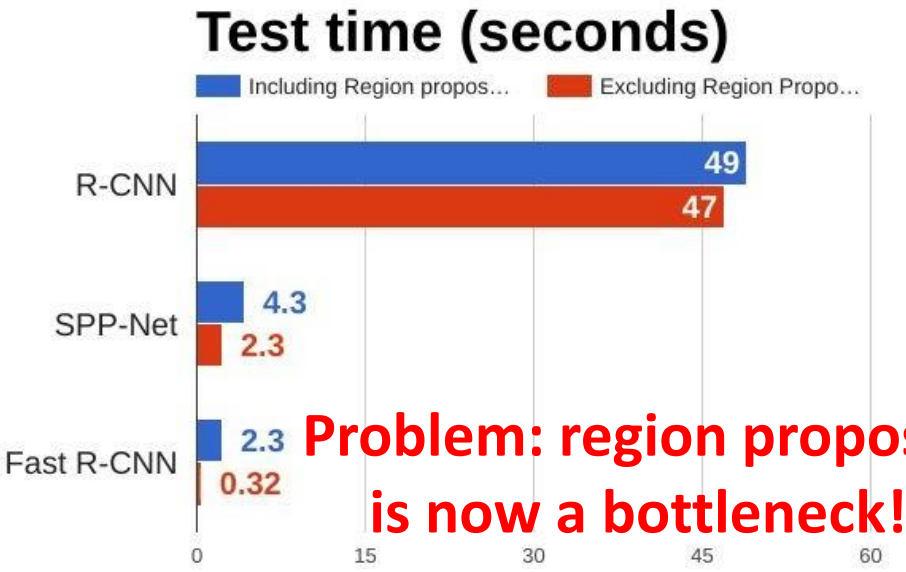


R. Girshick, "Fast R-CNN," ICCV 2015.

Fast R-CNN

- Problem: R-CNN is super slow! (has to run CNN 2000 times)
- Enhancements
  - ✓ Apply CNN on an whole image, pool ROI regions for each proposal
  - ✓ Replace SVM to fully connected + softmax layers





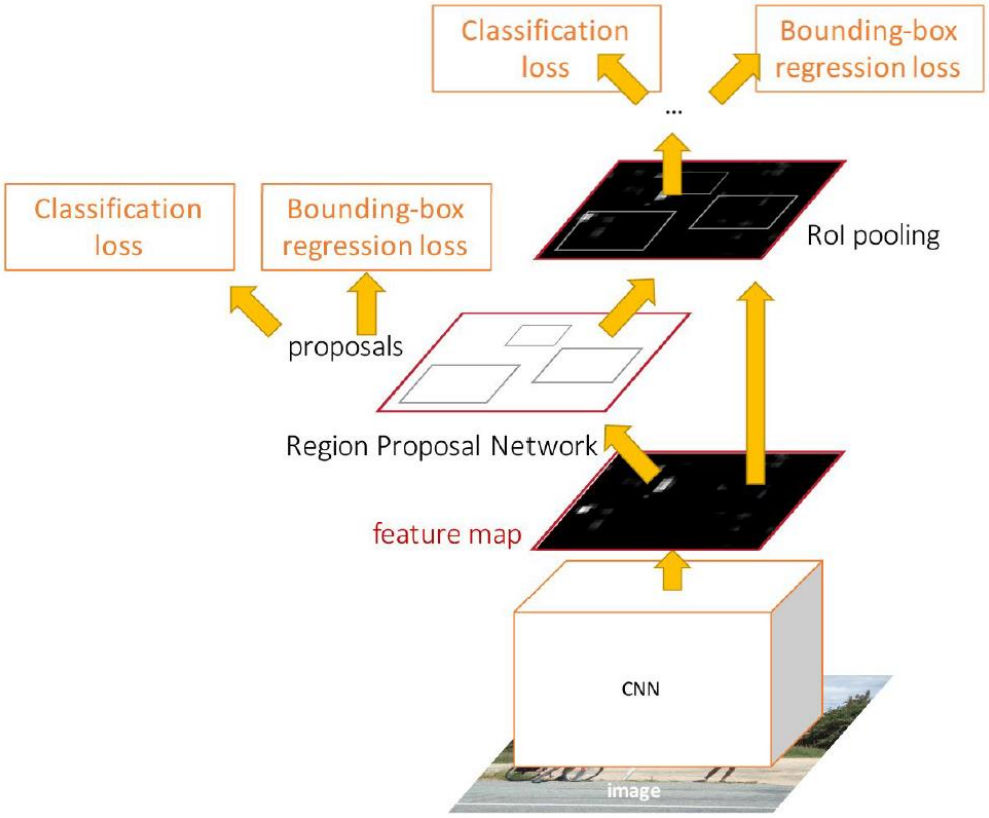
Problem: region proposal is now a bottleneck!

R. Girshick, "Fast R-CNN," ICCV 2015.

Faster R-CNN

- Integrate region proposal into the CNN

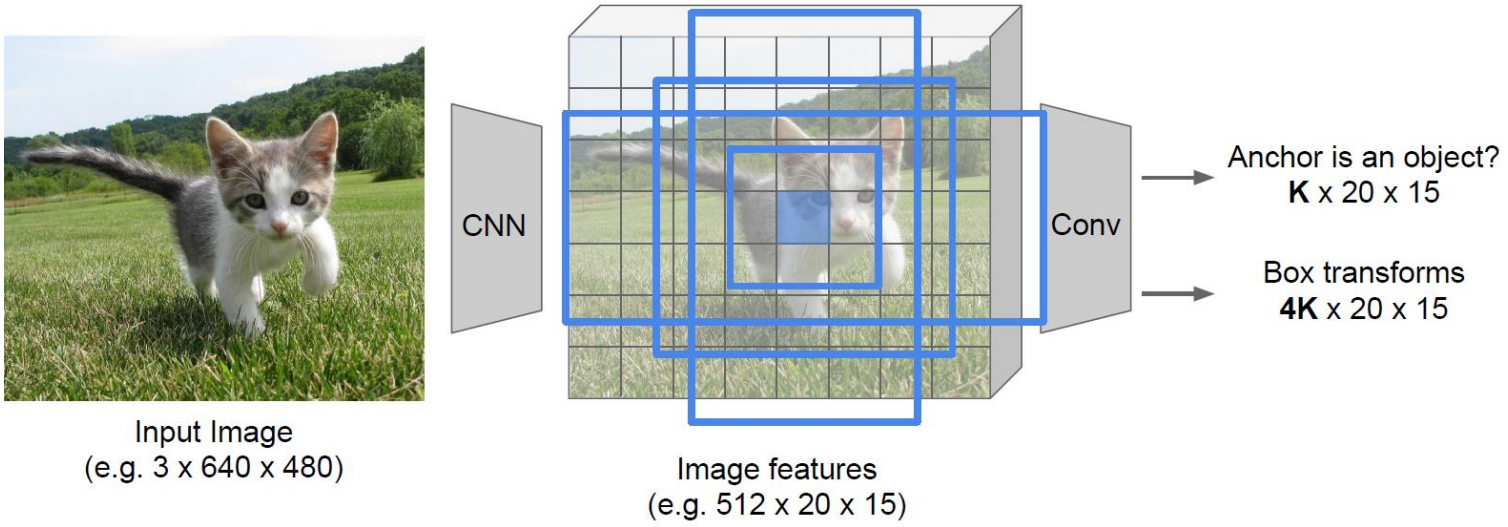
  → Region Proposal Network (RPN)



S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," NIPS 2015.

Faster R-CNN

- Region Proposal Network (RPN)
  - ✓Predefine K anchor boxes
  - ✓Extract image feature
  - ✓For each pixel (i.e., center), classify whether it is an object + regress the bounding box

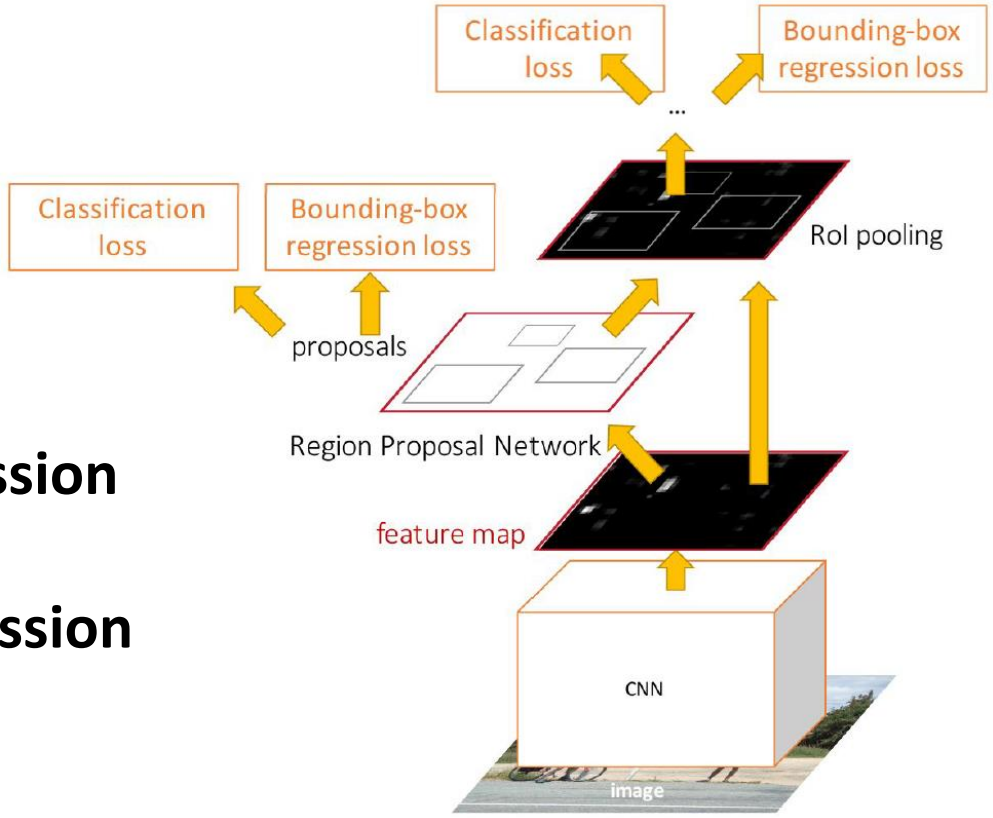


S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," NIPS 2015.

Faster R-CNN

- Integrate region proposal into the CNN

  → Region Proposal Network (RPN)



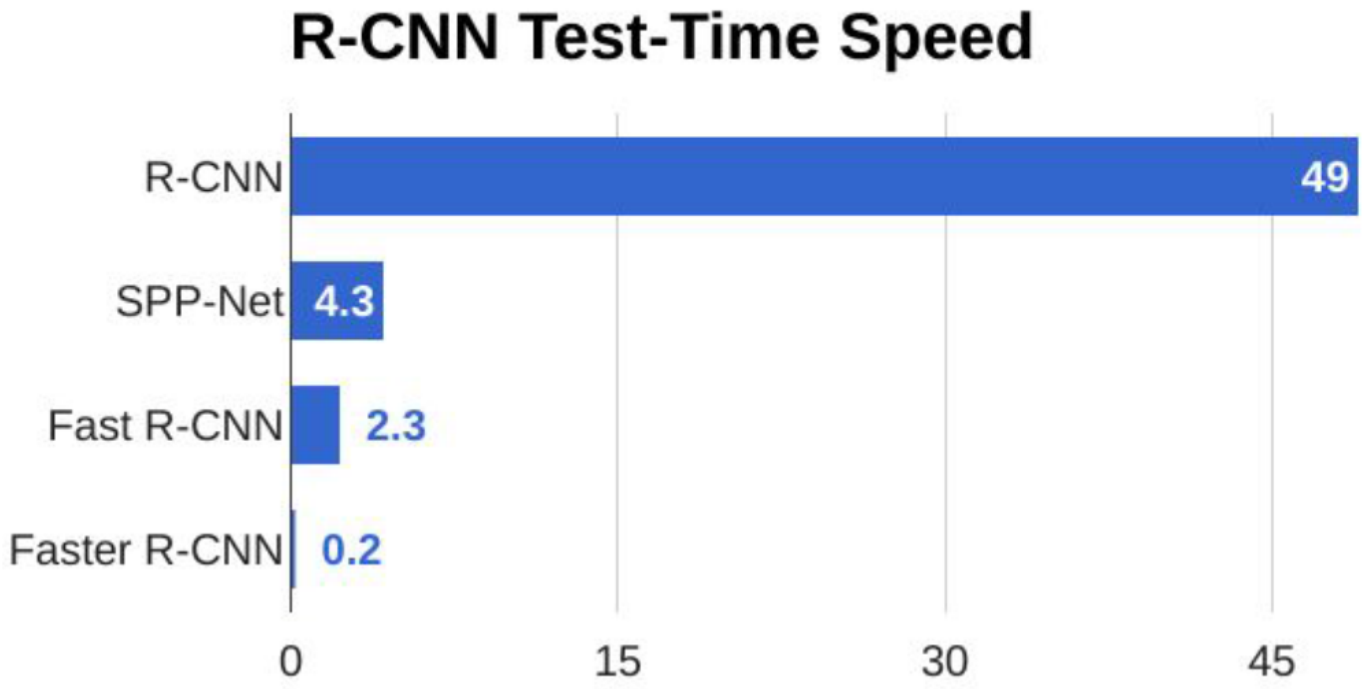
4 training losses

1. **RPN object classification**
2. **RPN box coordinate regression**
3. **Final classification**
4. **Final box coordinate regression**

S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," NIPS 2015.

Faster R-CNN

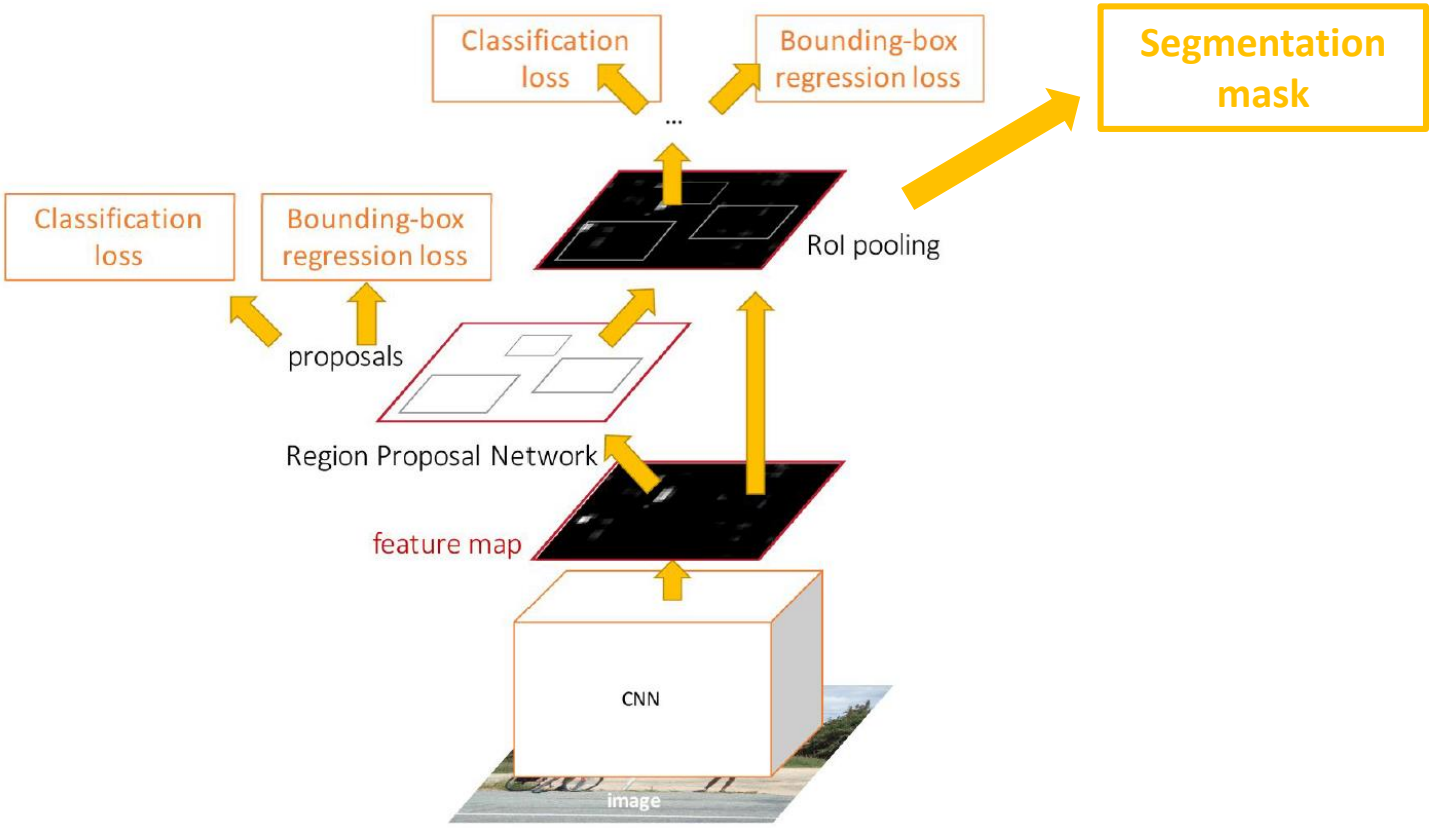
- Performance



S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," NIPS 2015.

Mask R-CNN

- Generate pixel-level masks for instance segmentation
  - ✓Uses ResNeXt as backbone network



K. He et al., "Mask R-CNN," ICCV 2017.

Mask R-CNN

- Generate pixel-level masks for instance segmentation
  - ✓Uses ResNeXt as backbone network

Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [15], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

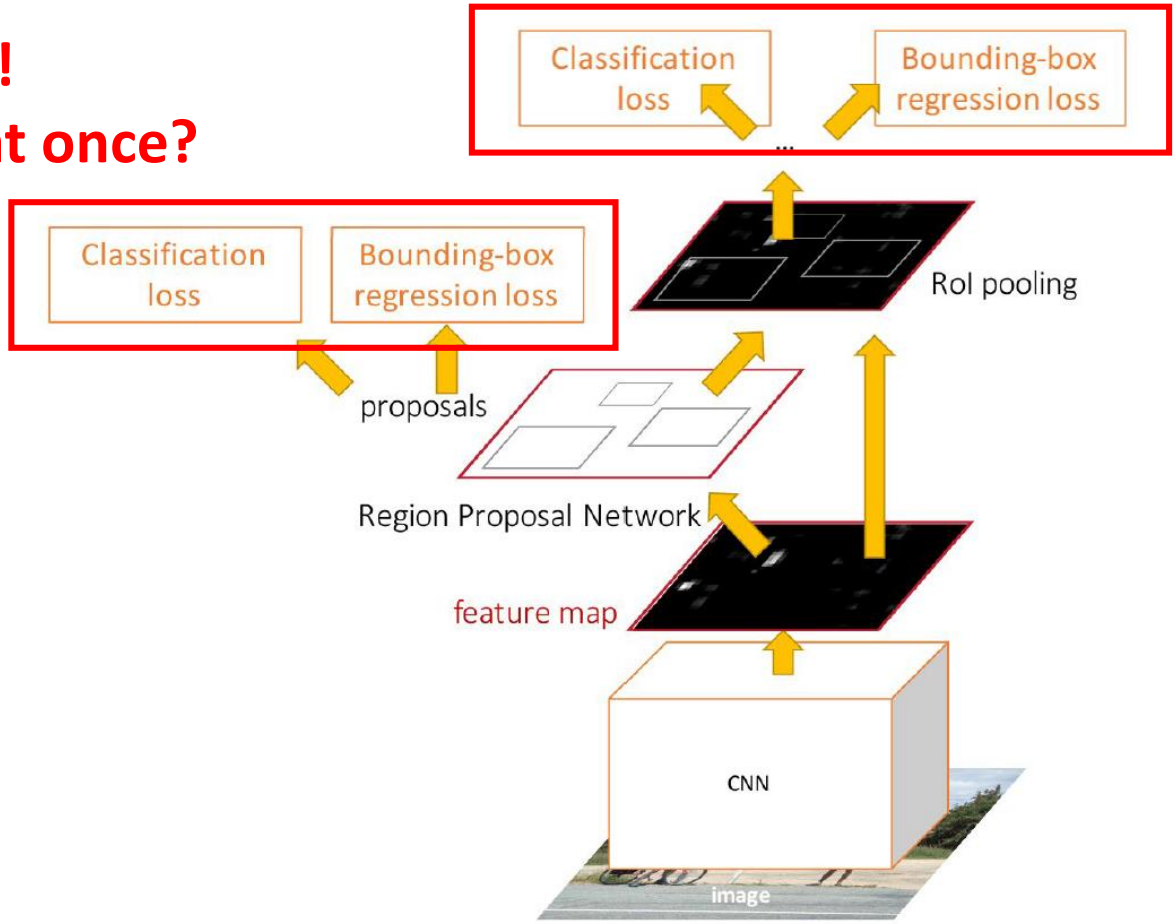
K. He et al., "Mask R-CNN," ICCV 2017.

YOLO v1

- Motivation

Redundant!

Can we just do it at once?

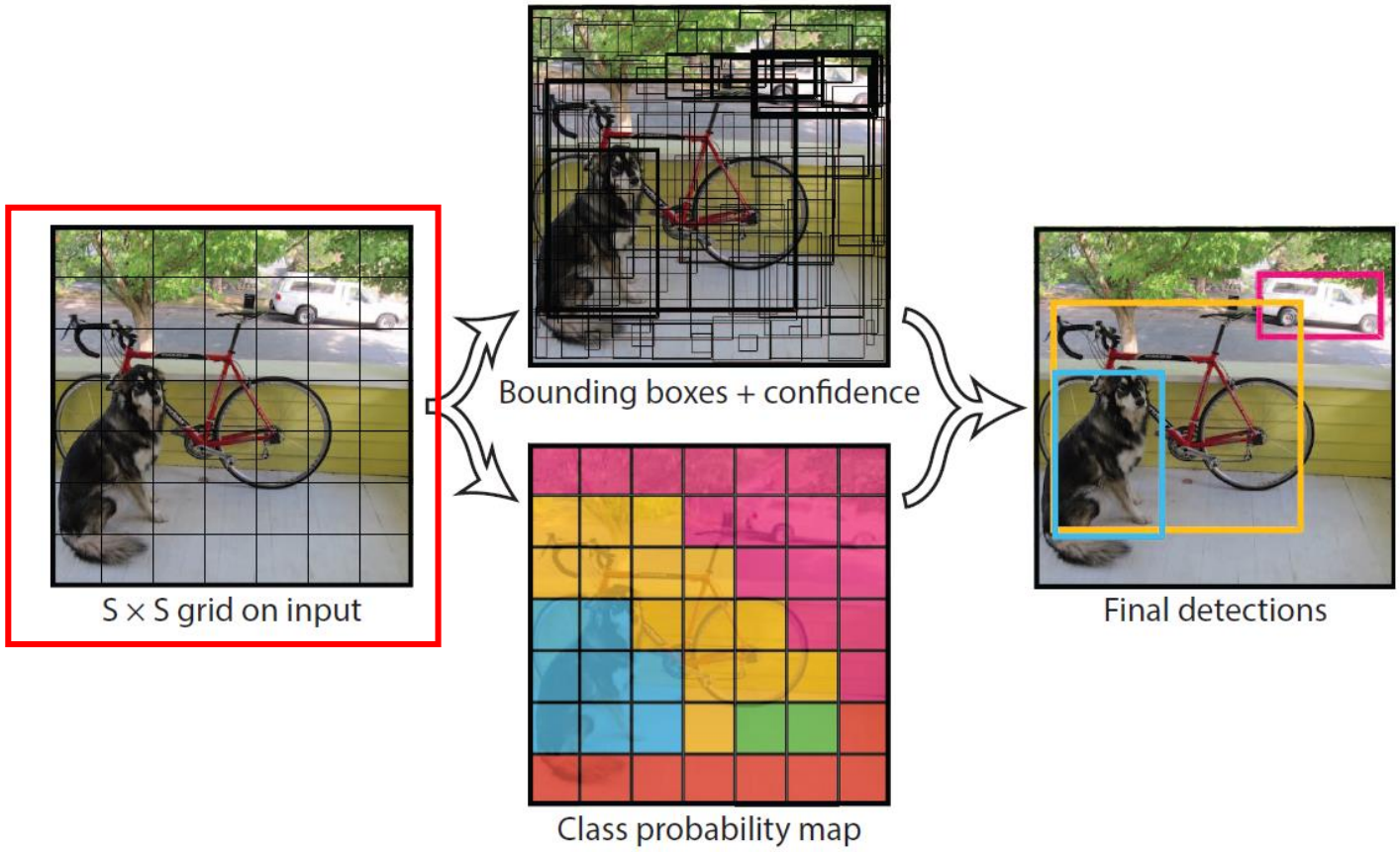


J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," CVPR 2016.

YOLO v1

Also have a look at this slide for more detailed explanations

- Divide image into $S \times S$ grids

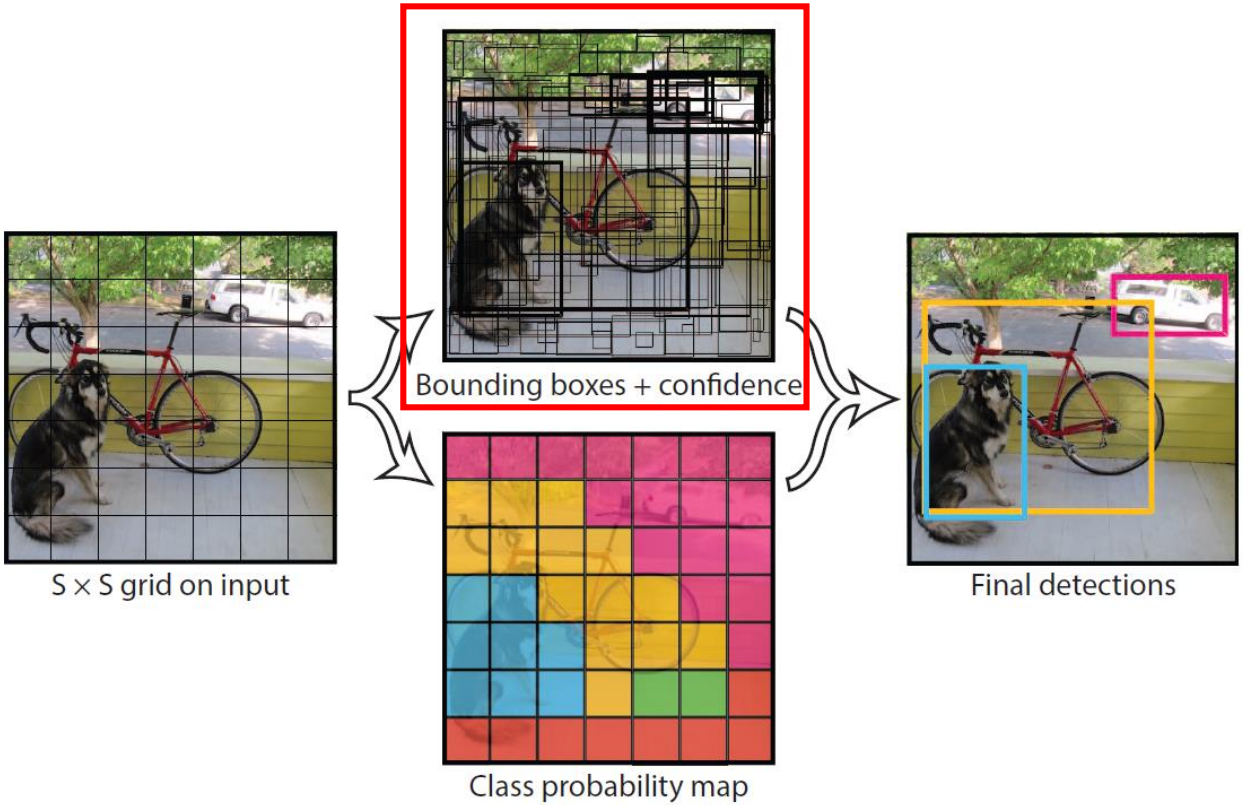


J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," CVPR 2016.

YOLO v1

Also have a look at this slide for more detailed explanations

- Each grid has
  - ✓B bounding boxes (x,y,w,h) + confidence $P(grid\ is\ an\ object)$

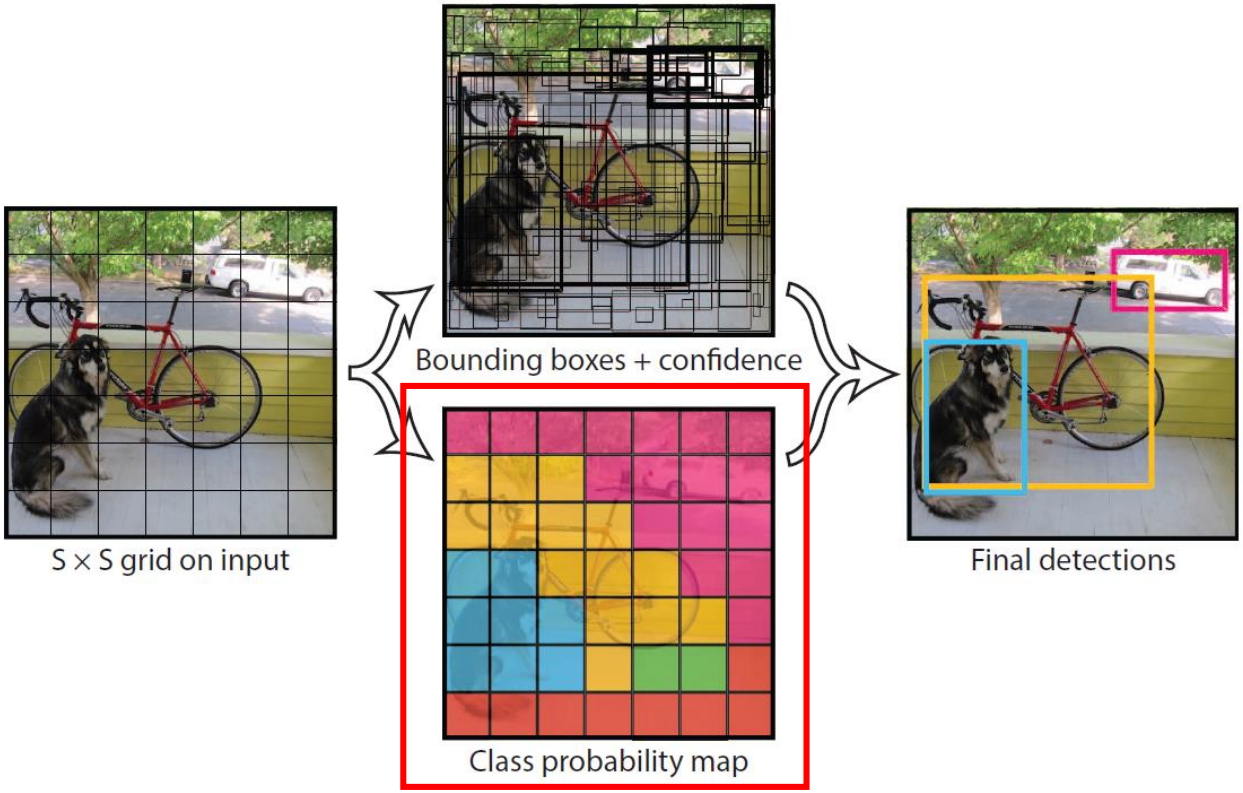


J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," CVPR 2016.

YOLO v1

Also have a look at this slide for more detailed explanations

- Each grid has
  - ✓Conditional class probability map $P(Class_i|grid\ is\ an\ object)$

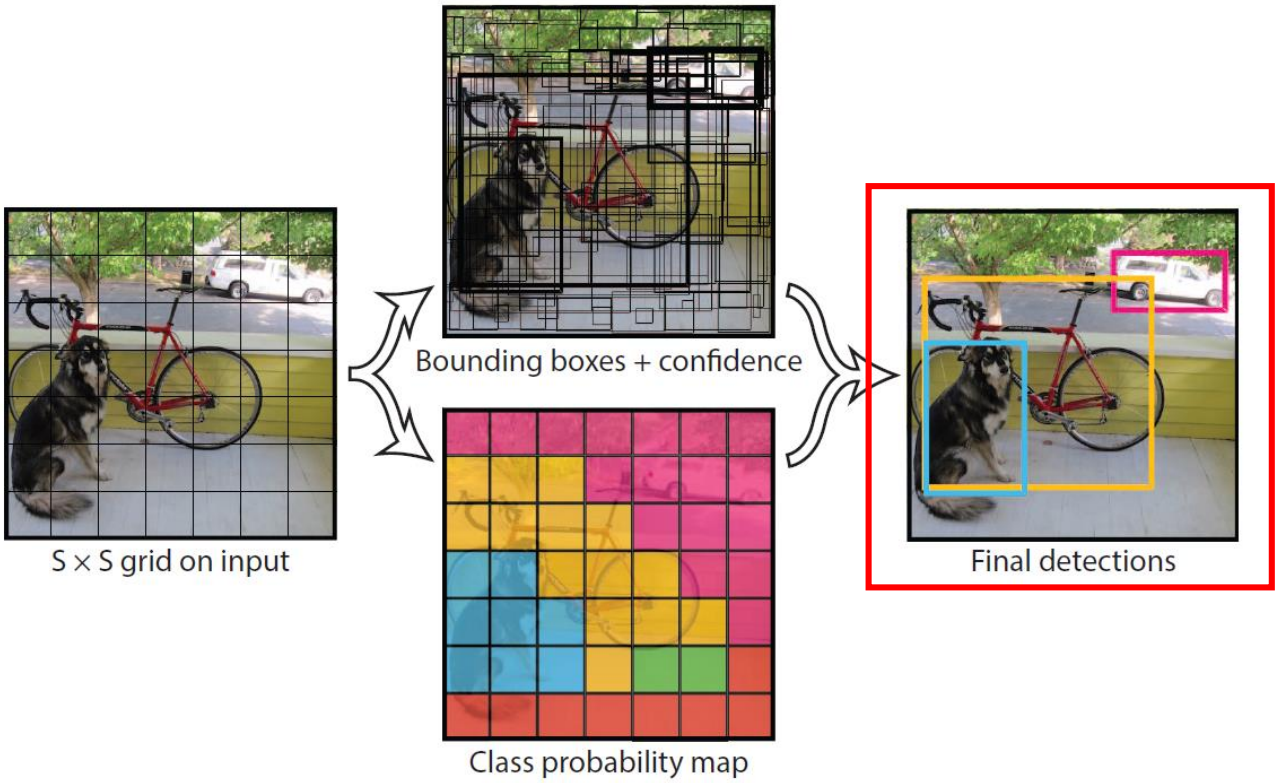


J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," CVPR 2016.

YOLO v1

Also have a look at this slide for more detailed explanations

- Multiply the two to find object class + bounding boxes
- Non-maximum suppression to remove overlapping boxes



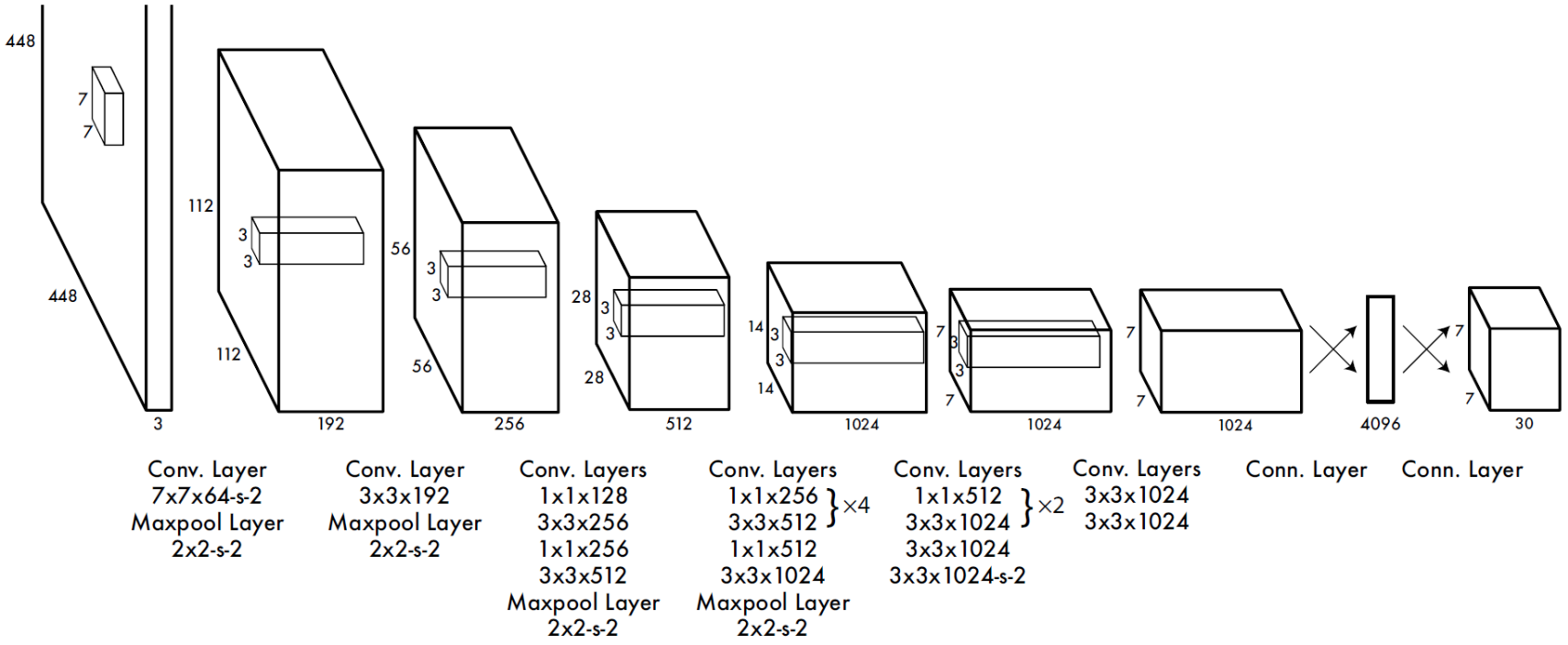
J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," CVPR 2016.

YOLO v1

Also have a look at this slide for more detailed explanations

- 7x7 grid, predict 2 boxes, 20 classes

  → output prediction becomes 7x7x(2*5+20) = 7x7x30



J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," CVPR 2016.

YOLO v1

- Performance

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

J. Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection,” CVPR 2016.

SSD

- Faster R-CNN is accurate but slow (mAP 73.2%, 7 fps)
- YOLO is fast but inaccurate (mAP 63.4%, 45 fps)
- Can we make the model both fast and accurate?

  → SSD achieves mAP 74.3%, 59 fps

W. Liu et al. “SSD: Single Shot MultiBox Detector,” ECCV 2016.

SSD

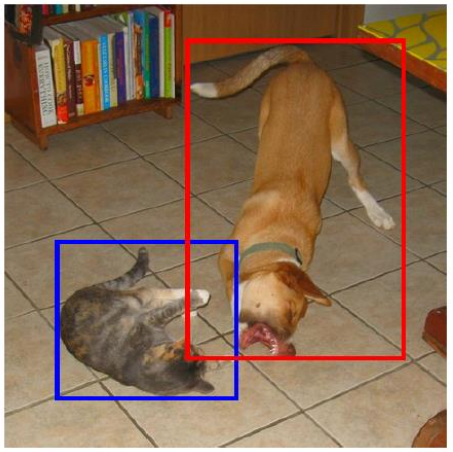
- Objects in an image have different sizes

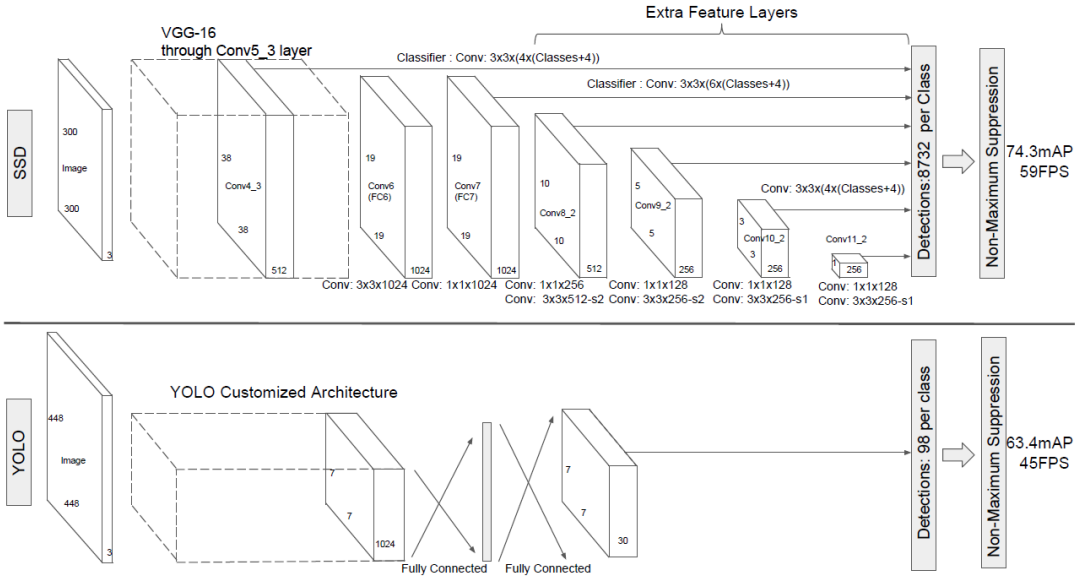
  → YOLO fails to detect various sized objects

- Multi-scale feature maps

  ✓ Latter feature maps detect large objects

  ✓ Earlier feature maps detect small objects





W. Liu et al. “SSD: Single Shot MultiBox Detector,” ECCV 2016.

SSD

- Performance

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast [6]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [2]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster [2]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [2]	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.6**	**86.6**	**88.3**	**82.4**	76.0	**66.3**	**88.6**	**88.9**	**89.1**	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	**81.2**

W. Liu et al. “SSD: Single Shot MultiBox Detector,” ECCV 2016.

YOLO v2

- Also known as YOLO9000 (classifies 9000 classes)
- 10 enhancements from YOLO

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

J. Redmon et al. "YOLO9000: Better, Faster, Stronger," CVPR 2017.

YOLO v2

- Noticeable changes
  - ✓Darknet-19 backbone
    - → Faster than VGG-16 used in SSD

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 6: Darknet-19.

J. Redmon et al. "YOLO9000: Better, Faster, Stronger," CVPR 2017.

YOLO v2

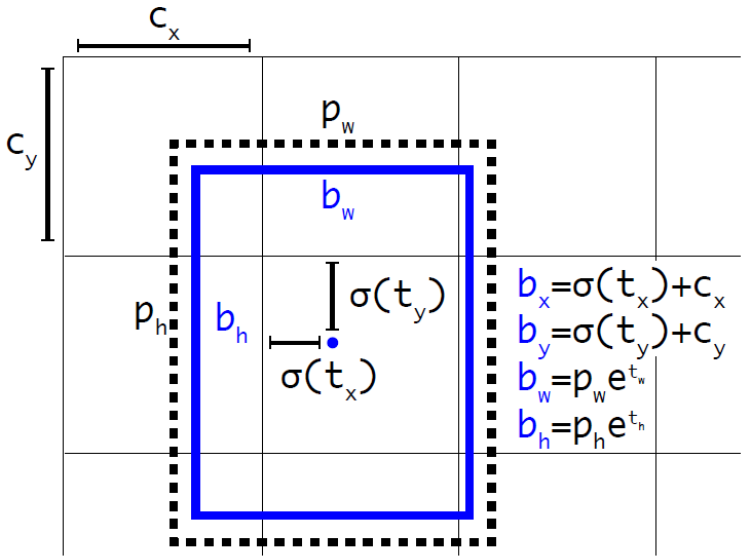
- Noticeable changes
  - ✓Fully convolutional network
    - → Can train on multiple image sizes for easy tradeoff between speed and accuracy

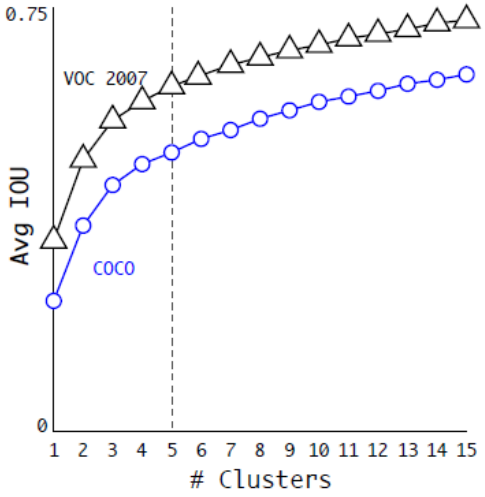
Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

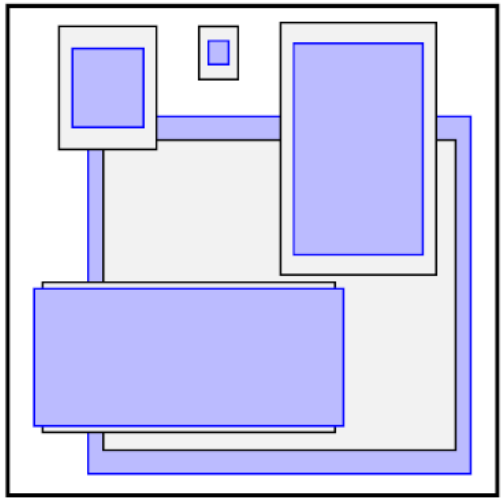
J. Redmon et al. "YOLO9000: Better, Faster, Stronger," CVPR 2017.

YOLO v2

- Noticeable changes
  - ✓YOLO v1 directly predicts bounding box coordinates (x,y,w,h)
    - → Training can be unstable
  - ✓YOLO v2 predicts offsets from anchor boxes
  - ✓Anchors are chosen from K-means clustering on COCO dataset



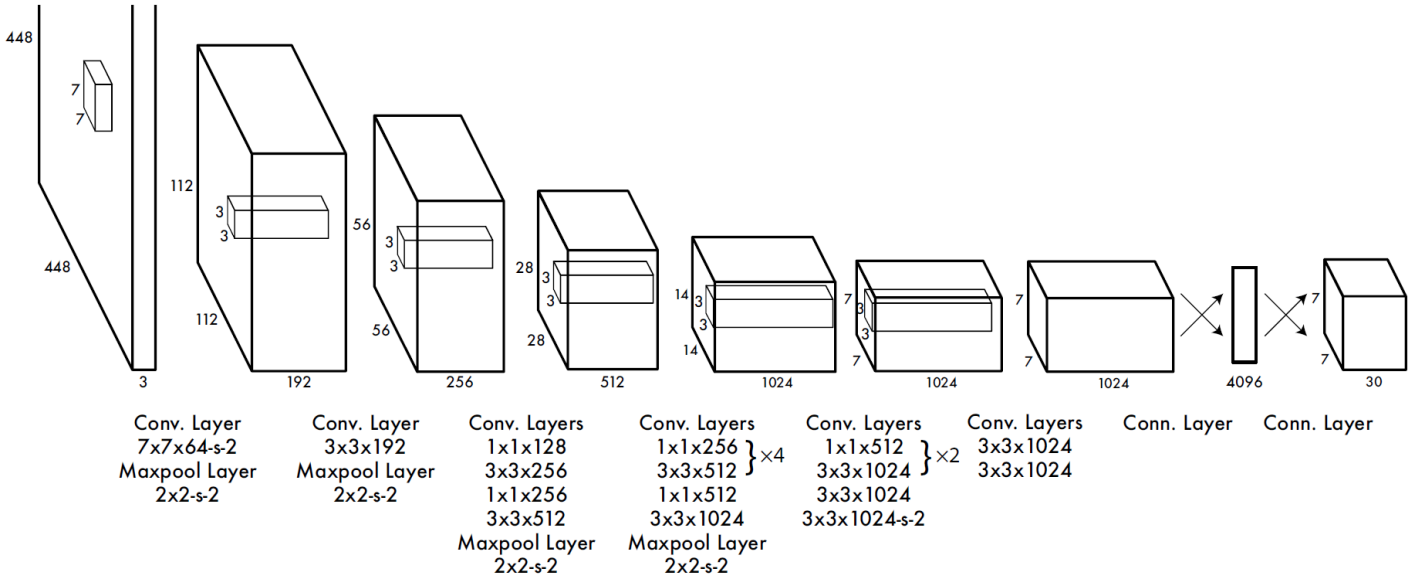




J. Redmon et al. "YOLO9000: Better, Faster, Stronger," CVPR 2017.

YOLO v2

- Noticeable changes
  - ✓YOLO v1 fails on detecting small objects



  - ✓YOLO v2 adds passthrough (i.e., adds intermediate features to final layer) to enhance accuracy

J. Redmon et al. "YOLO9000: Better, Faster, Stronger," CVPR 2017.

YOLO v2

- Performance

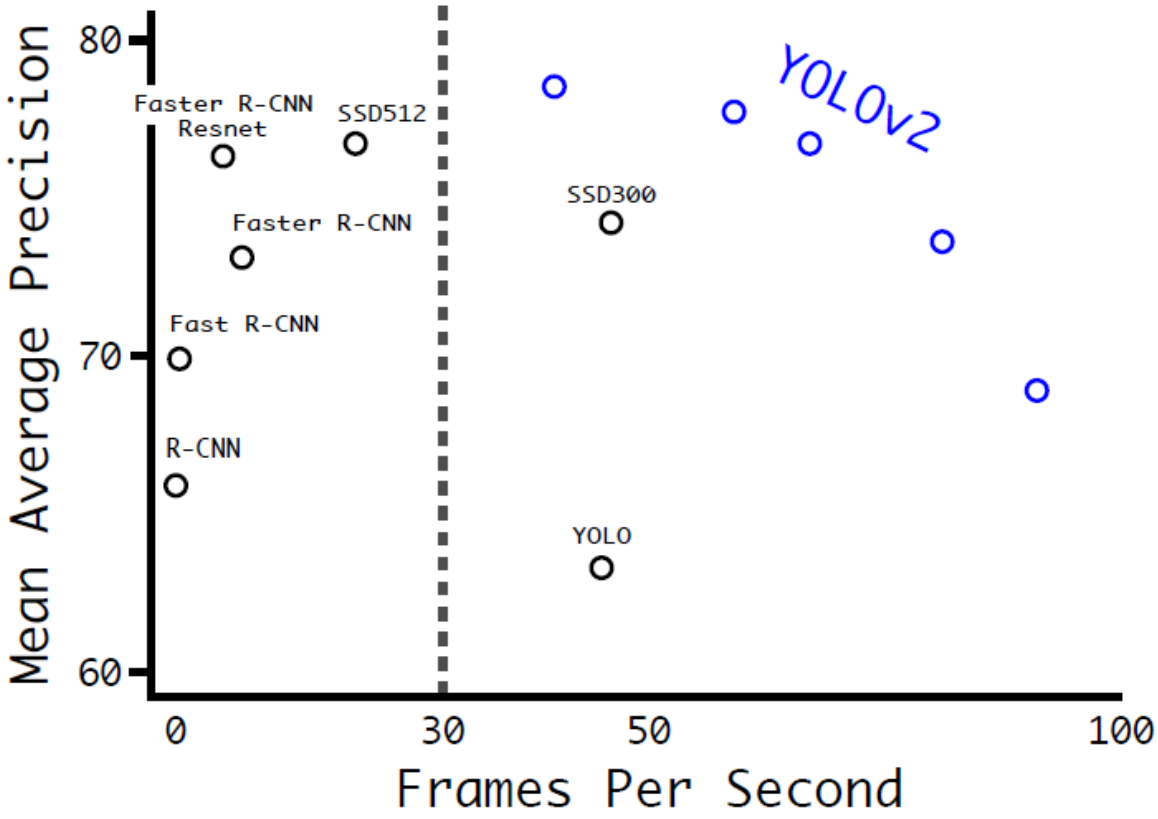
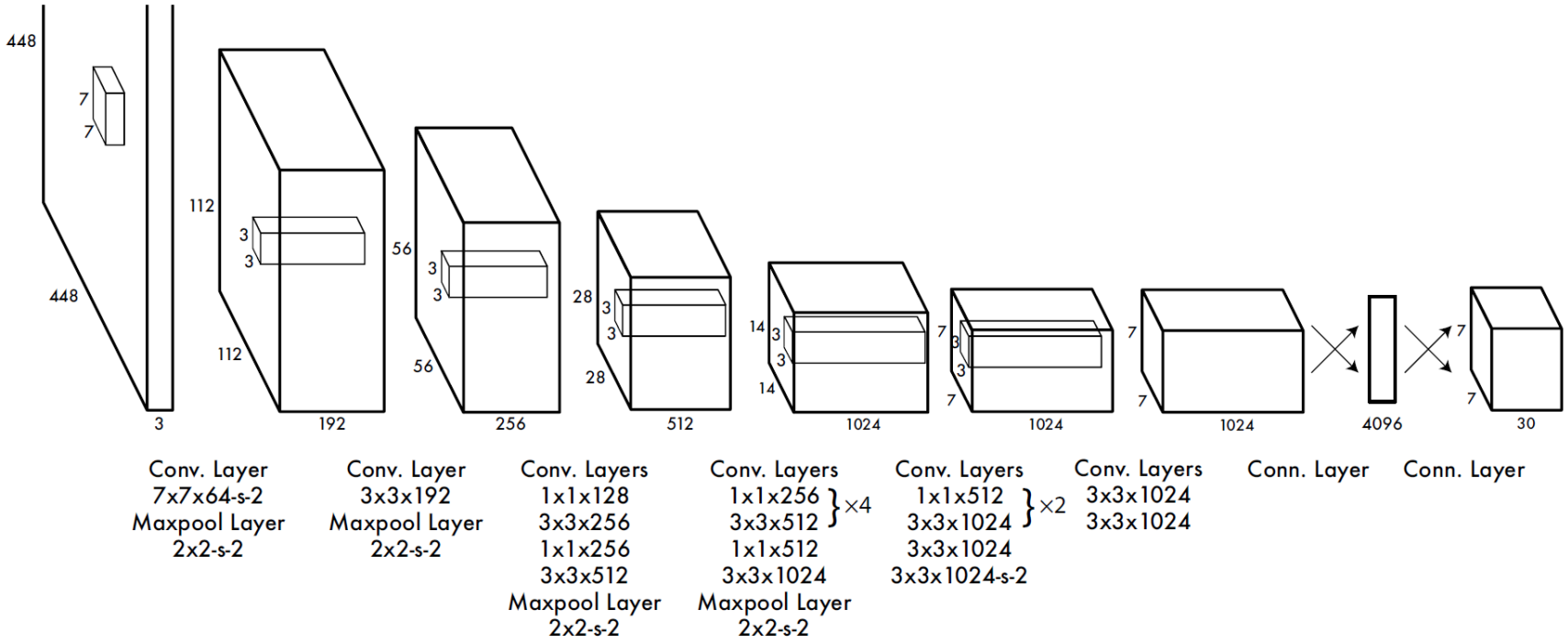


Figure 4: Accuracy and speed on VOC 2007.

J. Redmon et al. "YOLO9000: Better, Faster, Stronger," CVPR 2017.

Trend: Small Object Detection

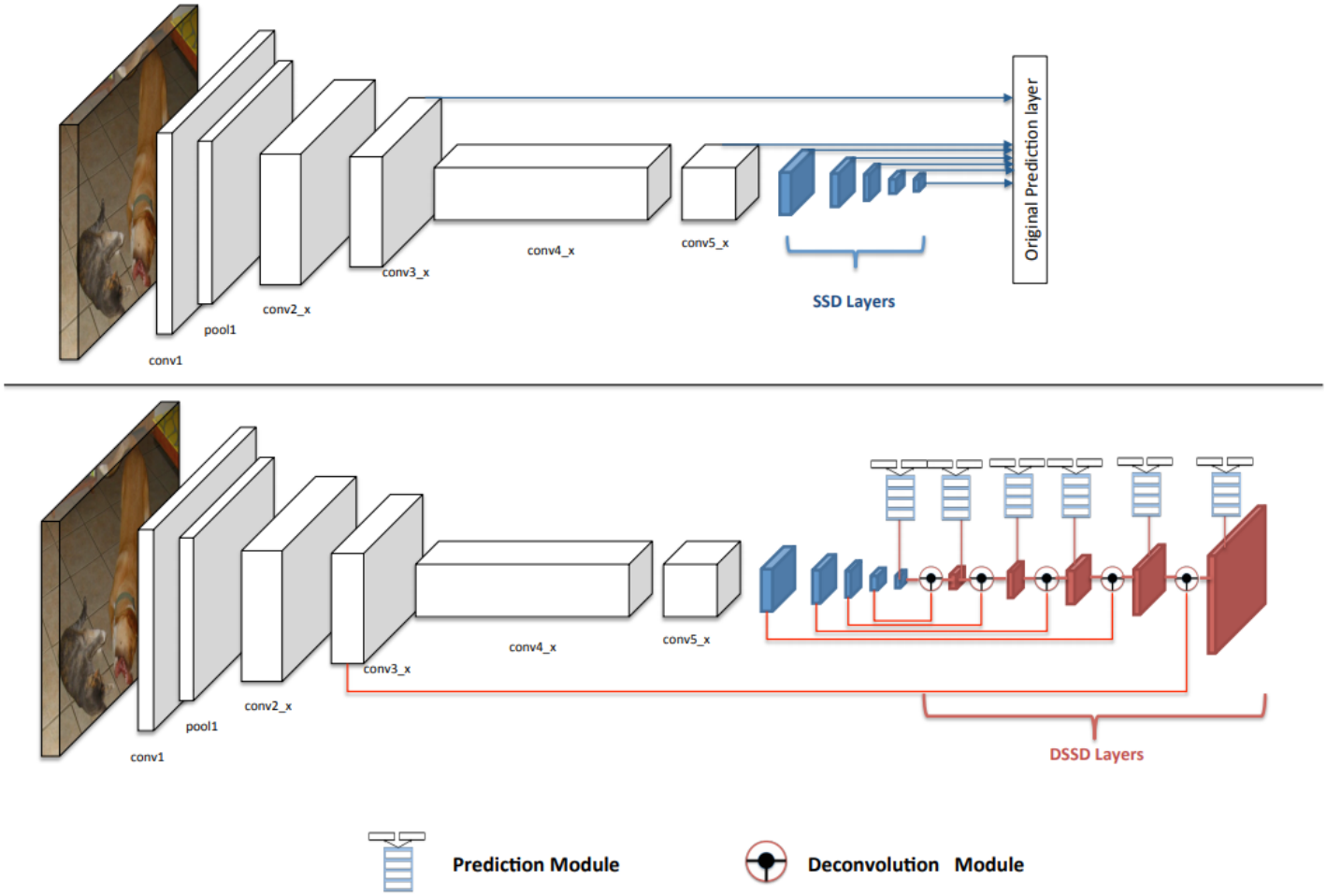
- Conventional CNNs “summarize” image through layers
- There are no features left for small objects!



J. Redmon et al. “YOLO9000: Better, Faster, Stronger,” CVPR 2017.

Trend: Small Object Detection

- Deconvolutional SSD (DSSD)

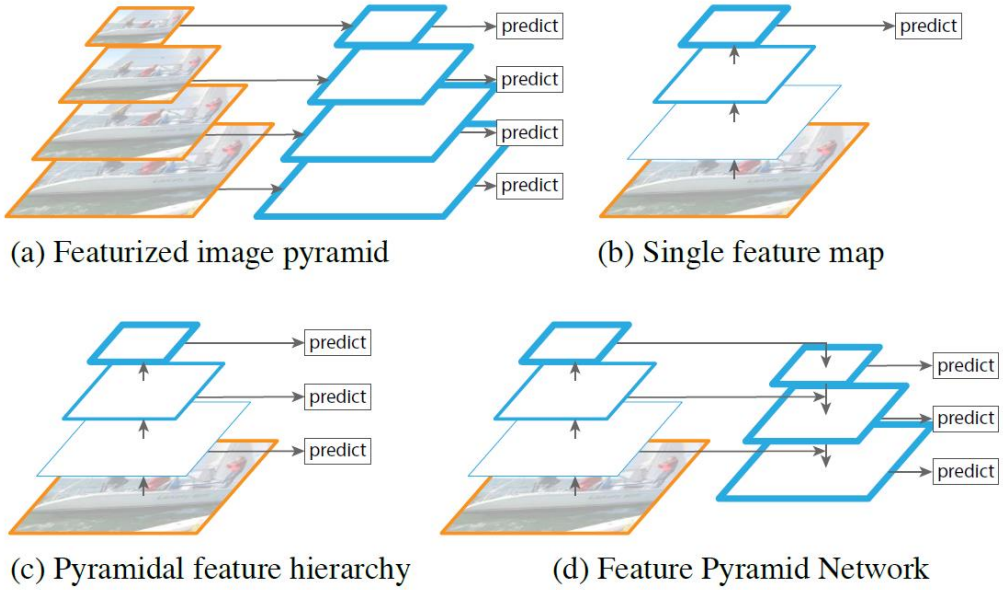


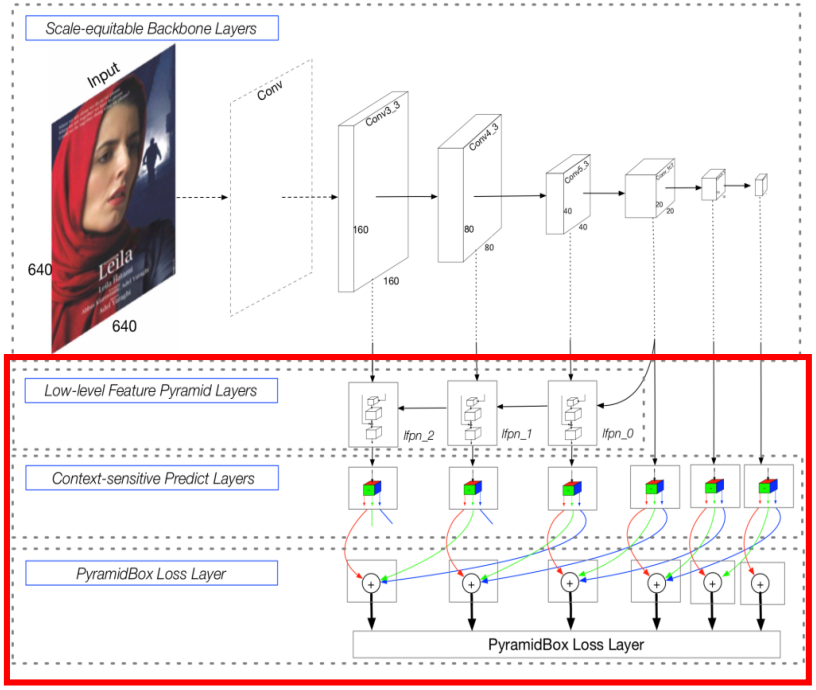
C. Fu et al. "DSSD : Deconvolutional Single Shot Detector," arXiv 2017.

Trend: Small Object Detection

- Feature pyramid network

  → De-facto standard for SOTA face detectors





T. Lin et al. "Feature Pyramid Networks for Object Detection," CVPR 2017.
X. Tang et al. "PyramidBox: A Context-assisted Single Shot Face Detector." ECCV 2018.

Thank You!