Convolutional Architectures

Tripp Deep Learning F23

UNIVERSITY OF WATERLOO

TODAY'S GOAL

By the end of the class, you should be able to explain the innovations introduced by various convolutional network architectures and their significance.

Summary

1. The earliest convolutional networks were LeNets, developed for handwritten digit recognition
2. AlexNet ignited interest in deep learning by combining convolutional networks with GPUs and big data
3. VGG networks added simplicity and depth
4. Inception networks introduced parallel paths with different kernel sizes
5. All-convolutional networks removed max-pooling and fully connected layers
6. ResNets added residual connections to facilitate training of very deep networks

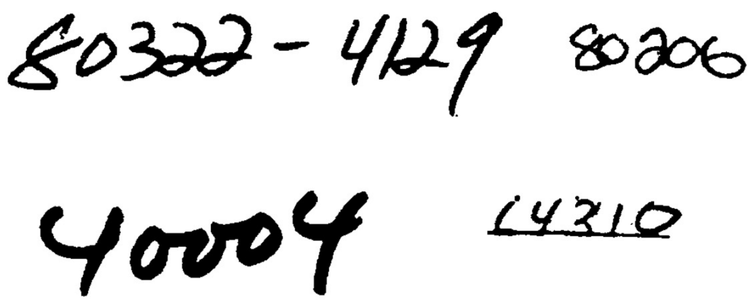
Summary

7. DenseNets included all possible skip connections
8. Squeeze & excitation networks applied a channel-wise gain that was input-dependent and learned
9. MobileNets were optimized for edge computing
10. EfficientNet used a scaling heuristic
11. U-Net combined low and high-level features to produce structured output
12. Convolutional networks continue to improve

THE EARLIEST CONVOLUTIONAL NETWORKS WERE LENETS, DEVELOPED FOR HANDWRITTEN DIGIT RECOGNITION

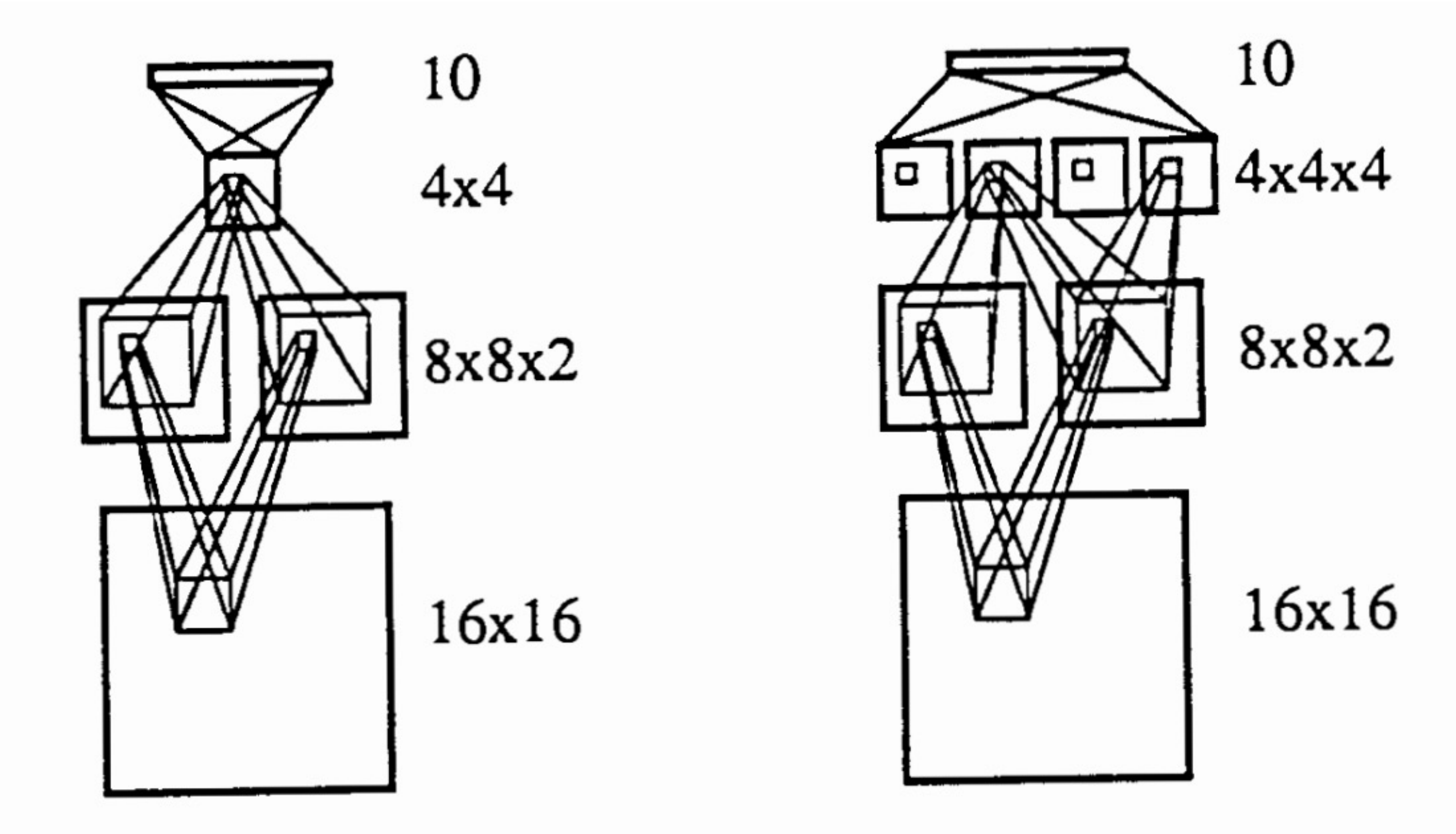
LeNet

- Named after primary creator, Yann LeCun
- Motivated by the problem of automatically reading ZIP codes (first paper preceded MNIST dataset)
- References
  - LeCun, 1989, Generalization and network design strategies. Connectionism in perspective, 19(143-155), 18.
  - LeCun et al., 1989, Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, vol. 1, no. 4, pp. 541–551,
  - LeCun et al., 1998, Gradient-based learning applied to document recognition, *Proc IEEE*



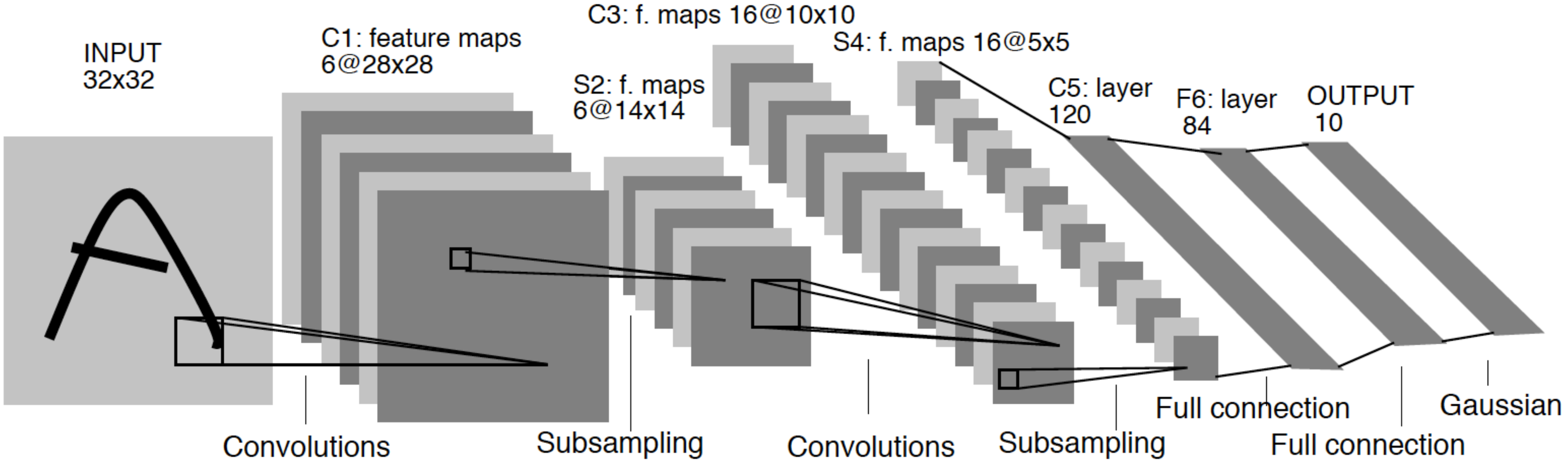
LeCun et al. (1989)

First versions of LeNet



LeCun (1989)

LeNet-5



LeCun et al. (1998)

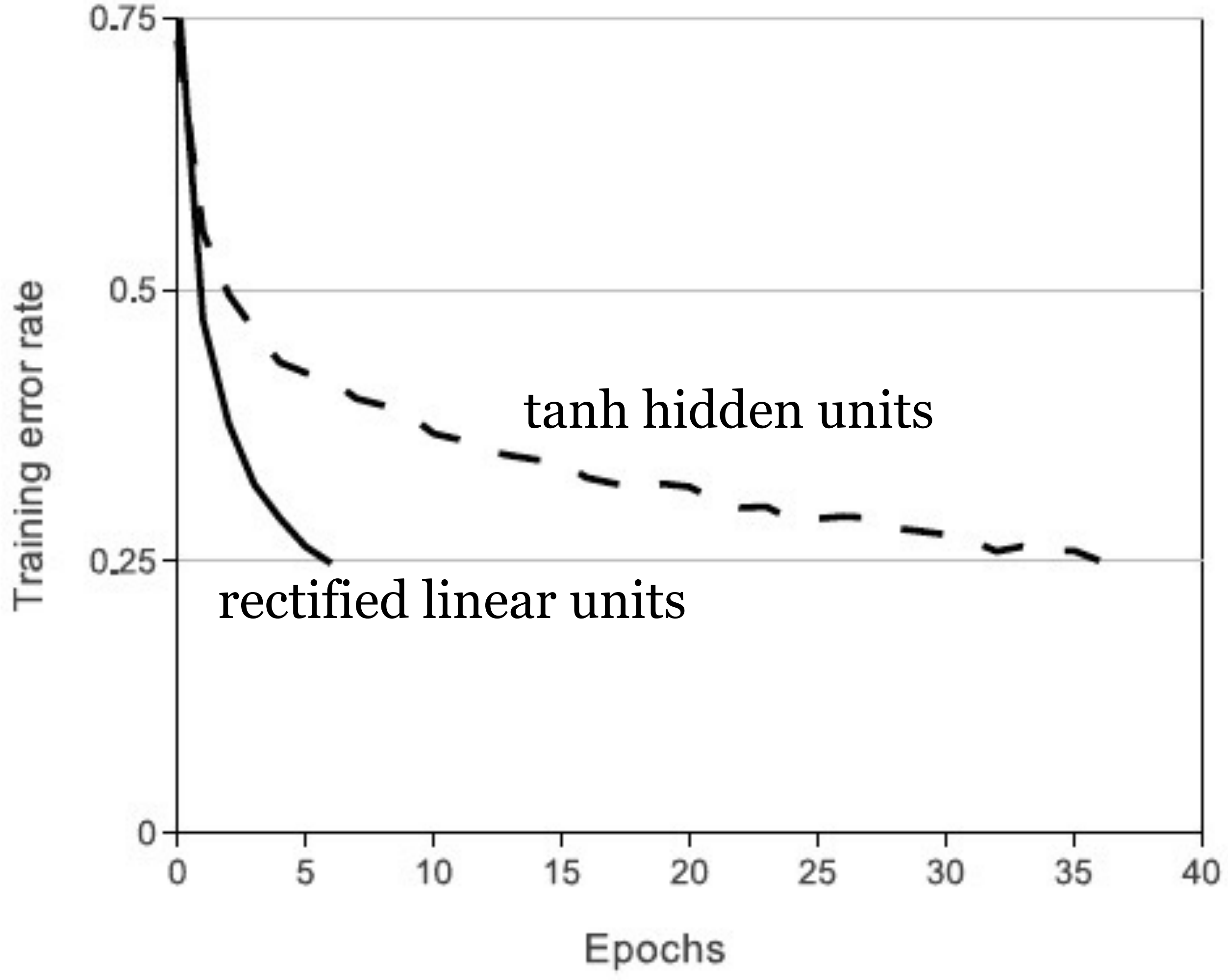
ALEXNET IGNITED INTEREST IN DEEP LEARNING BY COMBINING CONVOLUTIONAL NETWORKS WITH GPUS AND BIG DATA

AlexNet

- Named after primary creator, Alex Krizhevsky
- Reference:
  - Krizhevsky et al., 2012, ImageNet Classification with Deep Convolutional Neural Networks, NIPS
- A large convolutional network for the time (60 million parameters; 1376 kernels) trained on lots of data (1.2 million labelled images)
- By far the best performance to that point on ImageNet Large-Scale Visual Recognition Challenge (LSVRC); 1000 categories, top-5 error rate 17%
- Made possible by GPUs and Amazon's Mechanical Turk

AlexNet

- This paper also popularized rectified linear units
- Shown here is faster learning with rectified linear units in a smaller network



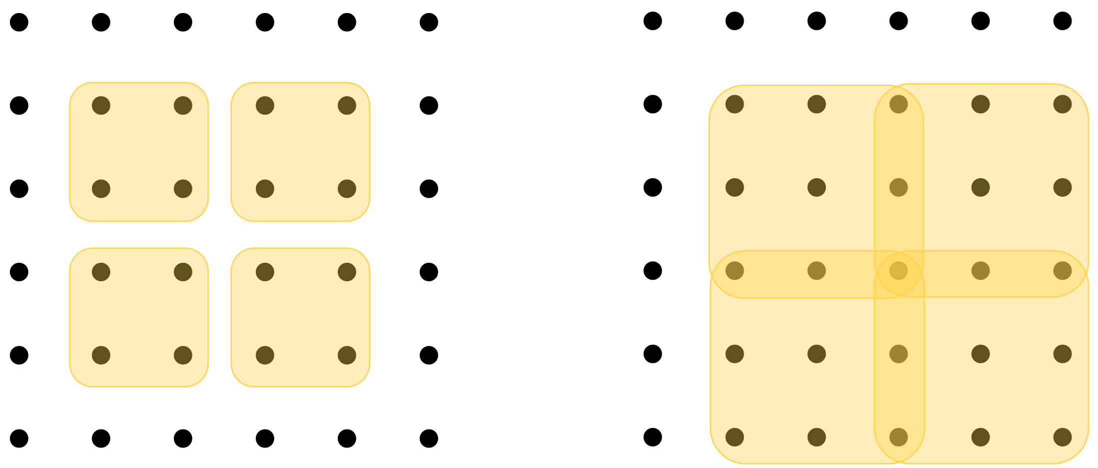
AlexNet

- Other tricks that reduce error rates
  - Multiple GPUs
  - Local response normalization
  - Overlapping pooling

non-normalized (rectified linear) output

$$b_{x,y}^{i} = \frac{a_{x,y}^{i}}{\left(k + \alpha \sum_{j=i-n/2}^{i+n/2} \left(a_{x,y}^{j}\right)^{2}\right)^{\beta}}$$

other feature maps



AlexNet

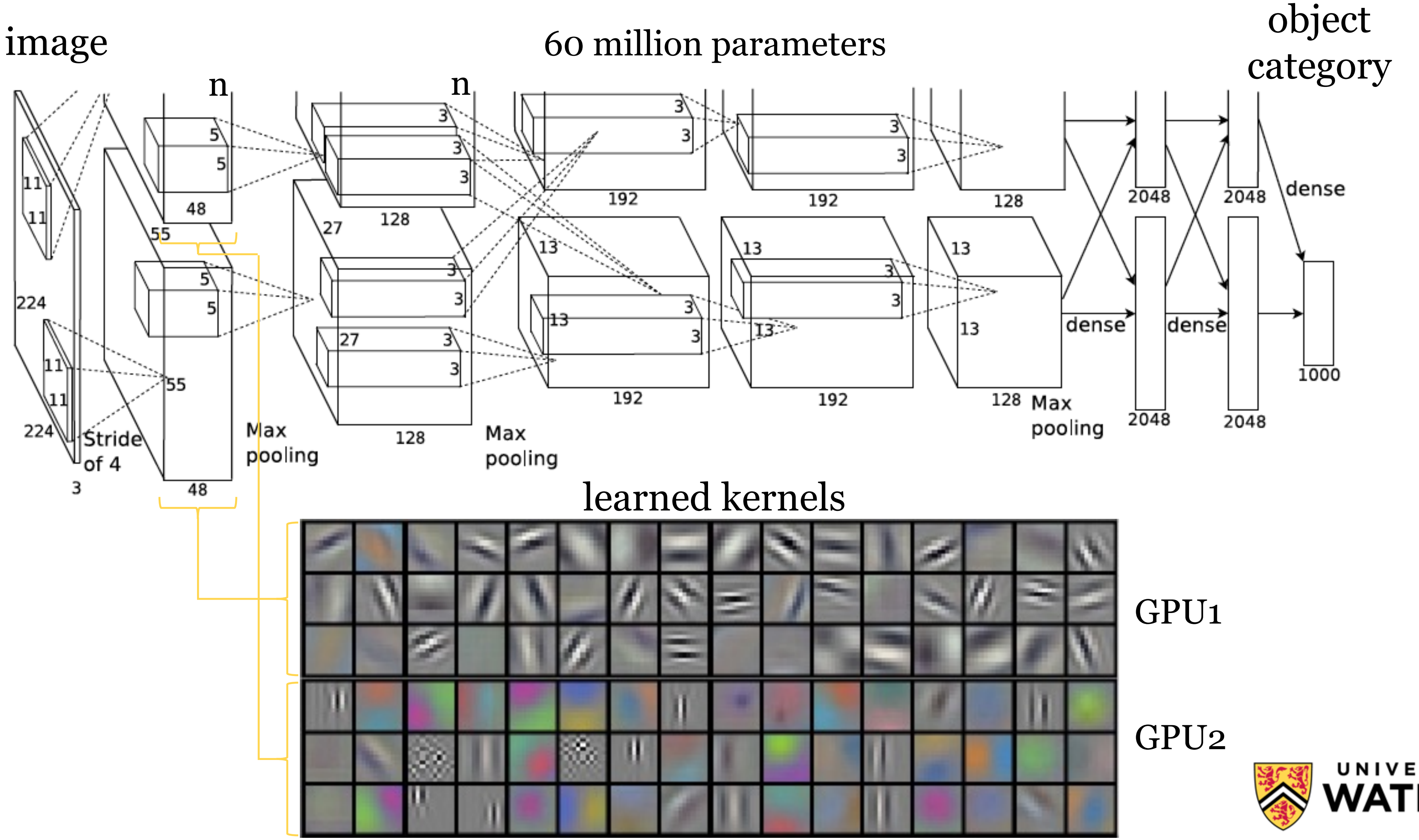
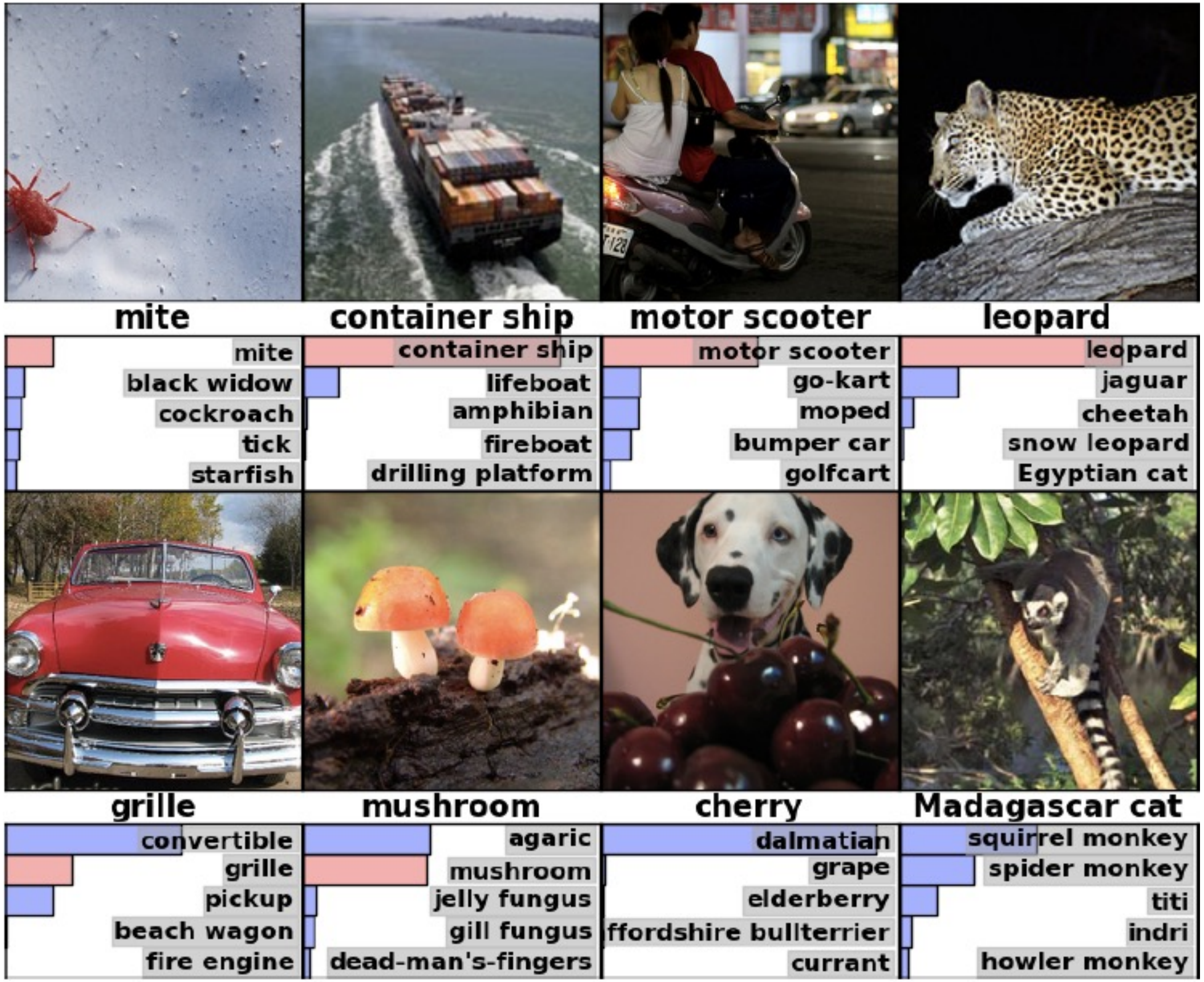


image
60 million parameters
object category
n
n
11
11
224
224
3
Stride of 4
55
55
48
48
5
5
27
27
128
128
3
Max pooling
Max pooling
13
13
192
192
128
dense
dense
dense
2048
2048
2048
2048
1000
learned kernels
GPU1
GPU2

Reducing overfitting

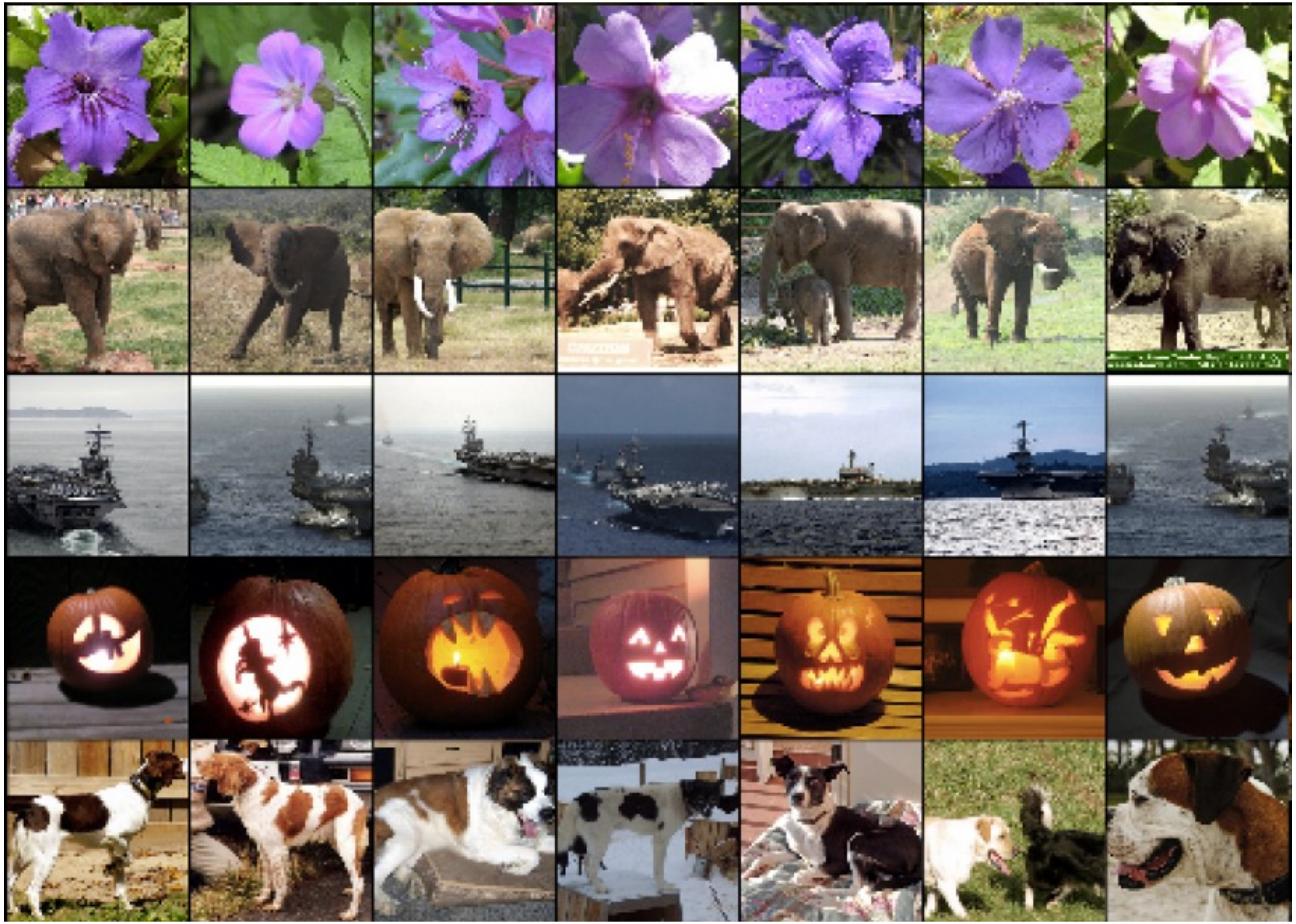
- Data augmentation
  - Offset and reflected image copies
  - Random colour offsets
- Dropout
  - Half the neurons in fully-connected hidden layers turned off randomly in each forward/backward pass



Krizhevsky et al. (2012)

UNIVERSITY OF WATERLOO

random image other images with most similar last-hidden-layer activity



Krizhevsky et al. (2012)

VGG NETWORKS ADDED SIMPLICITY AND DEPTH

VGG Networks

- Simonyan & Zisserman (2015) *ICLR*
- Particularly VGG-16 and VGG-19
- Compared to AlexNet
  - Deeper (up to 19 weight layers)
  - Smaller kernels (3x3 throughout)
  - More parameters (VGG-16 has 138M vs. 60M for AlexNet)
  - Similar otherwise (alternating convolution and max-pooling layers, increasing numbers of channels, a few fully-connected layers at the end)
- Main ideas: depth, small kernels
- Best performance to that point on ImageNet (with multiple crops, fusion of multiple network instances): Top-5 error 6.8%

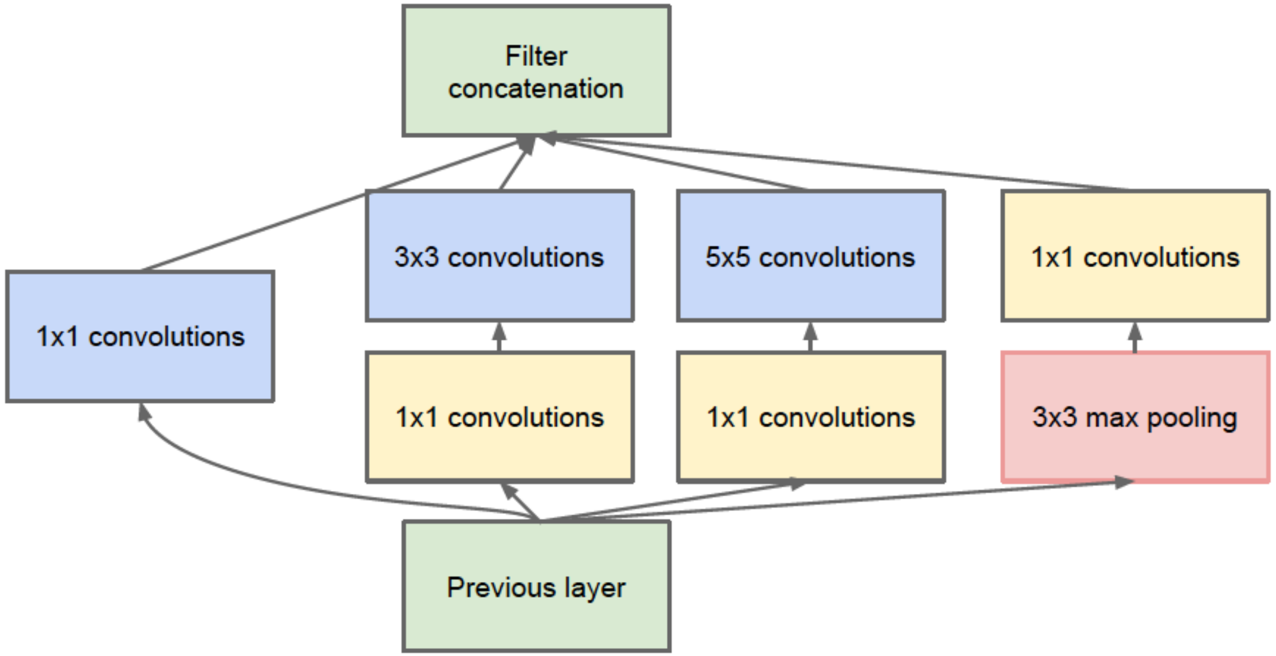
16 weight layers	19 weight layers
conv3-64 conv3-64	conv3-64 conv3-64
maxpool	
conv3-128 conv3-128	conv3-128 conv3-128
conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool	
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool	
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool	
FC-4096	
FC-4096	
FC-1000	
softmax	

Simonyan & Zisserman (2015) ICLR

INCEPTION NETWORKS INTRODUCED PARALLEL PATHS WITH DIFFERENT KERNEL SIZES

Parallel convolutions

- Multiple filter scales in parallel: 1x1, 3x3, and 5x5 convolutions
- This idea builds on an earlier "network-in-network" architecture (Lin et al., 2013), which also involved 1x1 convolutions
- Large network depth (# layers) and width (channels per layer) without lots of parameters (12x fewer than AlexNet)
- Fewer parameters because:
  - Fewer channels per path, fewer large kernels
  - Smaller fully-connected layers
- Fewer parameters reduces overfitting and computation



Szegedy et al. (2015) CVPR

1x1 convolutions

Recall that 2D convolution with multiple channels is,

$$S_{l,i,j} = \sum_{k=1}^{K} (I_k \star K_{k,l})_{i,j} = \sum_{k=1}^{K} \sum_{m=-M}^{M} \sum_{n=-N}^{N} I_{k,i+m,j+n} K_{k,l,m,n}$$

If the kernels are 1x1, this reduces to,

$$S_{l,i,j} = \sum_{k=1}^{K} I_{k,i,j} K_{k,l}$$

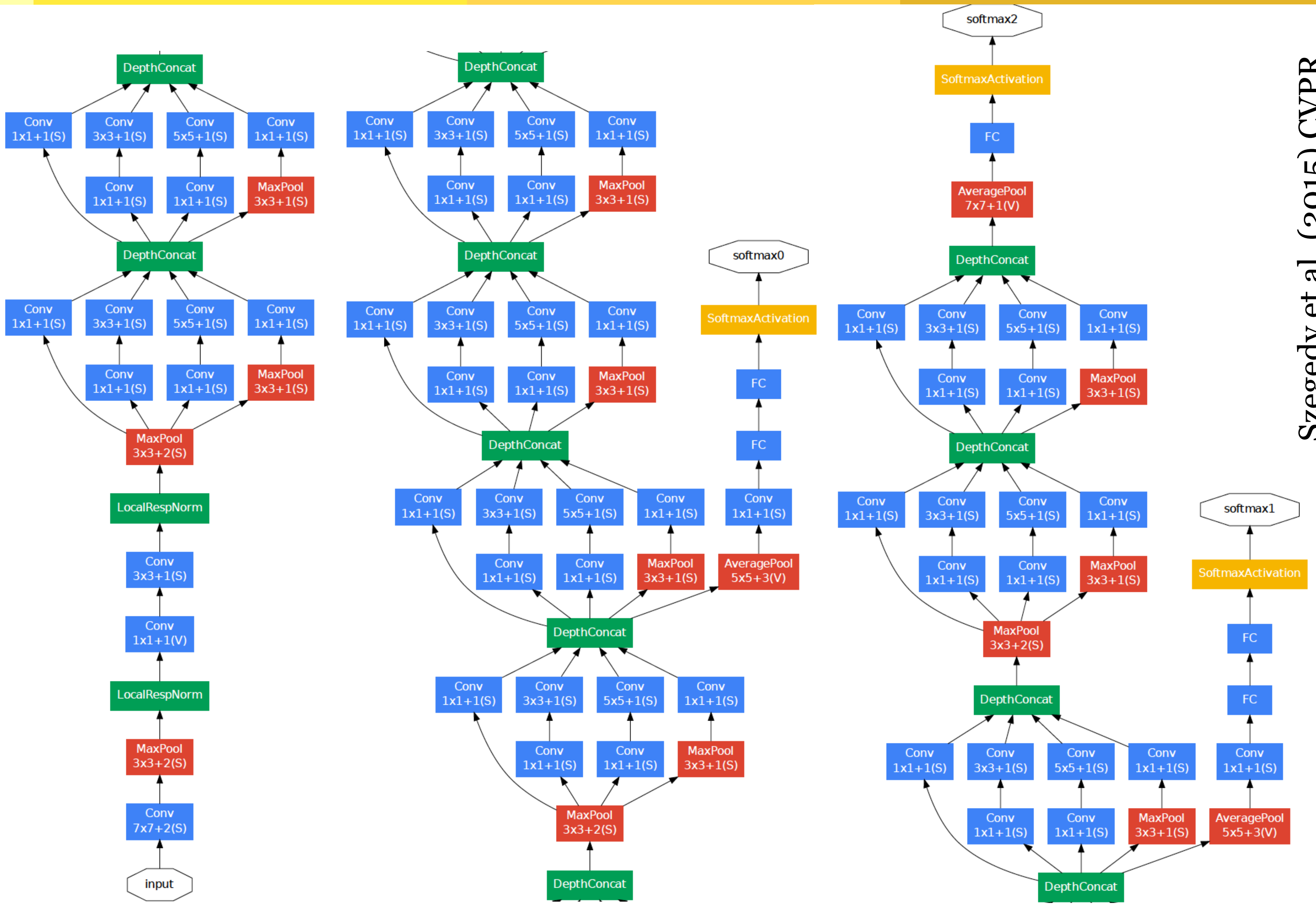
This is like operating on each multi-channel pixel of I with a fully-connected layer.
The weights of this fully-connected layer are shared by all the pixels, as usual.

Other elements

- Between Inception modules are max-pooling layers with stride 2, to reduce feature-map dimension
- Auxiliary classifiers try to classify images using mid-level features, to improve propagation of errors deep into the network (this helps performance slightly)

GoogLeNet

- An inception architecture that won the 2014 ImageNet LSVRC
- 6.67% top-5 error with ensemble of 7 networks



Szegedy et al. (2015) CVPR

Later inception networks

- Inception V2 is a slight variation introduced in the batch normalization paper (Ioffe & Szegedy, 2015)
  - 5x5 convolutions are replaced with a sequence of two 3x3 convolutions (same receptive field but fewer parameters)
- Inception V3 (Szegedy et al., 2016)
  - Builds on V2 and replaces some nxn convolutions with 1xn then nx1 convolution
  - Replaces some max pooling after Inception blocks with strided convolution within Inception blocks
- Inception V4 (Szegedy et al., 2016b)
  - Larger network that uses mostly the same ideas as V3
  - This paper also explores adding residual connections (see later slides) to Inception networks

ALL-CONVOLUTIONAL NETWORKS REMOVED MAX-POOLING AND FULLY CONNECTED LAYERS

All-CNN

- Springenberg et al. (2015) ICLR
- Eliminated some standard parts of convolutional networks
- Found that max pooling could be replaced with an additional strided convolutional layer without loss of accuracy
- Also replaced fully-connected layers with 1x1 convolutions and global average pooling
- State-of-the-art results on CIFAR-10 and CIFAR-100

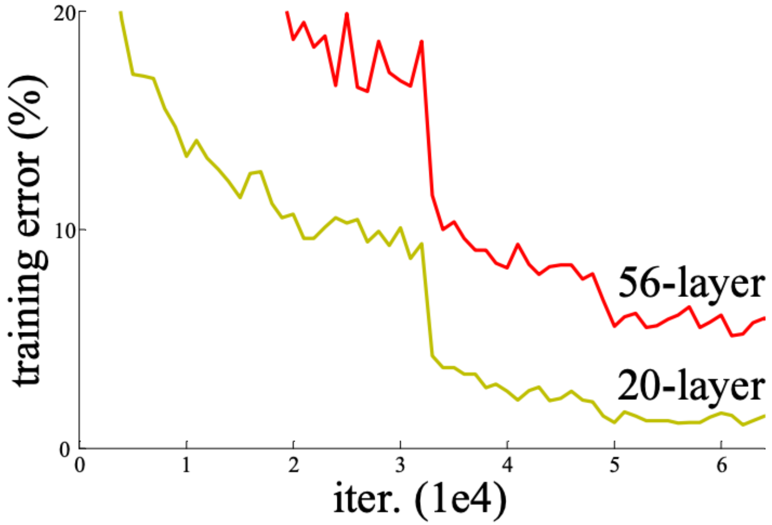
ImageNet model

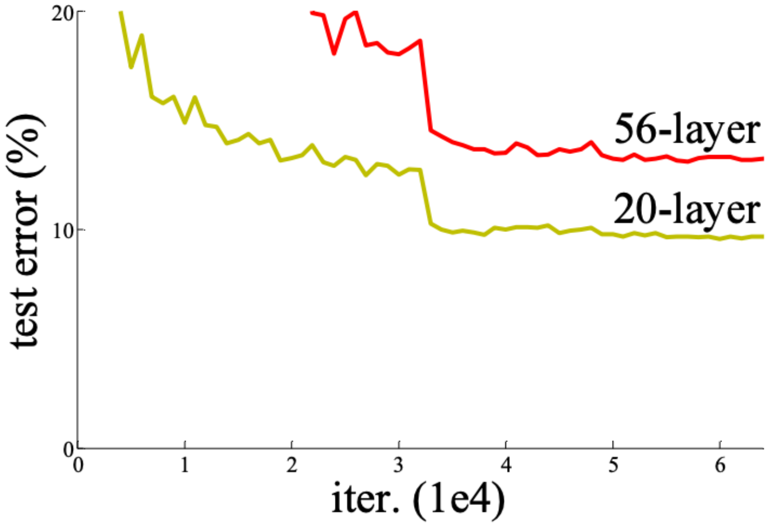
Layer name	Layer description
input	Input 224×224 RGB image
conv1	11×11 conv. 96 ReLU units, stride 4
conv2	1×1 conv. 96 ReLU, stride 1
conv3	3×3 conv. 96 ReLU, stride 2
conv4	5×5 conv. 256 ReLU, stride 1
conv5	1×1 conv. 256 ReLU, stride 1
conv6	3×3 conv. 256 ReLU, stride 2
conv7	3×3 conv. 384 ReLU, stride 1
conv8	1×1 conv. 384 ReLU, stride 1
conv9	3×3 conv. 384 ReLU, stride 2, dropout 50 %
conv10	3×3 conv. 1024 ReLU, stride 1
conv11	1×1 conv. 1024 ReLU, stride 1
conv12	1×1 conv. 1000 ReLU, stride 1
global_pool	global average pooling (6×6)
softmax	1000-way softmax

RESNETS ADDED RESIDUAL CONNECTIONS TO FACILITATE TRAINING OF VERY DEEP NETWORKS

Motivation

- It had previously been hard to train very deep networks
- It was possible at this point given Kaiming initialization and batch norm, but it was found that performance *on training data* degraded with large depth
- This is an issue of optimization; a deeper network is in principle capable of same performance by performing an identity mapping in the extra layers

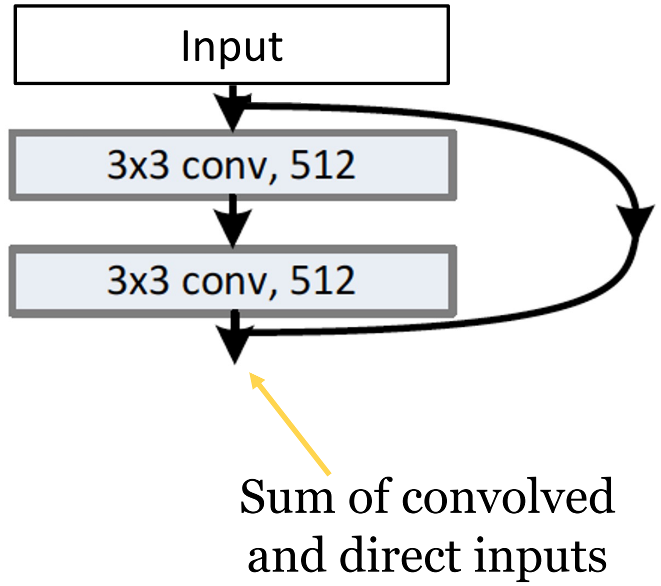




He et al., 2016, CVPR

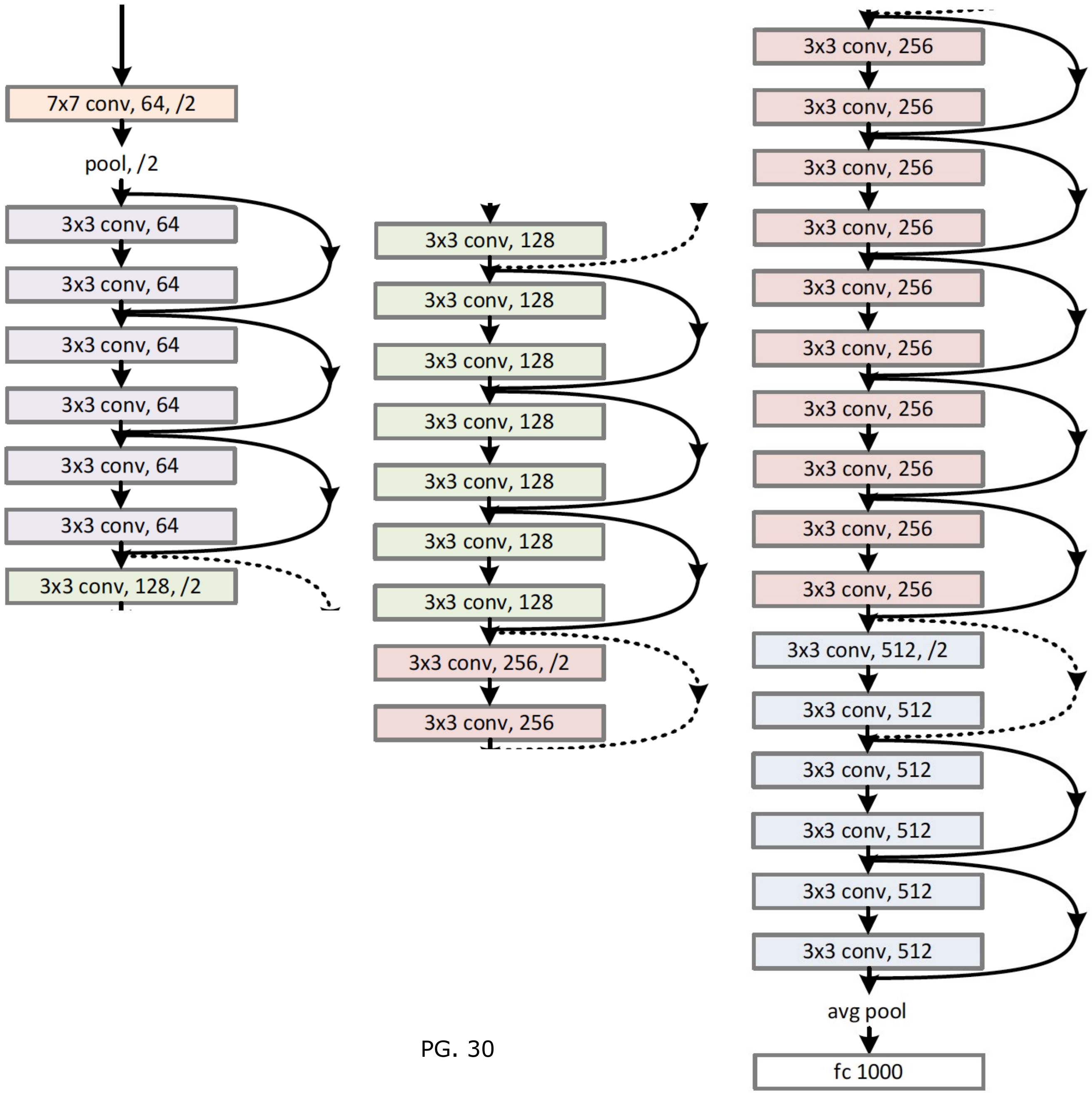
ResNet

- The main idea is to incorporate “skip connections” in parallel with blocks of convolutional layers, which pass on the previous layer’s outputs unchanged
- If the ideal mapping in a part of the network is $f(x)$, the network must then learn $f(x) - x$, which may be easier
- The skip connections do not have parameters, so no increase in total parameters
- Ensemble of six networks including two very deep networks (152 layers) had 3.57% top-5 error on ImageNet LSVRC
- They also trained a 1202-layer network on CIFAR-10, and the training error was very low showing that optimization worked (although it overfit)



ResNet

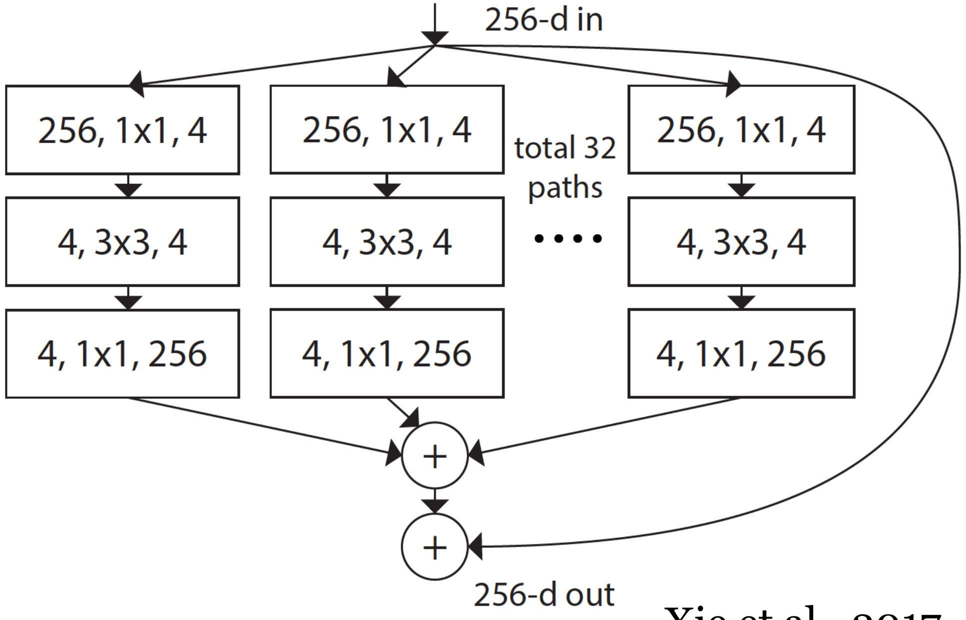
- Here is a relatively small 34-layer ResNet



He et al. (2016) CVPR

Later variations

- Zagoruyko & Komodakis (2016, *arXiv*) showed better performance with shallower but wider networks (more channels), leading to Wide ResNets
- Xie et al. (2017, *CVPR*) modified ResNets to have parallel groups of convolutional layers (like Inception networks but with more parallel paths all with the same kernel sizes, and summed) leading to ResNeXt networks
- Both approaches showed improved performance for a given number of parameters

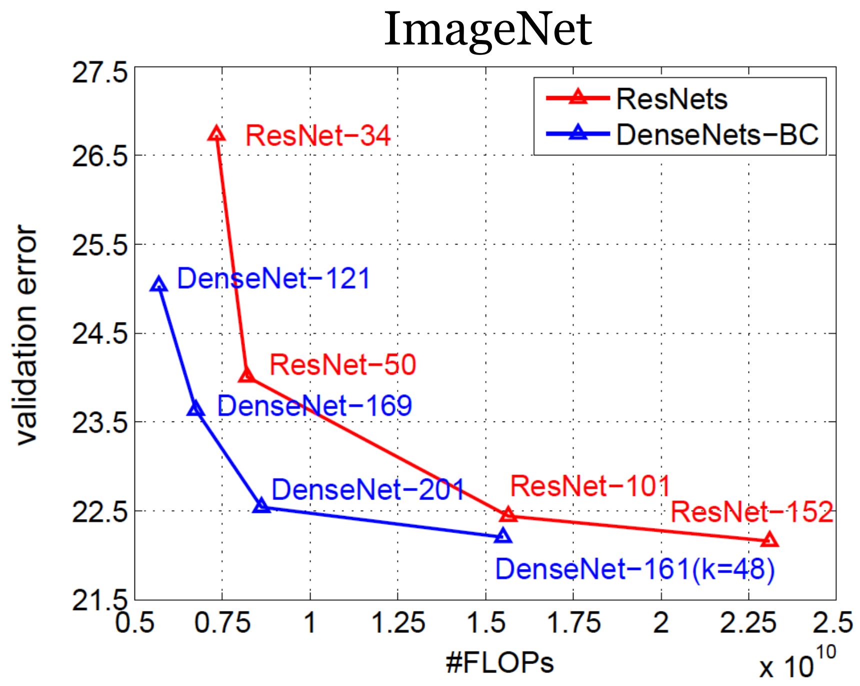


Xie et al., 2017

DENSENETS INCLUDED ALL POSSIBLE SKIP CONNECTIONS

DenseNet

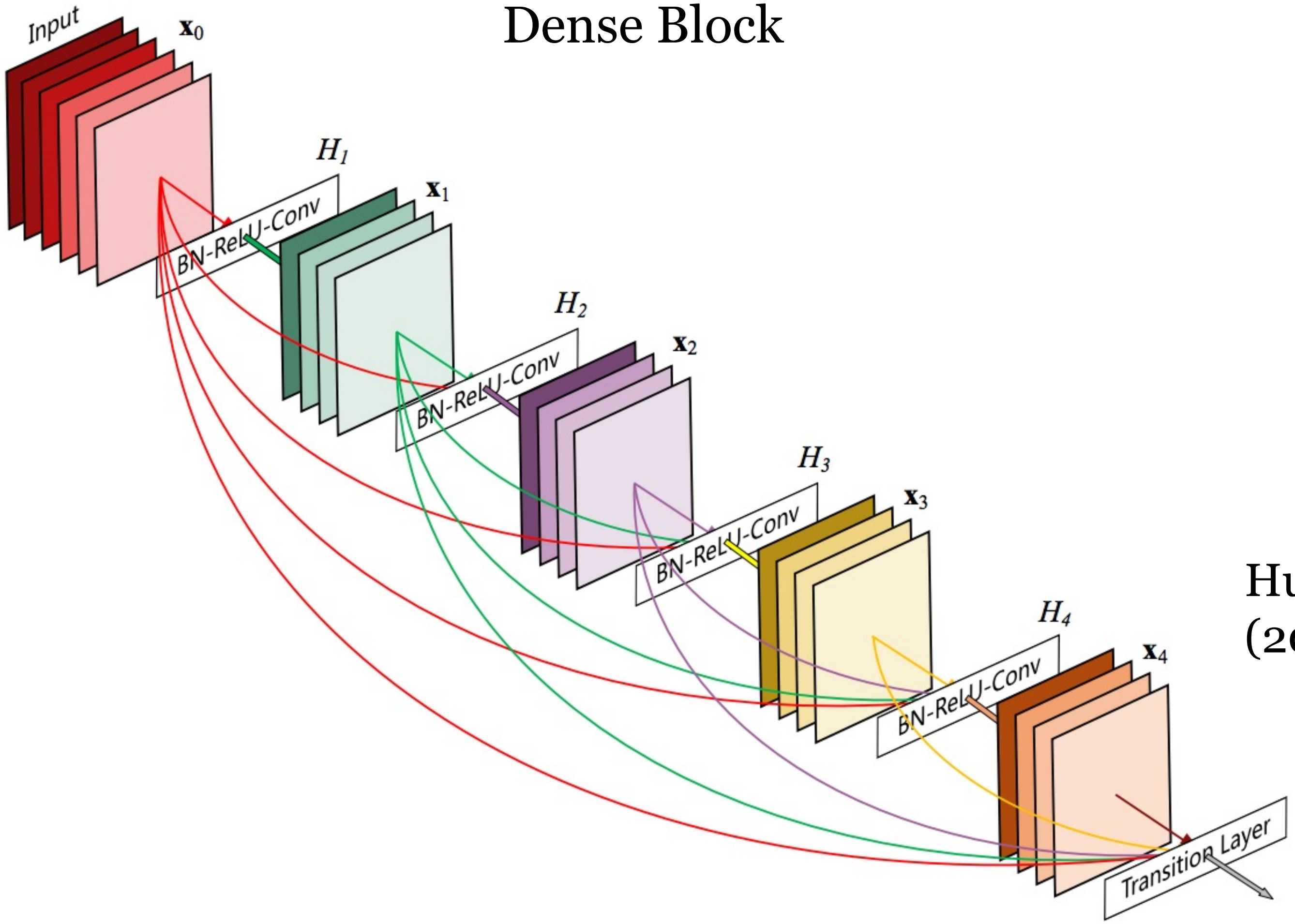
- Huang et al. (2017) *CVPR*
- Prior networks, particularly ResNets, had benefitted from skip connections (shortcuts through the network)
- DenseNet carried this idea to an extreme:
  - Each layer gets input from all previous layers
  - Each of these paths is combined by concatenation (rather than summation)
- Similar performance to ResNets with fewer parameters and floating-point operations



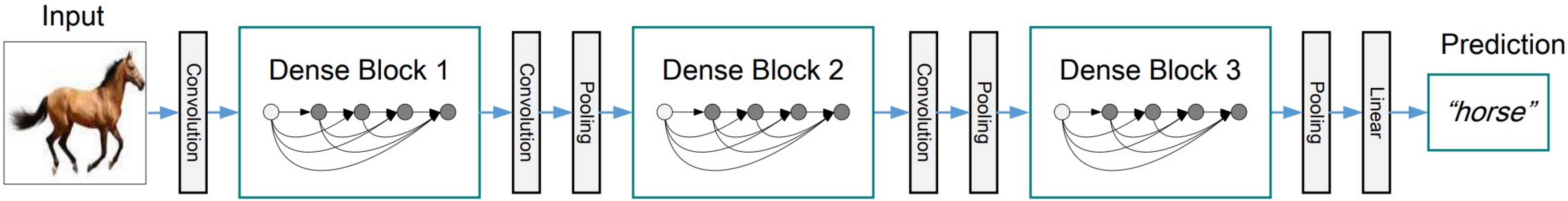
UNIVERSITY OF WATERLOO

DenseNet

- Each layer x_l in a block gets input from all previous layers, $x_l = H_l([x_1, x_2, \dots, x_{l-1}])$
- Chaining dense blocks together allows stride>1 at intermediate "transition layers"



Huang et al. (2017) *CVPR*



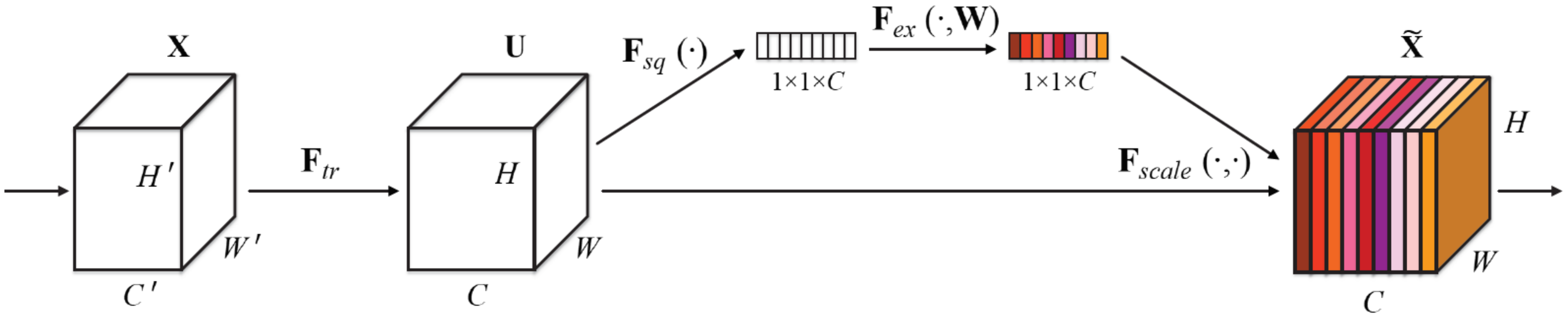
DenseNet

- Layer l gets $k_0 + k(l-1)$ channels of input, where k_0 is the # of channels of input to the block and k is the number of new filters in each layer
- There are not many parameters overall:
  - Layers have fewer filters than other convolutional networks, e.g., 12
  - This probably works because the network doesn't need redundant representations of lower-level feature maps in order to propagate their features forward
- Later layers in a block have many input channels, so in some versions:
  - They use a 1x1 bottleneck convolution to map a layer's inputs to $4k$ channels
  - They reduce the number of channels at transition layers (between dense blocks)

SQUEEZE & EXCITATION NETWORKS APPLIED A CHANNEL-WISE GAIN THAT WAS INPUT-DEPENDENT AND LEARNED

Squeeze and excitation

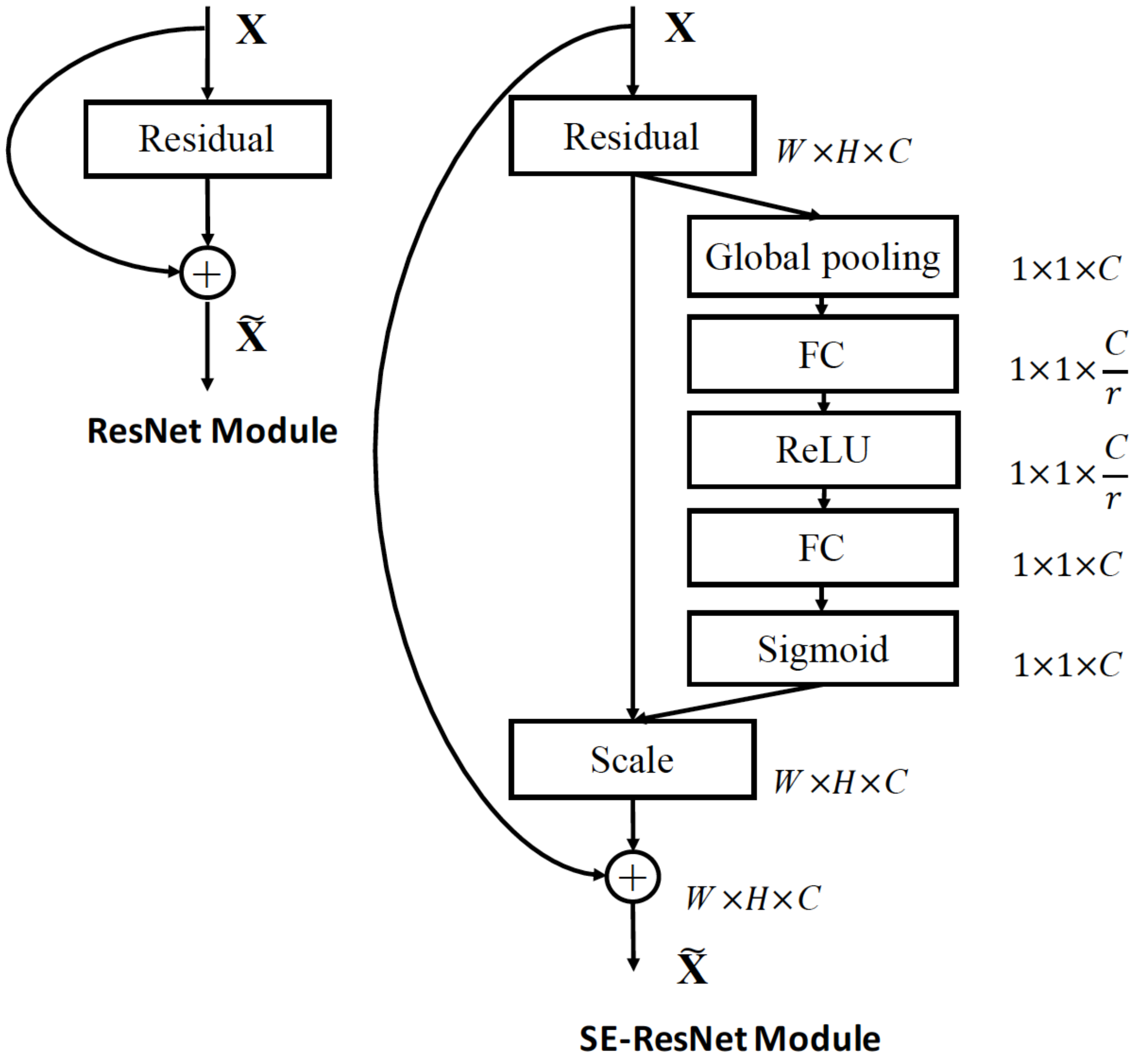
- Rescales each channel in a data-dependent way, to emphasize important features
- Scaled channel $\tilde{X}_c = s_c U_c$, where U_c is the channel and $0 \leq s_c \leq 1$
- "Squeeze" operation $z_c = F_{sq}(U_c)$ is the mean of layer outputs U_c across channel c
- Scale factors are $\boldsymbol{s} = \sigma(W_2 \delta(W_1 \boldsymbol{z}))$, where δ is ReLU and σ is sigmoid
- W_1 reduces the dimension (typically 16x)



Hu et al. (2018) CVPR

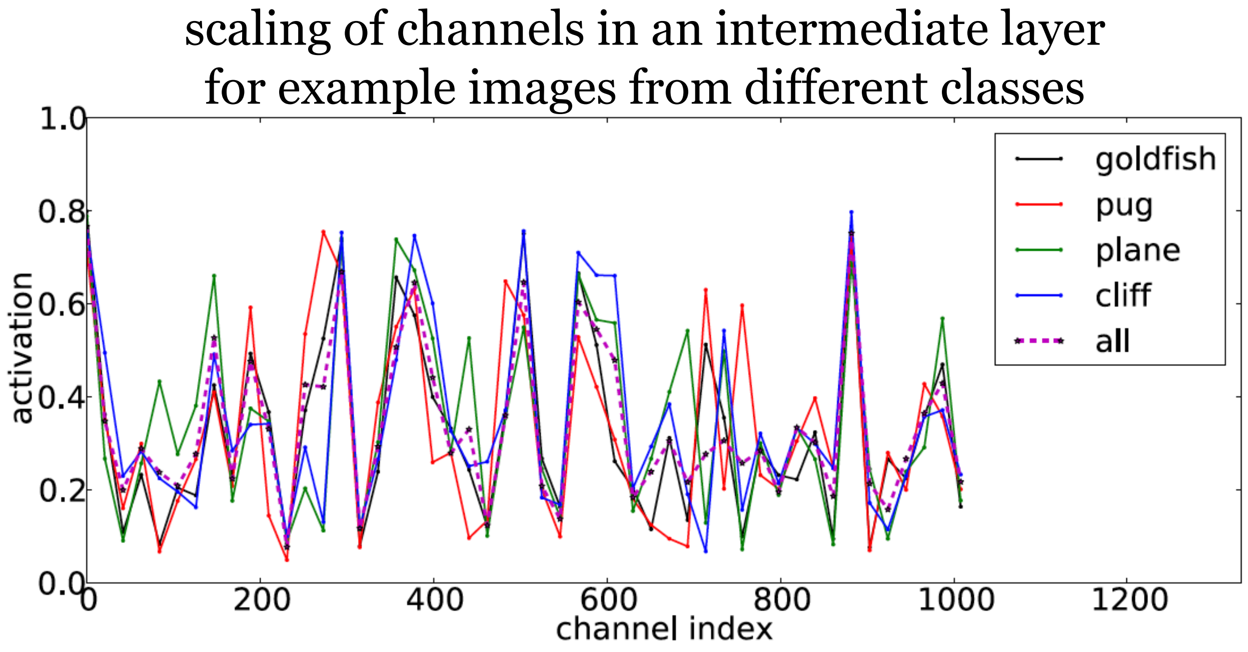
Squeeze and excitation

- Can be applied to any convolutional network; they tested with Inception networks and ResNets
- A small ensemble of different SE networks achieved top-5 error 2.251% on ILSVRC 2017



Squeeze and excitation

- In early layers channel scales are fairly static
- In later layers channel scales become class dependent

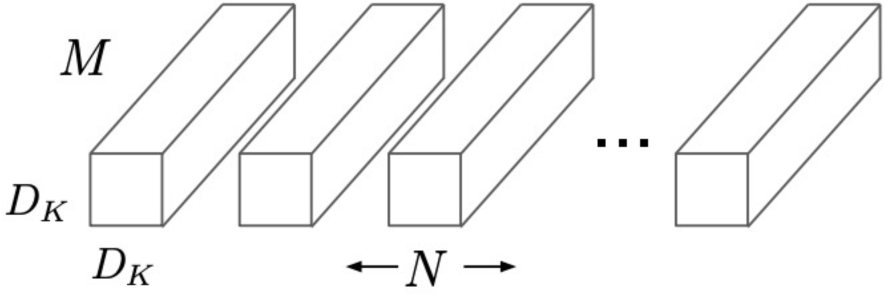


UNIVERSITY OF WATERLOO

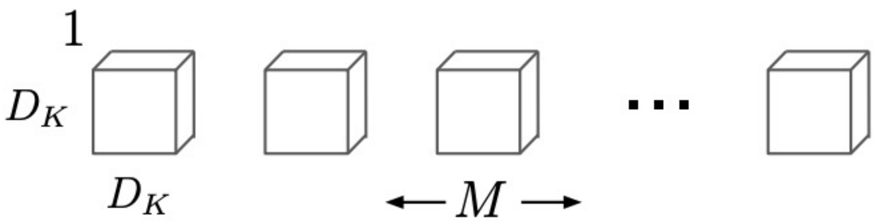
MOBILENETS WERE OPTIMIZED FOR EDGE COMPUTING

MobileNet V1

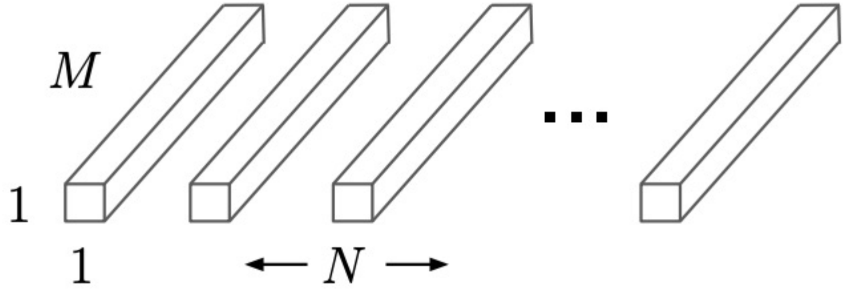
- Howard et al., (2017) arXiv
- Depthwise-separable convolutions
  - Replaces a standard convolutional layer with D_K by D_K kernels (usually 3x3), M input channels and N output channels (MND_K^2 parameters; $MND_K^2D_F^2$ operations; where D_F is the width and height of the feature map)
  - First convolve each channel with a different filter ($\mathrm{M}D_k^2$ parameters; $\mathrm{M}D_k^2D_F^2$ operations)
  - Next apply 1x1 convolution to combine channels (MN parameters; MND_F^2 operations)
  - Nearly 9x more efficient with 3x3 kernels



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



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

MobileNet V1

- Small loss of accuracy despite greater efficiency

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

MobileNet V1

- Batch norm and ReLU after each depthwise convolution and each 1x1 convolution

Depthwise convolution

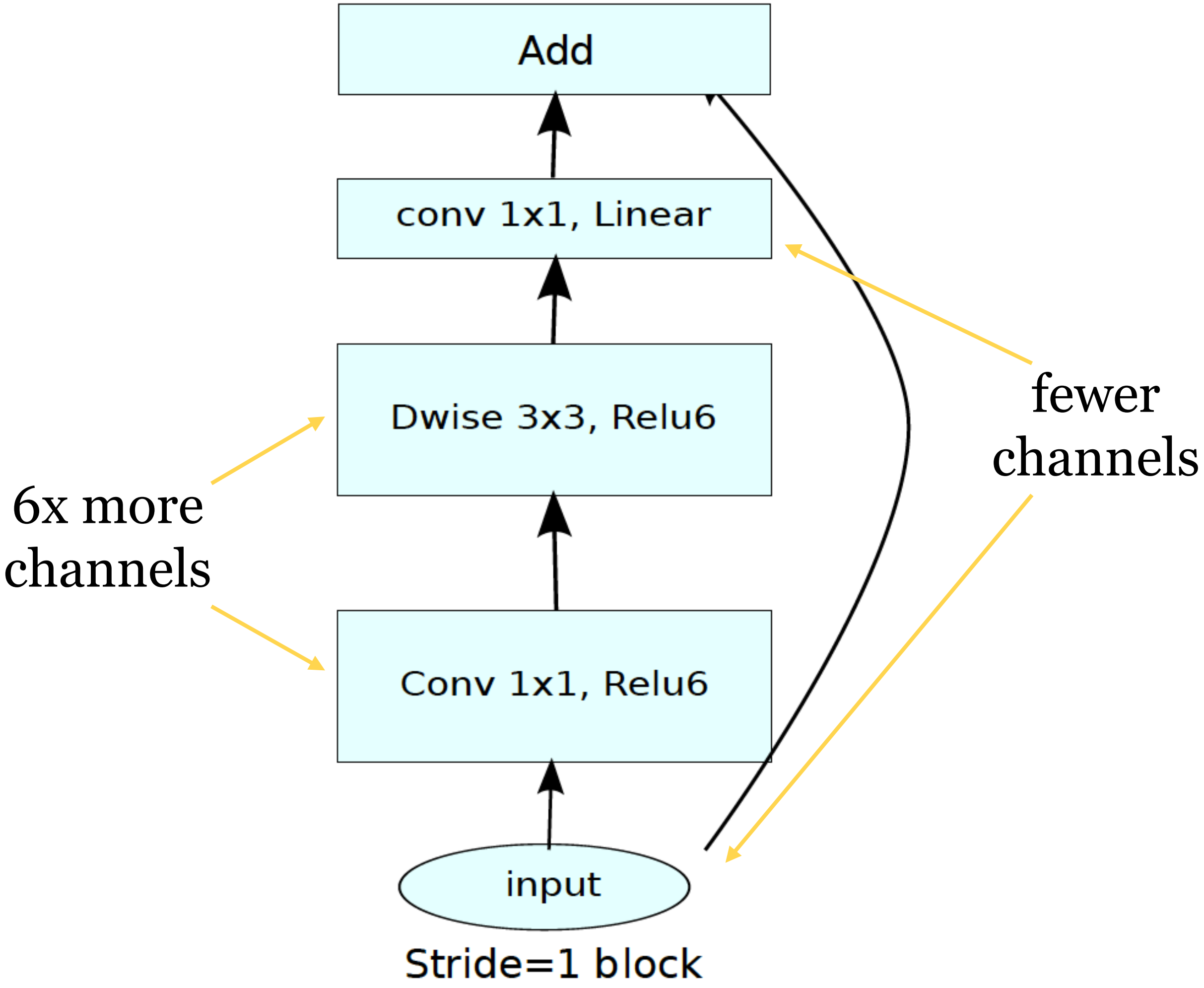
1x1 convolution

Stride 2

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	$1 \times 1 \times 1000$

MobileNet V2 & V3

- Two key ideas in V2 (Sandler et al., 2018) versus V1:
  - Bottlenecks: Some layers have fewer channels
  - Inverted residuals: Some past networks had used residual connections in parallel with bottlenecks; this model used residual connections between bottleneck layers
- These improve both performance and runtime over V1
- There is also a MobileNet V3 (Howard et al., 2019), with further improved performance via squeeze & excitation layers and new architecture search



EFFICIENTNET USED A SCALING HEURISTIC

EfficientNet

- Tan & Le (2019) *PMLR*
- Intuition:
  - Given a baseline network, previous approaches had improved performance by scaling up depth (# layers) or width (# channels) or less commonly resolution
  - Maybe better to scale all three up together, e.g., higher resolution may require more depth to increase the receptive field size and more width to model a greater variety of fine-grained patterns
  - Start with an optimized small network
  - Scale depth, width, and resolution together with an empirical ratio
- 2.9% top-5 error on ImageNet with a single network; smaller & faster than previous top networks
- State-of-art performance in transfer learning to other vision tasks

Scaling method

Start with a small network of the form,

$$\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i \rangle}\right),$$

where $\odot$ denotes composition, $\mathcal{F}_i^{L_i}$ means the i$^{\text{th}}$ kind of layer is repeated L_i times in a block, with input to the block of shape H_i by W_i with C_i channels.

Scale the network as,

$$\mathcal{N}(d, w, r) = \bigodot_{i=1\ldots s} \mathcal{F}_i^{dL_i}\left(X_{\langle rH_i, rW_i, wC_i \rangle}\right),$$

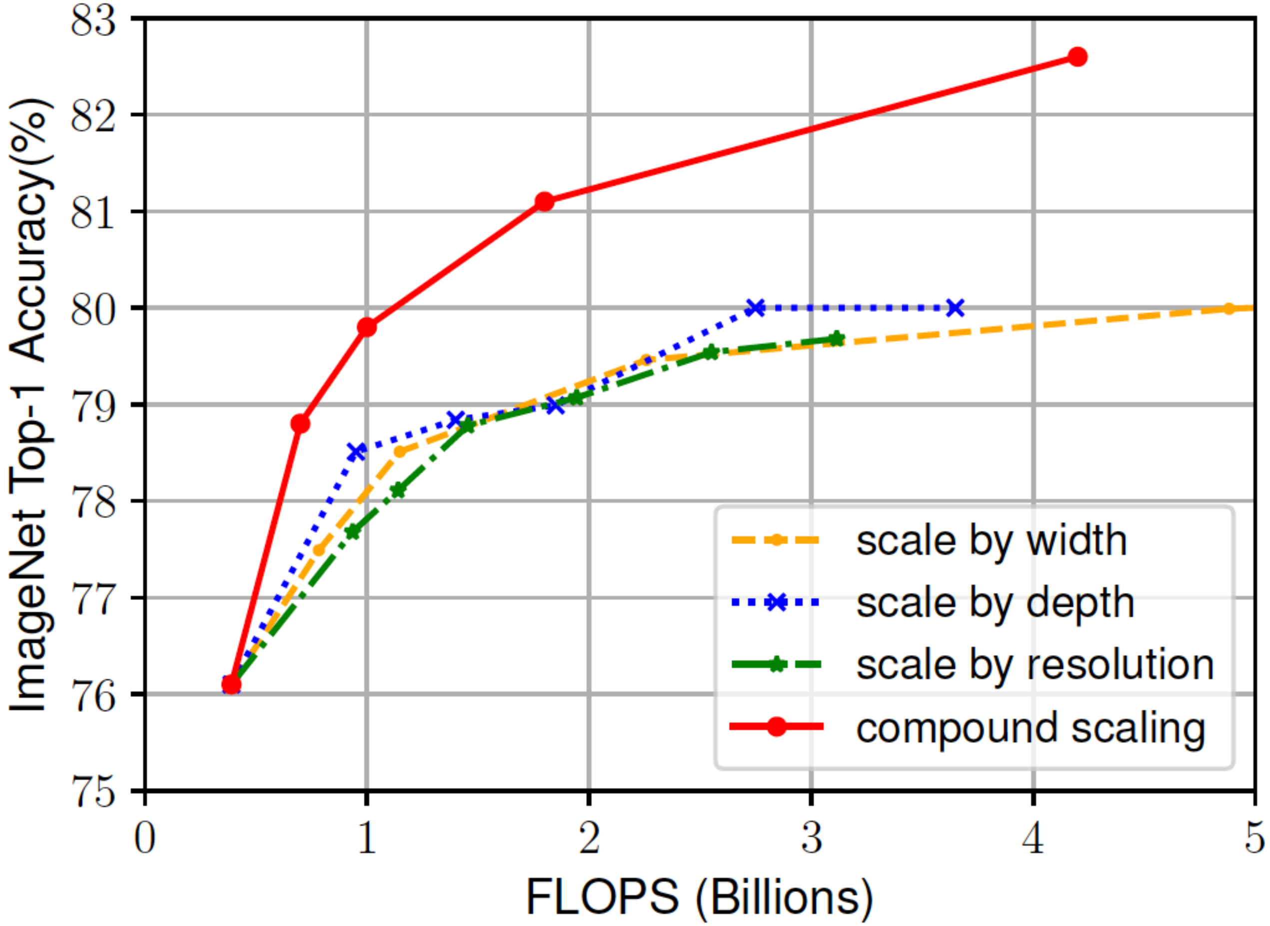
subject to constraints on memory and floating-point operations.

Scaling method

- Scale d, w, r together as $d = \alpha^{\phi}$, $\mathrm{w} = \beta^{\phi}$, $\mathrm{r} = \gamma^{\phi}$ such that $\alpha, \beta, \gamma \geq 1$ and $\alpha\beta^2\gamma^2 \approx 2$
- Note that the floating-point operations (FLOPS) scale with $\alpha\beta^2\gamma^2$, so they increase by a factor of 2^{ϕ}
- Starting with a small network, fix $\phi = 1$ (allow twice as many FLOPS) and do a grid search to find best-performing α, β, γ
- Fix α, β, γ and set $\phi > 1$ to use up a given FLOPS budget

Scaling method

- It is possible that better parameters could be found by grid search at the large scale but
  - That would be more expensive
  - The proposed scaling works well empirically

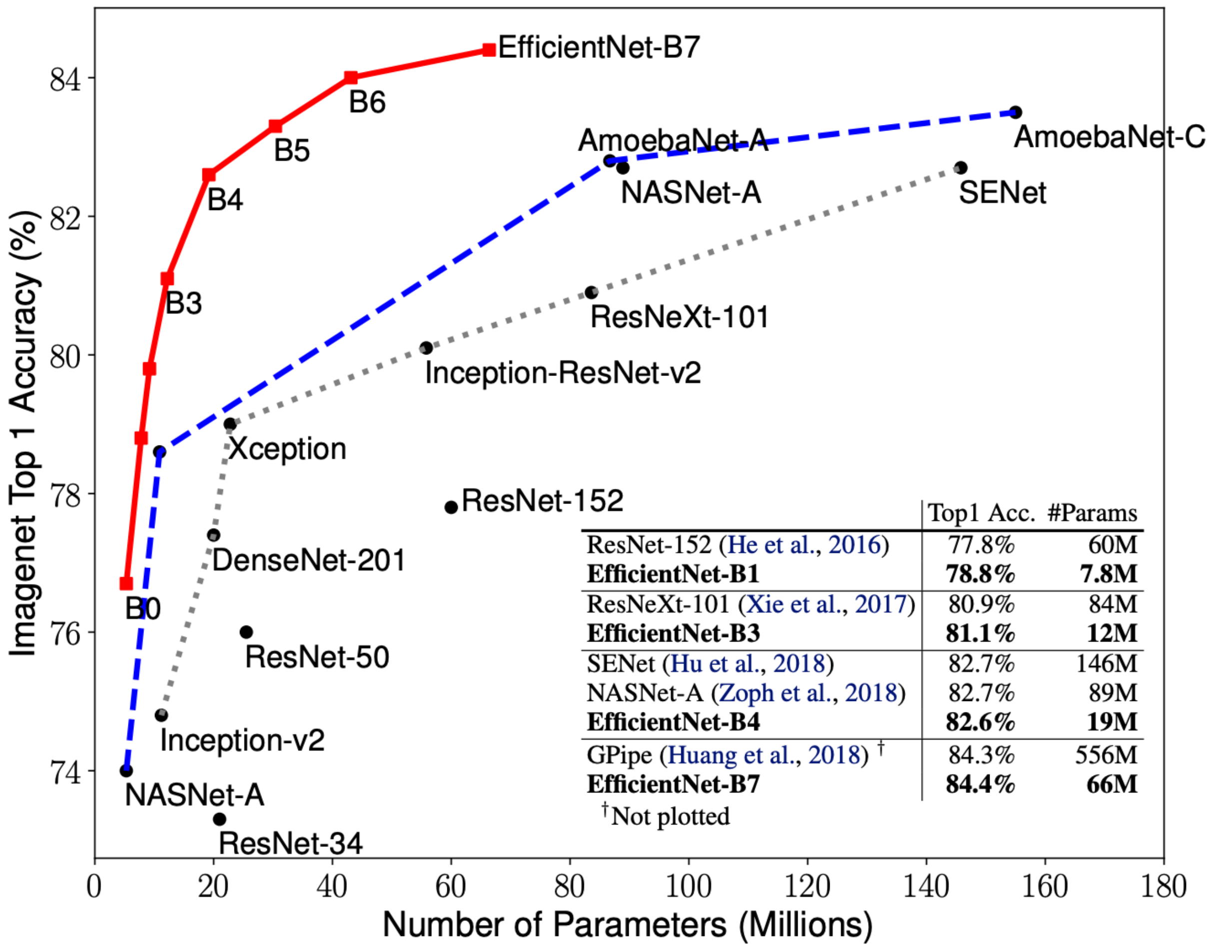


Baseline network

- Found through a hyperparameter search
- MBConv is mobile inverted bottleneck convolution from MobileNet V2; number indicates expansion ratio
- They add a squeeze and excitation to each layer, from Squeeze & Excitation Networks (Hu et al., 2017)

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

EfficientNet



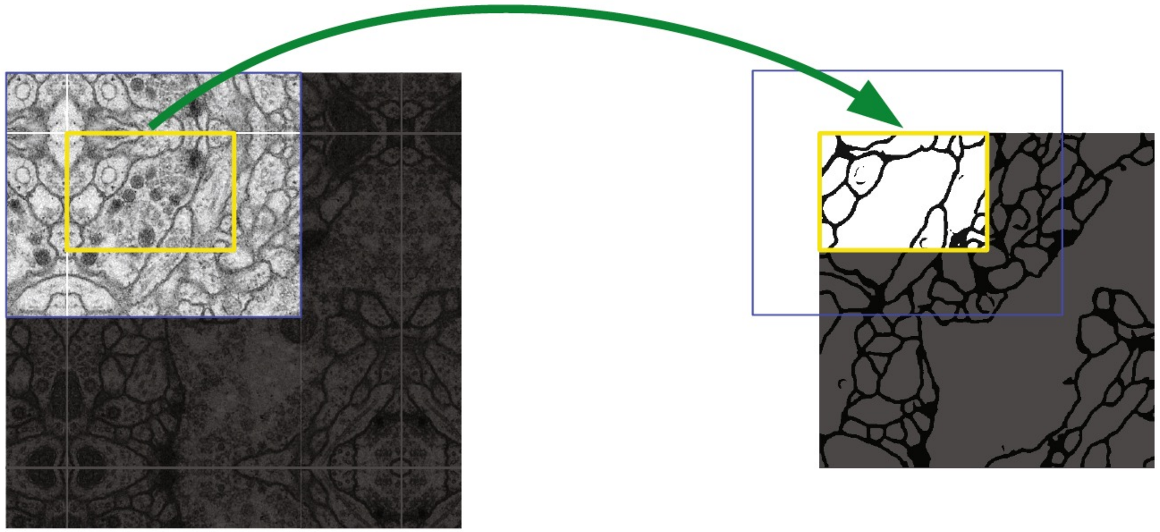
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018)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

†Not plotted

U-NET COMBINED LOW AND HIGH-LEVEL FEATURES TO PRODUCE STRUCTURED OUTPUT

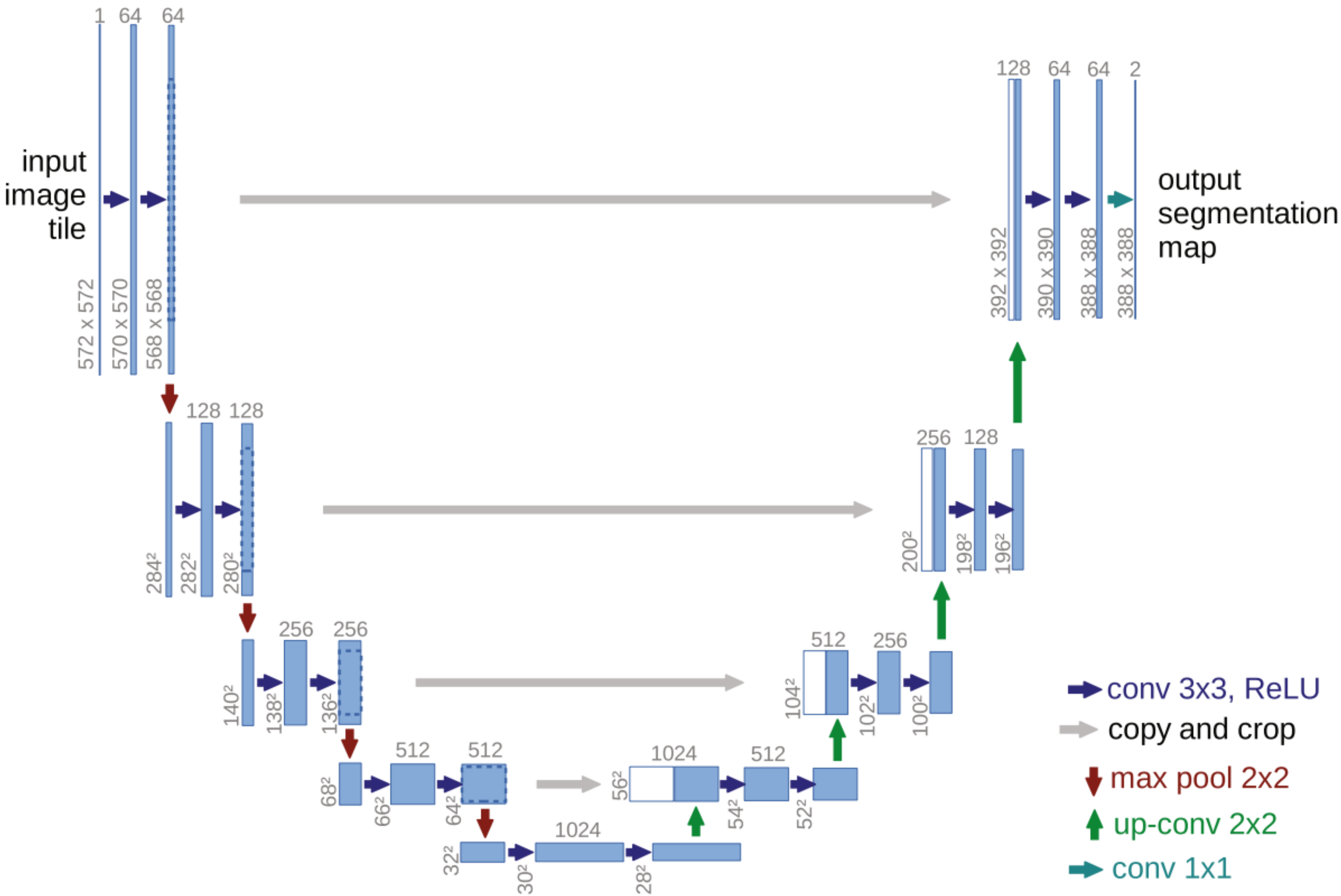
U-Net

- Ronneberger et al. (2015) *International Conference on Medical Image Computing and Computer-Assisted Intervention*
- Designed to segment images of cells in overlapping tiles
- Used valid convolutions with mirror padding at edges of image
- Built on fully-convolutional networks (Long et al., 2015, CPVR) (not all-convolutional networks)

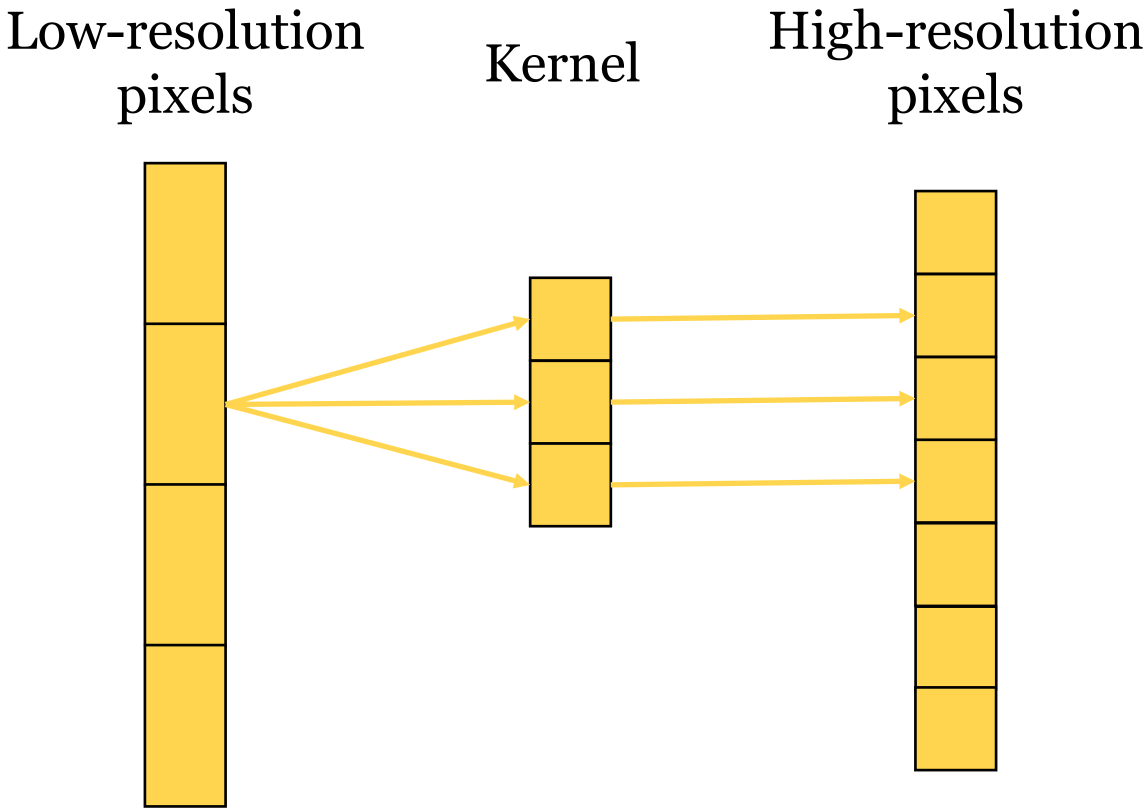


U-Net

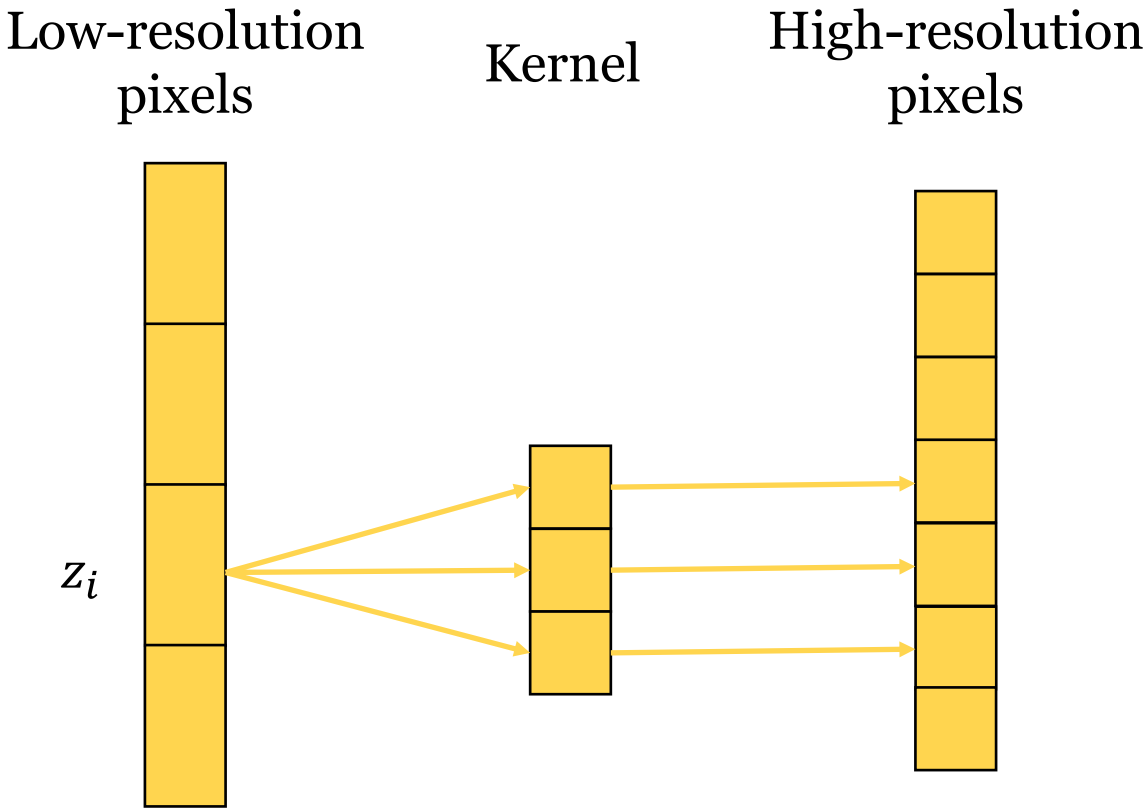
- Contractive and expansive parts
- Upsampling in expansive part via convolution
- Shortcut connections combine lower and higher- level features



Upsampling via convolution



Upsampling via convolution



CONVOLUTIONAL NETWORKS CONTINUE TO IMPROVE

Recent Advances

- Ding et al. (2021) trained a branched architecture (parallel paths with kernels of different sizes) and merged parallel kernels to produce a more efficient VGG-like architecture with the same performance at inference time
- Liu et al. (2022) achieved state-of-art performance by updating a ResNet in several ways that were inspired by transformers (larger kernels, layer norm rather than batch norm, using image patches at input rather than pooling overlapping kernels, GELU rather than ReLU, fewer nonlinearities, etc.)
- Ding et al. (2022) improved performance through careful use of large (31x31) kerels
- Liu et al. (2023) showed that convolutional networks with very large (51x51) sparse kernels outperformed smaller kernels in image recognition, object detection, and semantic segmentation.

Summary

1. The earliest convolutional networks were LeNets, developed for handwritten digit recognition
2. AlexNet ignited interest in deep learning by combining convolutional networks with GPUs and big data
3. VGG networks added simplicity and depth
4. Inception networks introduced parallel paths with different kernel sizes
5. All-convolutional networks removed max-pooling and fully connected layers
6. ResNets added residual connections to facilitate training of very deep networks

Summary

7. DenseNets included all possible skip connections
8. Squeeze & excitation networks applied a channel-wise gain that was input-dependent and learned
9. MobileNets were optimized for edge computing
10. EfficientNet used a scaling heuristic
11. U-Net combined low and high-level features to produce structured output
12. Convolutional networks continue to improve

References

- Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021). RepVGG: Making VGG-style convnets great again. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13733-13742).
- Ding, X., Zhang, X., Han, J., & Ding, G. (2022). Scaling up your kernels to 31x31: Revisiting large kernel design in CNNs. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11963-11975).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
- Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).
- Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
- Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.
- Krizhevsky et al., 2012, ImageNet Classification with Deep Convolutional Neural Networks, NIPS
- LeCun, 1989, Generalization and network design strategies. Connectionism in perspective, 19(143-155), 18.
- LeCun et al., 1989, Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, vol. 1, no. 4, pp. 541–551,
- Le Cun et al., 1998, Gradient-based learning applied to document recognition, Proc IEEE
- Liu, S., Chen, T., Chen, X., Chen, X., Xiao, Q., Wu, B., ... & Wang, Z. (2022). More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. arXiv preprint arXiv:2207.03620.

UNIVERSITY OF WATERLOO

References

- Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11976-11986).
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18 (pp. 234-241). Springer International Publishing.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
- Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *ICLR*. arXiv preprint arXiv:1409.1556.
- Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
- Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).
- Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).
- Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.