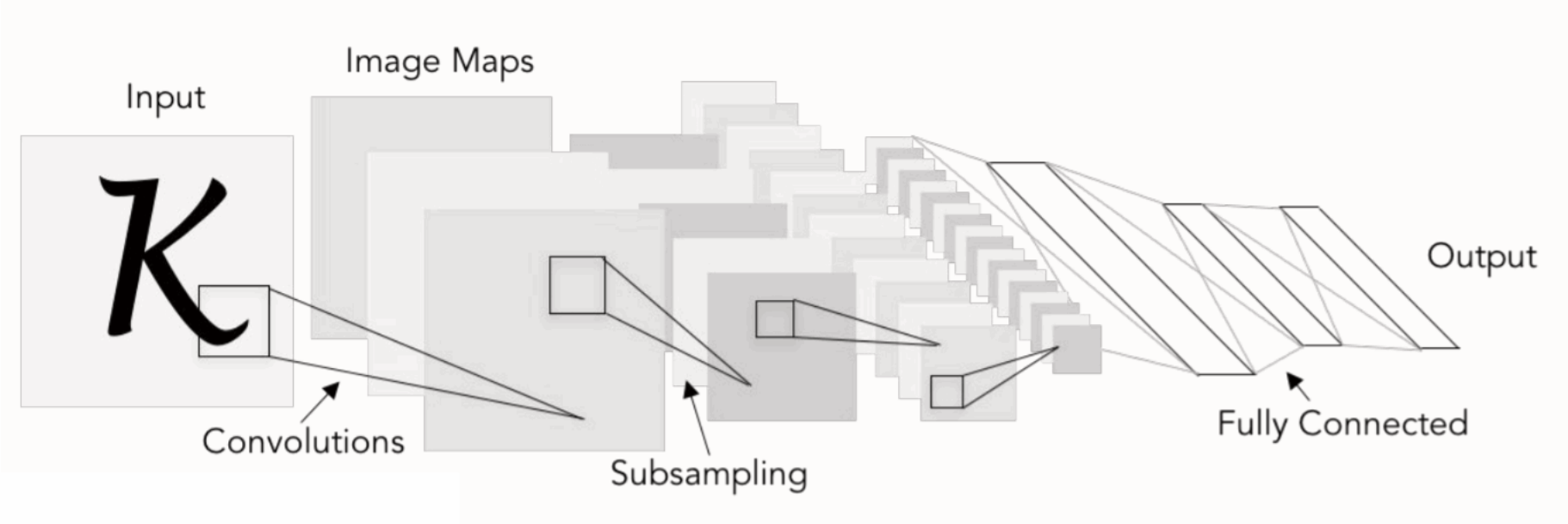
lecture 7

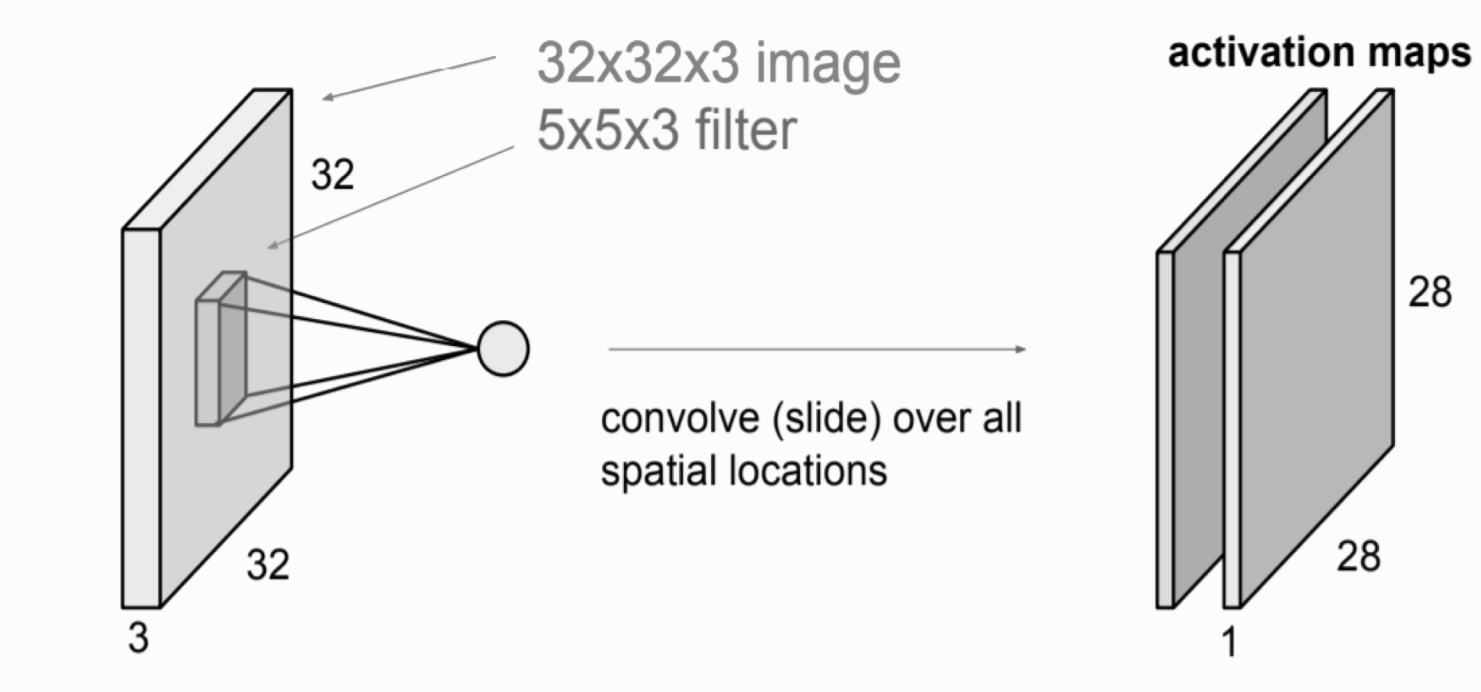
Dong-Geol Choi
Hanbat Nat'l Univ.

Review: LeNet-5

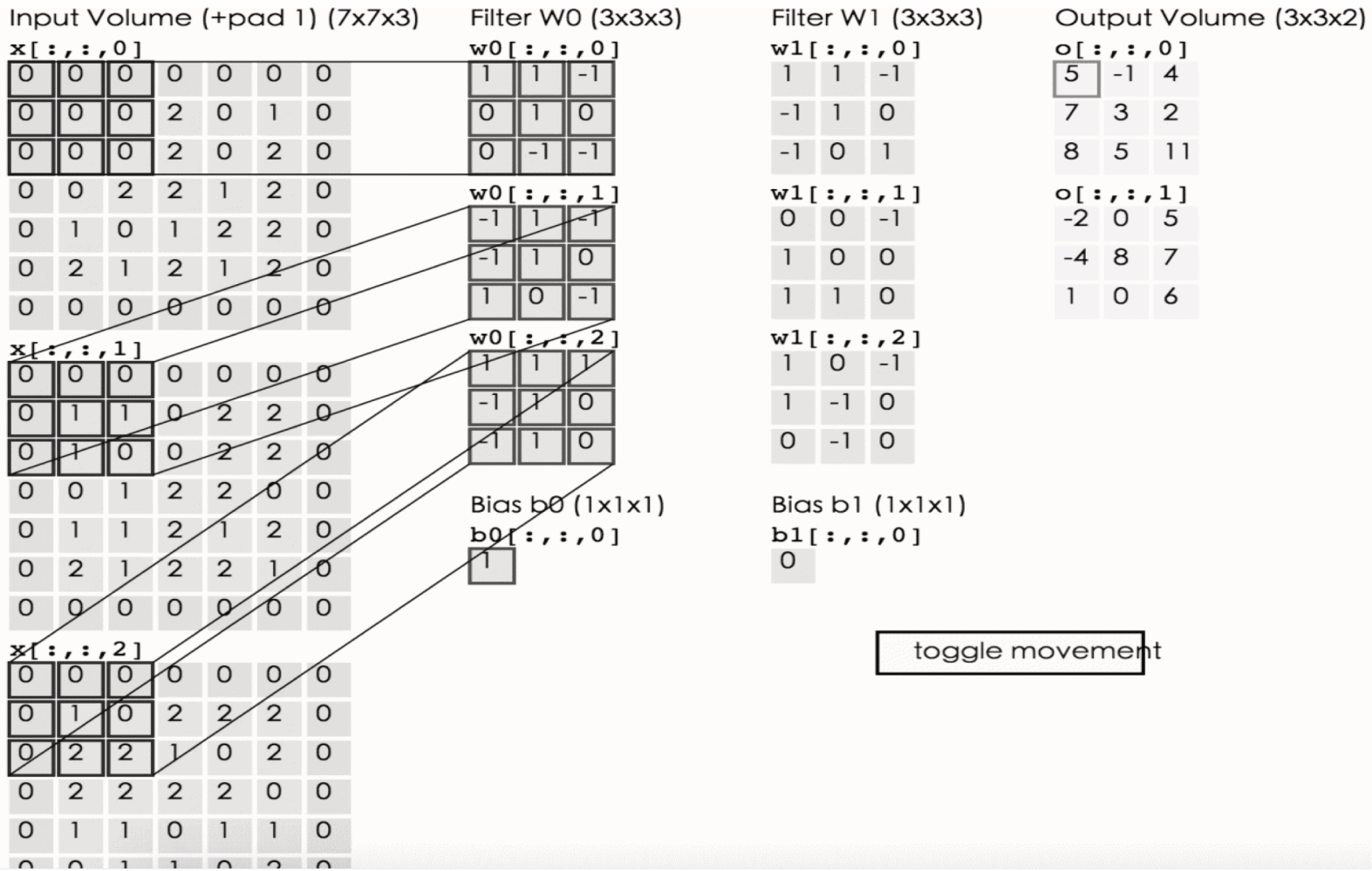


LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Review: Convolution

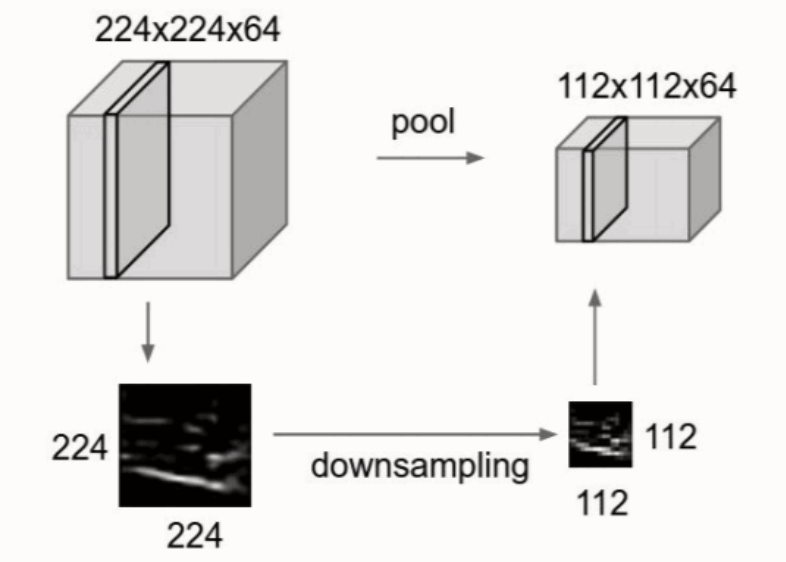


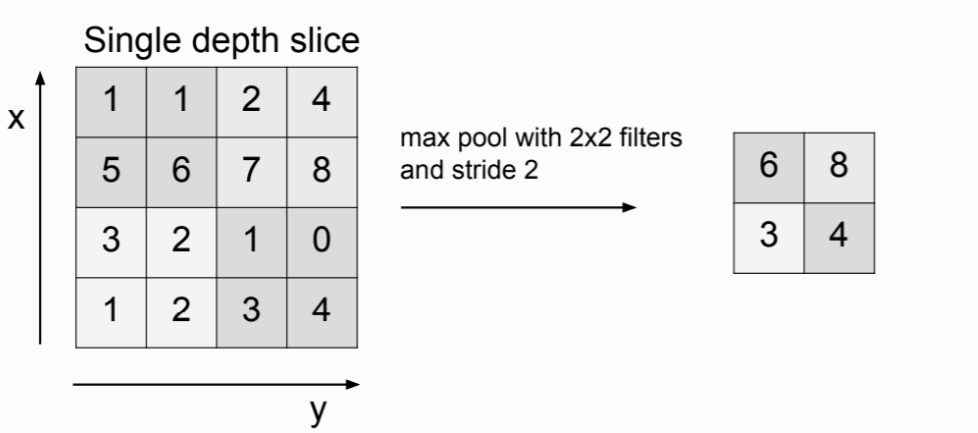
Review: Convolution



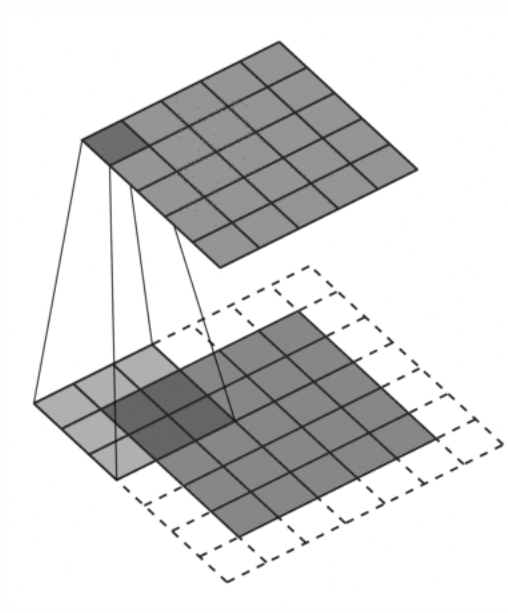
Review: Pooling

- makes the representations smaller and more manageable
- operates over each activation map independently:

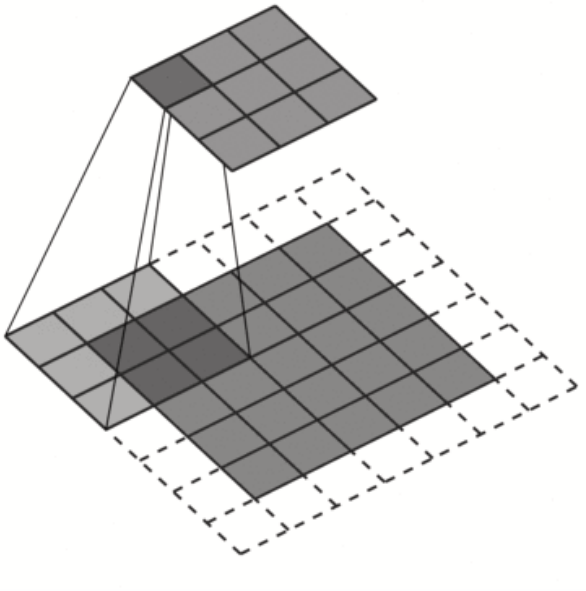




Review: Stride

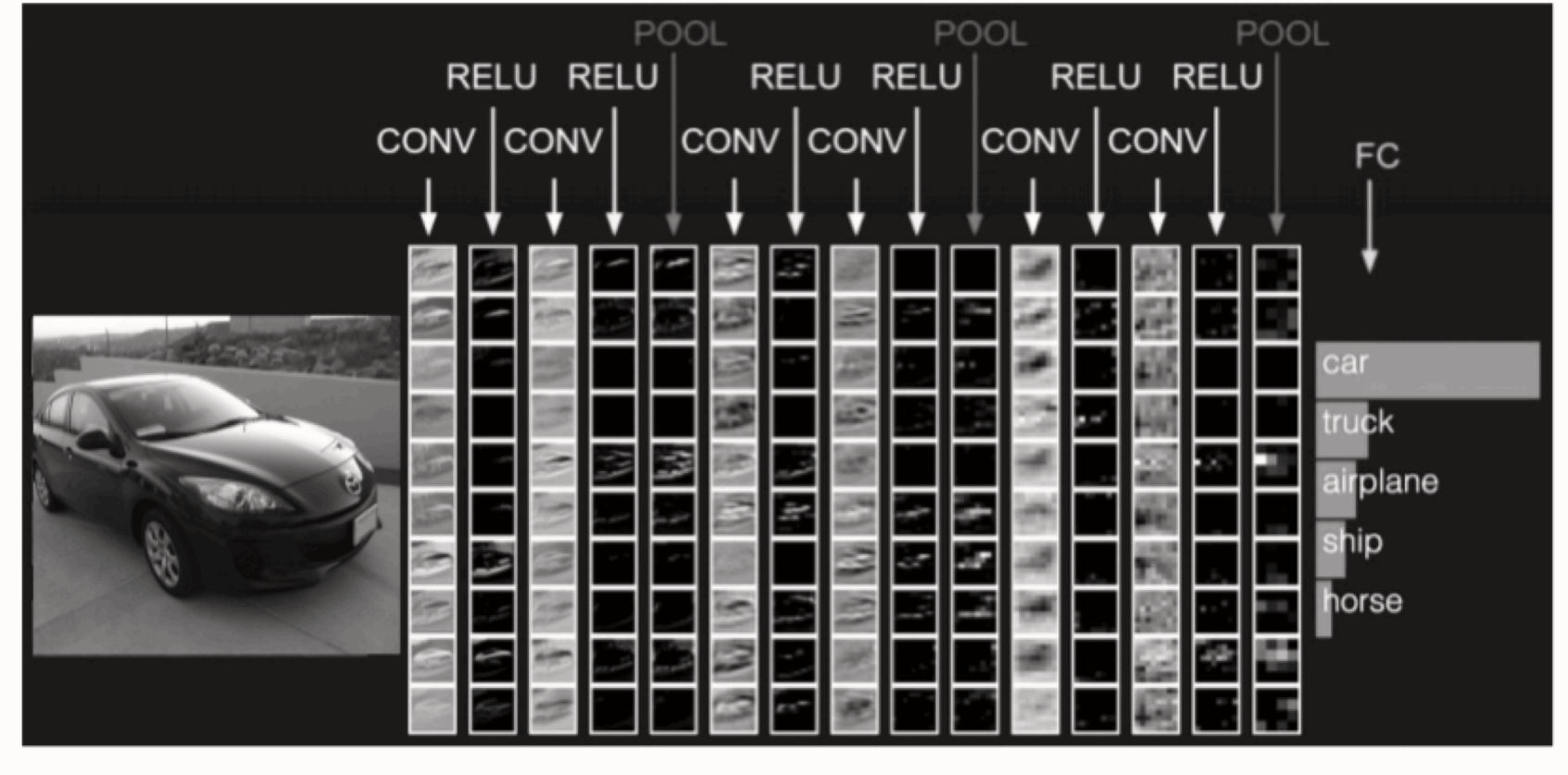


stride 1



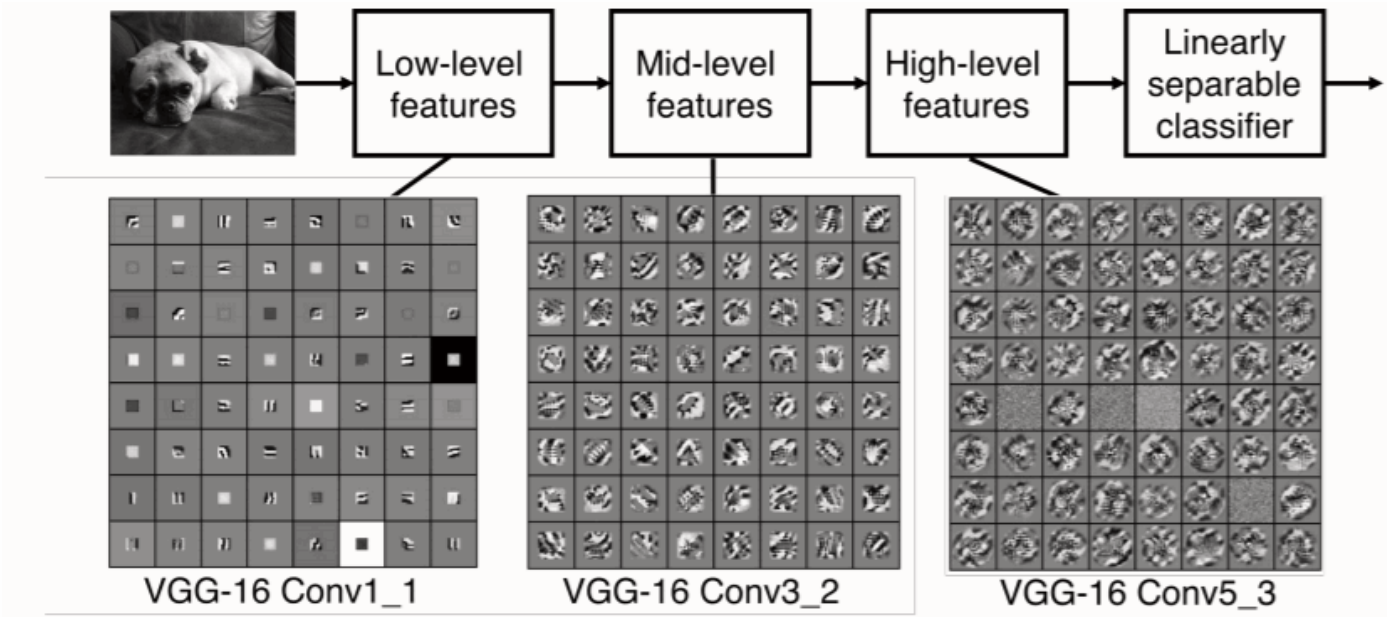
stride 2

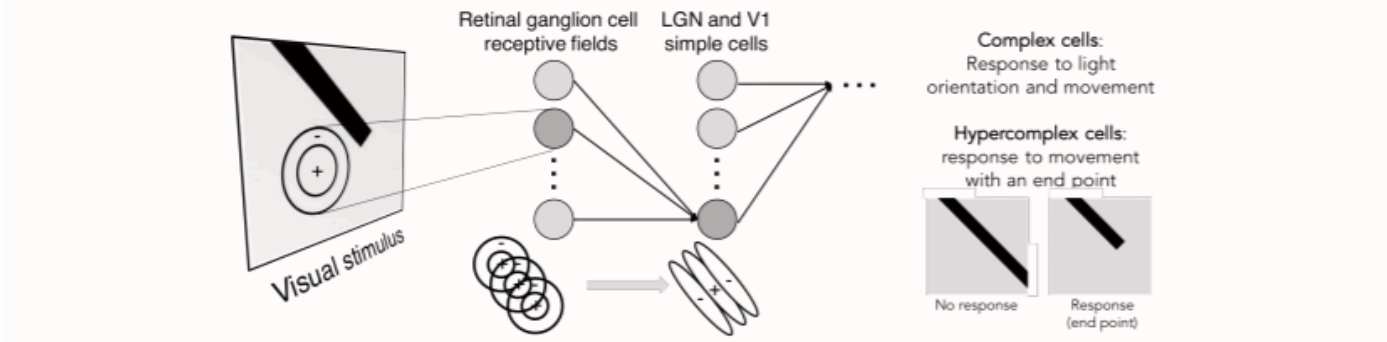
Today's generic CNN architecture.



Today's generic CNN architecture.

HW Review: Stride vs Pooling





HW Review: Stride vs Pooling

Model		
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
Input 32×32 RGB image		
3×3 conv. 96 ReLU 3×3 conv. 96 ReLU with stride $r = 2$	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU 3×3 conv. 96 ReLU	3×3 conv. 96 ReLU 3×3 conv. 96 ReLU
	3×3 max-pooling stride 2	3×3 conv. 96 ReLU with stride $r = 2$
3×3 conv. 192 ReLU 3×3 conv. 192 ReLU with stride $r = 2$	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU 3×3 conv. 192 ReLU	3×3 conv. 192 ReLU 3×3 conv. 192 ReLU
	3×3 max-pooling stride 2	3×3 conv. 192 ReLU with stride $r = 2$

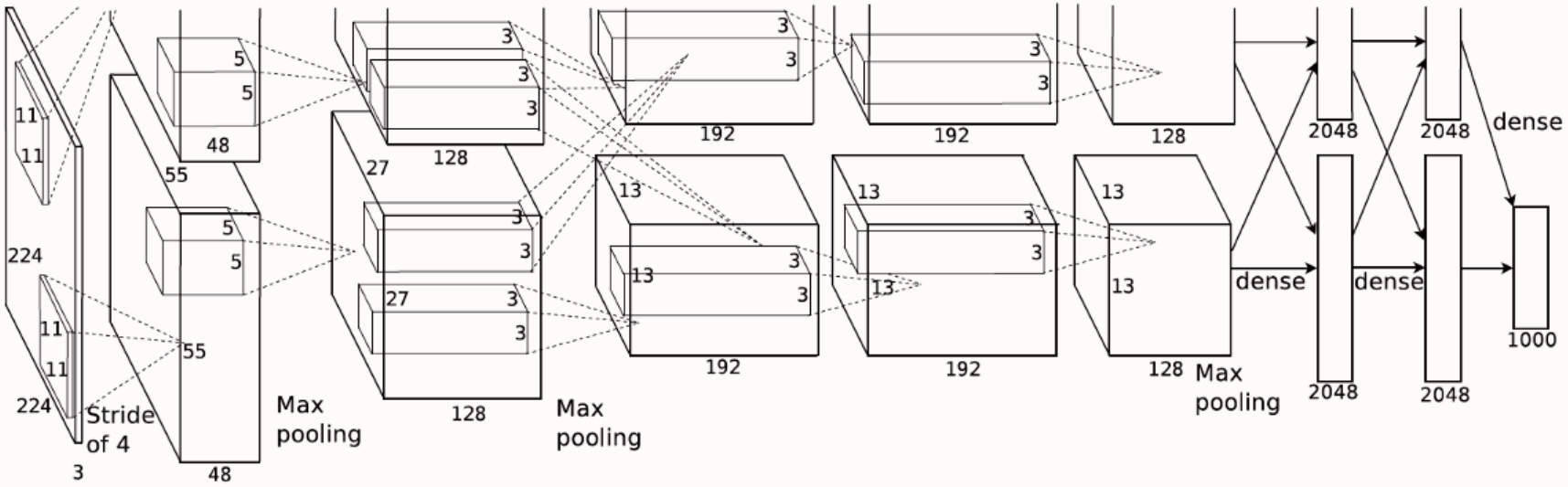
CIFAR-10 classification error

Model	Error (%)	# parameters
without data augmentation		
Model A	12.47%	$\approx$ 0.9 M
Strided-CNN-A	13.46%	$\approx$ 0.9 M
ConvPool-CNN-A	**10.21%**	$\approx$ 1.28 M
ALL-CNN-A	10.30%	$\approx$ 1.28 M
Model B	10.20%	$\approx$ 1 M
Strided-CNN-B	10.98%	$\approx$ 1 M
ConvPool-CNN-B	9.33%	$\approx$ 1.35 M
ALL-CNN-B	**9.10%**	$\approx$ 1.35 M
Model C	9.74%	$\approx$ 1.3 M
Strided-CNN-C	10.19%	$\approx$ 1.3 M
ConvPool-CNN-C	9.31%	$\approx$ 1.4 M
ALL-CNN-C	**9.08%**	$\approx$ 1.4 M

Striving for Simplicity: The All Convolutional Net **(CVPR 2014)**

CNN-Architecture: AlexNet

[Krizhevsky et al. 2012]



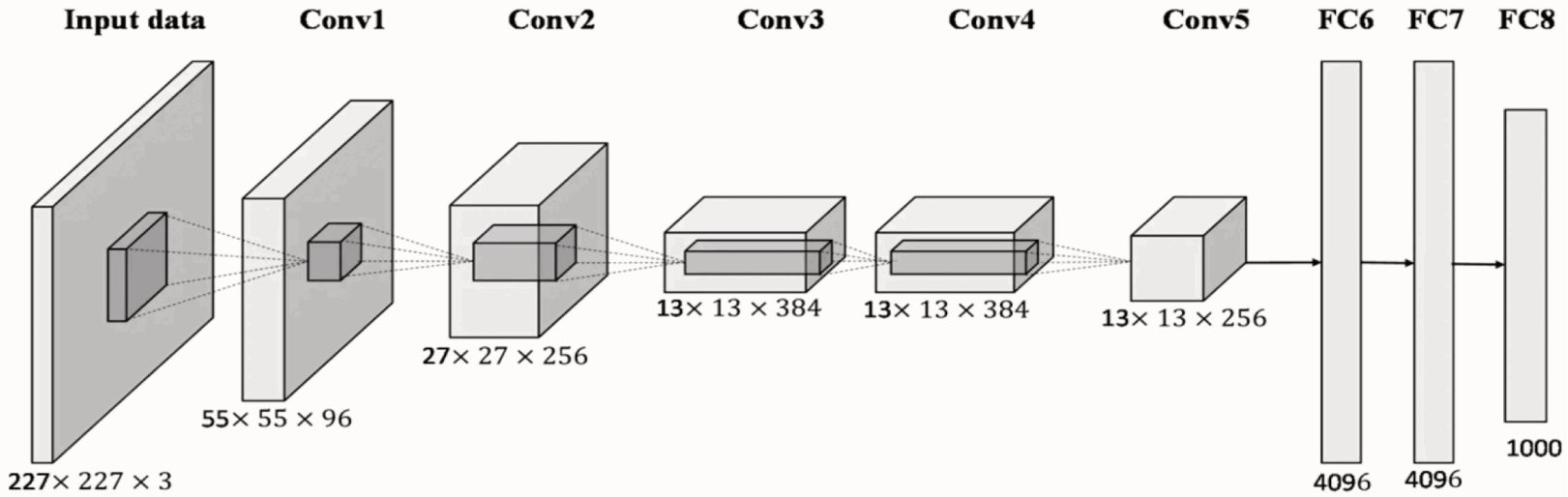
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "**Imagenet classification with deep convolutional neural networks."** Advances in neural information processing systems. 2012.

CNN-Architecture: AlexNet

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Architecture:
Layer1: Conv → ReLU → Max Pooling
Layer2: Conv → ReLU → Max Pooling
Layer3: Conv → ReLU →
Layer4: Conv → ReLU →
Layer5: Conv → ReLU → Max Pooling
Layer6: FC → ReLU → FC → ReLU → FC



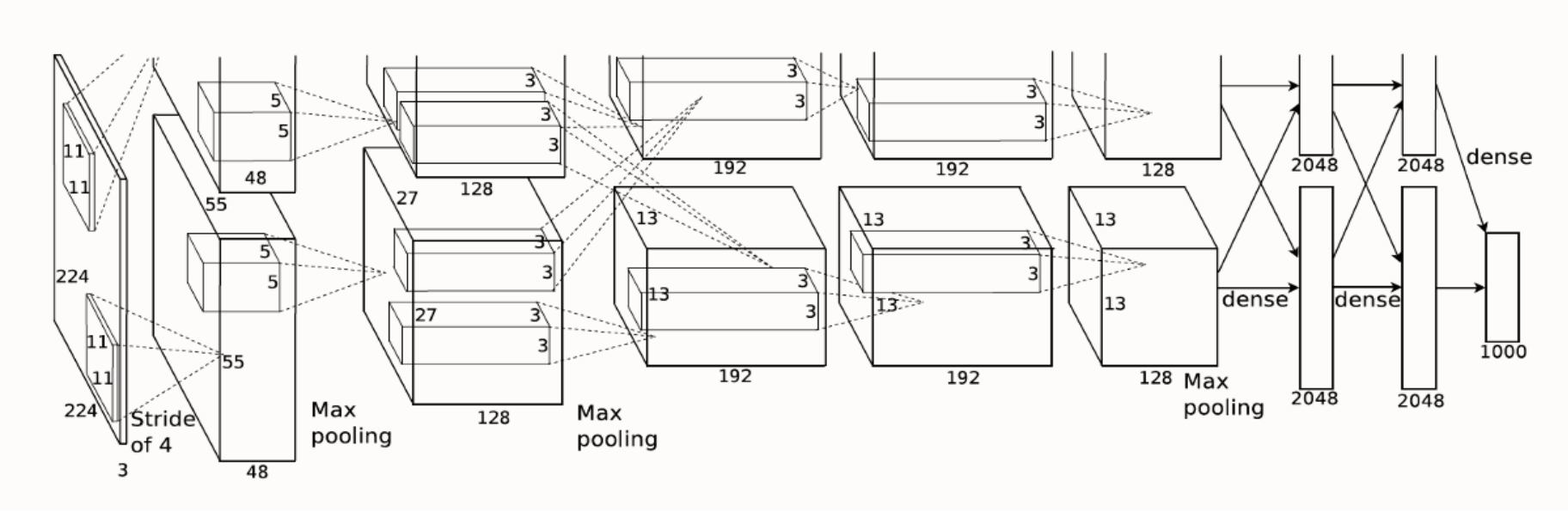
CNN-Architecture: AlexNet

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Architecture:

Layer1: Conv → ReLU → Max Pooling
Layer2: Conv → ReLU → Max Pooling
Layer3: Conv → ReLU →
Layer4: Conv → ReLU →
Layer5: Conv → ReLU → Max Pooling
Layer6: FC → ReLU → FC → ReLU → FC

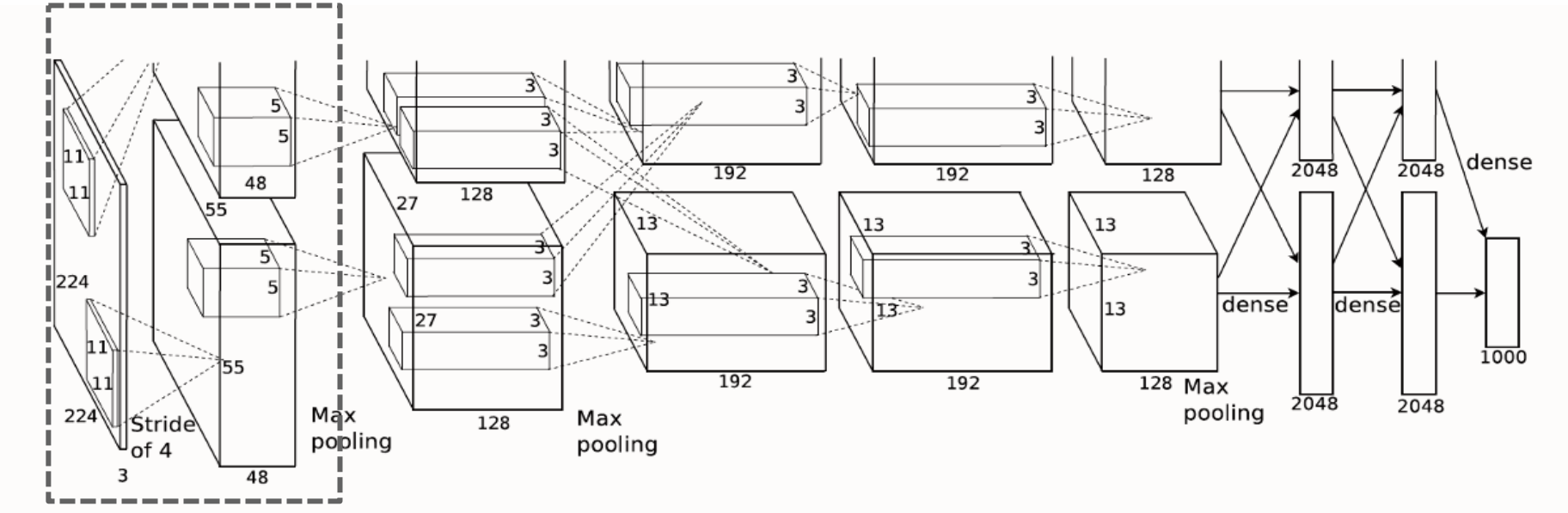


CNN-Architecture: AlexNet Layer1

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2)

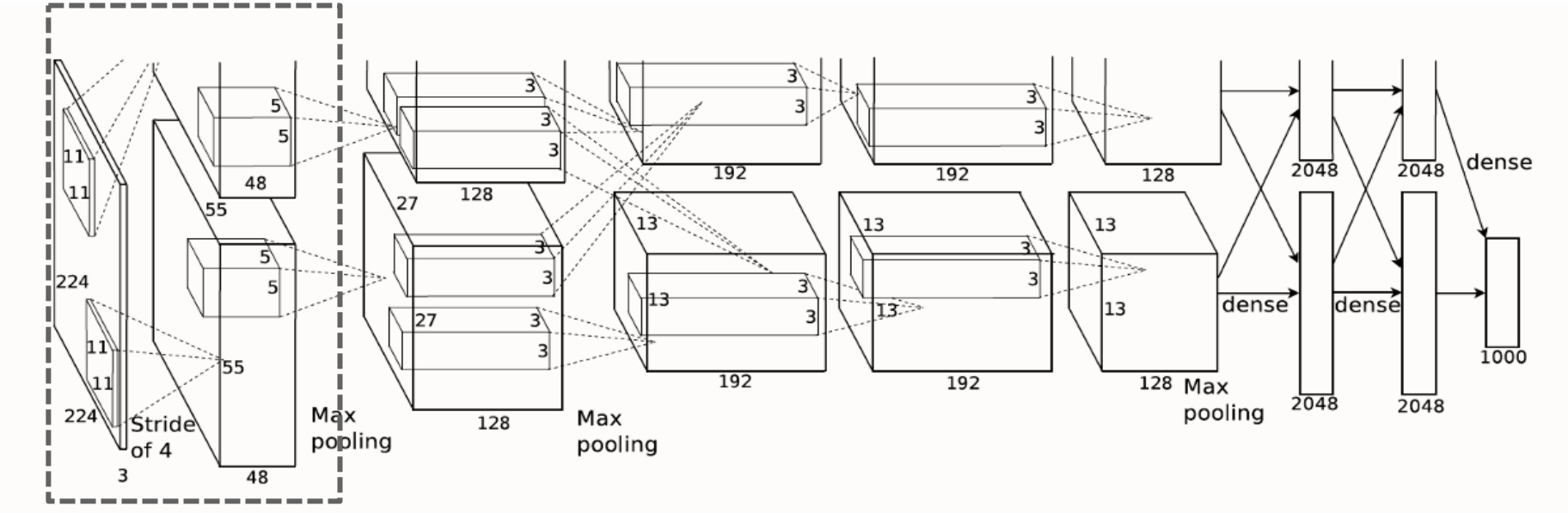


CNN-Architecture: AlexNet Layer1 (Conv)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: **Conv(11x11x96, stride 4)** → ReLU → Max Pooling(3x3, stride 2)

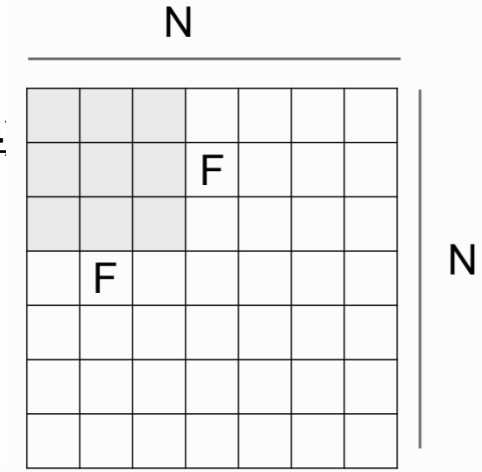


CNN-Architecture: AlexNet Layer1 (Conv)

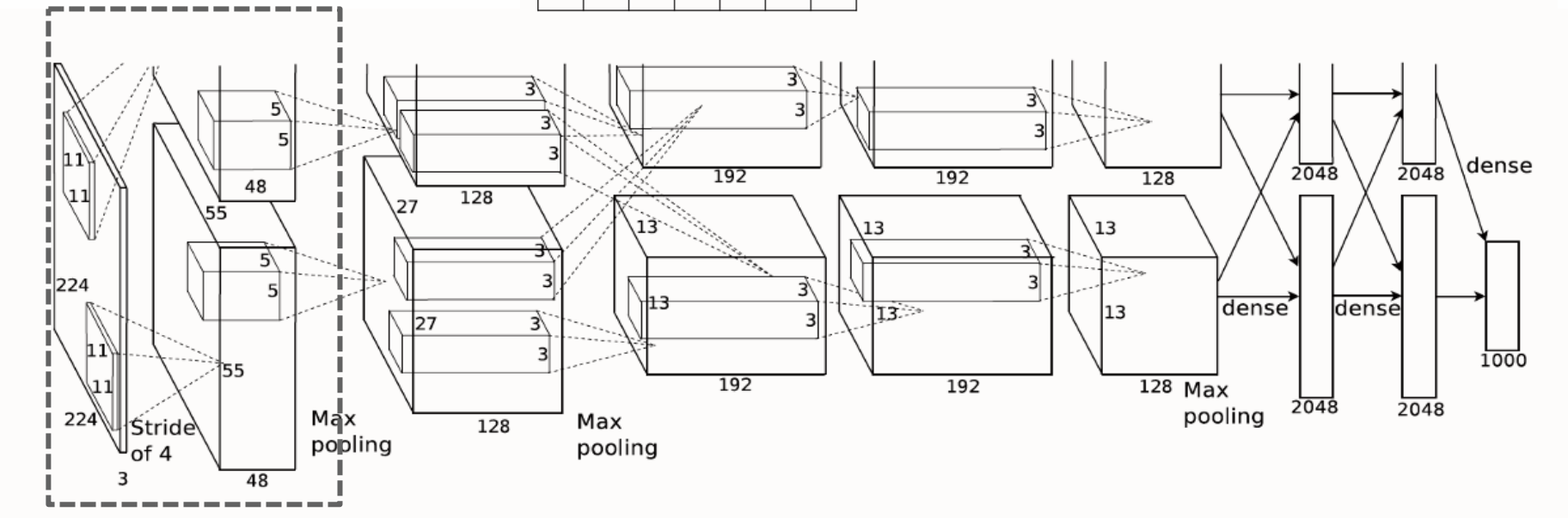
[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: **Conv(11x11x96, stride 4**



Output size:
(N - F) / stride + 1



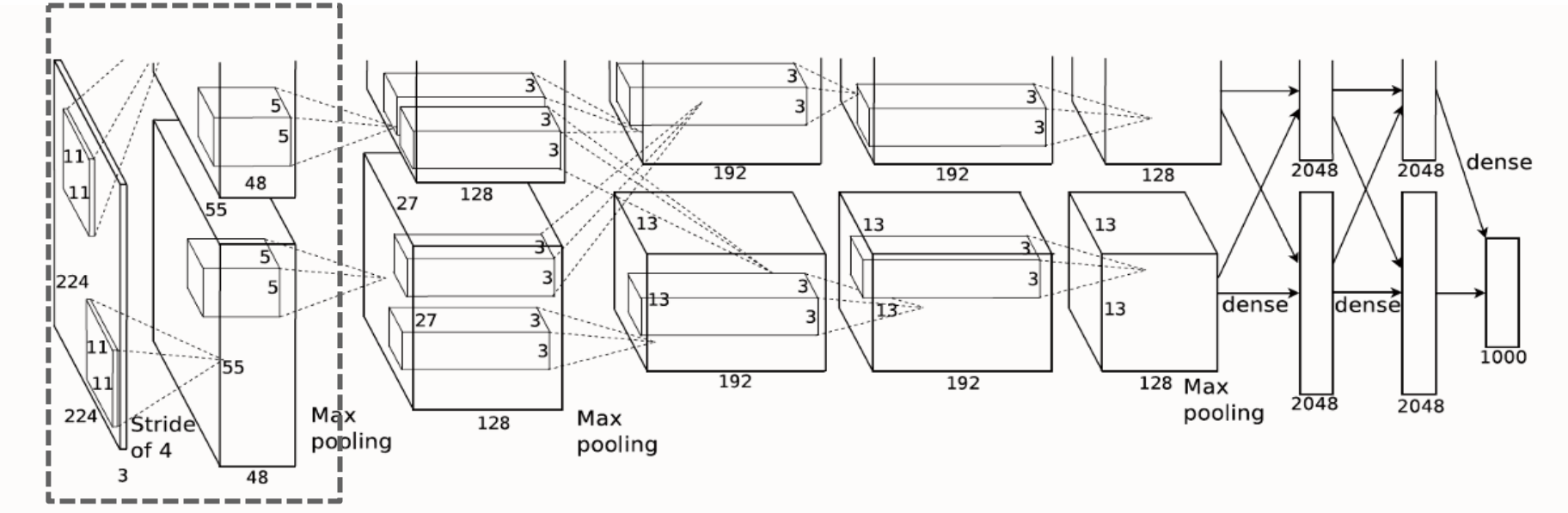
CNN-Architecture: AlexNet Layer1 (Conv)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: **Conv(11x11x96, stride 4)** → ReLU → Max Pooling(3x3, stride 2)

output: (N – F) / stride + 1 = (227 – 11) / 4 + 1 = **55 X 55 X 96**



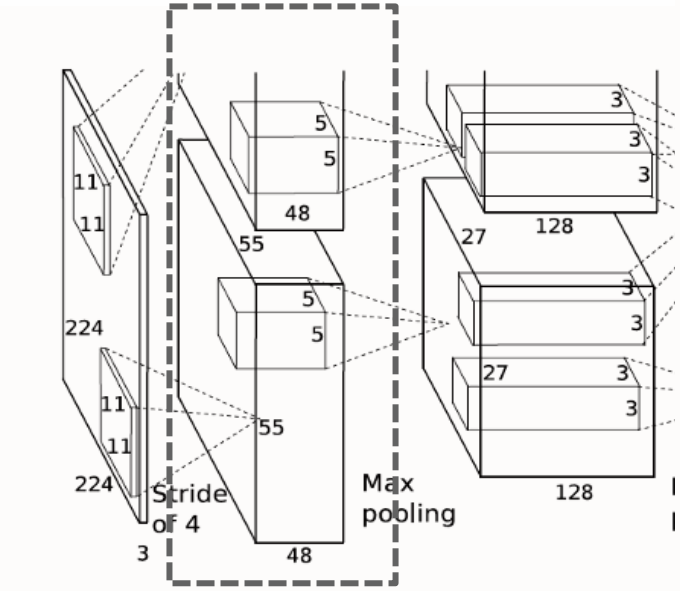
CNN-Architecture: AlexNet Layer1 (MaxPool)

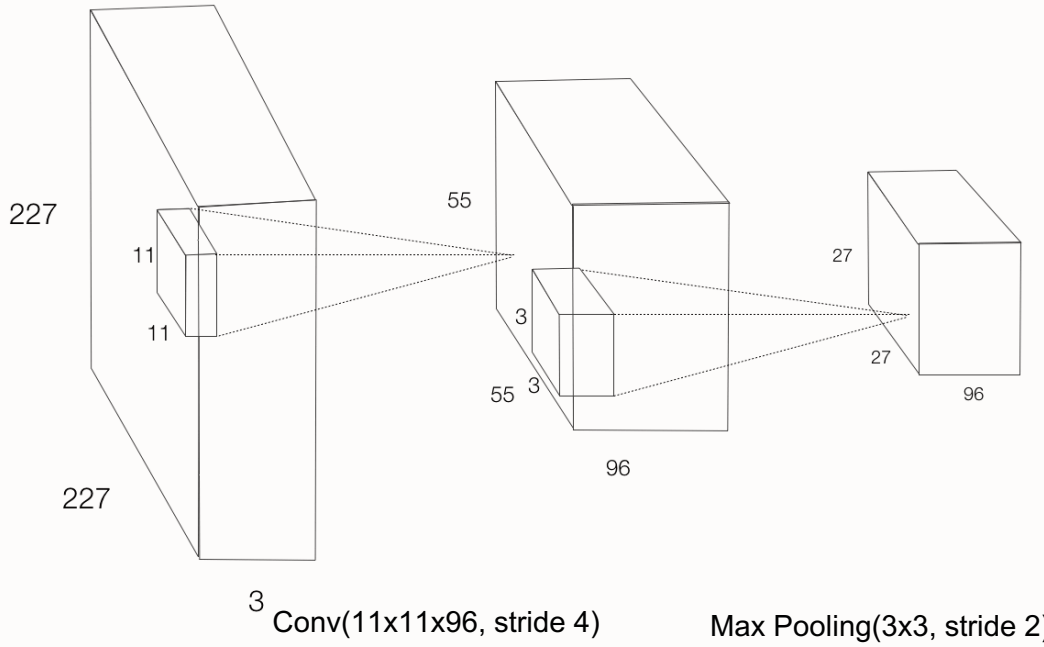
[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → **Max Pooling(3x3, stride 2)**

output: (N – F) / stride + 1 = (227 – 11) / 4 + 1 = **55 X 55 X 96**
output: (55 – 3) / 2 + 1 = **27 X 27 X 96**



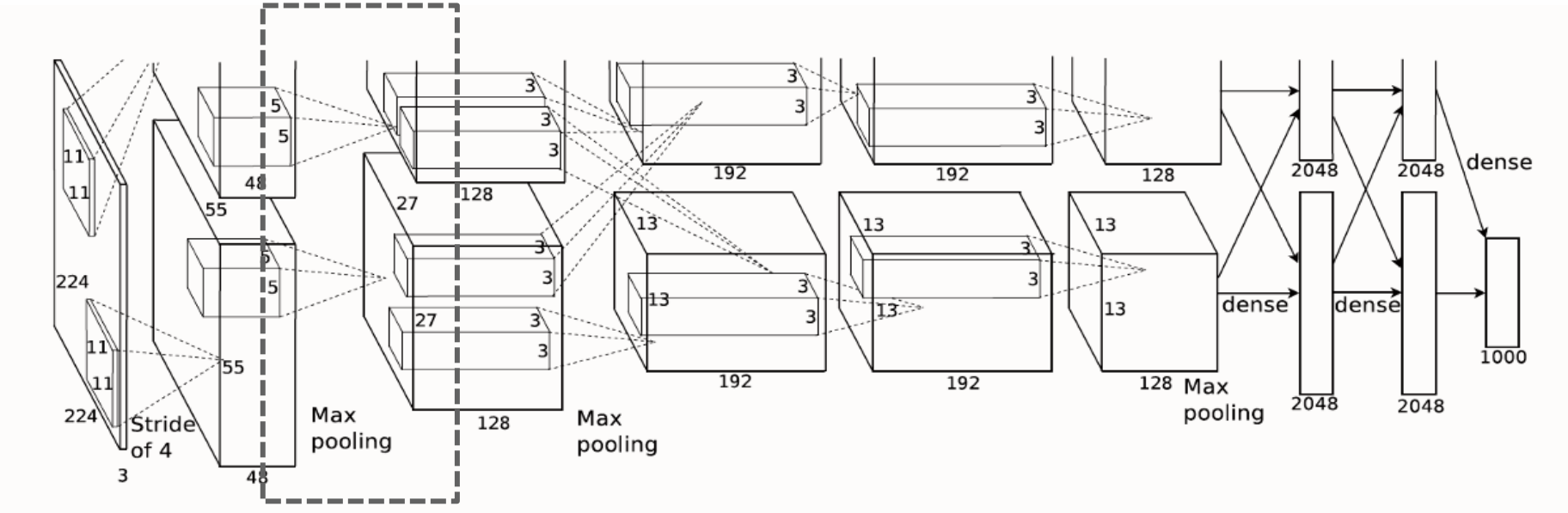


CNN-Architecture: AlexNet Layer2

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2)

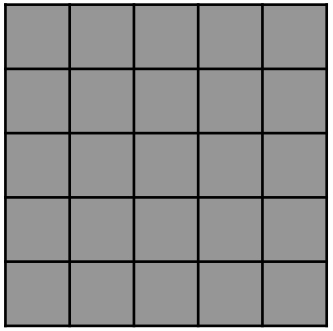


CNN-Architecture: AlexNet Layer2 (Padding)

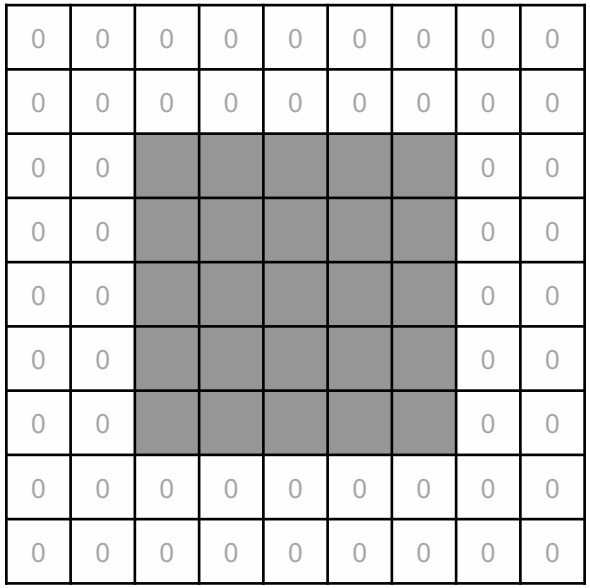
[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, **pad 2**) → ReLU → Max Pooling(3x3, stride 2)



5 X 5,
padding 0



5 X 5,
padding 2

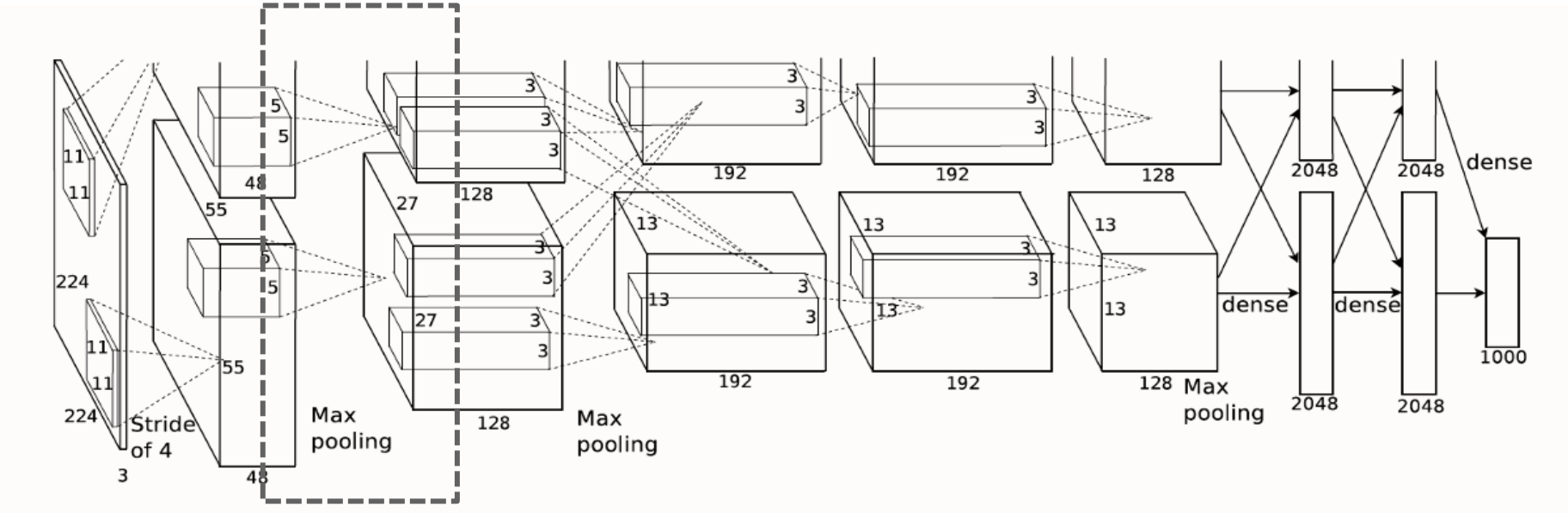
CNN-Architecture: AlexNet Layer2 (Conv)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: **Conv(5x5x256, stride 1, pad 2)** → ReLU → Max Pooling(3x3, stride 2)

output: (31 – 5) / 1 + 1 = **27 X 27 X 256**



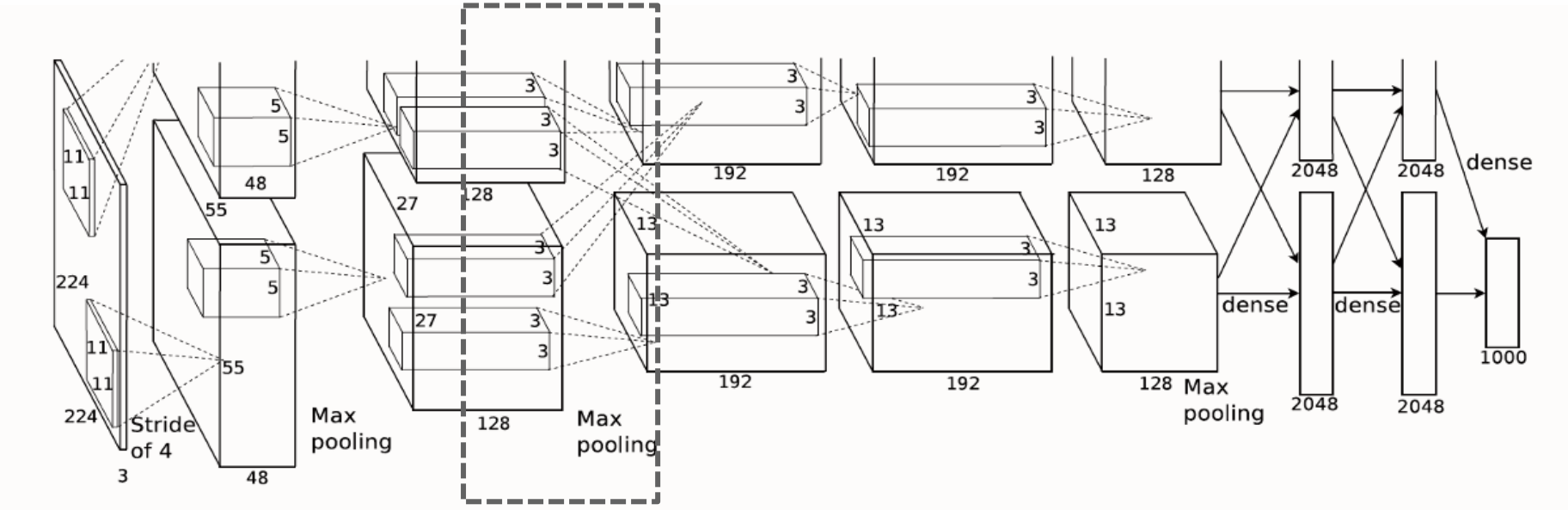
CNN-Architecture: AlexNet Layer2 (MaxPool)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → **Max Pooling(3x3, stride 2)**

output: (31 – 5) / 1 + 1 = **27 X 27 X 256**
output: (27 – 3) / 2 + 1 = **13 X 13 X 256**

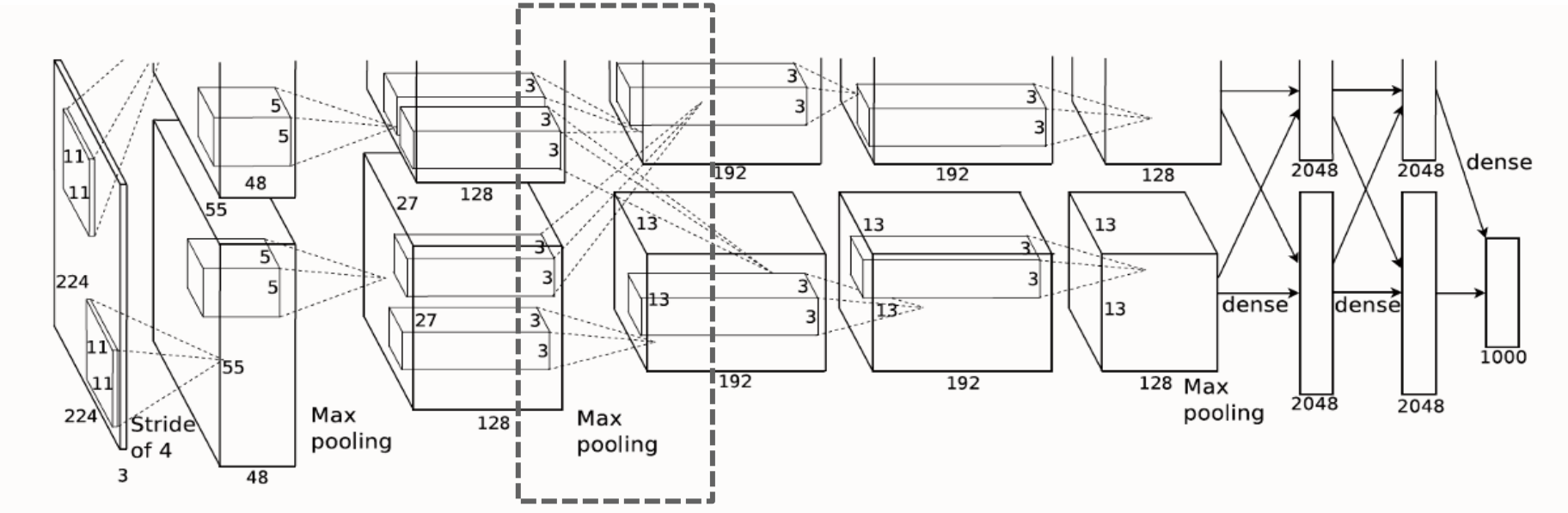


CNN-Architecture: AlexNet Layer3

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv → ReLU



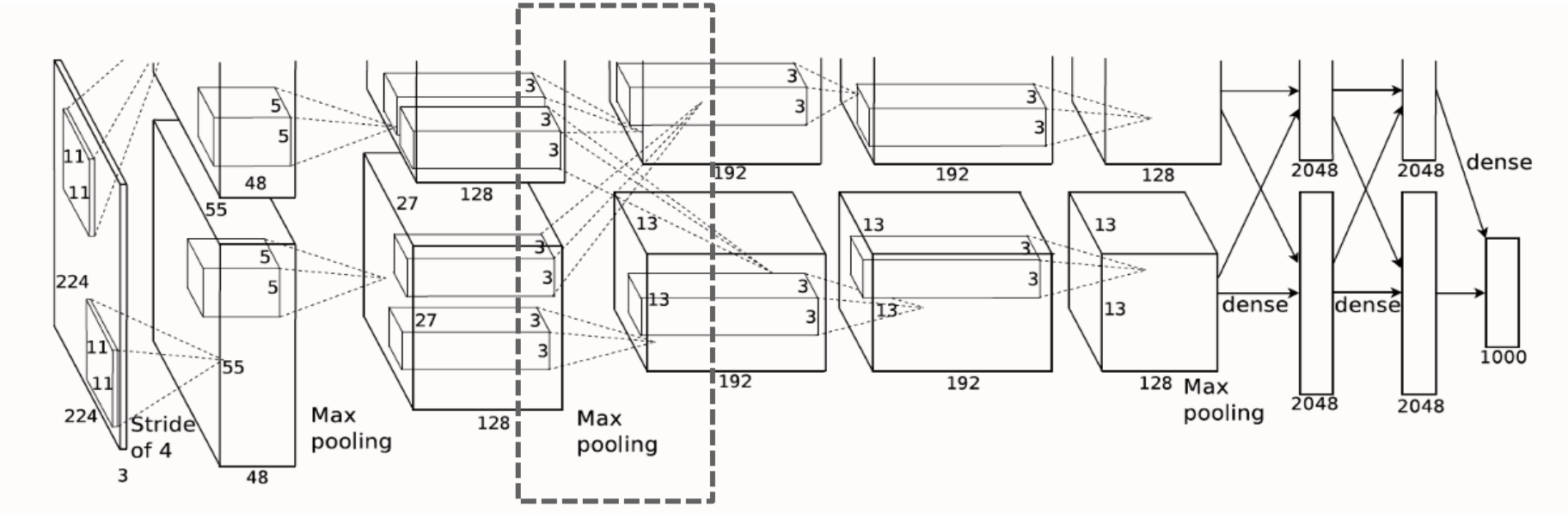
CNN-Architecture: AlexNet Layer3 (Conv)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: **Conv(3x3x384, stride 1, pad 1)** → ReLU

output: (15 – 3) / 1 + 1 = **13 X 13 X 384**

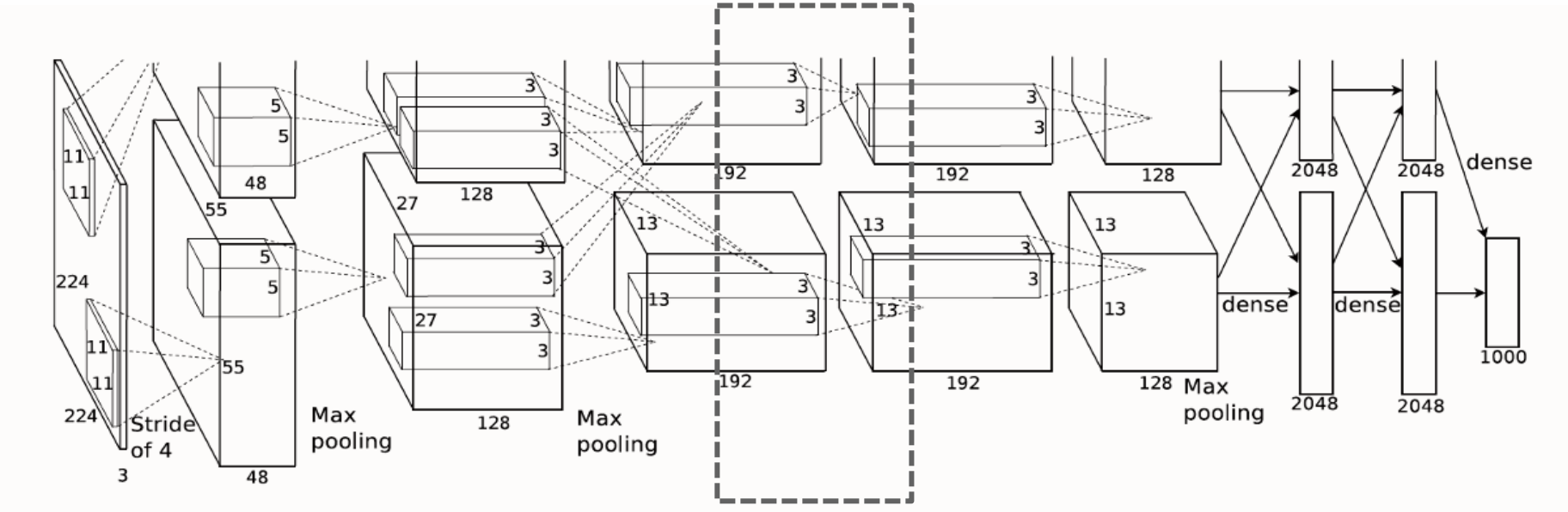


CNN-Architecture: AlexNet Layer4

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU



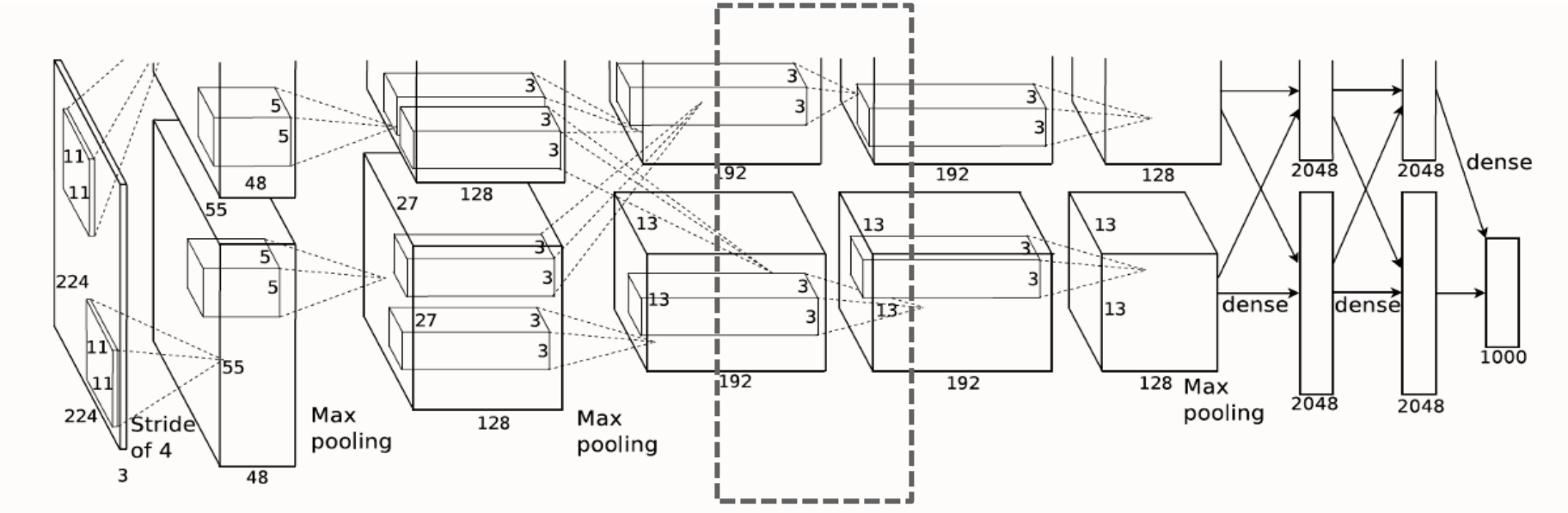
CNN-Architecture: AlexNet Layer4 (Conv)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: **Conv(3x3x384, stride 1, pad 1)** → ReLU

output: (15 – 3) / 1 + 1 = **13 X 13 X 384**

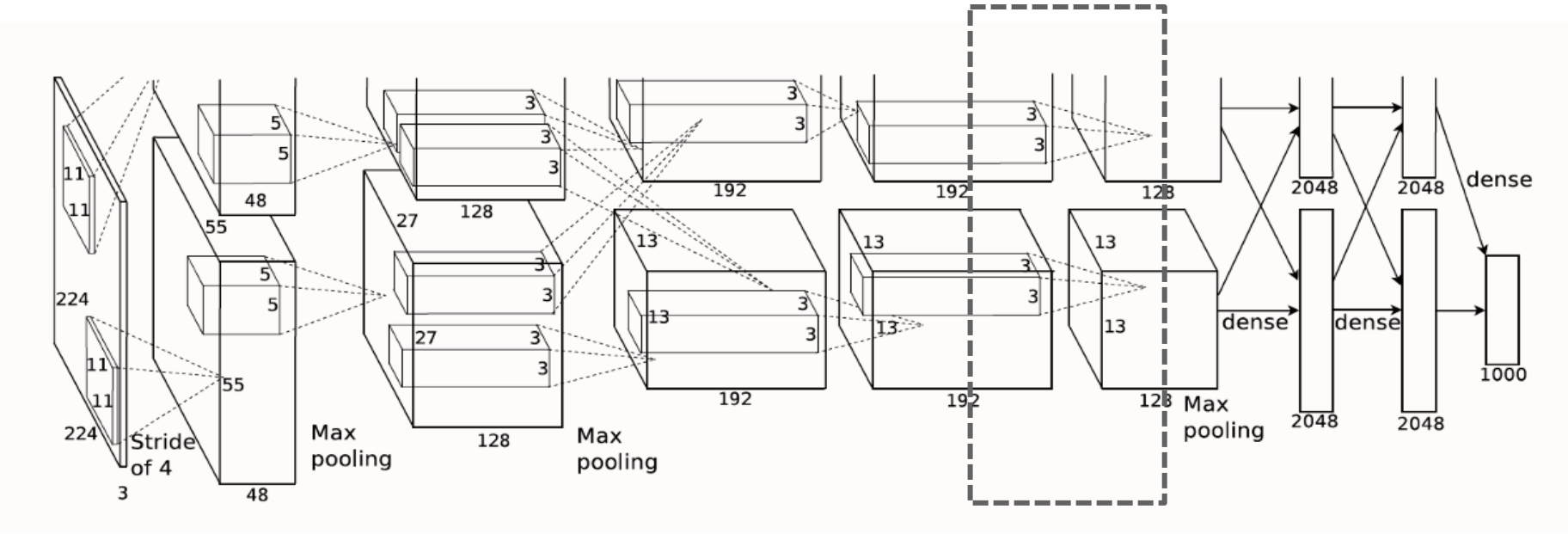


CNN-Architecture: AlexNet Layer5

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: Conv(3x3x256, stride 1, pad 1) → ReLU → Max Pooling(3x3, stride 2)



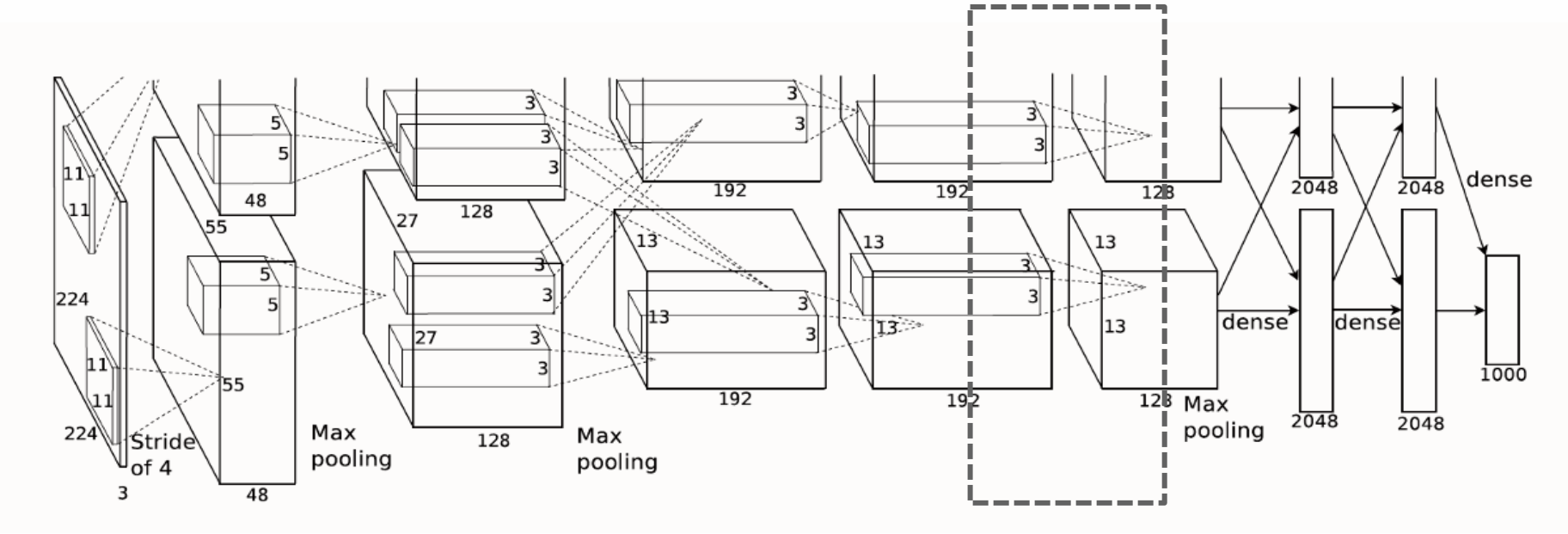
CNN-Architecture: AlexNet Layer5 (Conv)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: **Conv(3x3x256, stride 1, pad 1)** → ReLU → Max Pooling(3x3, stride 2)

output: (15 – 3) / 1 + 1 = **13 X 13 X 256**



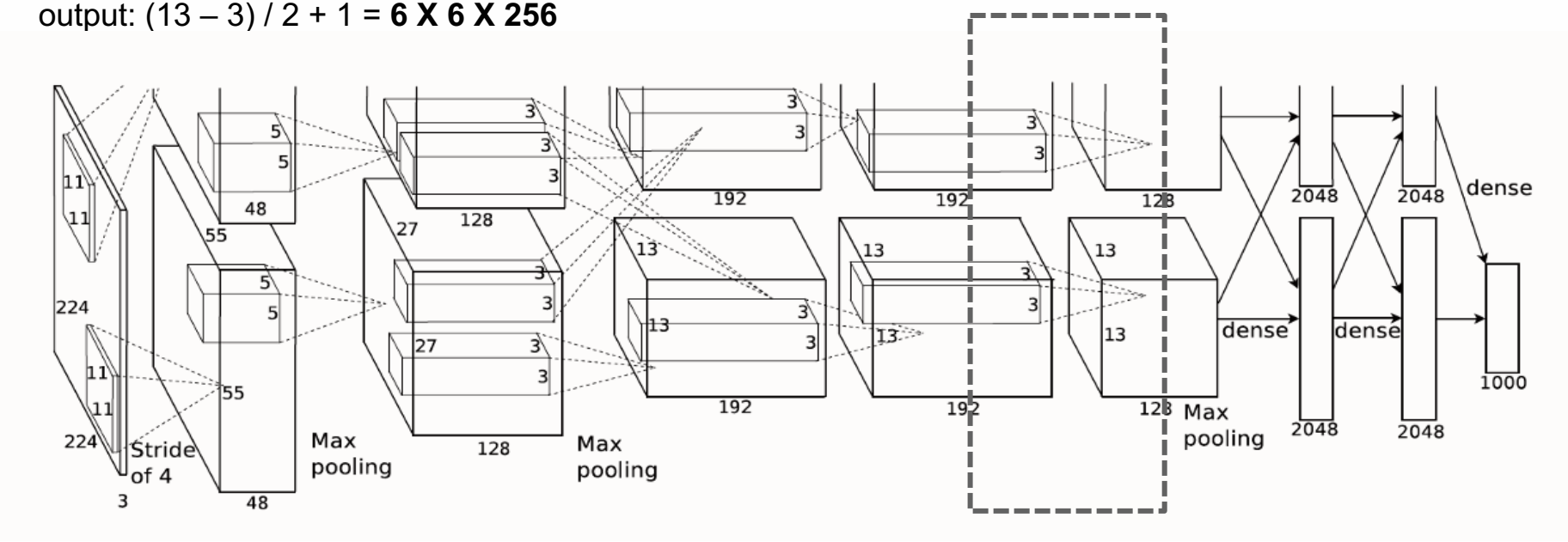
CNN-Architecture: AlexNet Layer5 (MaxPool)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: Conv(3x3x256, stride 1, pad 1) → ReLU → **Max Pooling(3x3, stride 2)**

output: (15 – 3) / 1 + 1 = **13 X 13 X 256**
output: (13 – 3) / 2 + 1 = **6 X 6 X 256**

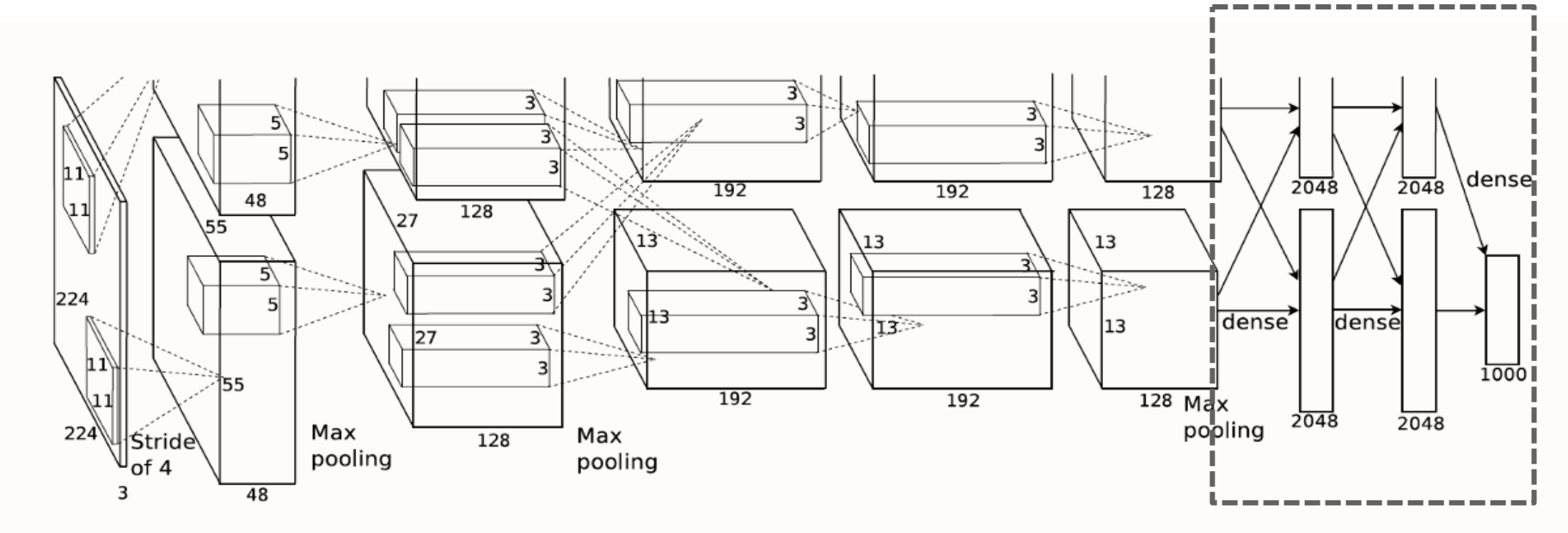


CNN-Architecture: AlexNet Layer6(FC)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: Conv(3x3x256, stride 1, pad 1) → ReLU → Max Pooling(3x3, stride 2): 6 X 6 X 256
Layer6: FC → ReLU → FC → ReLU → FC



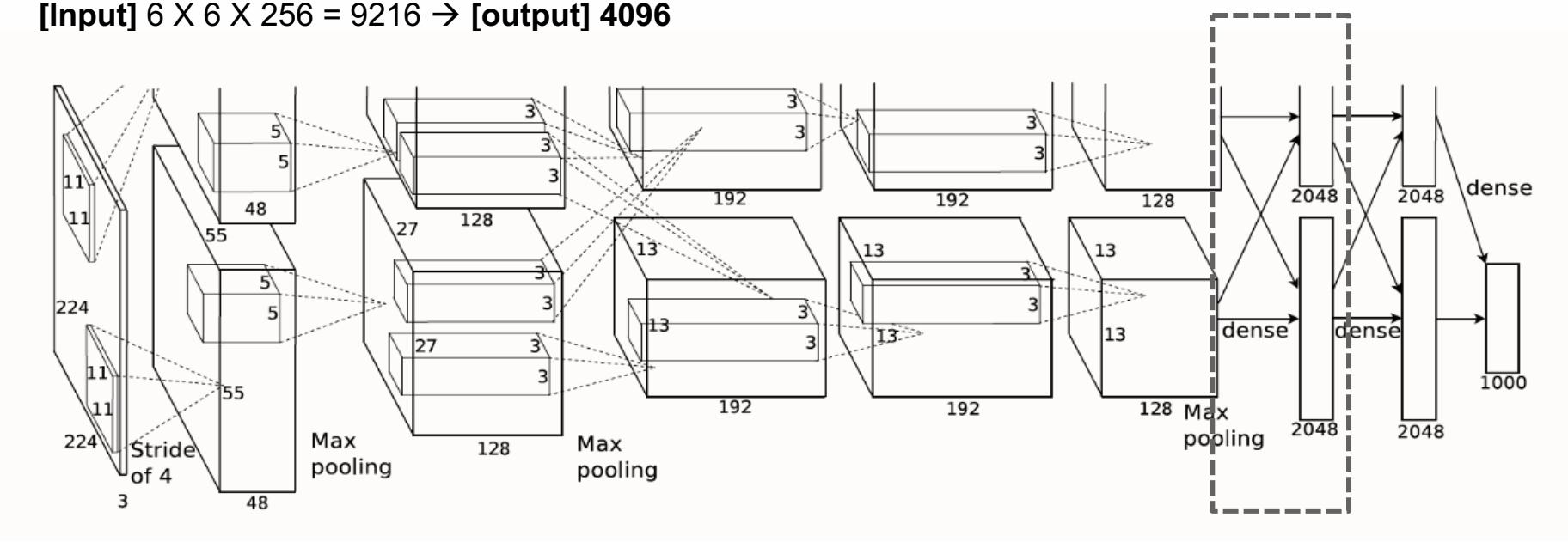
CNN-Architecture: AlexNet Layer6(FC)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: Conv(3x3x256, stride 1, pad 1) → ReLU → Max Pooling(3x3, stride 2): 6 X 6 X 256
Layer6: **FC** → ReLU → FC → ReLU → FC

[Input] 6 X 6 X 256 = 9216 → **[output] 4096**



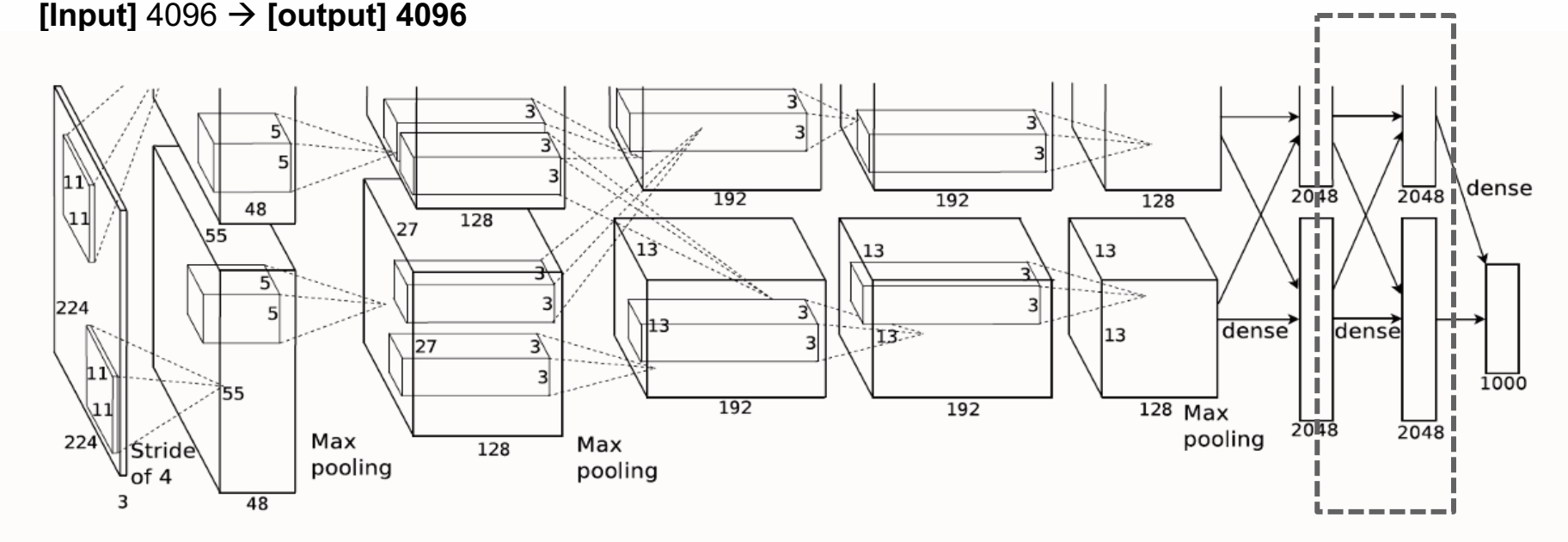
CNN-Architecture: AlexNet Layer6(FC)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: Conv(3x3x256, stride 1, pad 1) → ReLU → Max Pooling(3x3, stride 2): 6 X 6 X 256
Layer6: FC → ReLU → **FC** → ReLU → FC

[Input] 4096 → **[output] 4096**



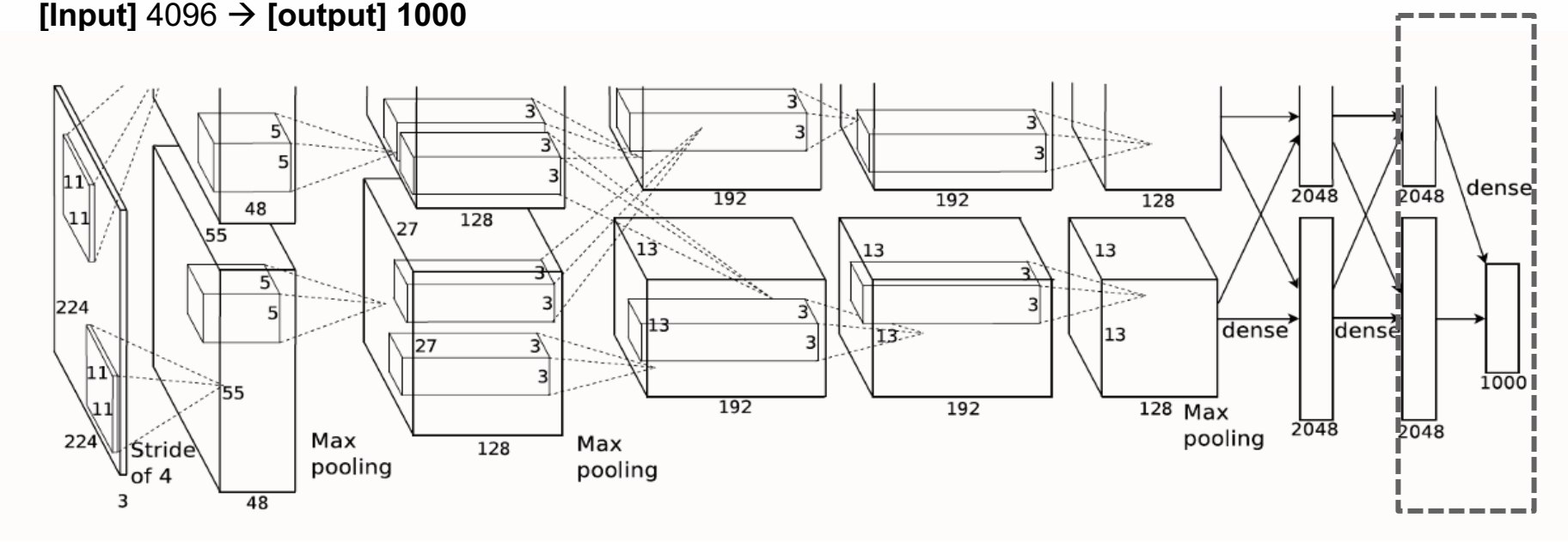
CNN-Architecture: AlexNet Layer6(FC)

[Krizhevsky et al. 2012]

Input: 227 X 227 X 3(rgb) images

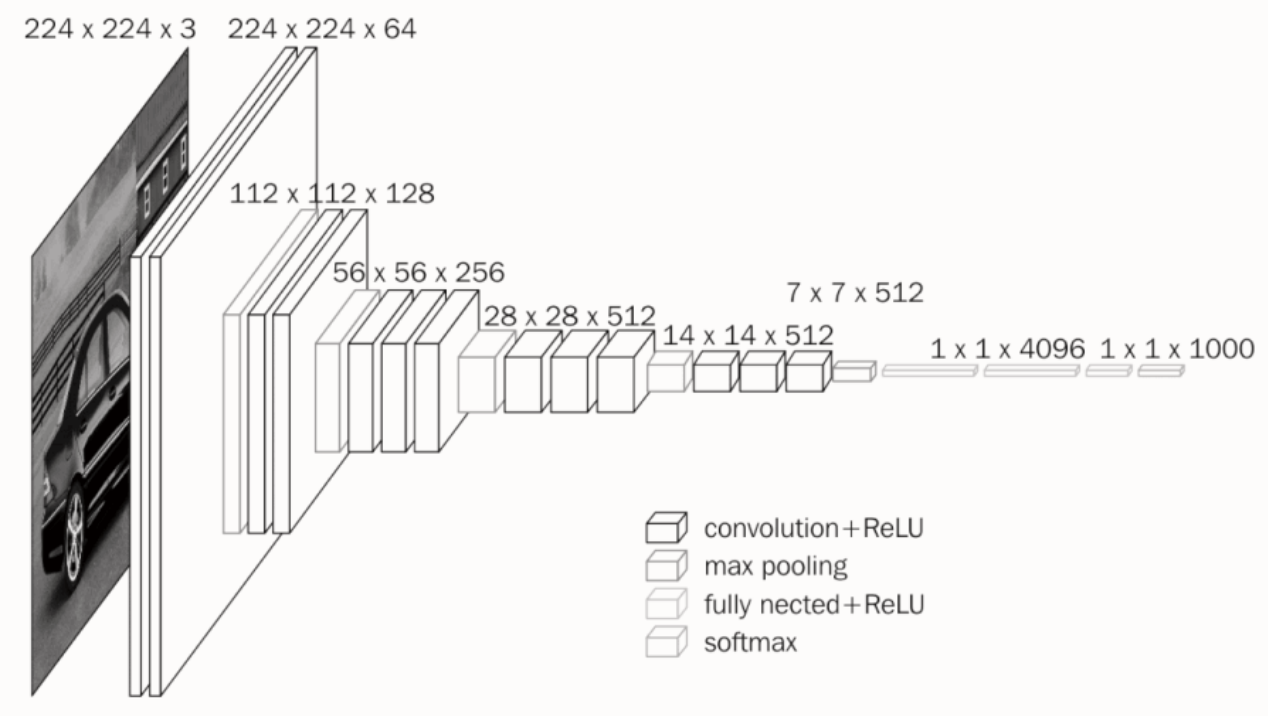
Layer1: Conv(11x11x96, stride 4) → ReLU → Max Pooling(3x3, stride 2): 27 X 27 X 96
Layer2: Conv(5x5x256, stride 1, pad 2) → ReLU → Max Pooling(3x3, stride 2): 13 X 13 X 256
Layer3: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer4: Conv(3x3x384, stride 1, pad 1) → ReLU: 13 X 13 X 384
Layer5: Conv(3x3x256, stride 1, pad 1) → ReLU → Max Pooling(3x3, stride 2): 6 X 6 X 256
Layer6: FC → ReLU → FC → ReLU → **FC**

[Input] 4096 → **[output] 1000**



CNN-Architecture: VGG

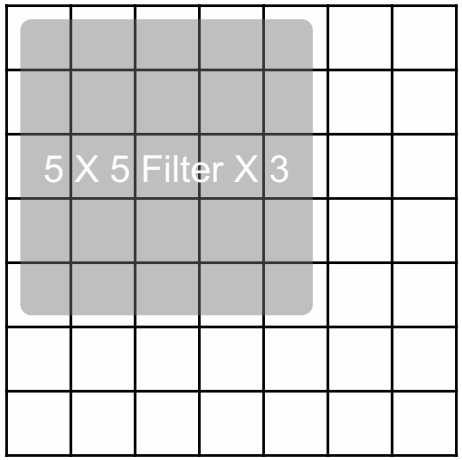
[Simonyan et al. 2014]

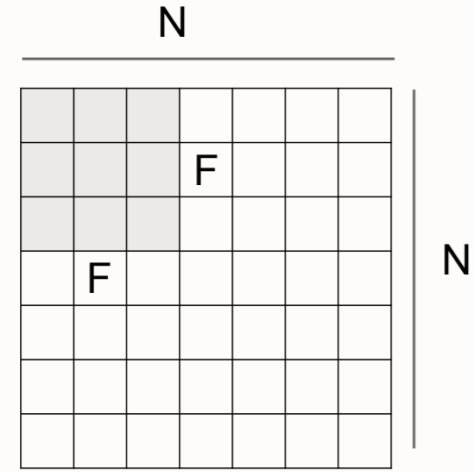


Simonyan, Karen, and Andrew Zisserman. **"Very deep convolutional networks for large-scale image recognition."** arXiv preprint arXiv:1409.1556 (2014).

CNN-Architecture: VGG (5 x 5 vs 3 x 3 Filter)

[Simonyan et al. 2014]





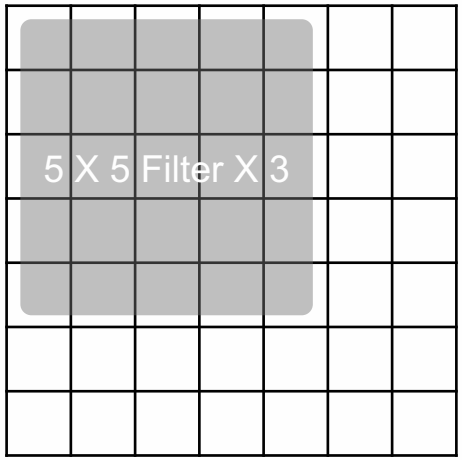
Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

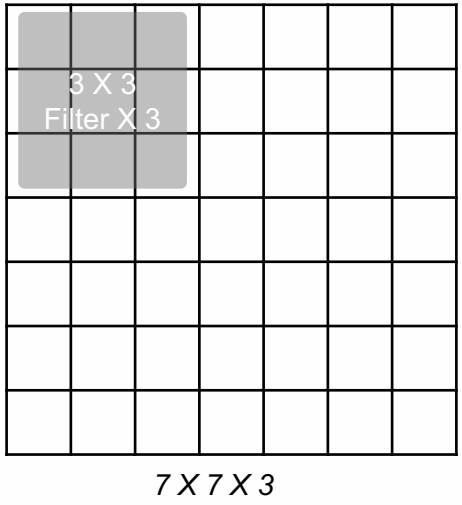
output: (7 – 5) / 1 + 1 = **3 X 3 X 3**

CNN-Architecture: VGG (5 x 5 vs 3 x 3 Filter)

[Simonyan et al. 2014]



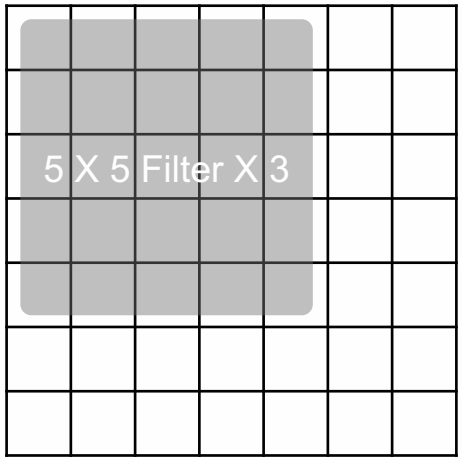
output: (7 – 5) / 1 + 1 = **3 X 3 X 3**



output: (7 – 3) / 1 + 1 = **5 X 5 X 3**

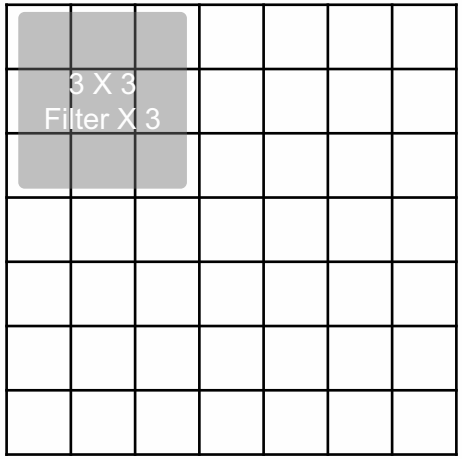
CNN-Architecture: VGG (5 x 5 vs 3 x 3 Filter)

[Simonyan et al. 2014]



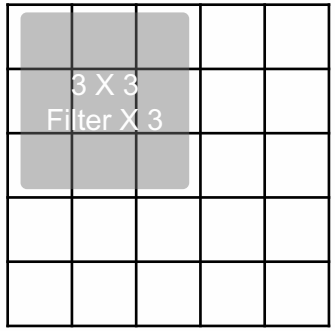
7 X 7 X 3

output: (7 – 5) / 1 + 1 = **3 X 3 X 3**



7 X 7 X 3

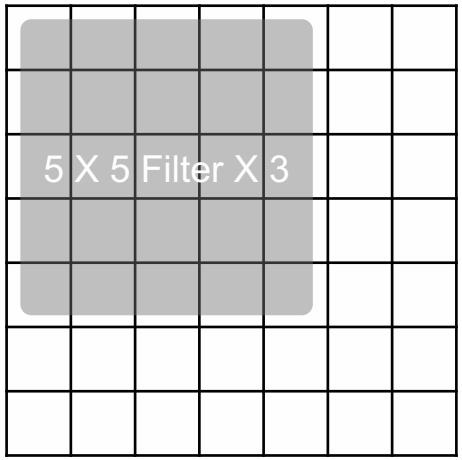
output: (7 – 3) / 1 + 1 = **5 X 5 X 3**



output: (5 – 3) / 1 + 1 = **3 X 3 X 3**

CNN-Architecture: VGG (5 x 5 vs 3 x 3 Filter)

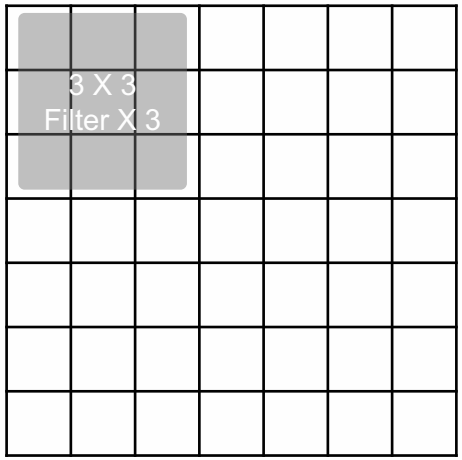
[Simonyan et al. 2014] comparison of Parameters



7 X 7 X 3

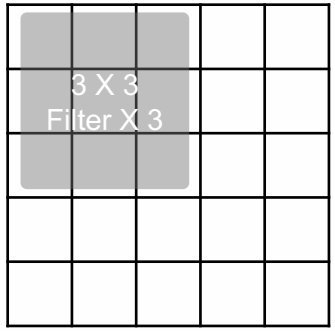
output: (7 – 5) / 1 + 1 = **3 X 3 X 3**

params: 5 x 5 x 3 x 3 = **225**



7 X 7 X 3

output: (7 – 3) / 1 + 1 = **5 X 5 X 3**

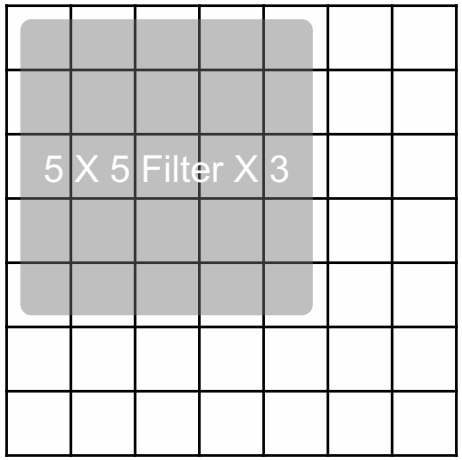


output: (5 – 3) / 1 + 1 = **3 X 3 X 3**

CNN-Architecture: VGG (5 x 5 vs 3 x 3 Filter)

[Simonyan et al. 2014]

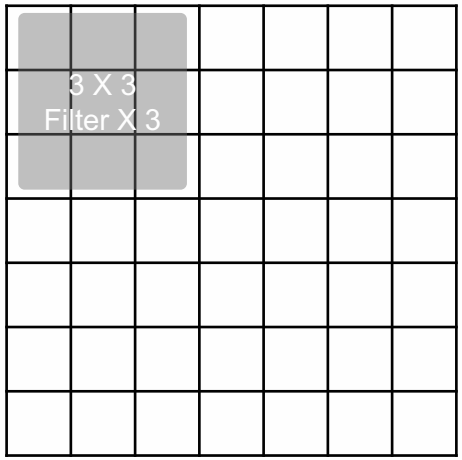
comparison of Parameters



7 X 7 X 3

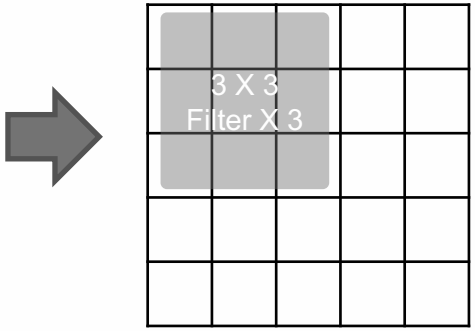
output: (7 – 5) / 1 + 1 = **3 X 3 X 3**

params: 5 x 5 x 3 x 3 = **225**



7 X 7 X 3

output: (7 – 3) / 1 + 1 = **5 X 5 X 3**



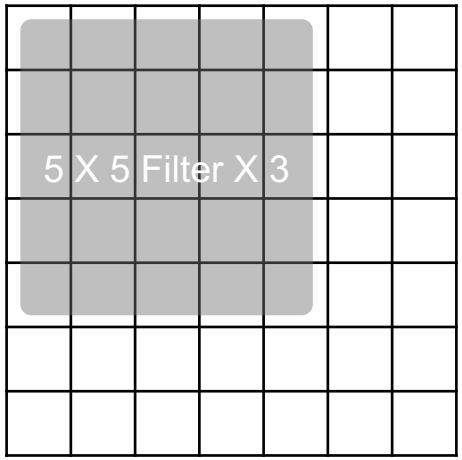
output: (5 – 3) / 1 + 1 = **3 X 3 X 3**

params: (3 x 3 x 3 x 3)+ (3 x 3 x 3 x 3) = **162**

CNN-Architecture: VGG (5 x 5 vs 3 x 3 Filter)

[Simonyan et al. 2014]

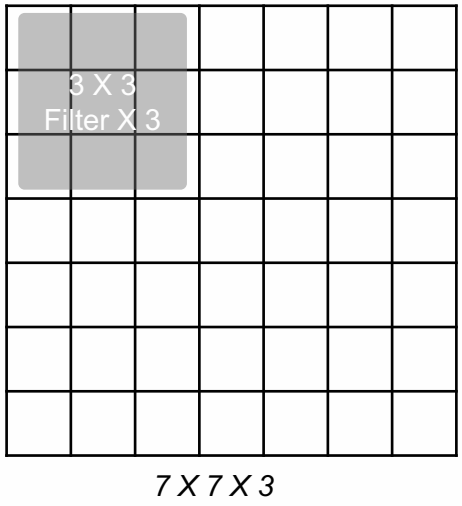
comparison of Parameters



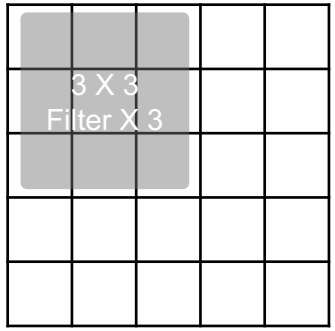
output: (7 – 5) / 1 + 1 = **3 X 3 X 3**

params: 5 x 5 x 3 x 3 = **225**

output = params



output: (7 – 3) / 1 + 1 = **5 X 5 X 3**



output: (5 – 3) / 1 + 1 = **3 X 3 X 3**

params: (3 x 3 x 3 x 3)+ (3 x 3 x 3 x 3) = **162**

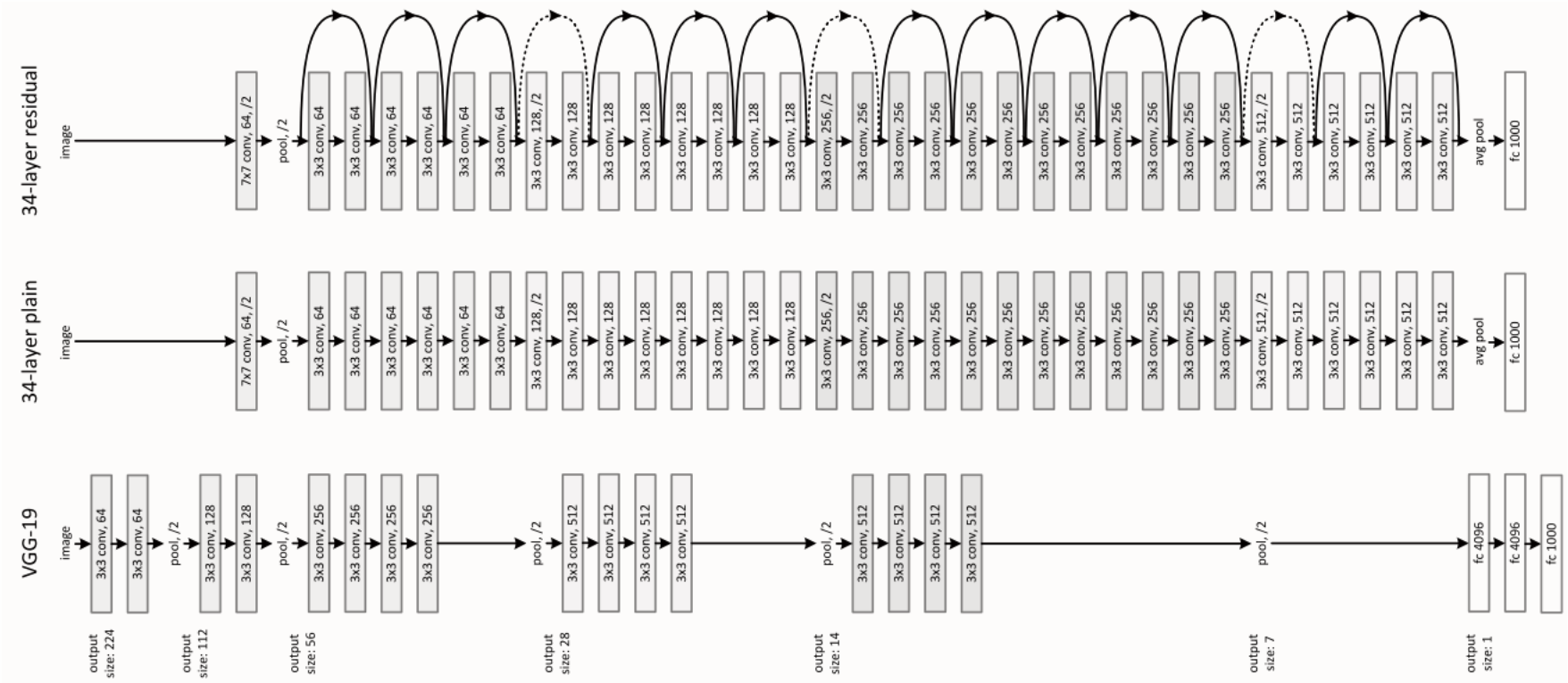
CNN-Architecture: VGG vs AlexNet

[Simonyan et al. 2014]

Model	Parameters	Top-1 error
AlexNet	62,378,344	42.90
VGG16	138,357,544	27.00
VGG19	143,667,240	27.30

CNN-Architecture: ResNet

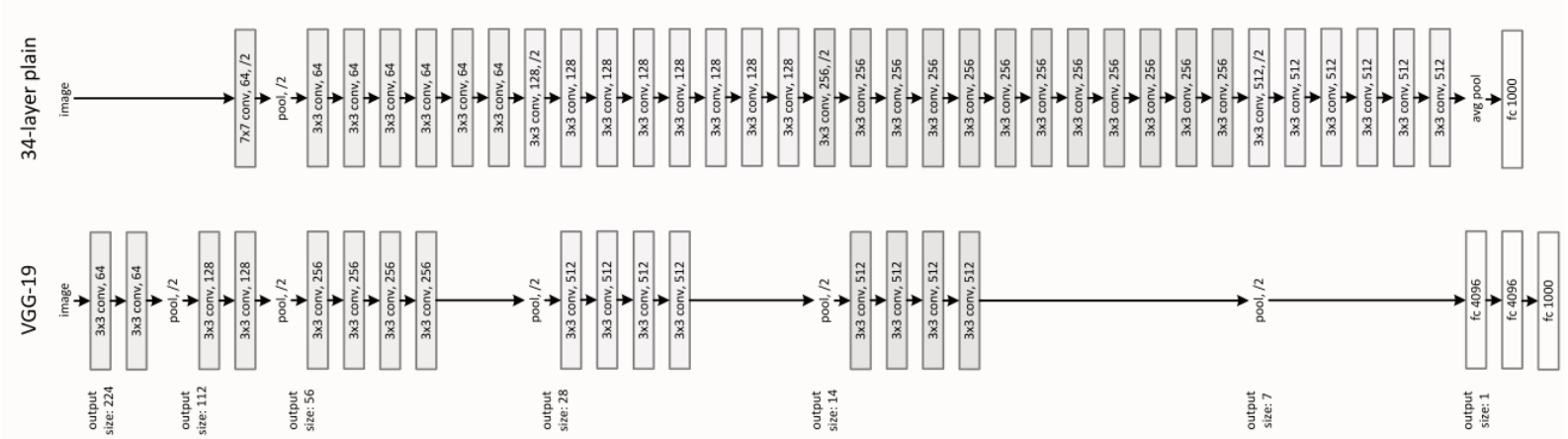
[He Kaiming et al. 2015]



He, Kaiming, et al. **"Deep residual learning for image recognition."** Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

CNN-Architecture: ResNet (vanishing gradient problem)

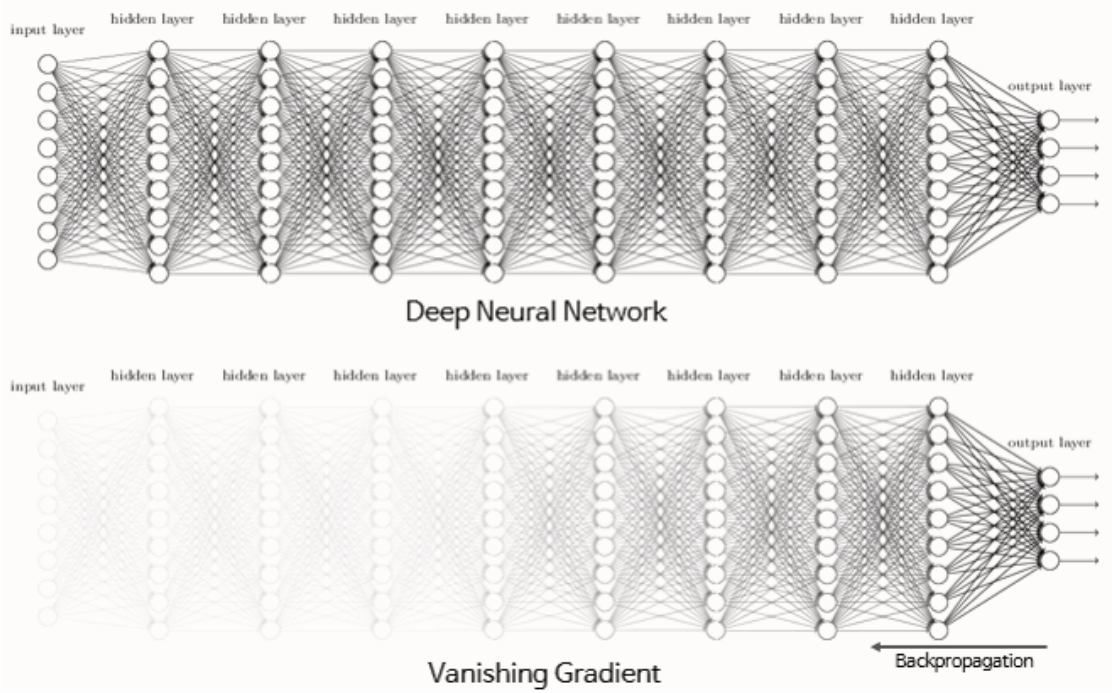
[He Kaiming et al. 2015]

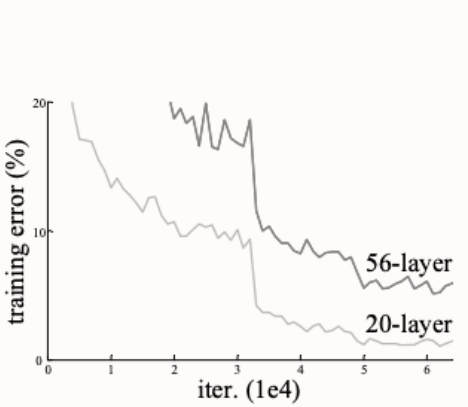


Model	Top-1 error	Top-5 error
VGG16	27.00	8.80
VGG19	27.30	9.00
Plain 34	28.54	10.02

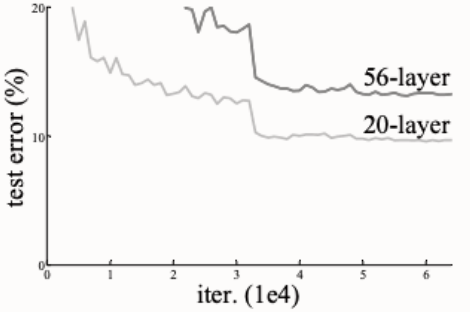
CNN-Architecture: ResNet (vanishing gradient problem)

[He Kaiming et al. 2015]





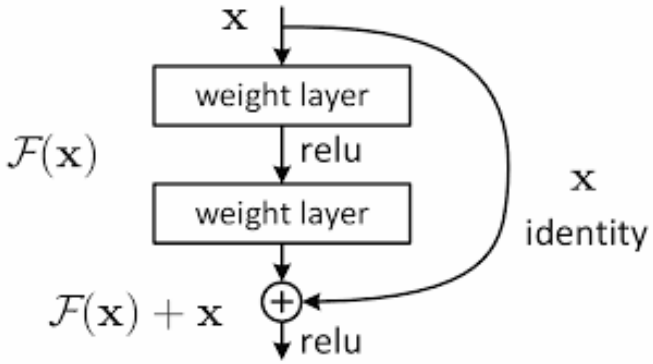
train error

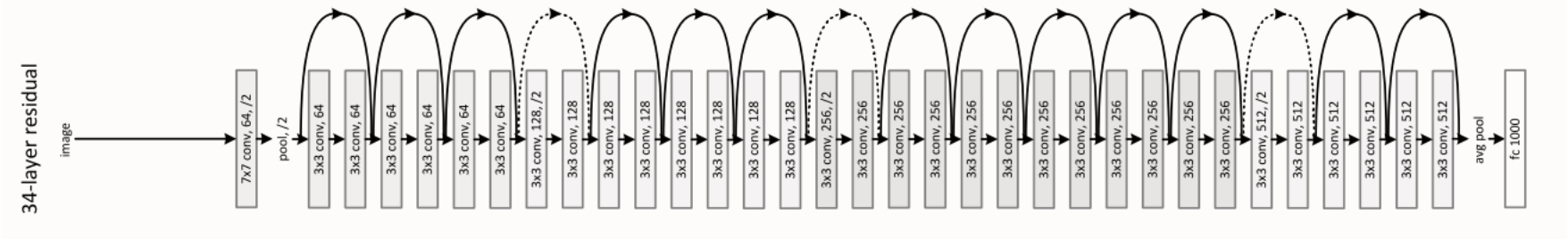


test error

CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]

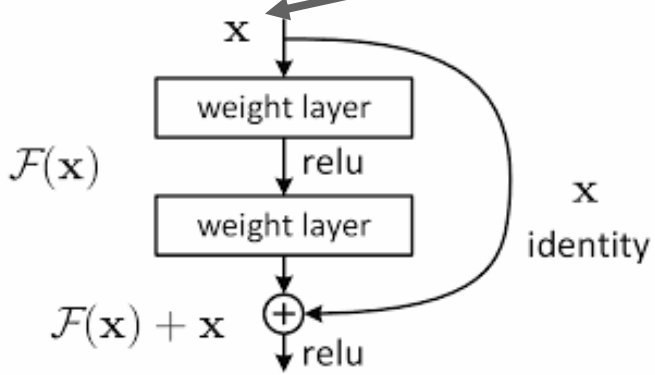


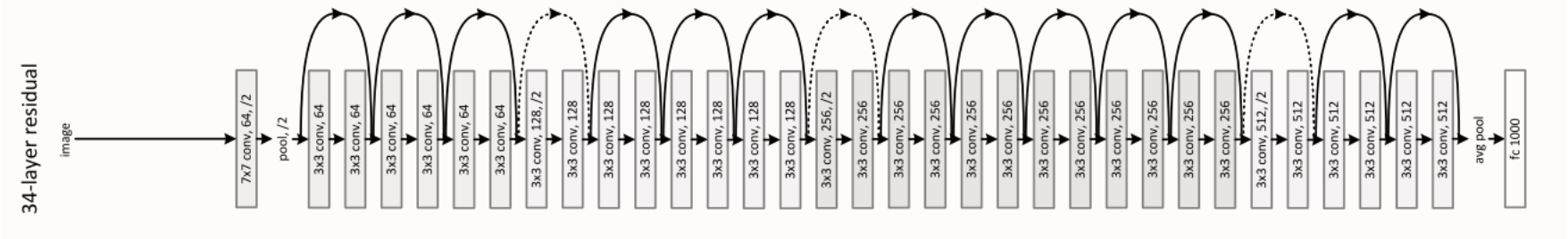


CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]

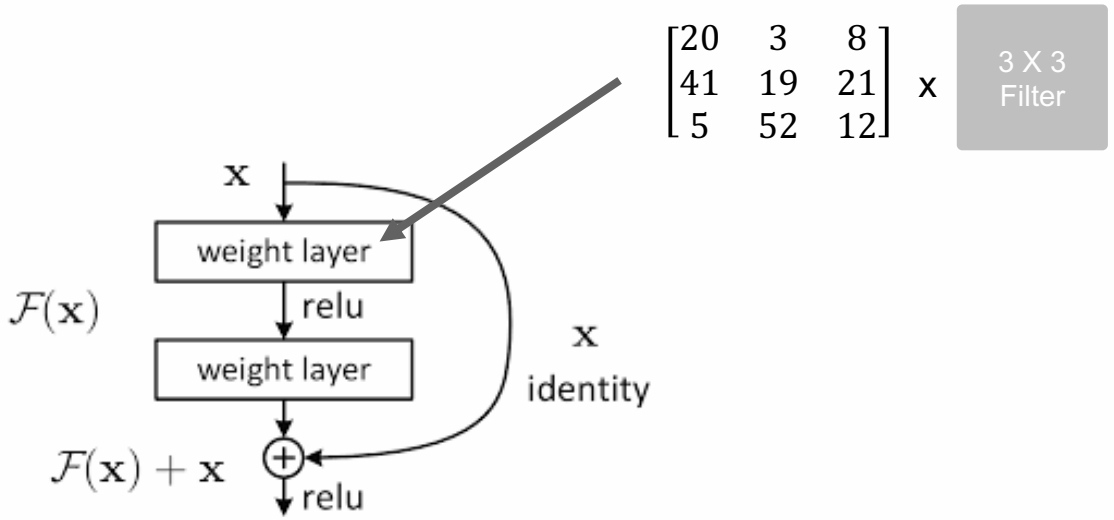
$$\begin{bmatrix} 20 & 3 & 8 \\ 41 & 19 & 21 \\ 5 & 52 & 12 \end{bmatrix}$$

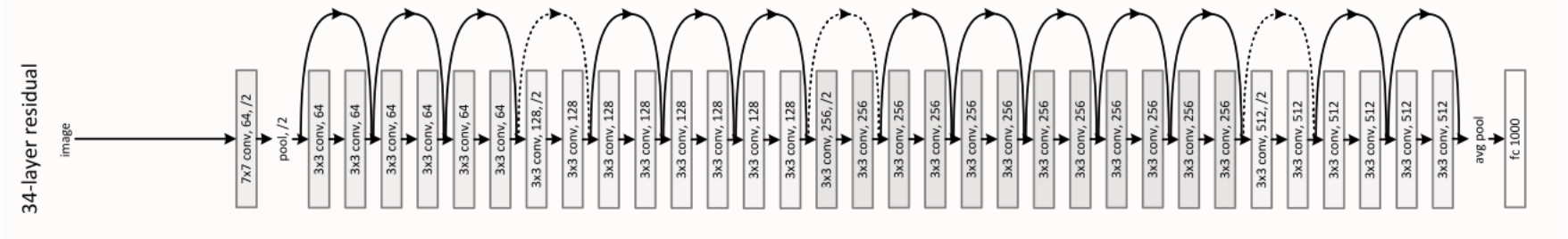




CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]

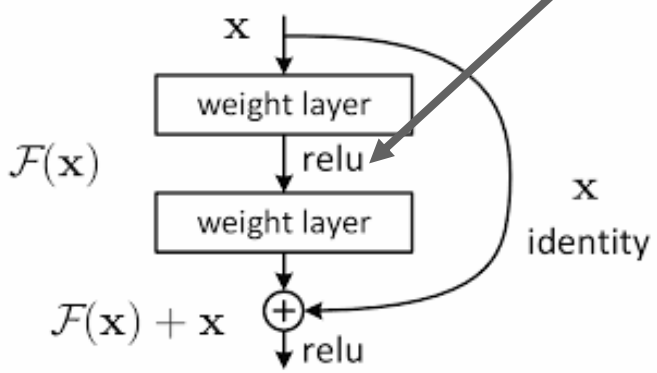


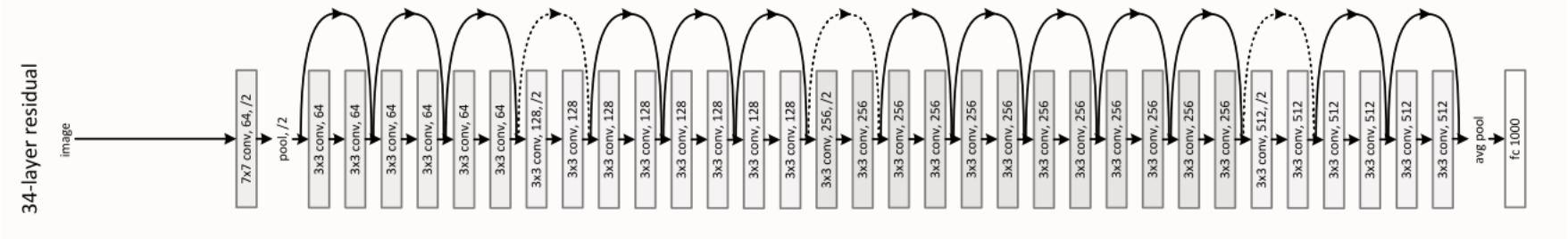


CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]

$$ReLU\left(\begin{bmatrix}20 & 3 & 8\\ 41 & 19 & 21\\ 5 & 52 & 12\end{bmatrix} \text{x } \boxed{\text{3 X 3 Filter}}\right)$$

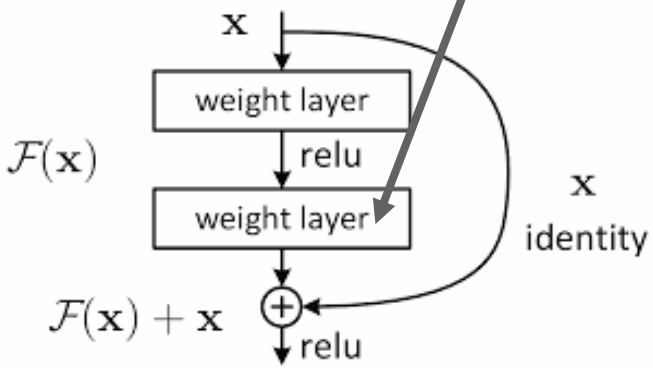


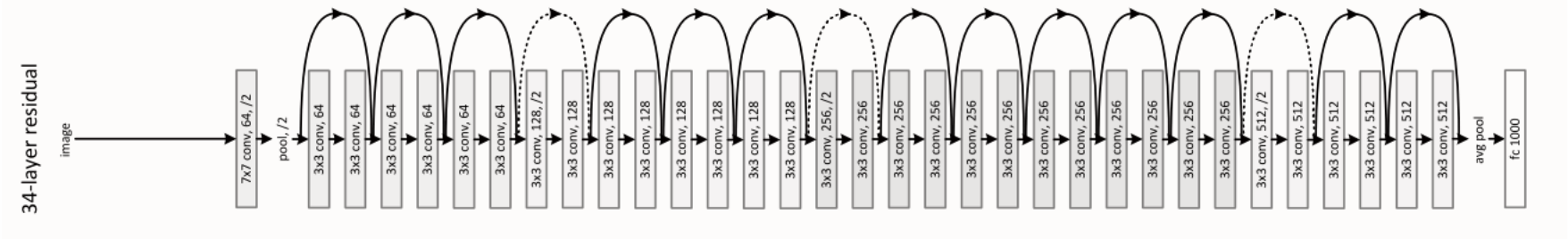


CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]

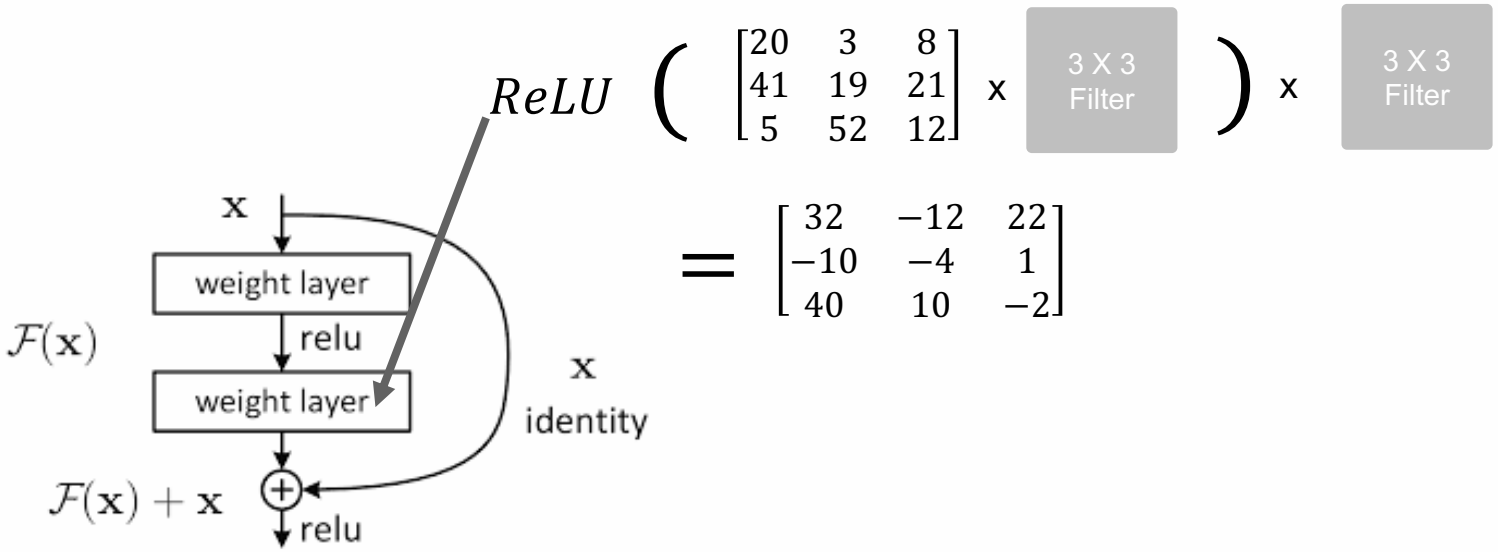
$$ReLU\left(\begin{bmatrix}20 & 3 & 8\\ 41 & 19 & 21\\ 5 & 52 & 12\end{bmatrix} \text{x } \boxed{\text{3 X 3 Filter}}\right) \text{x } \boxed{\text{3 X 3 Filter}}$$

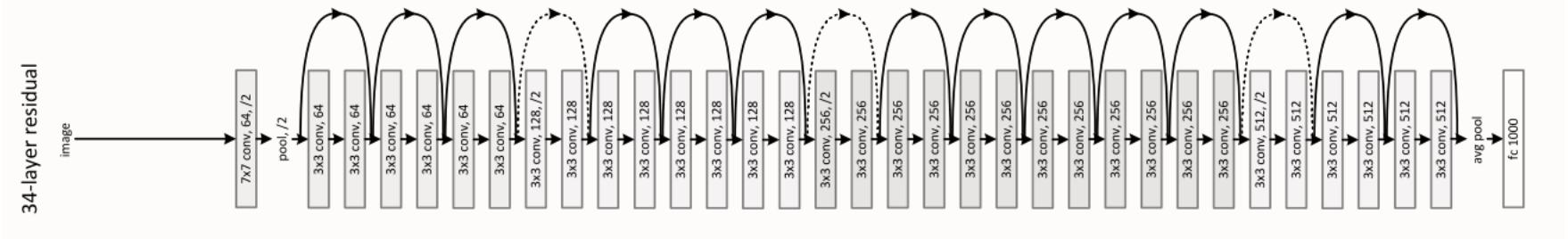




CNN-Architecture: ResNet (Skip Connection)

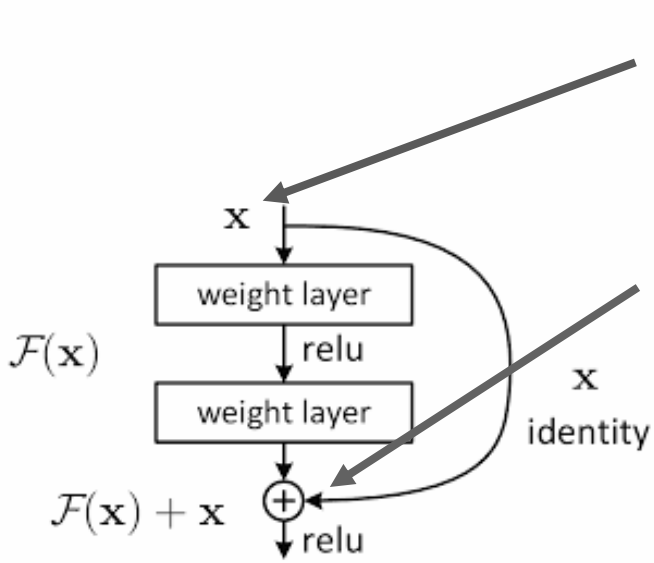
[He Kaiming et al. 2015]





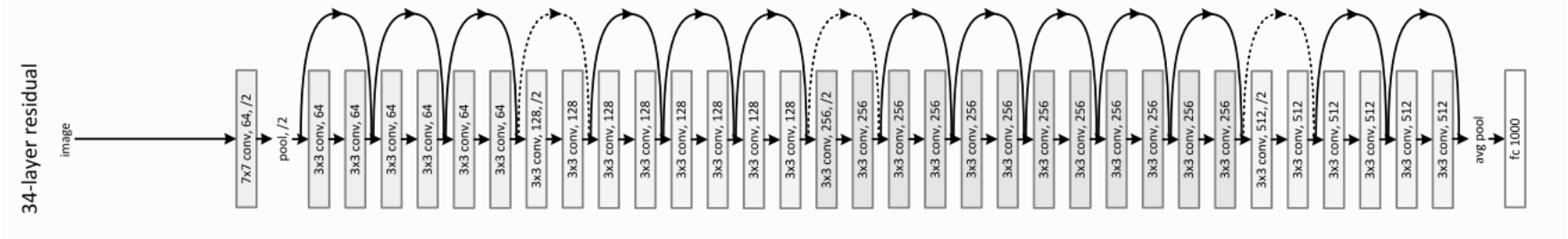
CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]



$$\begin{bmatrix} 20 & 3 & 8 \\ 41 & 19 & 21 \\ 5 & 52 & 12 \end{bmatrix}$$

$$\begin{bmatrix} 32 & -12 & 22 \\ -10 & -41 & 1 \\ -30 & 10 & -2 \end{bmatrix} + \begin{bmatrix} 20 & 3 & 8 \\ 41 & 19 & 21 \\ 5 & 52 & 12 \end{bmatrix} = \begin{bmatrix} 52 & -9 & 30 \\ 31 & -22 & 22 \\ -25 & 62 & 10 \end{bmatrix}$$

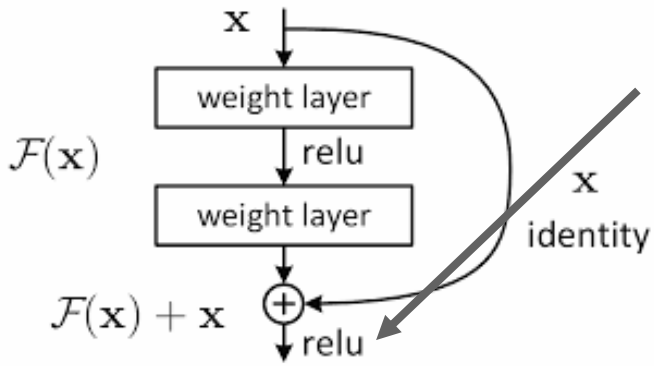


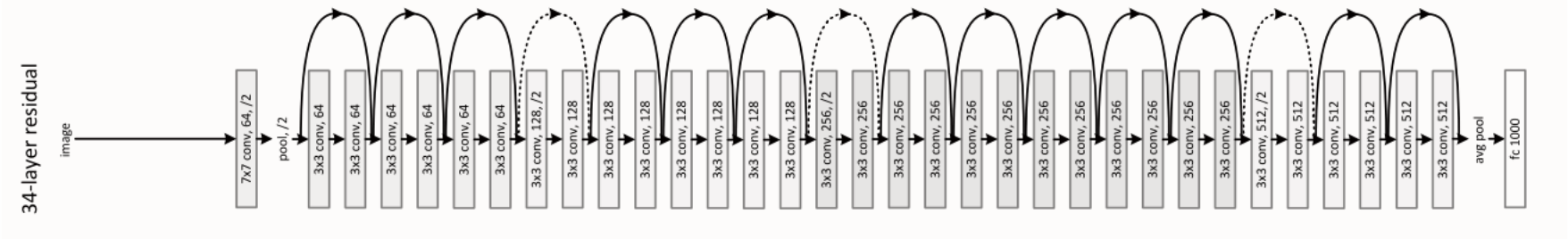
CNN-Architecture: ResNet (Skip Connection)

[He Kaiming et al. 2015]

$$\begin{bmatrix} 32 & -12 & 22 \\ -10 & -41 & 1 \\ -30 & 10 & -2 \end{bmatrix} + \begin{bmatrix} 20 & 3 & 8 \\ 41 & 19 & 21 \\ 5 & 52 & 12 \end{bmatrix} = \begin{bmatrix} 52 & -9 & 30 \\ 31 & -22 & 22 \\ -25 & 62 & 10 \end{bmatrix}$$

$$ReLU\left(\begin{bmatrix} 52 & -9 & 30 \\ 31 & -22 & 22 \\ -25 & 62 & 10 \end{bmatrix} \right) = \begin{bmatrix} 52 & 0 & 30 \\ 31 & 0 & 22 \\ 0 & 62 & 10 \end{bmatrix}$$



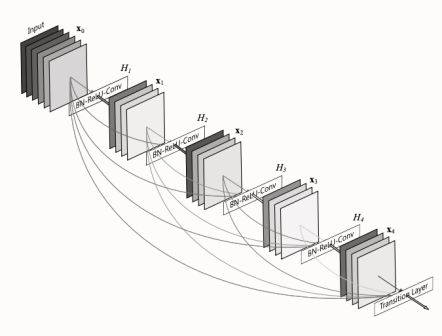


CNN-Architecture: ResNet (Performance)

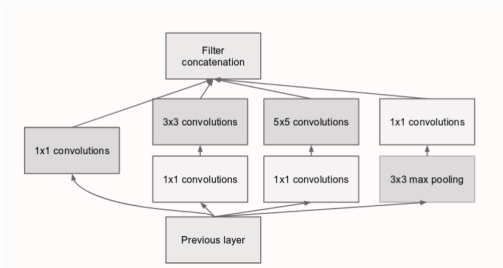
[He Kaiming et al. 2015]

Model	Top-1 error	Top-5 error
VGG16	27.00	8.80
VGG19	27.30	9.00
Plain 34	28.54	10.02
ResNet 34	26.73	8.74
ResNet 50	24.60	7.70
ResNet 101	23.40	7.00

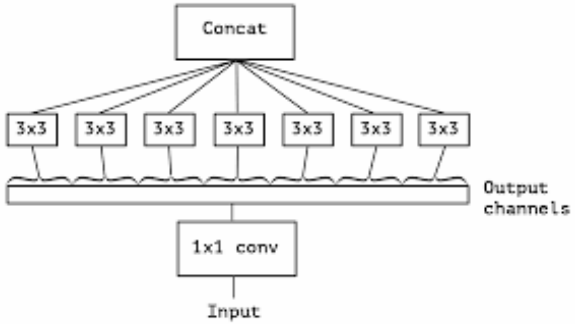
CNN-Architectures



Model: DenseNet
key Idea: Concat All of Layers



Model: Inception(GoogLeNet)
key Idea: 1x1 Convolution (redude params)



Model: xception
key Idea: extreme Inception