EN.601.482/682 Deep Learning

Network Architectures

Mathias Unberath, PhD

Assistant Professor

Dept of Computer Science

Johns Hopkins University

Reminder

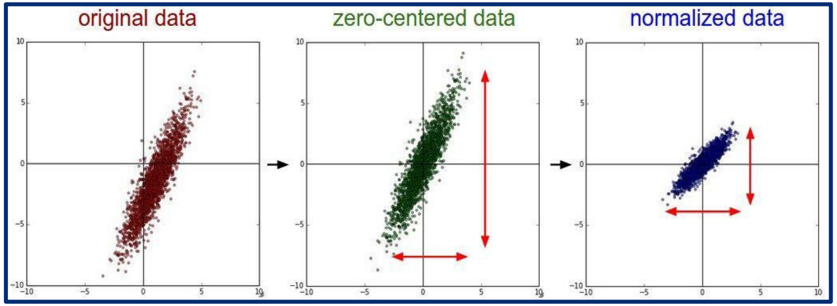
ConvNets

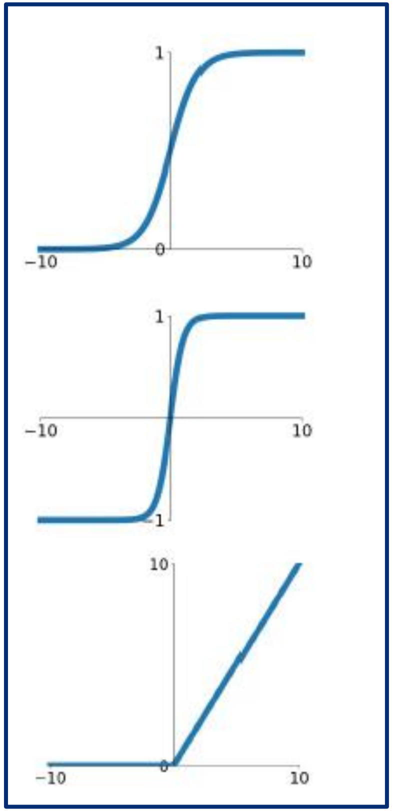
- One-time setup
  - Architecture (**TODAY**)
  - Activation functions (sigmoid, ReLU, …)
  - Regularization (batch norm, dropout)

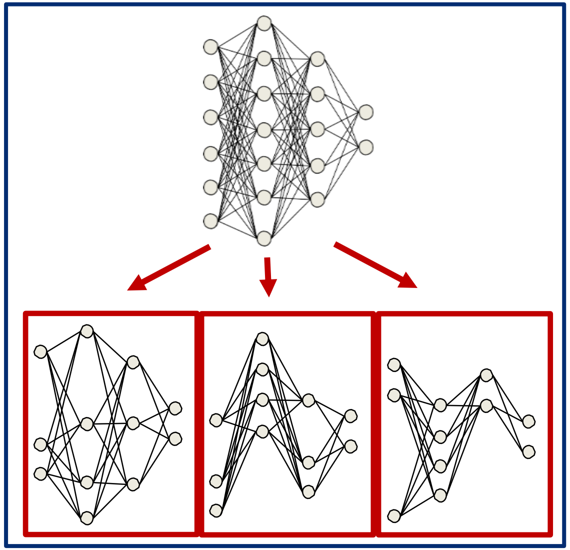
- Training
  - Data collection: Preprocessing, Augmentation
  - Training via SGD (update rules)

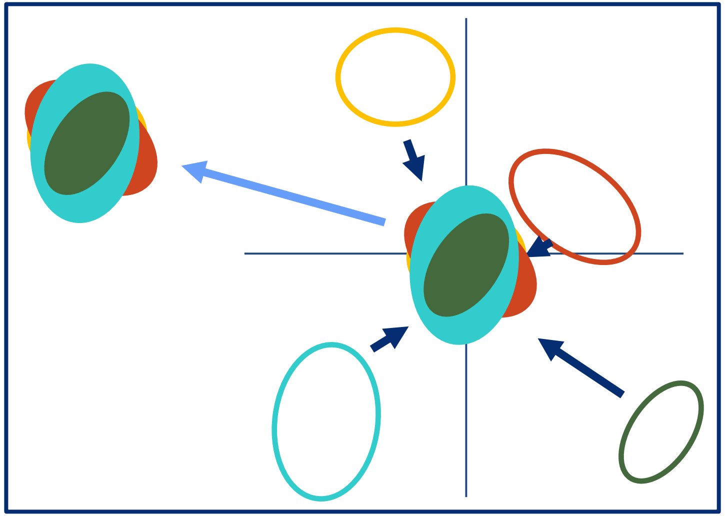
- Transfer learning

Reminder









Reminder

$$g_t = \nabla_W L(W_t)$$

Bias correction

Momentum

$$S_i^{(1)} = \left(\rho_1 S_i^{(1)} + (1-\rho_1)(g_t)_i\right)(1-\rho_1^t)^{(-1)}$$

RMSProp

$$S_i^{(2)} = \left(\rho_2 S_i^{(2)} + (1-\rho_2)(g_t)_i^2\right)(1-\rho_2^t)^{(-1)}$$

$$(\mathrm{d}W_t)_i = \frac{\alpha}{\sqrt{S_i^{(2)}} + \epsilon} S_i^{(1)}$$

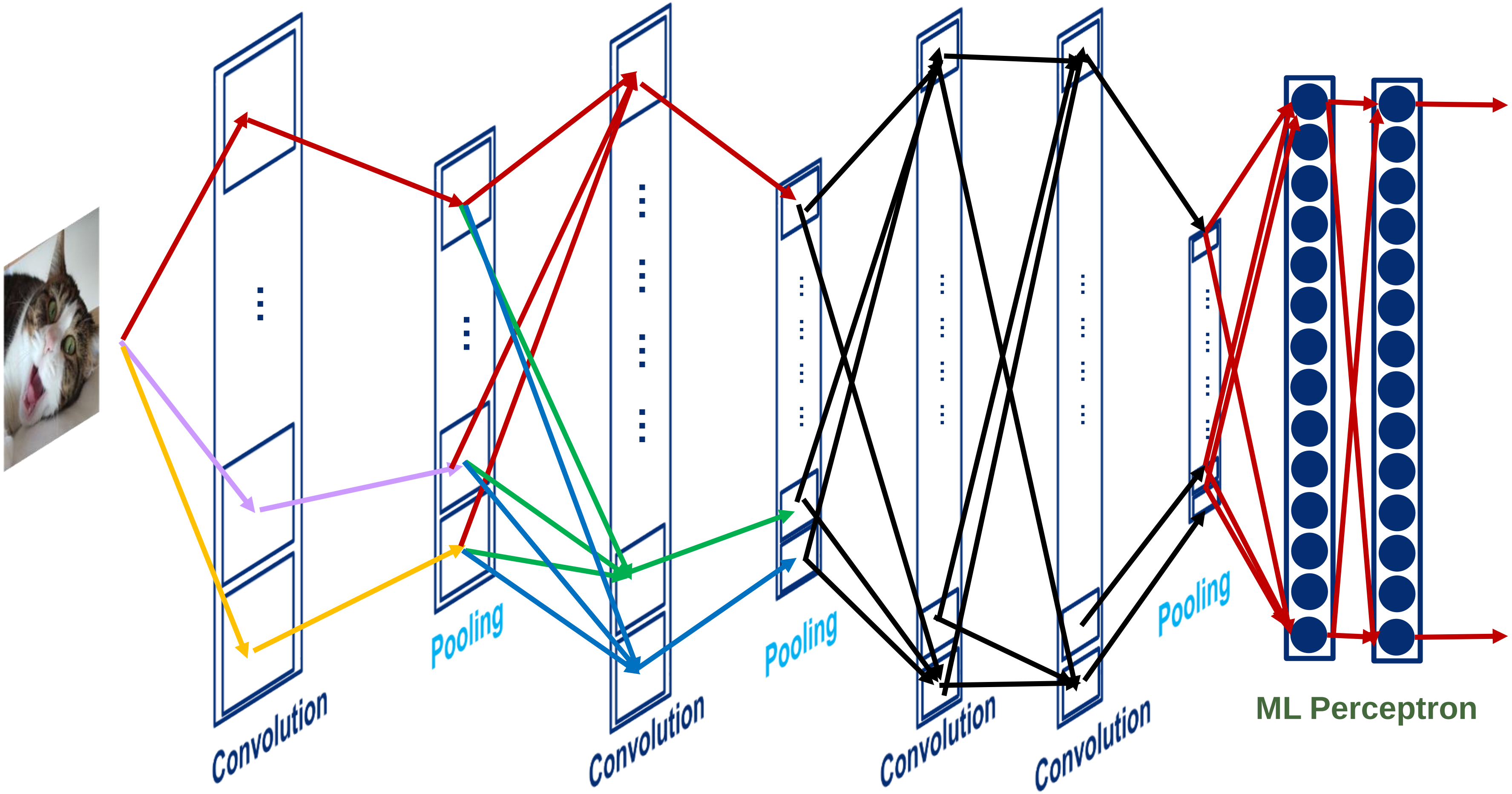
$$W_{t+1} = W_t - \mathrm{d}W_t$$

Adam: Combines SGD + Momentum and RMSProp

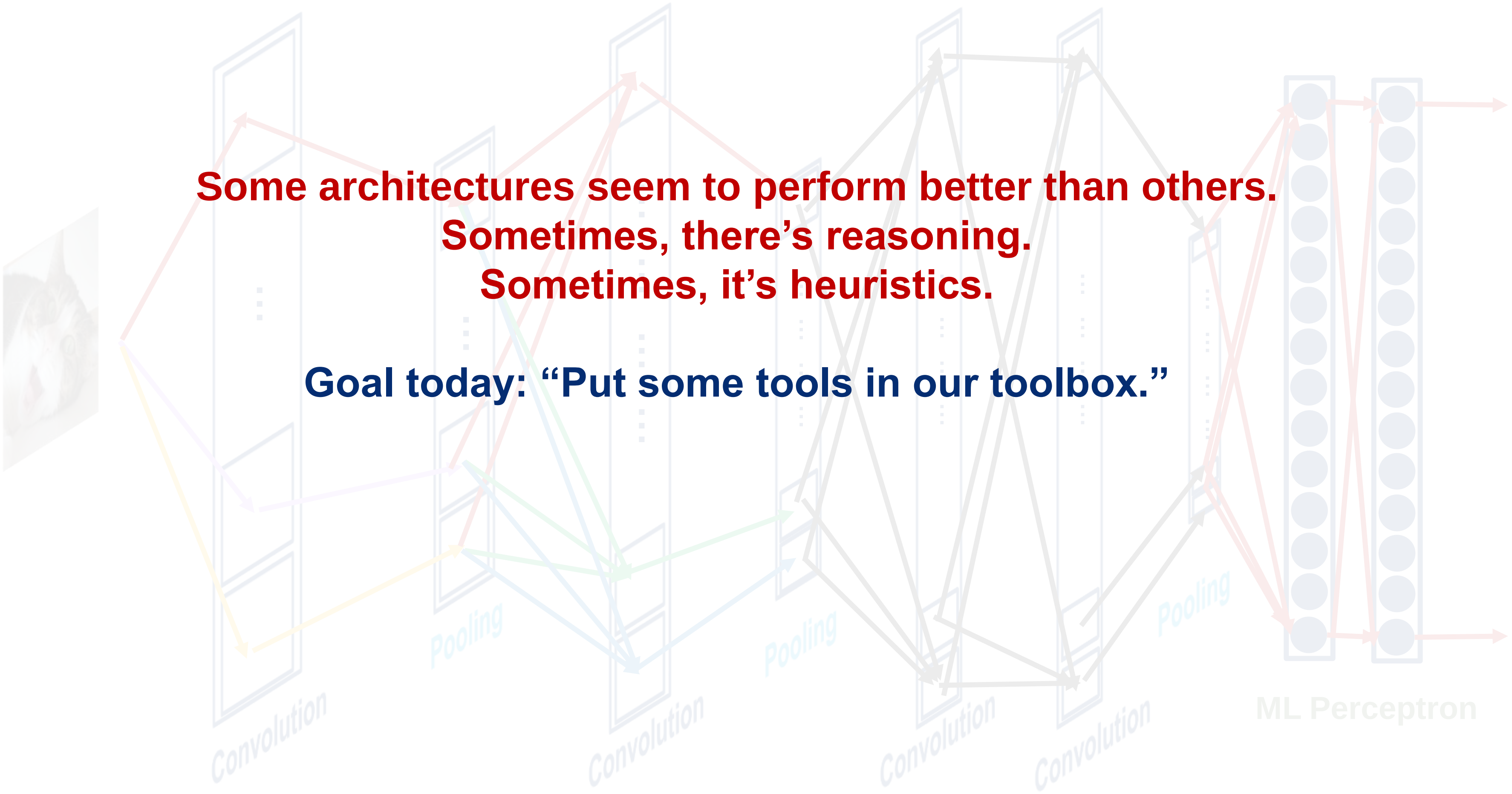
→ Optimization is complicated largely due to saddle points (not local minima)

→ 1st moment helps for minima, 2nd moment helps for saddle points

The General Architecture of ConvNets



The General Architecture of ConvNets



Some architectures seem to perform better than others.
Sometimes, there's reasoning.
Sometimes, it's heuristics.

Goal today: "Put some tools in our toolbox."

Today’s Lecture

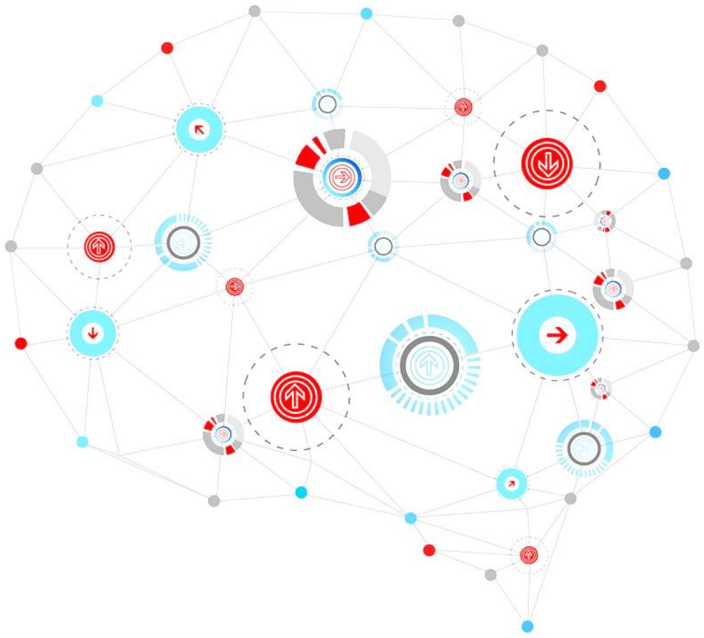
AlexNet

VGG

ResNet

U-Net

Transfer Learning



The Humble Beginnings

LeNet-5

State-of-the-art performance on MNIST digit recognition (< 1%)

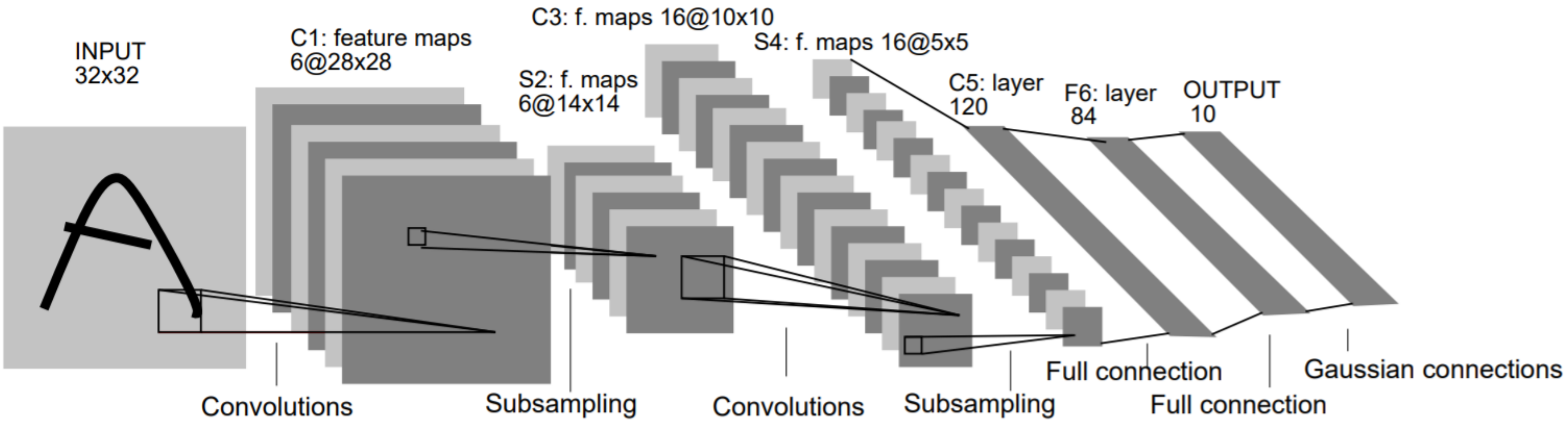


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

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

Network Architectures

AlexNet

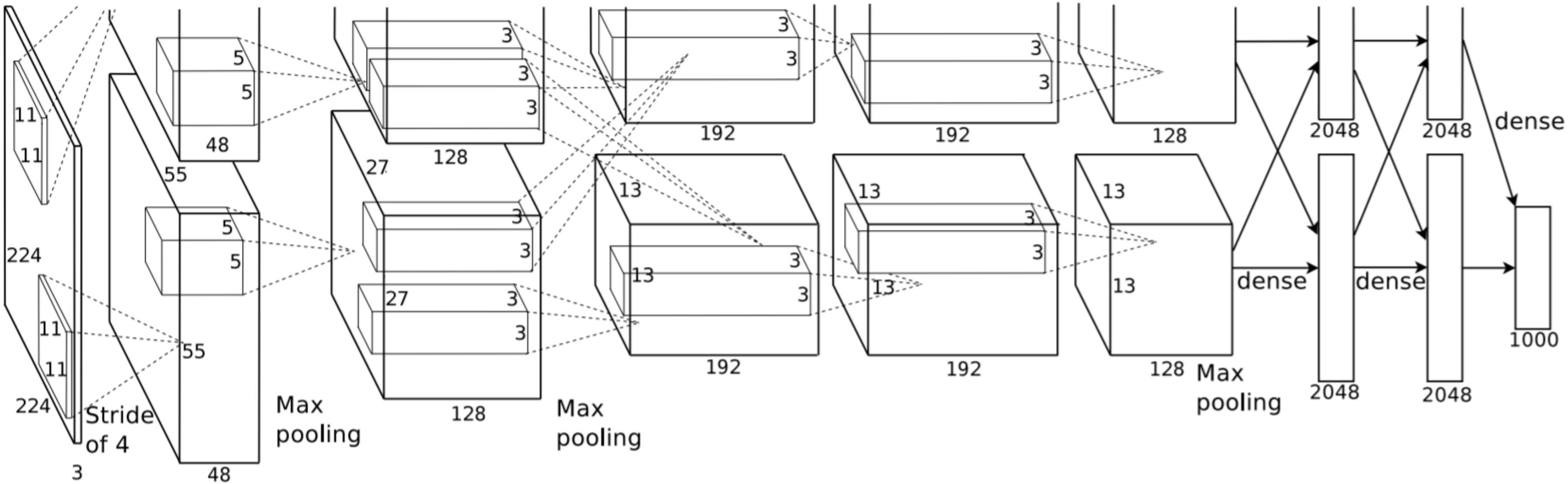
AlexNet

The start of the Deep Learning hype

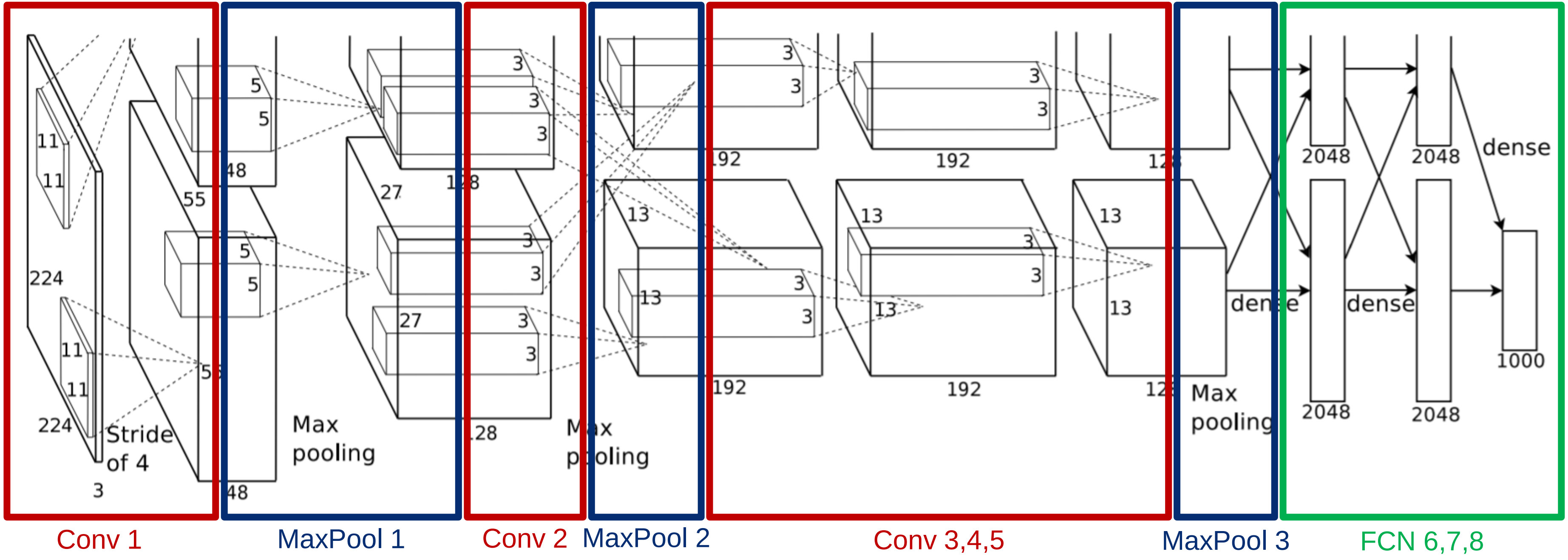
AlexNet outperforms all competing methods on ImageNet by a large margin

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

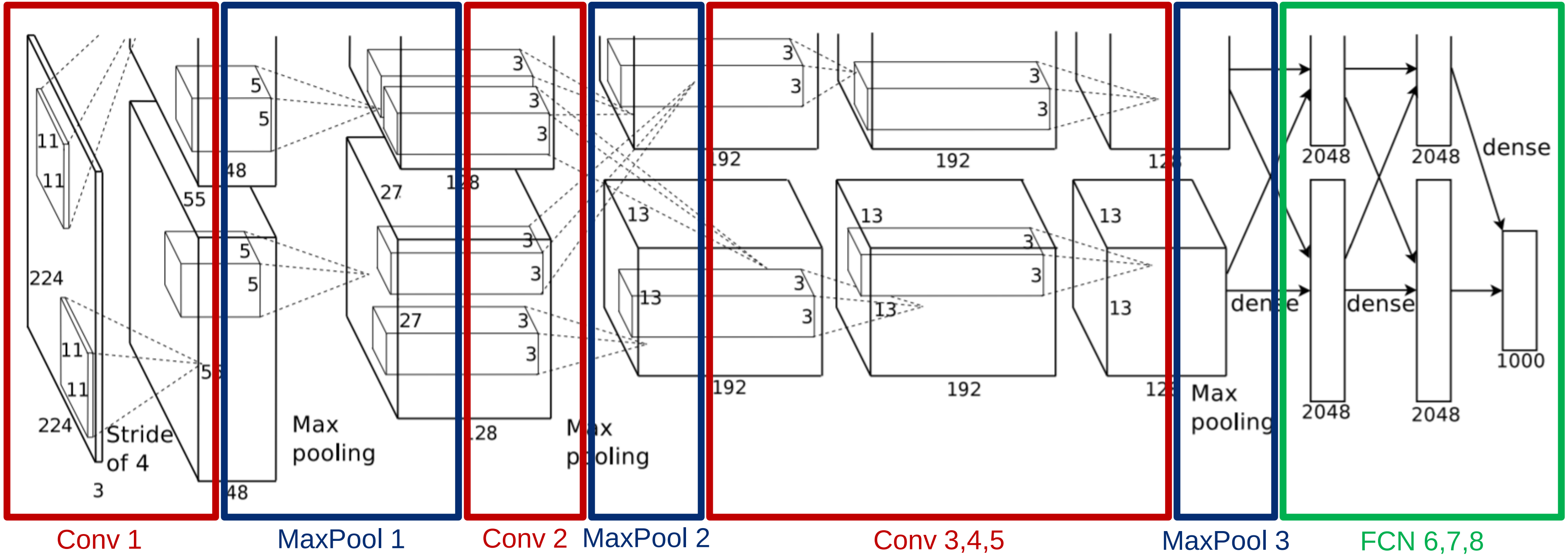
Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.



[Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).]



[Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).]

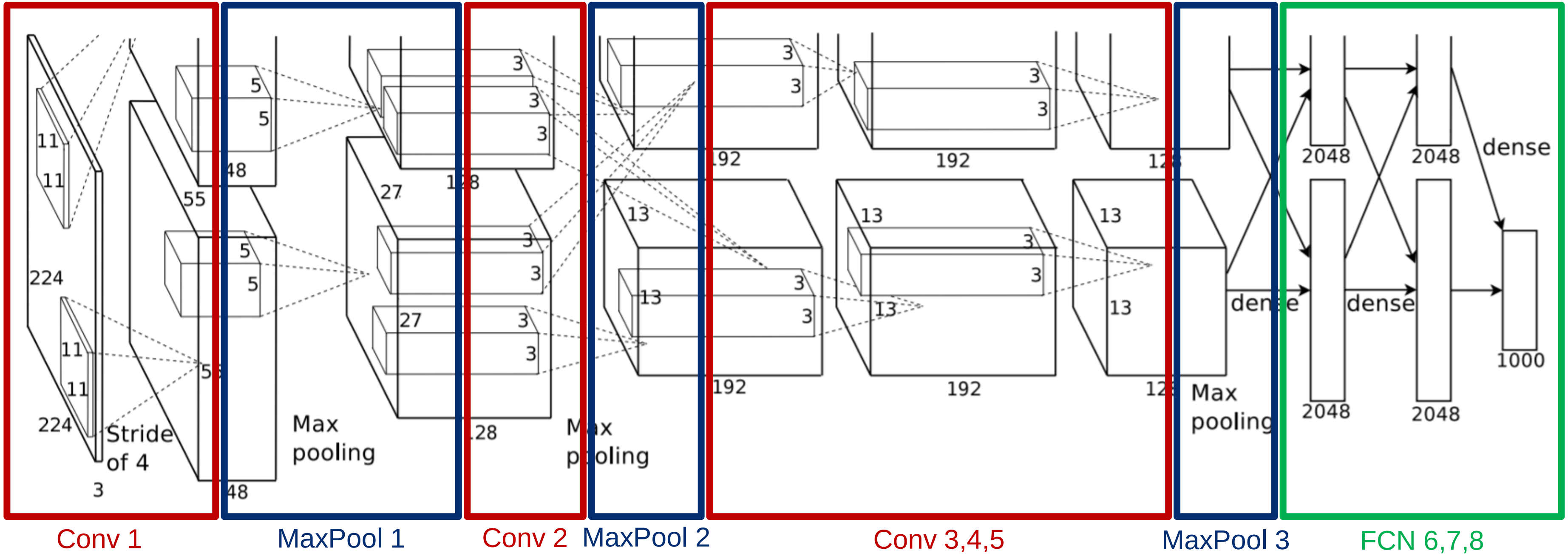


Input: 227 x 227 x 3

Conv 1: 96 11 x 11 filters applied at stride 4.

What is the output size?

[Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).]



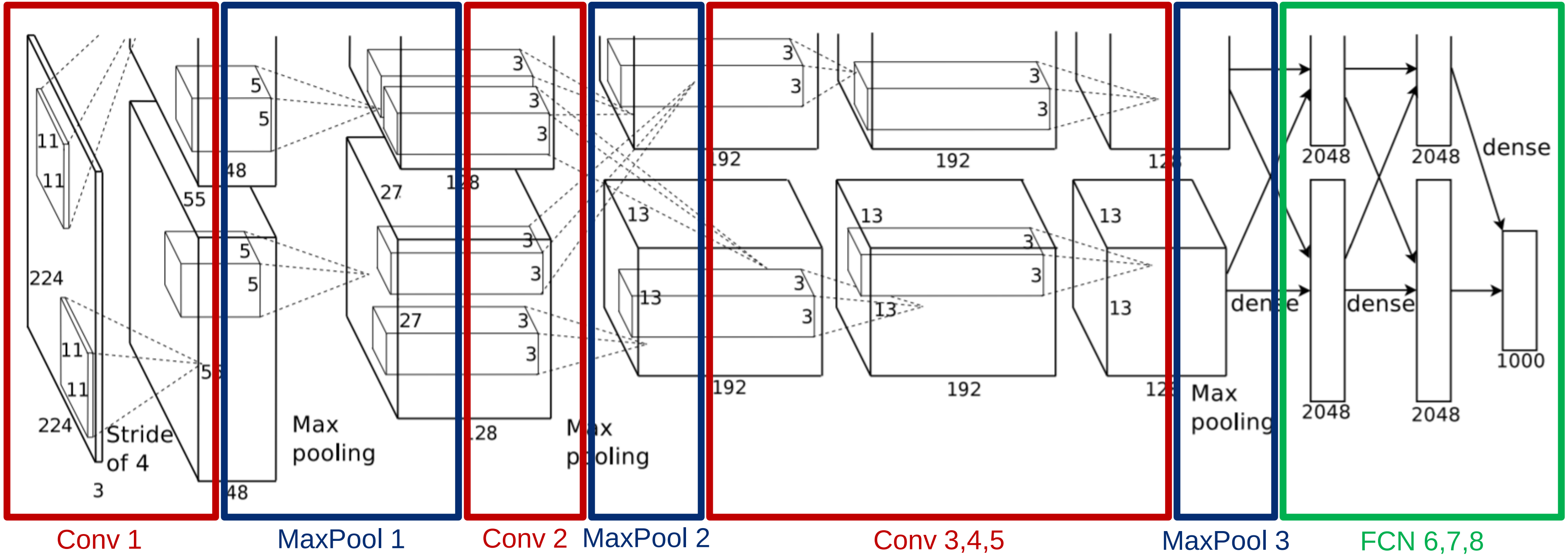
Input: 227 x 227 x 3

Conv 1: 96 11 x 11 filters applied at stride 4.

What is the output size? **55 x 55 x 96**

How many parameters?

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).



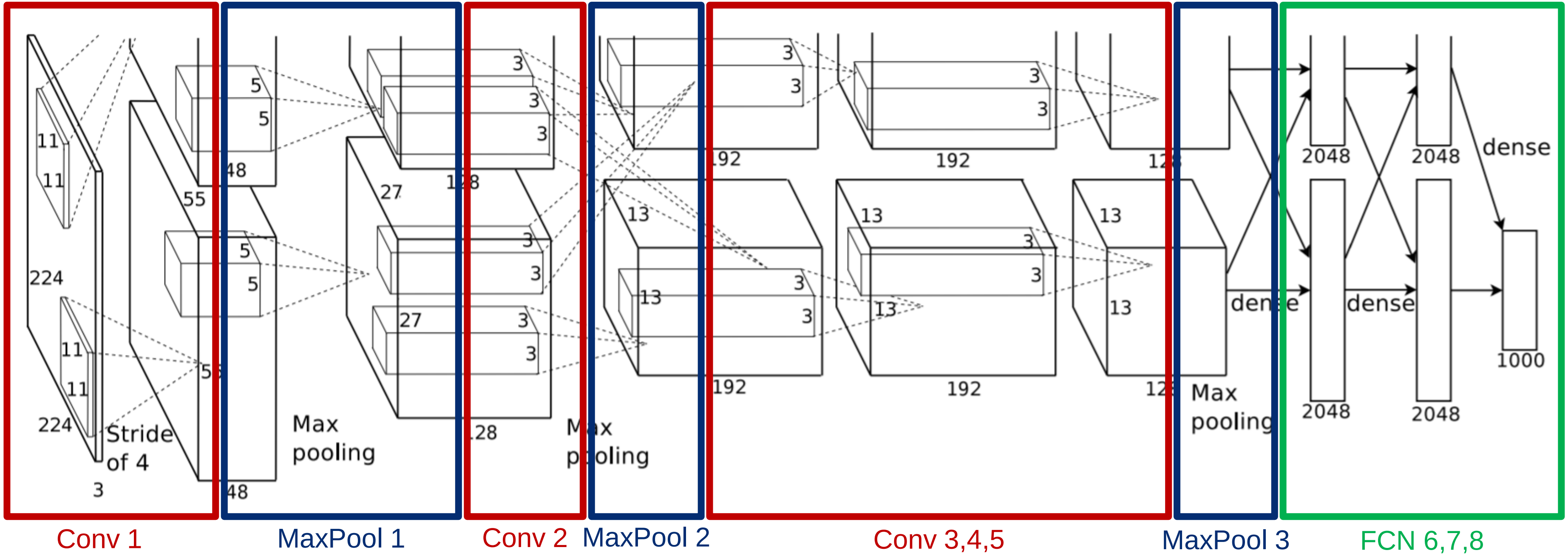
Input: 227 x 227 x 3

Conv 1: 96 11 x 11 filters applied at stride 4.

What is the output size? **55 x 55 x 96**

How many parameters? **34,944**

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).

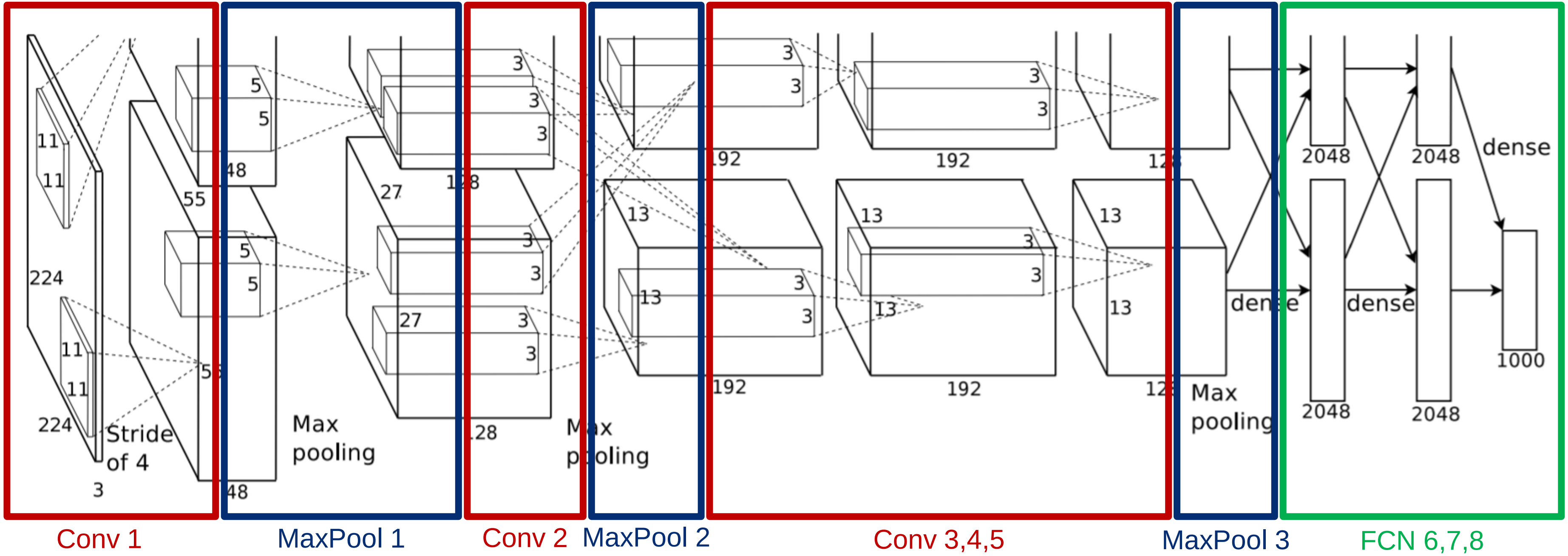


After Conv 1: 55 x 55 x 96

Max Pool 1: 3 x 3 filters applied at stride 2.

What is the output size?

[Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).]



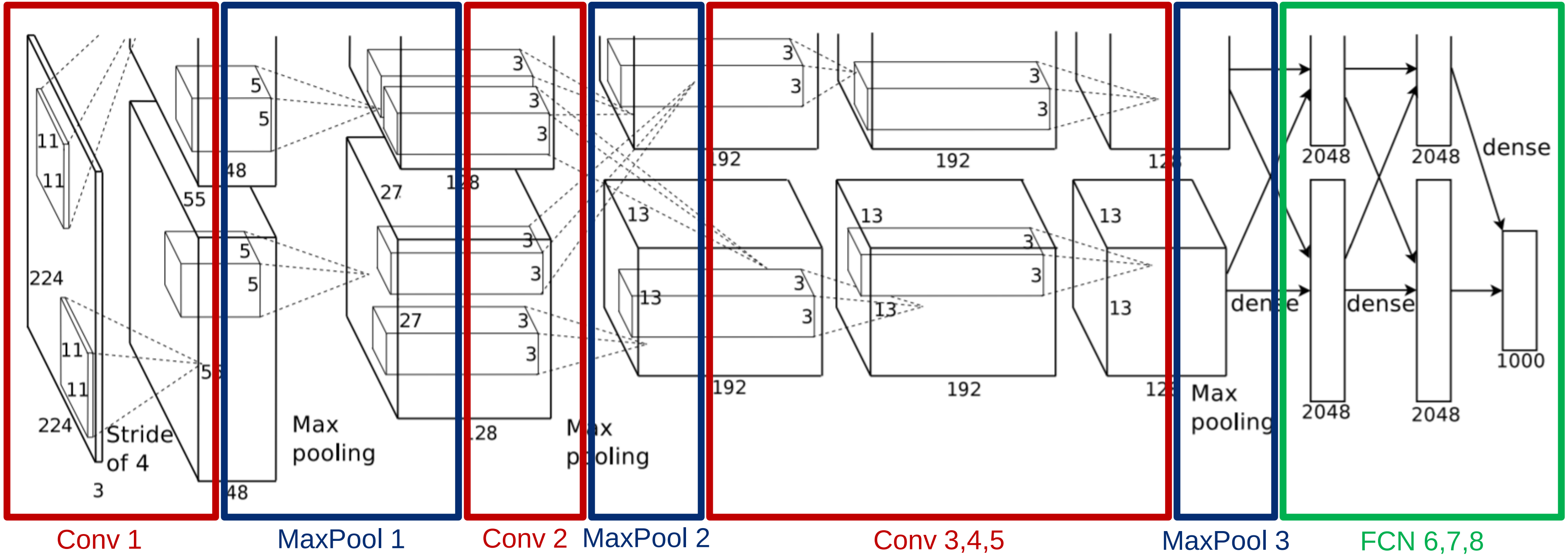
After Conv 1: 55 x 55 x 96

Max Pool 1: 3 x 3 filters applied at stride 2.

What is the output size? **27 x 27 x 96**

How many parameters?

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).



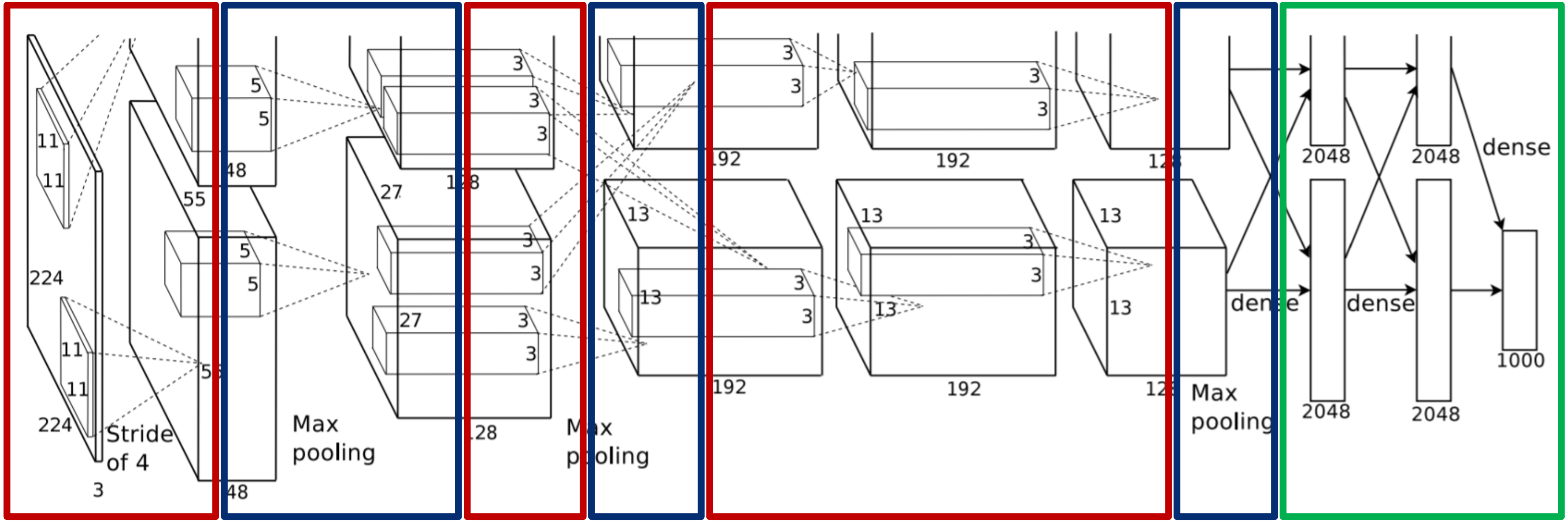
After Conv 1: 55 x 55 x 96

Max Pool 1: 3 x 3 filters applied at stride 2.

What is the output size? **27 x 27 x 96**

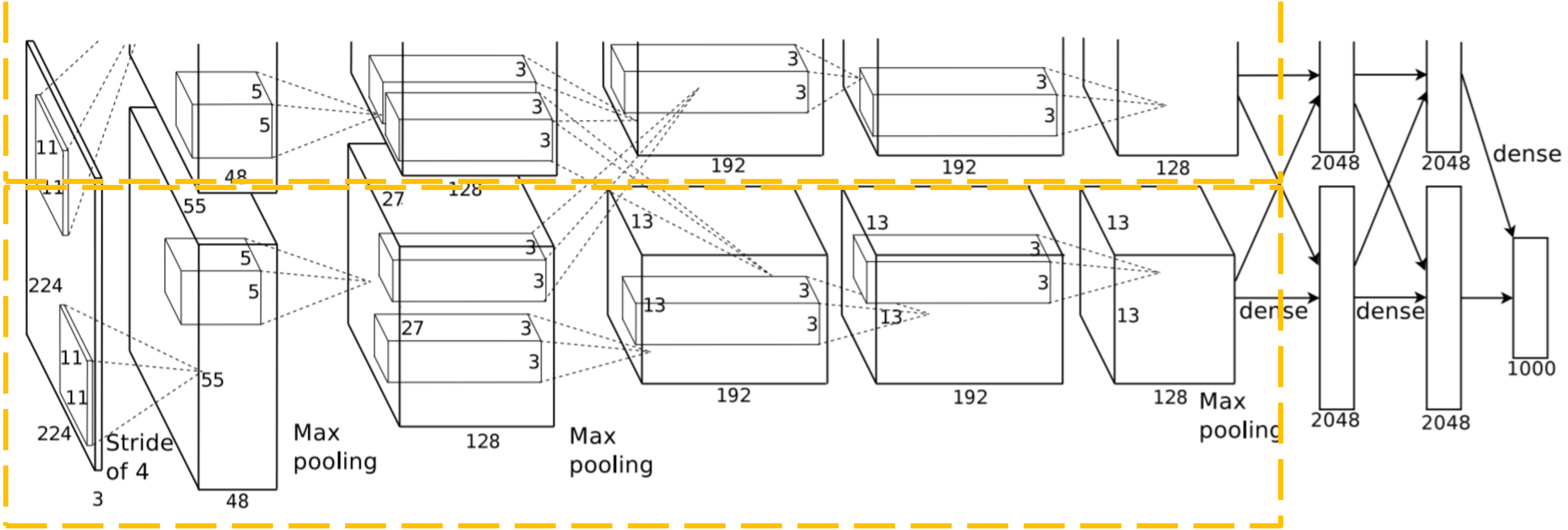
How many parameters? **0**

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).



Conv 1	MaxPool 1	Conv 2	MaxPool 2	Conv 3,4,5			MaxPool 3	FCN 6,7,8
96 11x11 Stride 4 Pad 0	3x3 Stride 2	256 5x5 Stride 1 Pad 2	3x3 Stride 2	384 3x3 Stride 1 Pad 1	384 3x3 Stride 1 Pad 1	256 3x3 Stride 1 Pad 1	3x3 Stride 2	4096 4096 1000
55x55x96	27x27x96	27x27x256	13x13x256	13x13x384	13x13x384	13x13x256	6x6x256	4096 4096 1000

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).



Conv 1

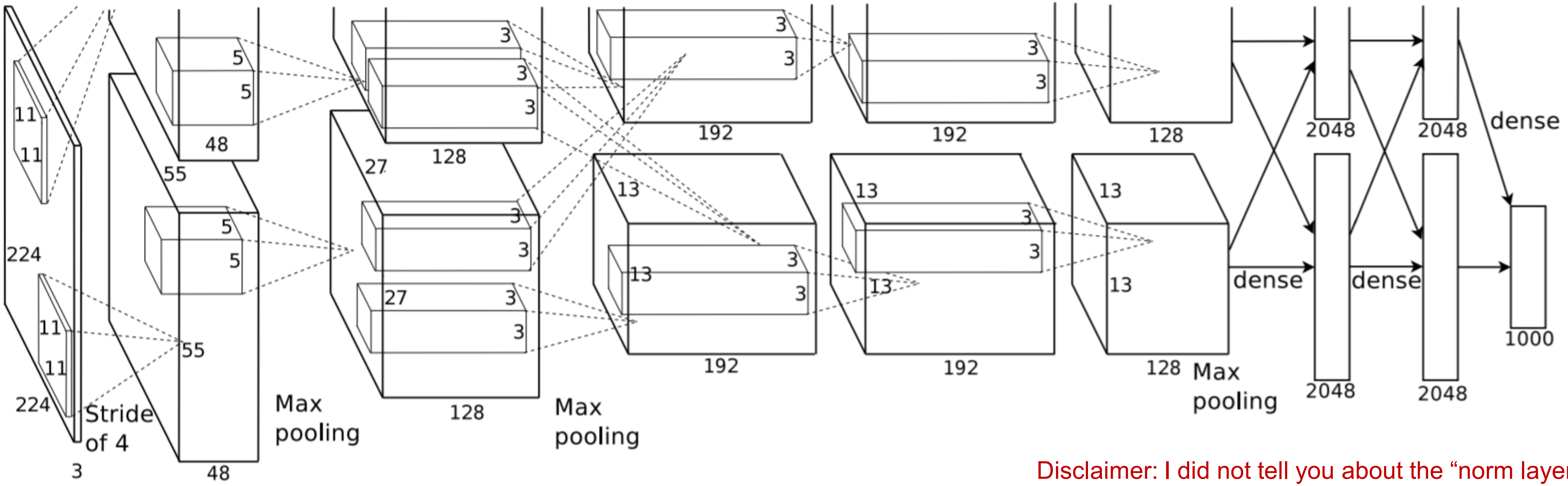
96 11x11
Stride 4
Pad 0

55x55x96

Actually: (55x55x48) x 2!

Historical reason: Trained on GTX 580 with 3GB memory
→ Did not fit! Network spread over 2 GPUs
→ Conv 1,2,4,5 only with feature maps on same GPU
→ Conv 3, FC 6,7,8 connection with all feature maps

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).



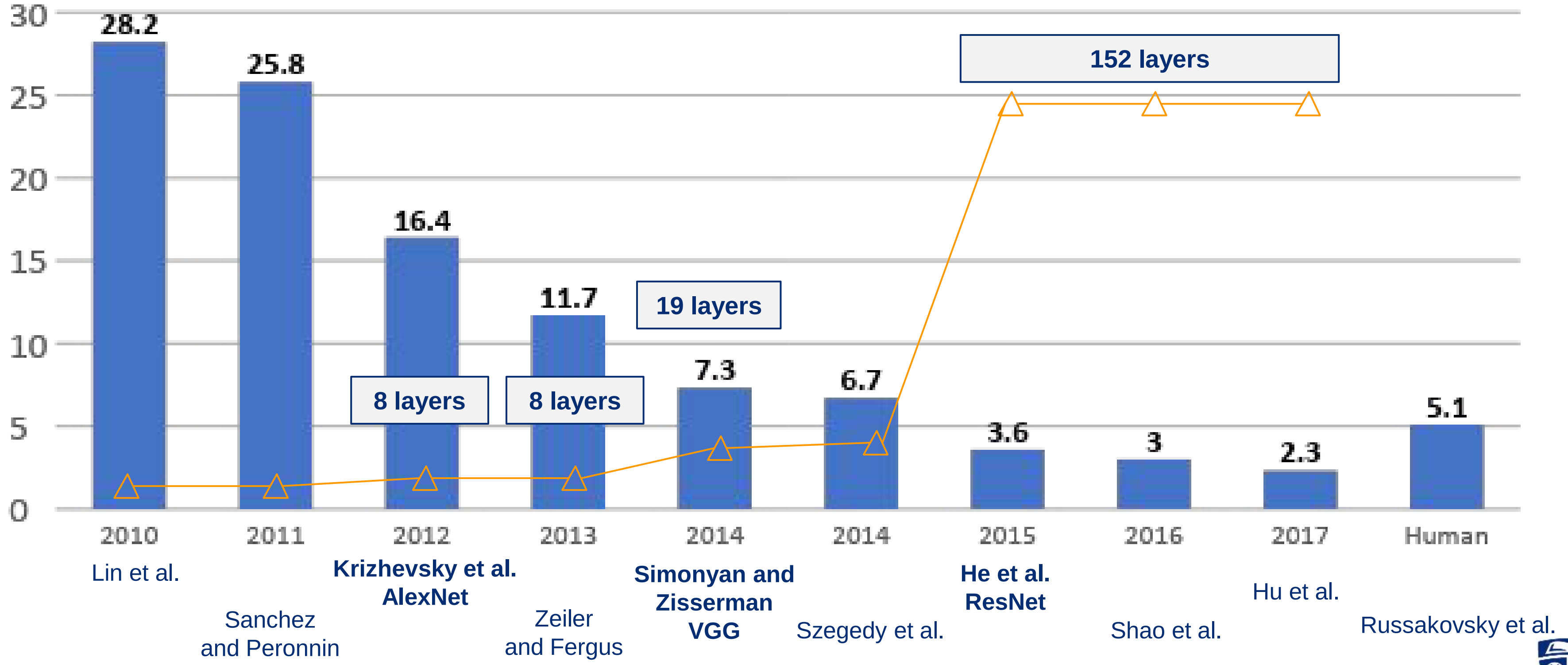
Disclaimer: I did not tell you about the “norm layers” but these are not important and not used anymore.

Details and interesting aspects

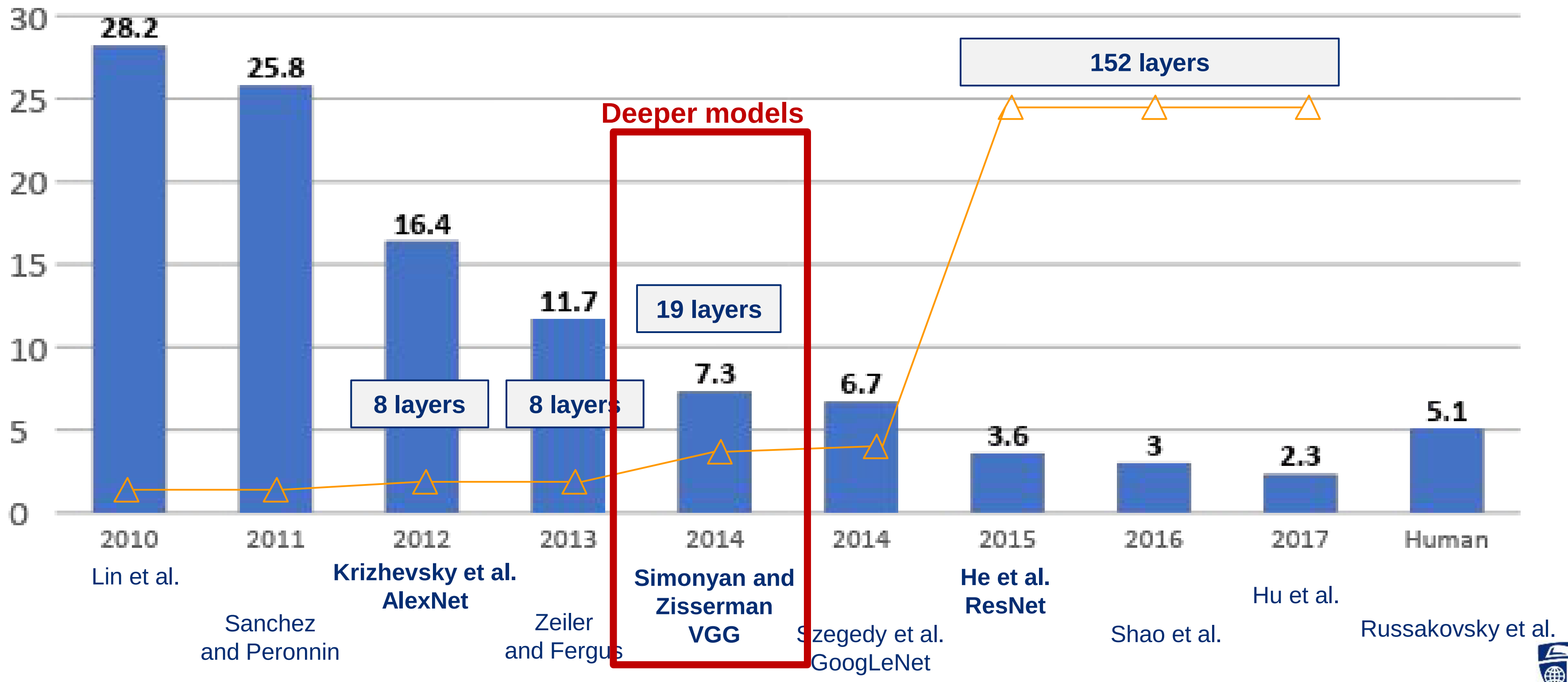
- First use of **ReLU**
- Already used **many tricks of the trade**: Heavy augmentation, dropout, SGD with momentum and manual learning rate decay, L2 weight decay (regularization!)
- Output is ensemble prediction over 7 CNNs: From 18.2% down to 16.4%

[Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).]

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Network Architectures

VGG

VGG

AlexNet (16.4% Top-5 error ILSVRC12)

8 layers

11x11 – 5x5 – 3x3

VGG (7.3% Top-5 error ILSVRC14)

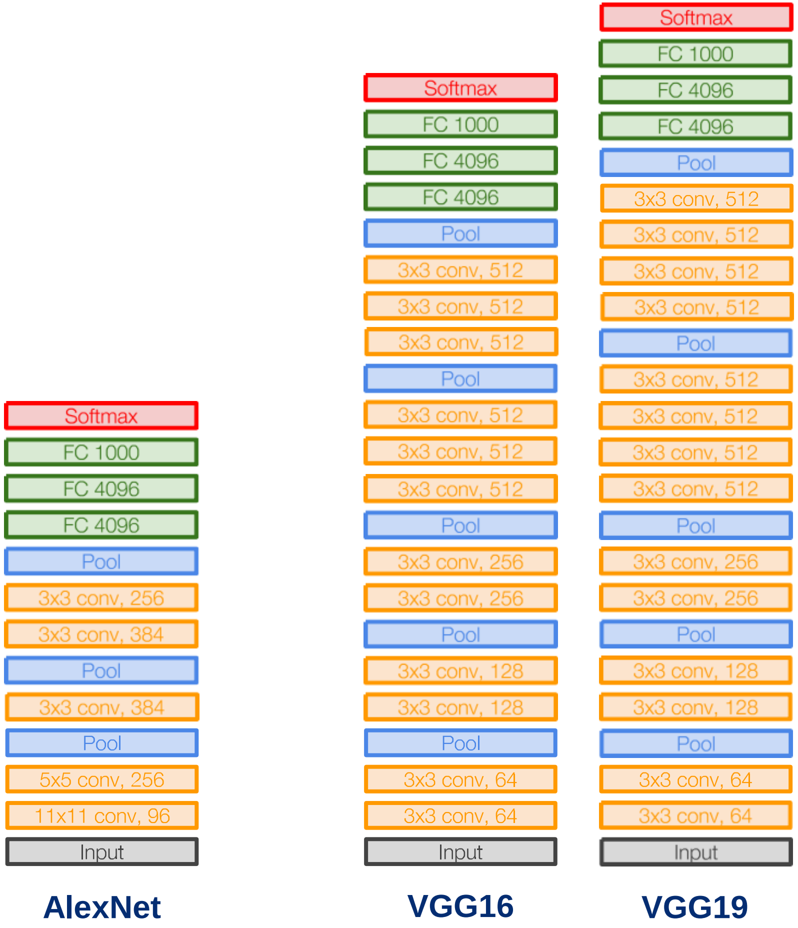
16 – 19 layers

3x3 conv with stride 1, pad 1

2x2 max pool, stride 2

→ **Smaller filters, deeper networks!**

Q: Why use smaller filter?



Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

VGG

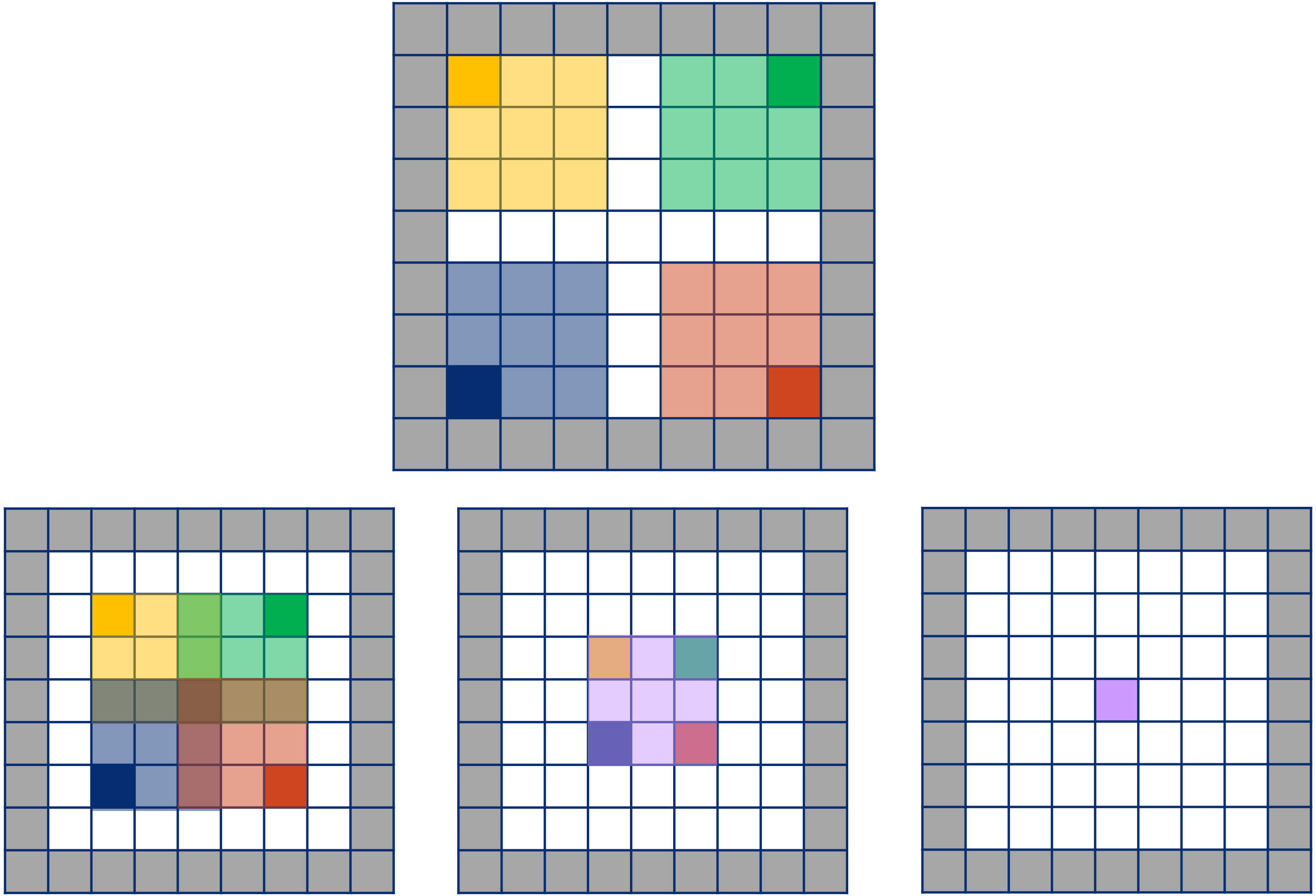
Q: Why use smaller filter?

Let’s do the other way around: Why user larger filters?
→ Receptive field!

Stack of 3 3x3 stride 1 convolutional layers has **same effective receptive field** as 7x7 layer!

But deeper → **More non-linearities**

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.



VGG

Q: Why use smaller filter?

Let's do the other way around: Why user larger filters?

→ Receptive field!

Stack of 3 3x3 stride 1 convolutional layers has **same effective receptive field** as 7x7 layer!

But deeper → **More non-linearities**

Fewer parameters!

$3*(3^2\,C^2)$ vs $7^2\,C^2$ for C channels per layer

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Disclaimer: Not counting biases

Input: [224x224x3]	memory: 224*224*3=150k	params: 0
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	params: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	params: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	params: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28*256=200K	params: 0
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	params: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	params: 0
FC: [1x1x4096]	memory: 4096	params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]	memory: 4096	params: 4096*4096 = 16,777,216
FC: [1x1x1000]	memory: 1000	params: 4096*1000 = 4,096,000

VGG16 **VGG19**

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Input: [224x224x3]	memory: 224*224*3=150k	params: 0
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	params: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	params: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	params: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28*256=200K	params: 0
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	params: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	params: 0
FC: [1x1x4096]	memory: 4096	params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]	memory: 4096	params: 4096*4096 = 16,777,216
FC: [1x1x1000]	memory: 1000	params: 4096*1000 = 4,096,000

Total Memory: 24M * 4 bytes ~ 96 MB / image (forward → ~ * 2 for backward!)

Total parameters: 138 Mio parameters

Input: [224x224x3]	memory: 224*224*3=150k	params: 0
CONV3-64: [224x224x64]	**memory: 224*224*64=3.2M**	params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	**memory: 224*224*64=3.2M**	params: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	params: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	params: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28*256=200K	params: 0
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	params: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	params: 0
FC: [1x1x4096]	memory: 4096	**params: 7*7*512*4096 = 102,760,448**
FC: [1x1x4096]	memory: 4096	**params: 4096*4096 = 16,777,216**
FC: [1x1x1000]	memory: 1000	**params: 4096*1000 = 4,096,000**

Most memory in early CONV

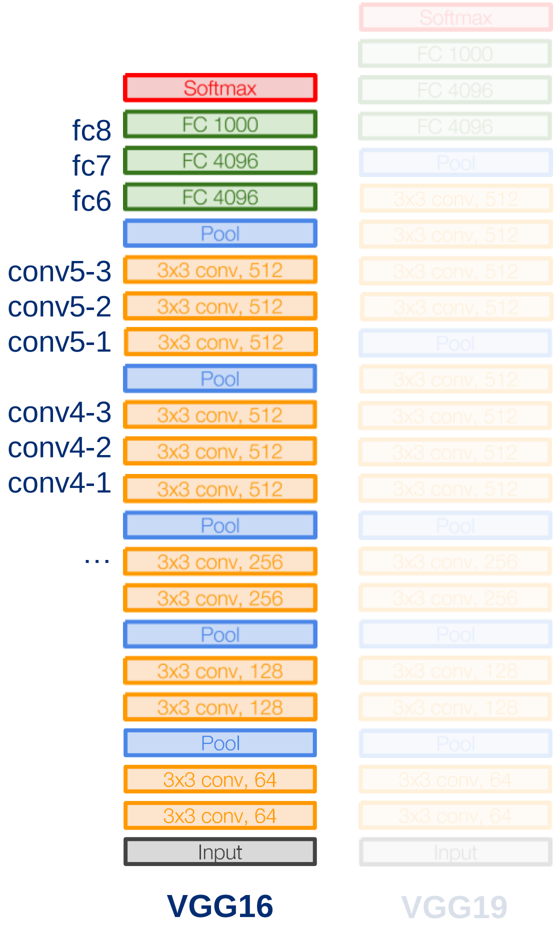
Most parameters in FCs

Total Memory: 24M * 4 bytes ~ 96 MB / image (forward → ~ * 2 for backward!)

Total parameters: 138 Mio parameters

Input: [224x224x3]	memory: 224*224*3=150k	params: 0
CONV3-64: [224x224x64]	**memory: 224*224*64=3.2M**	params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	**memory: 224*224*64=3.2M**	params: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	params: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	params: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	params: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	params: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28	
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	params: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	params: 0
FC: [1x1x4096]	memory: 4096	**params: 7*7*512*4096 = 102,760,448**
FC: [1x1x4096]	memory: 4096	**params: 4096*4096 = 16,777,216**
FC: [1x1x1000]	memory: 1000	**params: 4096*1000 = 4,096,000**

Common layer names



Total Memory: 24M * 4 bytes ~ 96 MB / image (forward → ~ * 2 for backward!)

Total parameters: 138 Mio parameters

VGG: Details

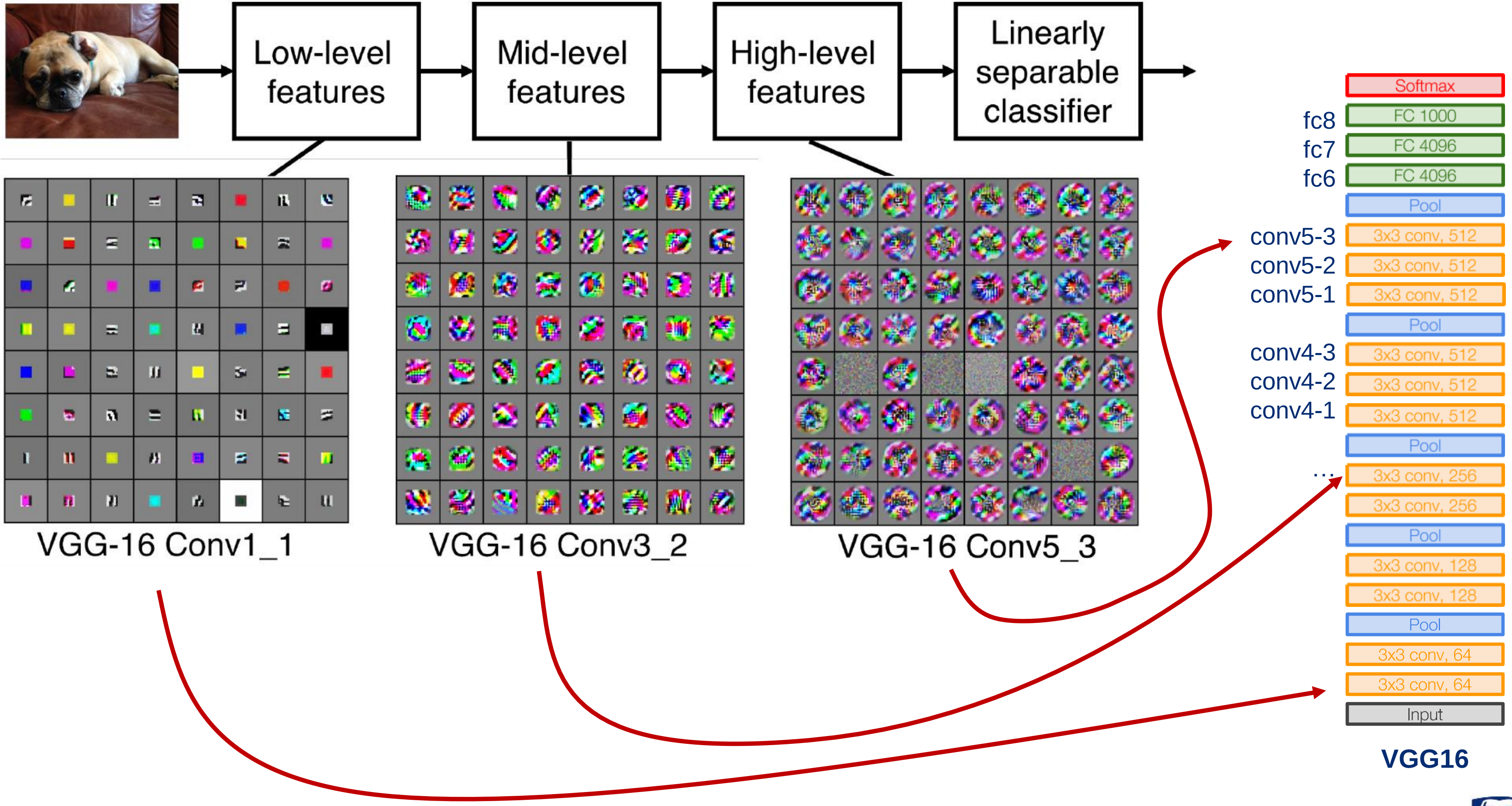
- ILSVRC’14: 2nd in classification and 1st in localization

- VGG19 only slightly better performance than VGG16, but more memory
  → Use VGG16

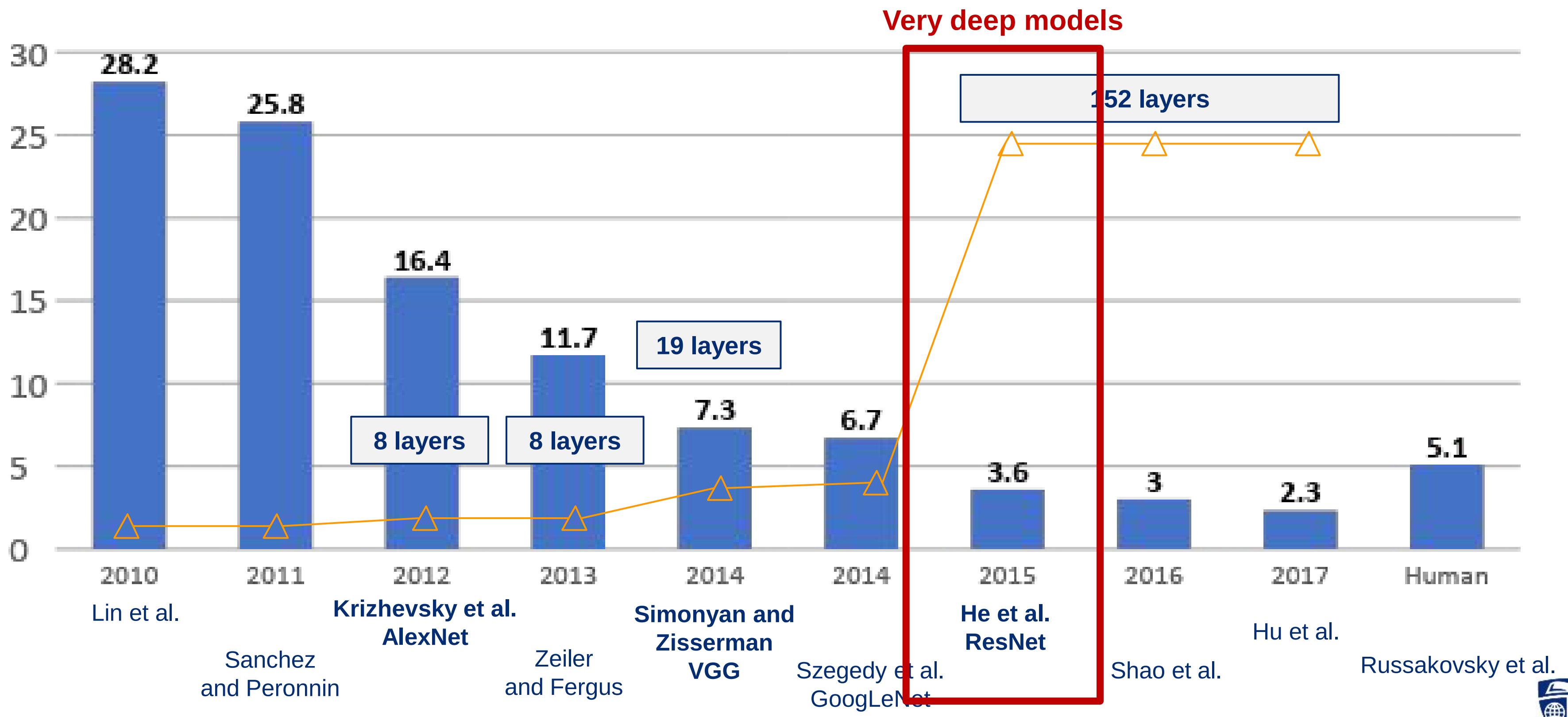
- Ensembles for better results (see AlexNet)

- fc7 features generalize well to other tasks
  → VGG19 is often used in transfer learning

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.


Low-level features
Mid-level features
High-level features
Linearly separable classifier
VGG-16 Conv1_1
VGG-16 Conv3_2
VGG-16 Conv5_3
Softmax
fc8
FC 1000
fc7
FC 4096
fc6
FC 4096
Pool
conv5-3
3x3 conv, 512
conv5-2
3x3 conv, 512
conv5-1
3x3 conv, 512
Pool
conv4-3
3x3 conv, 512
conv4-2
3x3 conv, 512
conv4-1
3x3 conv, 512
Pool
...
3x3 conv, 256
3x3 conv, 256
Pool
3x3 conv, 128
3x3 conv, 128
Pool
3x3 conv, 64
3x3 conv, 64
Input
VGG16

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



Network Architectures

ResNet

ResNet

So far: Increasing depth → increasing performance!
E.g. AlexNet with 8 layers to VGG19 with 19 layers

An idea: If the above is true, then let's stack more layers to get even better!
Let's start with stacking 56 layers.

Q: What do you think? How does the performance of a 56 layer network compare to a 20 layer network?

First, on test data.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognitionIEEE conference on computer vision and pattern recognition (pp. 770-778).

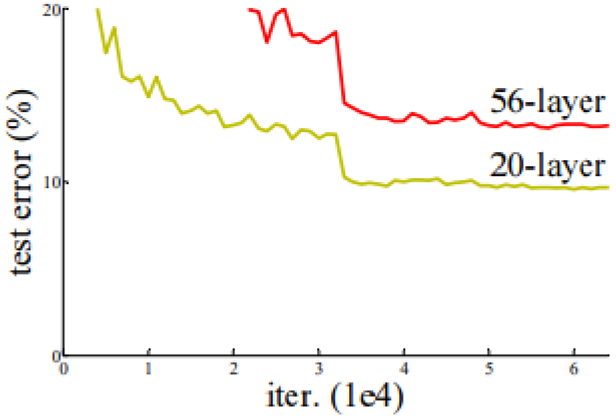
ResNet

Q: What do you think? How does the performance of a 56 layer network compare to a 20 layer network?

First, on test data.

Worse performance!

Hypothesis:
The amount of data is the same, but the deeper network has more free parameters.
→ **Strong overfitting**
→ **Check training data**

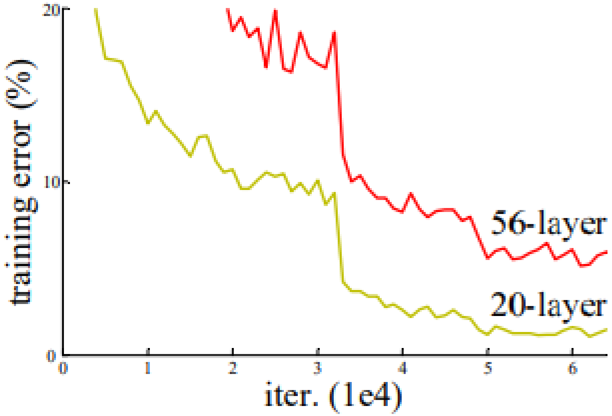


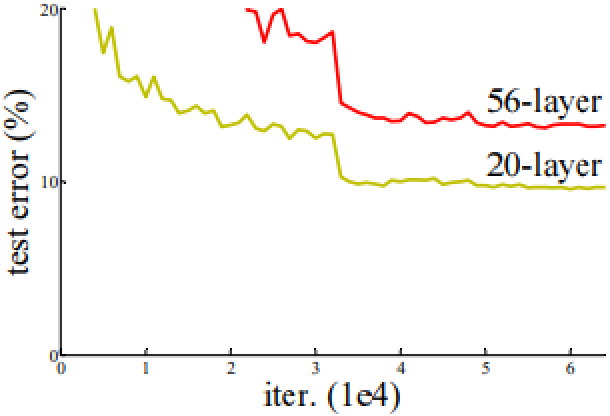
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognitionIEEE conference on computer vision and pattern recognition (pp. 770-778).

ResNet

Q: What do you think? How does the performance of a 56 layer network compare to a 20 layer network?

Training error also worse. This is not overfitting!





[He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognitionIEEE conference on computer vision and pattern recognition (pp. 770-778).]

ResNet

Another hypothesis

This observation is due to an optimization problem.
Deeper models are harder to optimize.

What intuition tells us

Deeper models should perform at least as well as the shallower model.

Designing such solution

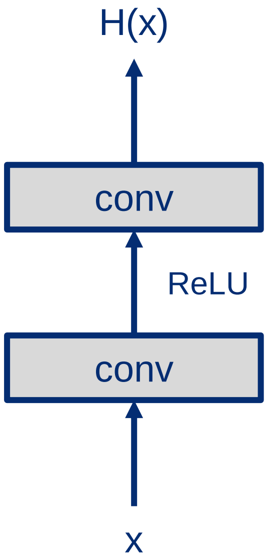
Copy learned layers from shadow model and set additional layers to identity

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognitionIEEE conference on computer vision and pattern recognition (pp. 770-778).

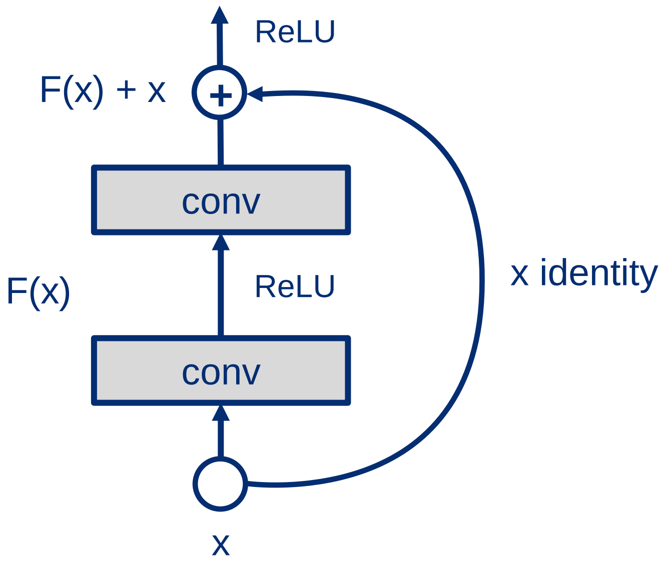
ResNet

Solution

Use network layers to fit residual mapping (rather than desired mapping directly)



“Plain” conv layers

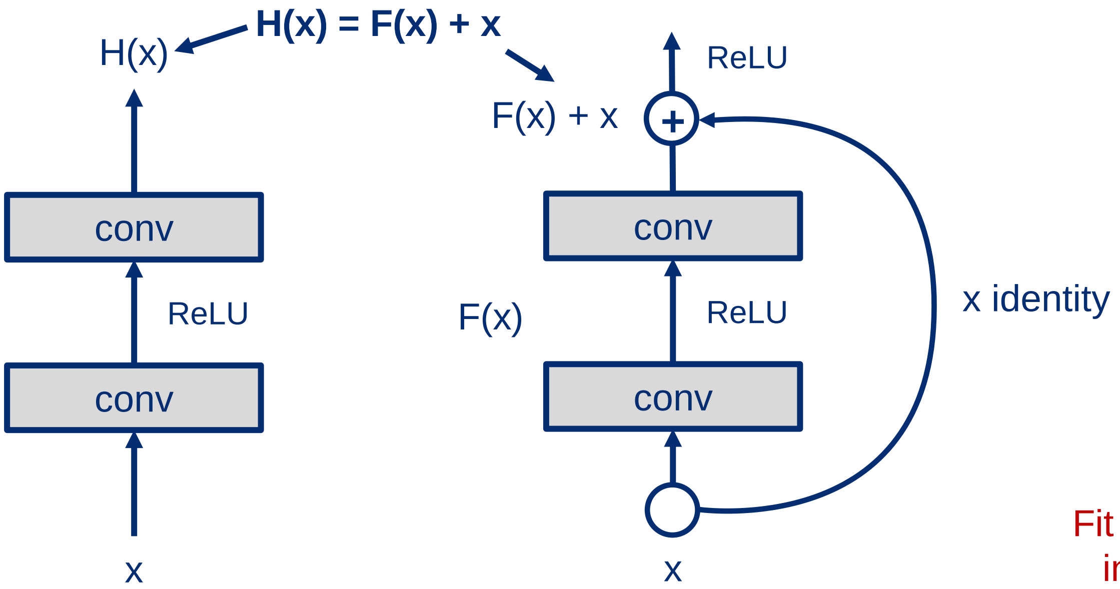


Residual block

ResNet

Solution

Use network layers to fit residual mapping (rather than desired mapping directly)



“Plain” conv layers

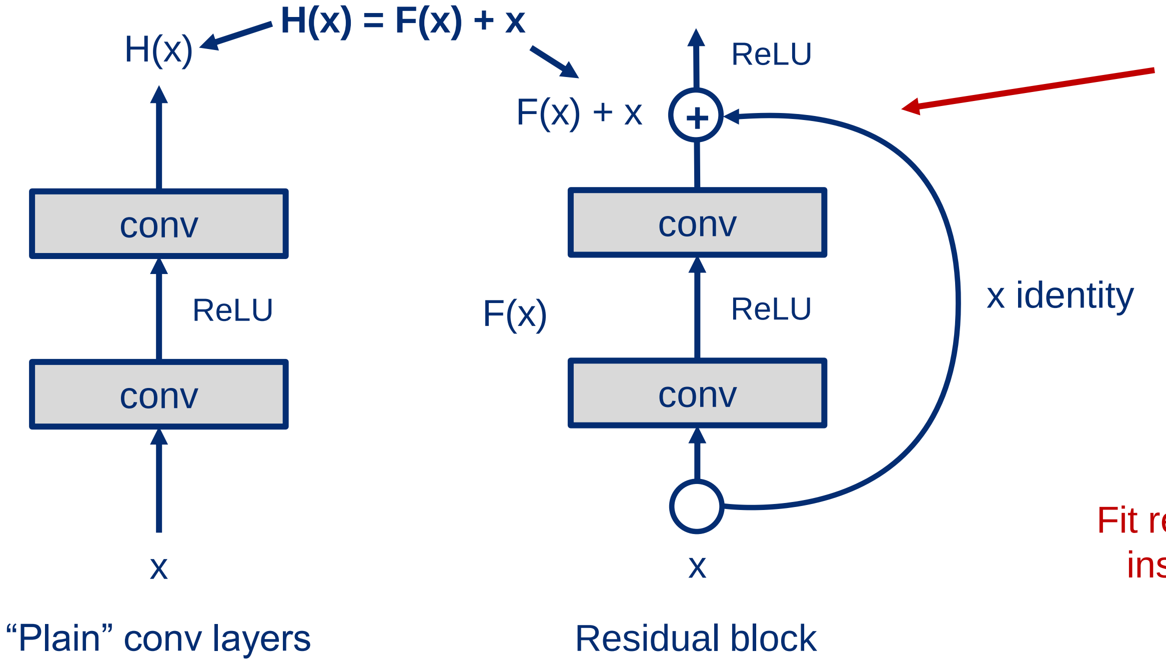
Residual block

Fit residual F(x) = H(x) – x
instead of H(x) directly

ResNet

Solution

Use network layers to fit residual mapping (rather than desired mapping directly)



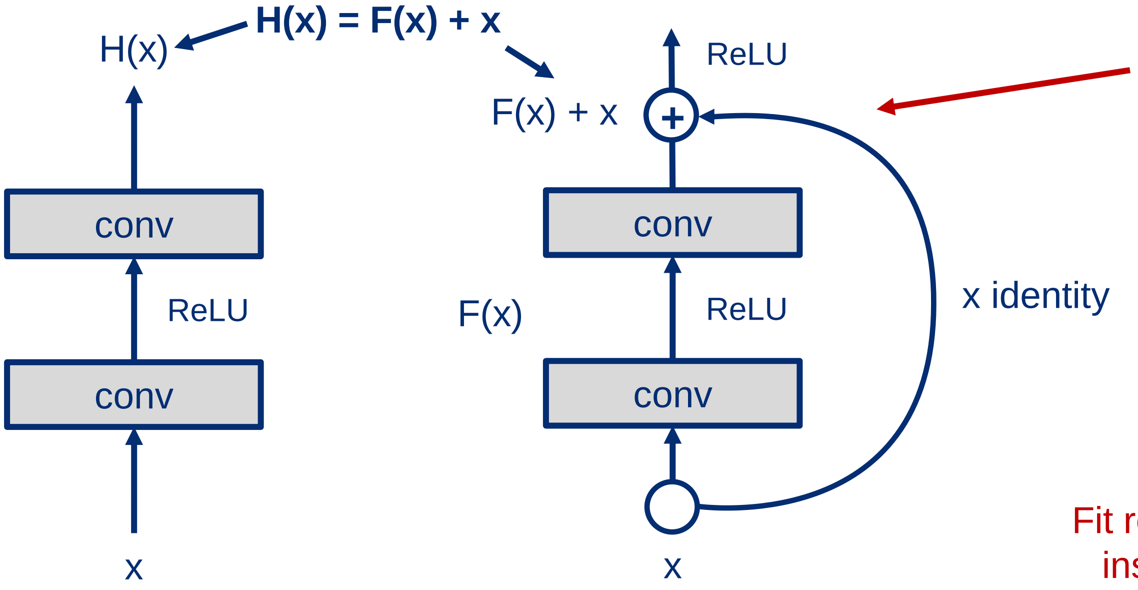
Q: What about gradients at addition nodes?

Fit residual $F(x) = H(x) - x$ instead of $H(x)$ directly

ResNet

Solution

Use network layers to fit residual mapping (rather than desired mapping directly)



“Plain” conv layers

Residual block

Q: What about gradients at addition nodes?

- Add: Distribute gradient!
- Gradient flows unhindered
- “Gradient highway”

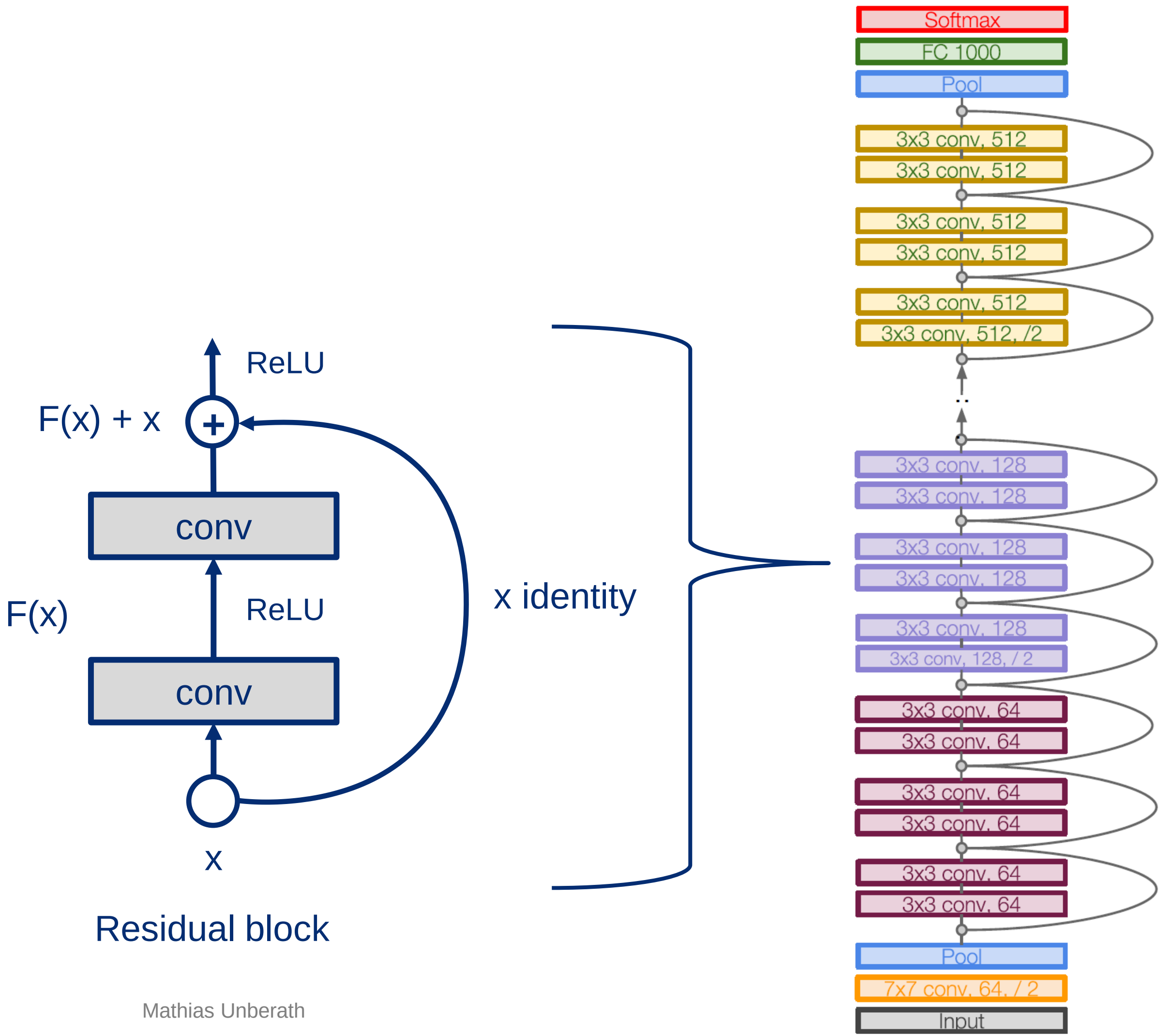
→ Makes training easier!

Fit residual F(x) = H(x) – x instead of H(x) directly

ResNet

Full architecture

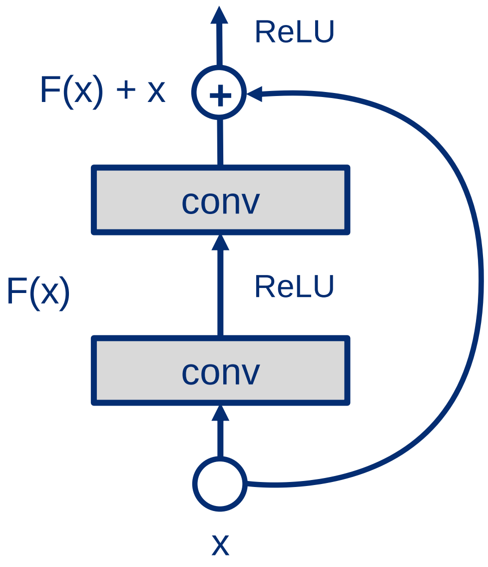
- Stack residual blocks
- Residual block has two 3x3 conv layers



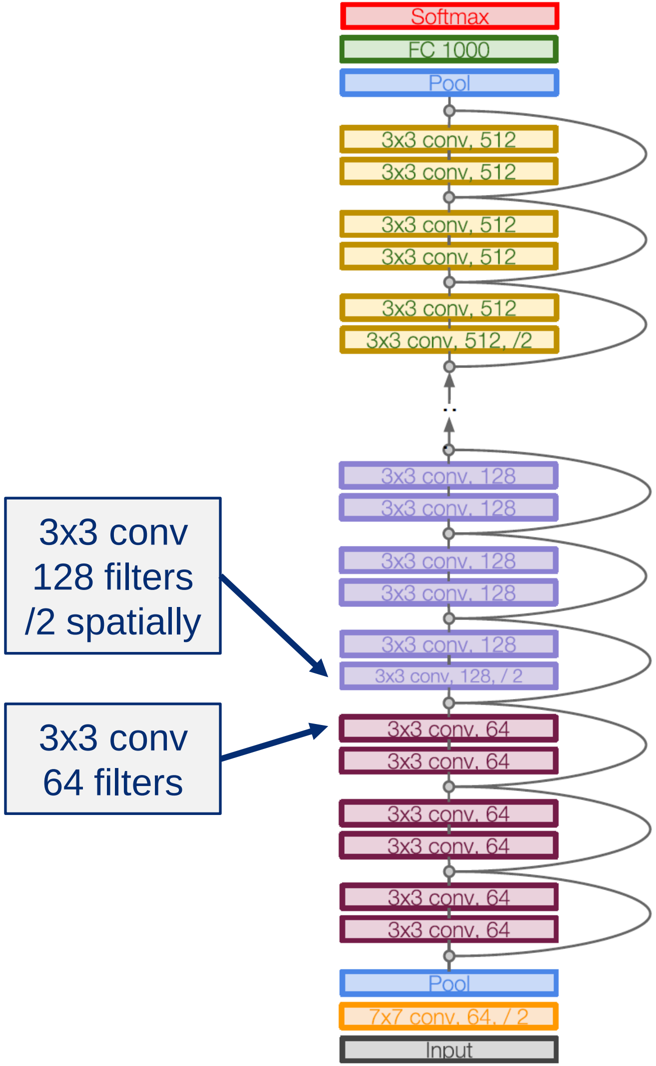
ResNet

Full architecture

- Stack residual blocks
- Residual block has two 3x3 conv layers
- Periodically, double # filters and down-sample with stride 2



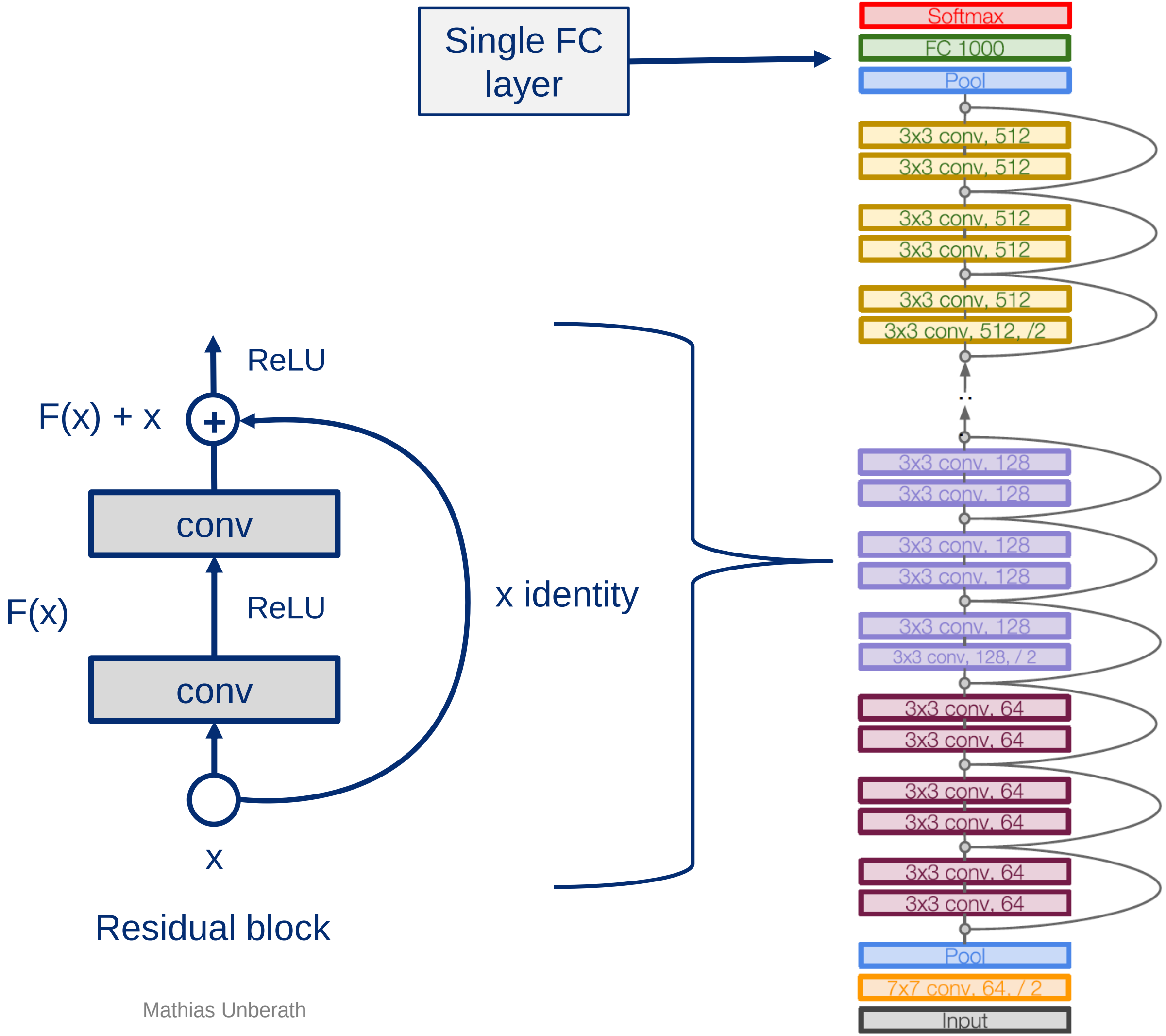
Residual block



ResNet

Full architecture

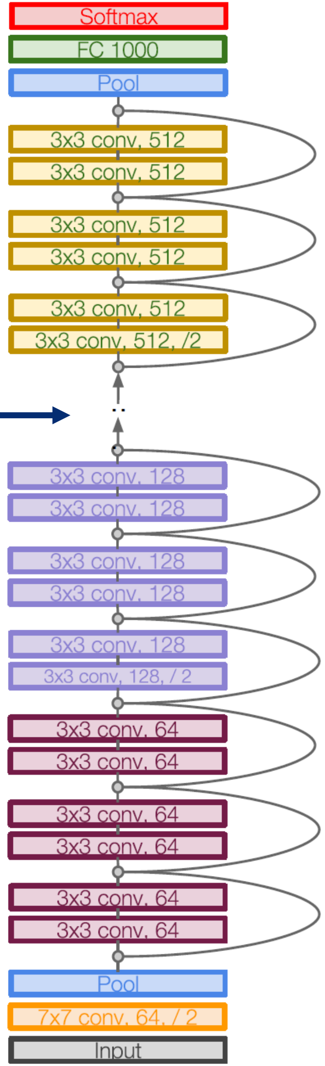
- Stack residual blocks
- Residual block has two 3x3 conv layers
- Periodically, double # filters and down-sample with stride 2
- Additional conv layer at beginning
- Only one FC layer



Residual block

ResNet

Total depths of 34, 50, 101, and even up to **152** for ImageNet



ResNet

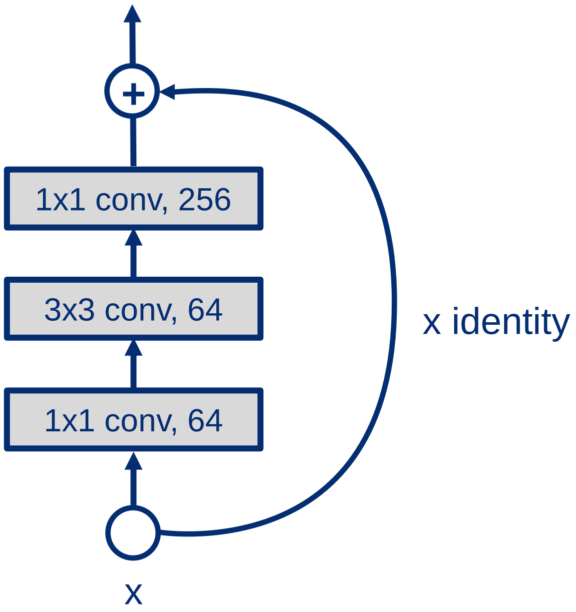
For very deep networks (> 50 layers) → “Bottleneck” layers to improve efficiency

Output: 28 x 28 x 256

1x1 conv, 256 filters
projects back to 256

3x3 conv operates on
only 64 feature maps

1x1 conv, 64 filters
→ Project to 28x28x64

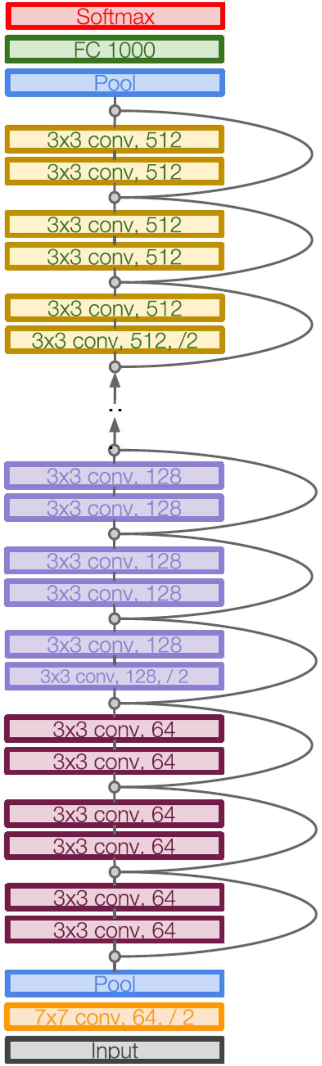


Input: 28 x 28 x 256

ResNet

Training ResNet and more details

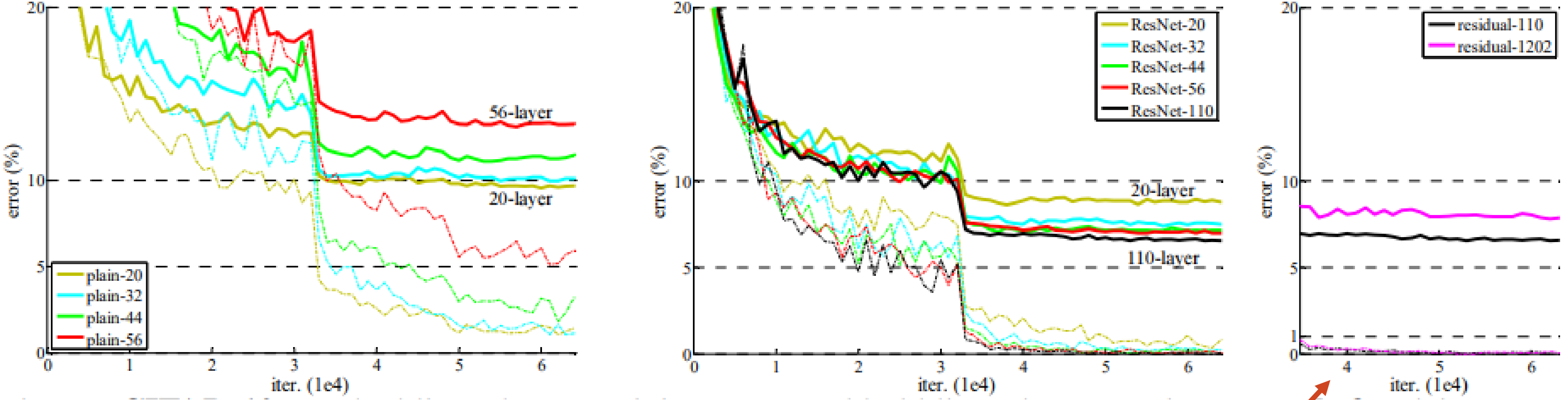
- Batch normalization after every conv layer
- Xavier initialization with factor 2 for ReLU (He initialization)
- SGD + momentum (0.9)
- Learning rate of 0.1 divided by 10 when validation plateaus
- Mini-batch size of 256
- Weight decay of 1e-5
- **No dropout**



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition IEEE CVPR (pp. 770-778).

ResNet

We finally see what is intuitive: Deeper networks perform better!



Q: What happens here?

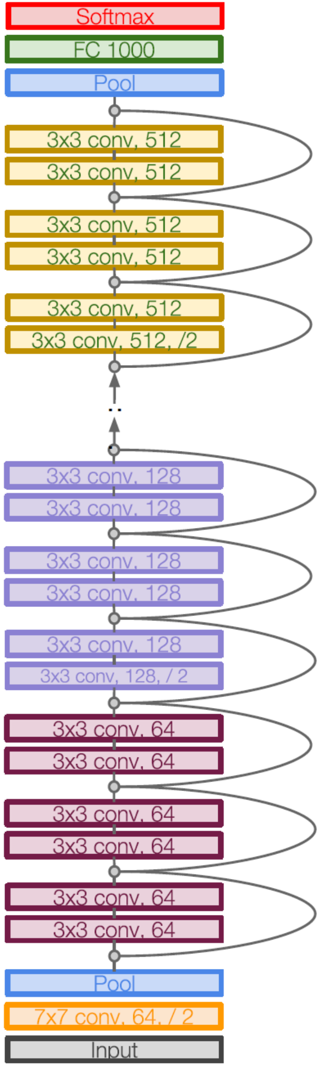
[He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition IEEE CVPR (pp. 770-778).]

ResNet

It is a very powerful architecture!

MSRA @ ILSVRC & COCO 2015 Competitions

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



He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition IEEE CVPR (pp. 770-778).

Network Architectures

U-Net

Towards Instance Segmentation with ConvNets

So far: ConvNets are used for classification / regression

High-dimensional input in → classification / regression values out

Much lower dimension
No spatial information

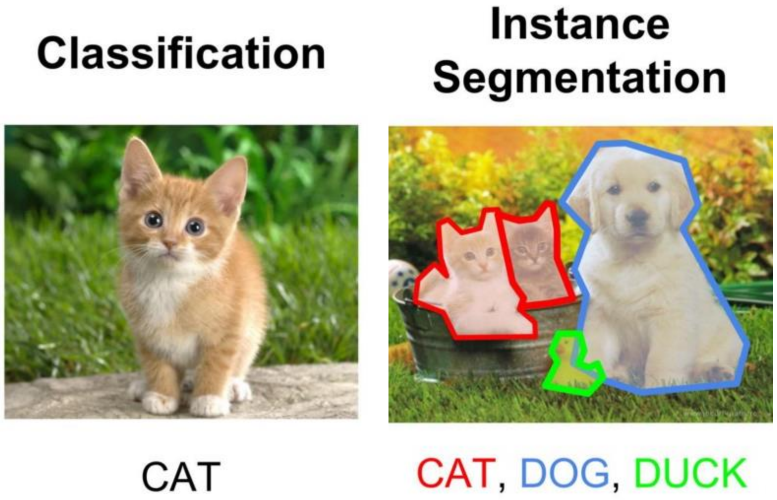
Classification

CAT

Towards Instance Segmentation with ConvNets

Now: ConvNets for instance segmentation

High-dimensional input in → $\underbrace{\text{high-dimensional output out}}_{\text{Same size as input data!}}$



Q: How do we achieve segmentation?

The Sliding-window Approach to Segmentation

A fairly early idea:

Classification is “understood”, so why not classify the central pixel of an image?

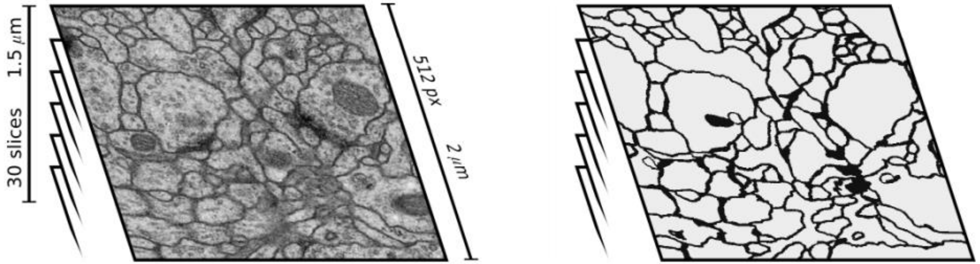


Figure 1: Left: the training stack (one slice shown). Right: corresponding ground truth; black lines denote neuron membranes. Note complexity of image appearance.

You may find it ironic, that this paper uses Artificial Neural Networks to analyze Anatomical Neural Networks.

Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. NeurIPS (pp. 2843-2851).

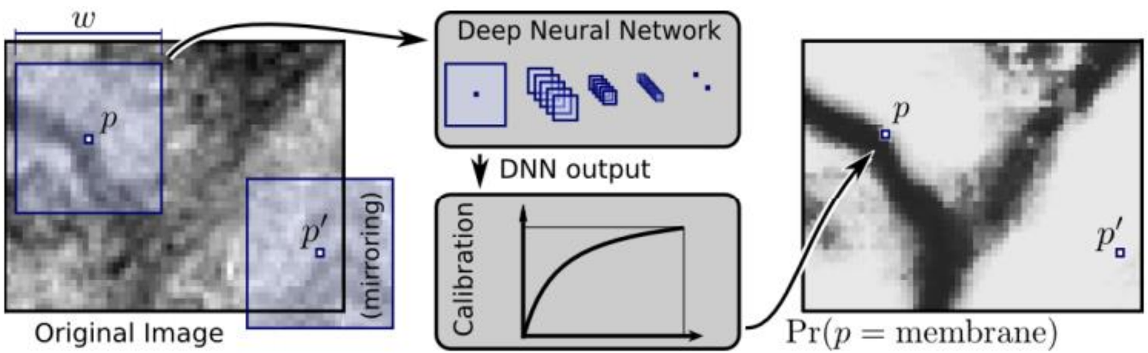
The Sliding-window Approach to Segmentation

A fairly early idea:

Classification is “understood”, so why not classify the central pixel of an image?

Some refinements:

- Images are large
  → Apply on small patches

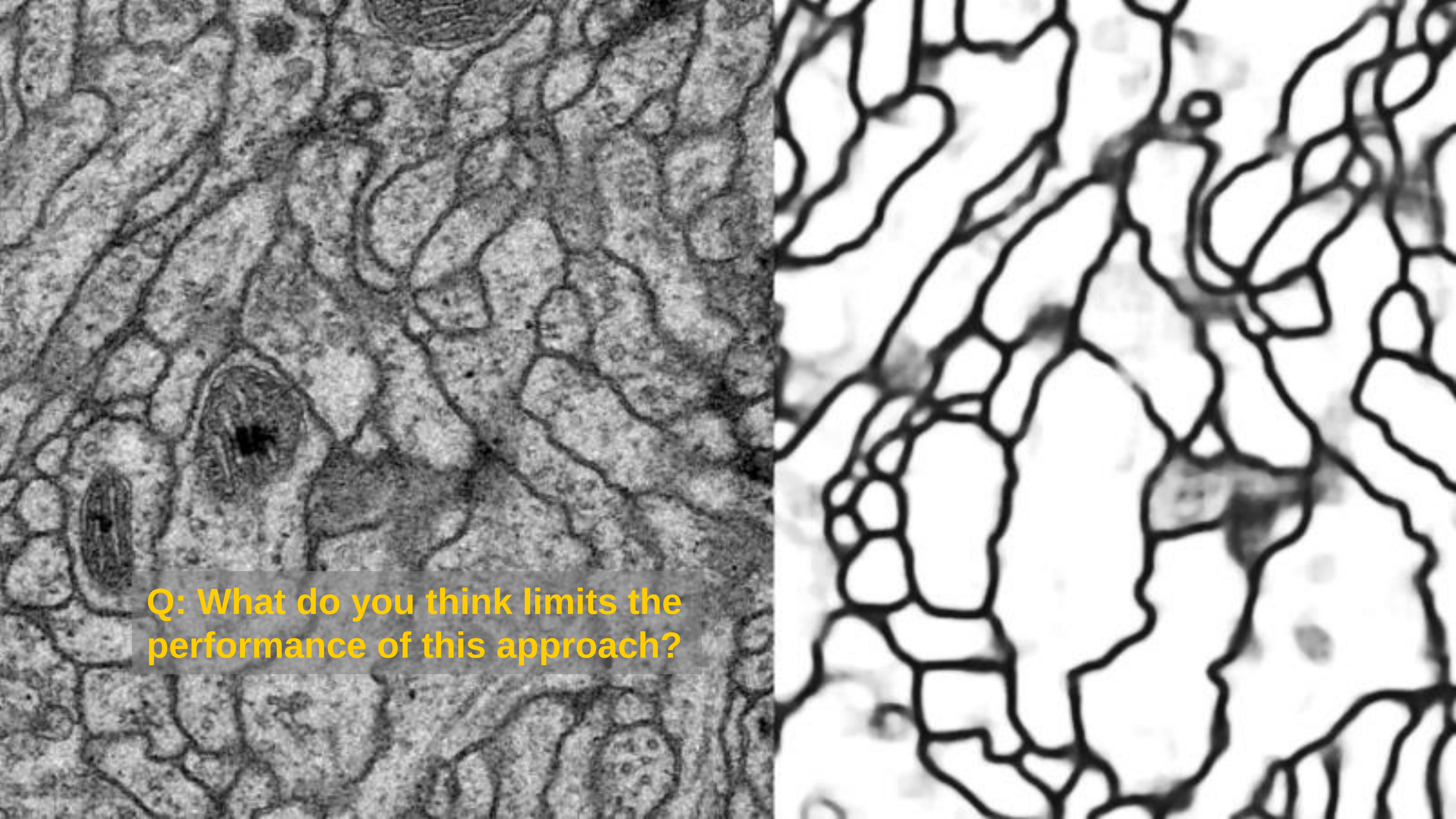


- Skewed class balance
  → Polynomial calibration

- Averaging over the output of 4 slightly different network architectures

You may find it ironic, that this paper uses Artificial Neural Networks to analyze Anatomical Neural Networks.

Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. NeurIPS (pp. 2843-2851).

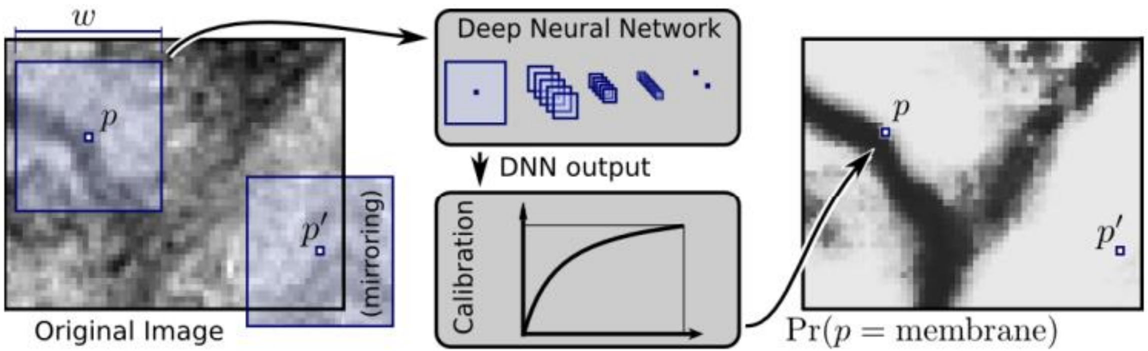

Q: What do you think limits the performance of this approach?

The Sliding-window Approach to Segmentation

"Inherent tension between semantics and location"

→ Global information: Resolves what

→ Local information: Resolves where



Sliding-window

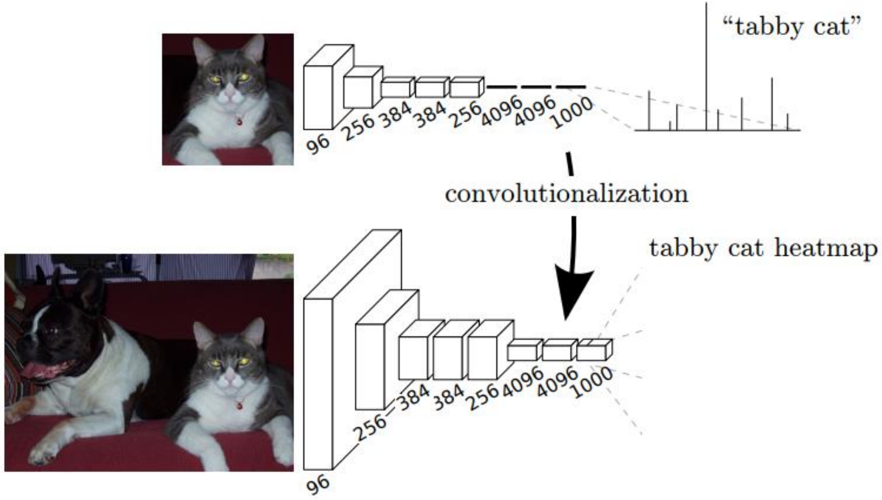
- Small models restricting capacity and receptive fields
- Application to every patch → Slow
- Pooling somewhat prevents "fast change" in output signal → Blurry edges

Ciresan, D., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. NeurIPS (pp. 2843-2851).

Fully Convolutional Neural Networks and the U-Net

An interesting observation

- Fully connected layers are no different from conv layers
- Convolutionize FC layers
  → Kernels that cover entire input region



→ Spatially resolved classification

→ Substantial speedups during both forward and backward pass

Q: Is this all?

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Fully Convolutional Neural Networks and the U-Net

Remember: “Inherent tension between semantics and location”

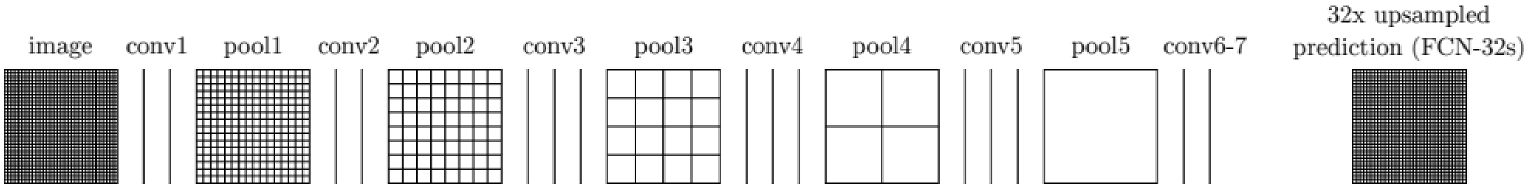
→ Global information: Resolves what

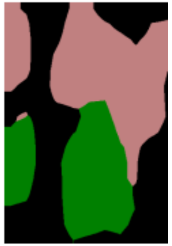
→ Local information: Resolves where

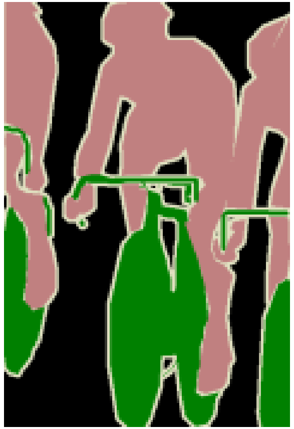
Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Fully Convolutional Neural Networks and the U-Net

Remember: “Inherent tension between semantics and location”





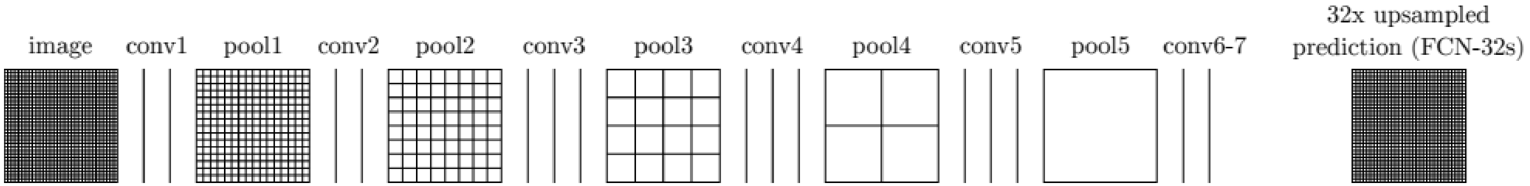


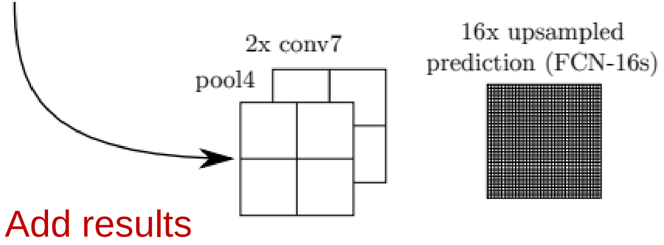
Target

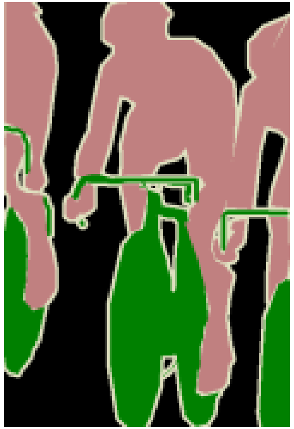
Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Fully Convolutional Neural Networks and the U-Net

Remember: “Inherent tension between semantics and location”





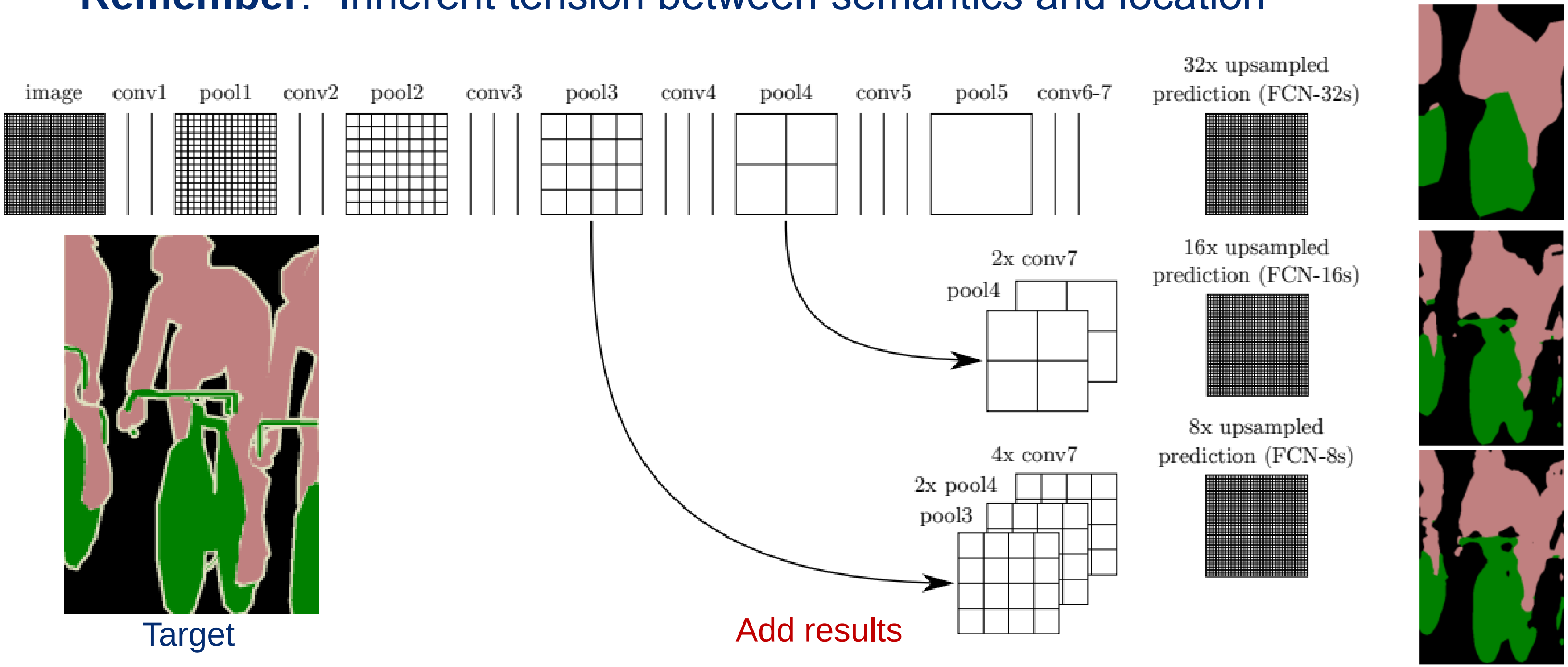


Target

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Fully Convolutional Neural Networks and the U-Net

Remember: "Inherent tension between semantics and location"



Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Fully Convolutional Neural Networks and the U-Net

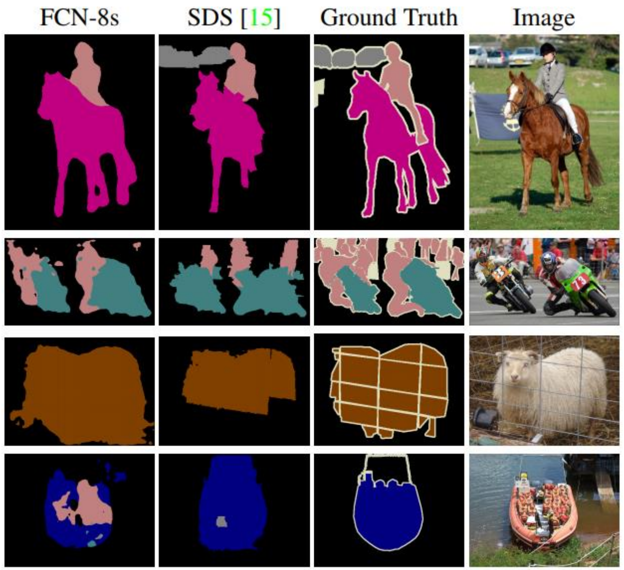
Remember: “Inherent tension between semantics and location”

Combining layers of feature hierarchy to refine spatial precision!

Can be trained end-to-end, achieves state-of-the-art results!

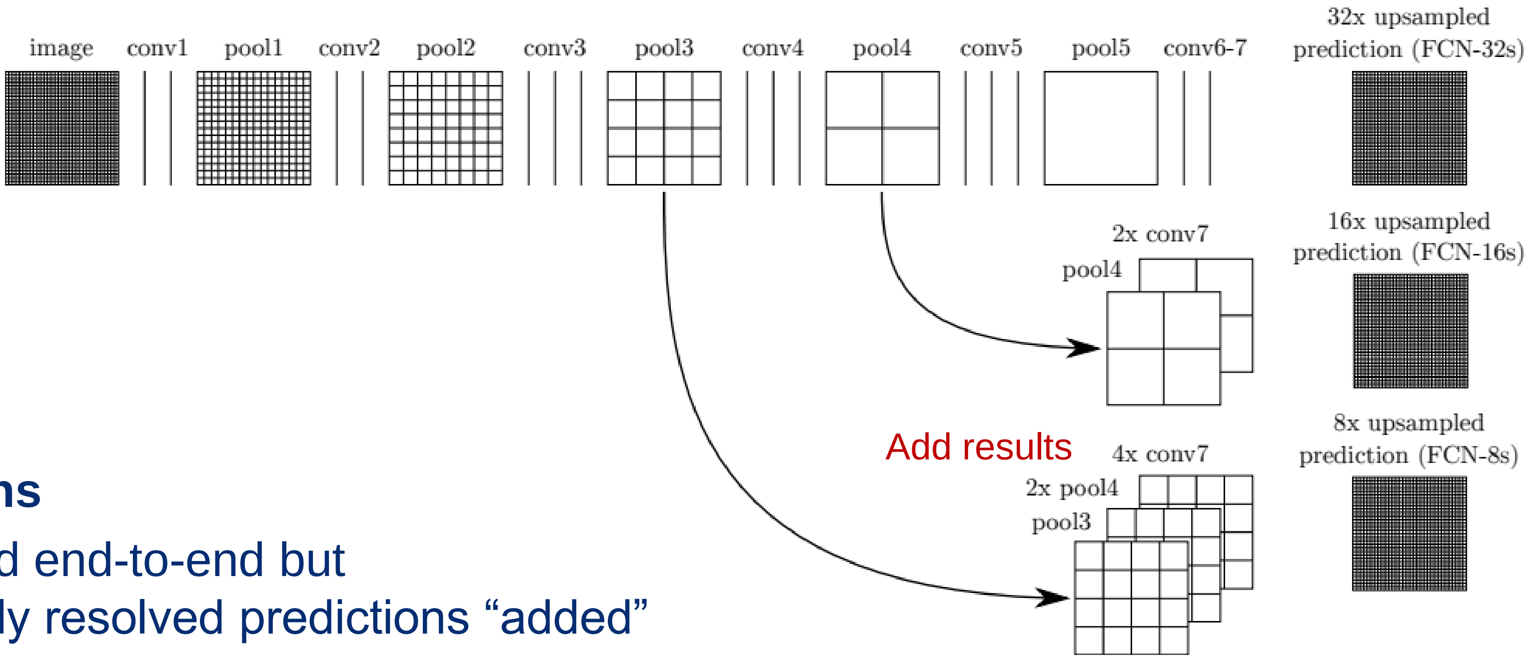
Table 2. Comparison of skip FCNs on a subset[7] of PASCAL VOC 2011 segval. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**



Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition (pp. 3431-3440).

Fully Convolutional Neural Networks and the U-Net



Limitations

- Trained end-to-end but spatially resolved predictions “added”
- Still fairly “low resolution”

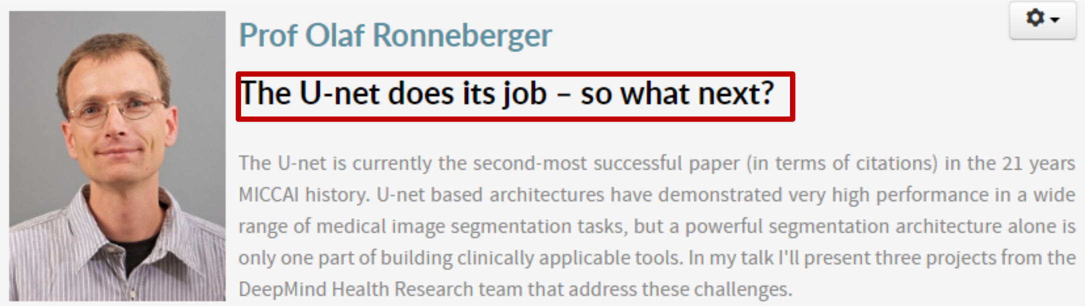
Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. MICCAI (pp. 234-241). Springer, Cham.

Fully Convolutional Neural Networks and the U-Net

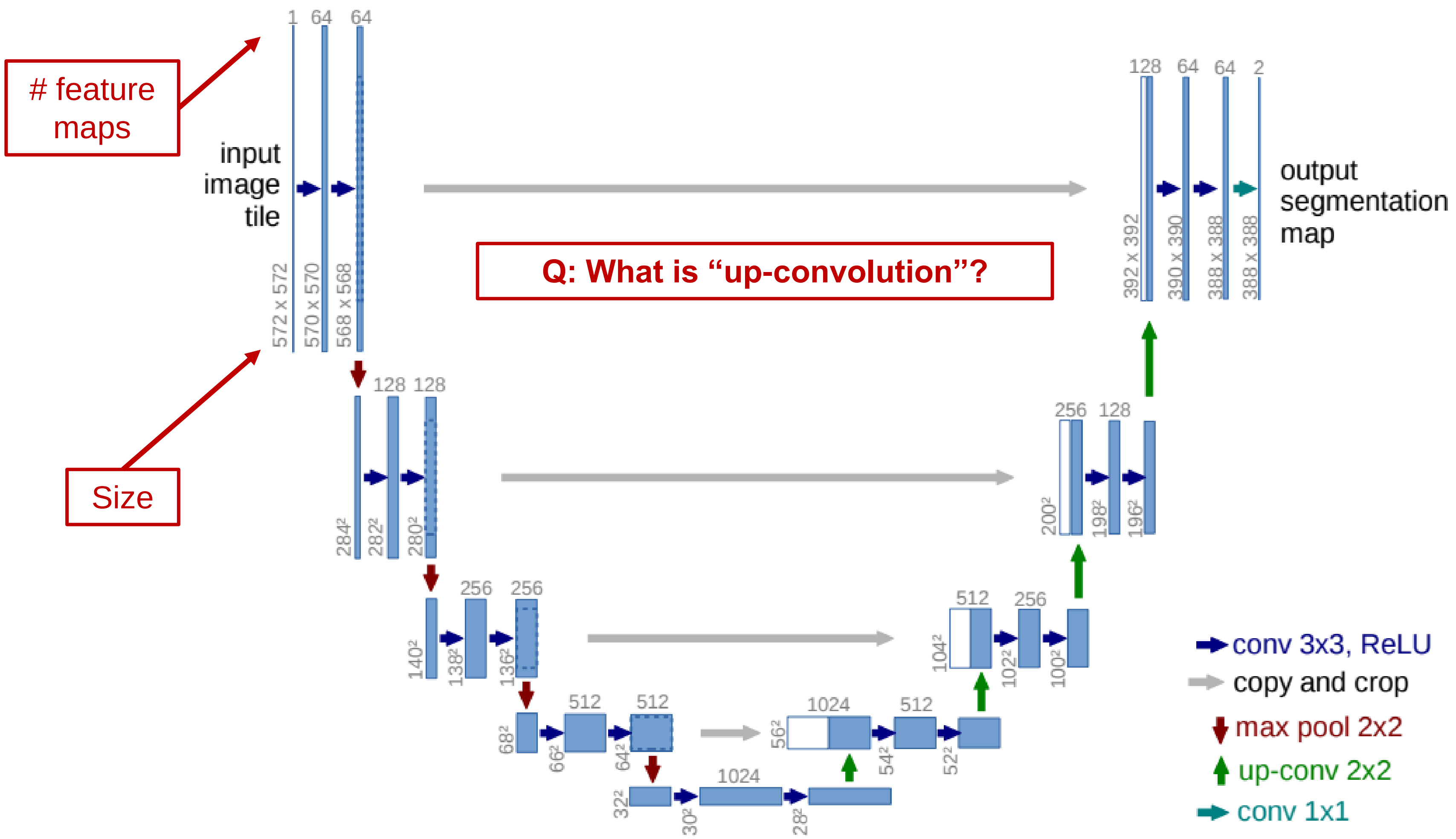
Enter the U-Net

- Designed for medical image segmentation
- Anecdotally:
  - Developed as “baseline” method adapted from Long et al. Fully Convolutional Networks
  - Ended up out-performing all “actually developed” methods
  - Paper then focused on the baseline method
- Now the most cited paper of MICCAI (>17k on google scholar as of 08/30/20)
  The second most cited: Frangi, A. et al. 1998 (3908 as of same day)

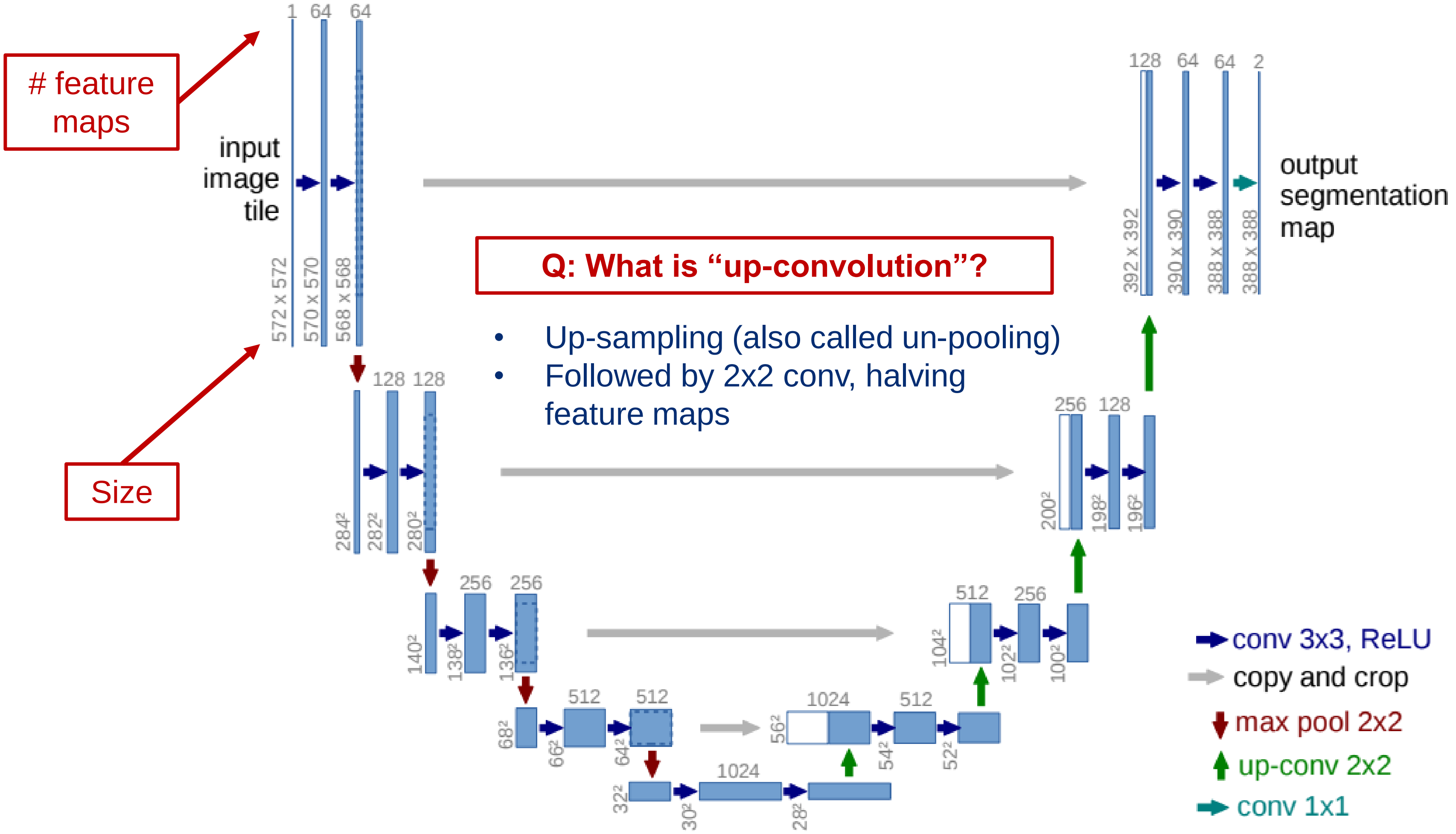
→ **Why all the fuzz?**



Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. MICCAI (pp. 234-241). Springer, Cham.



Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. MICCAI (pp. 234-241). Springer, Cham.

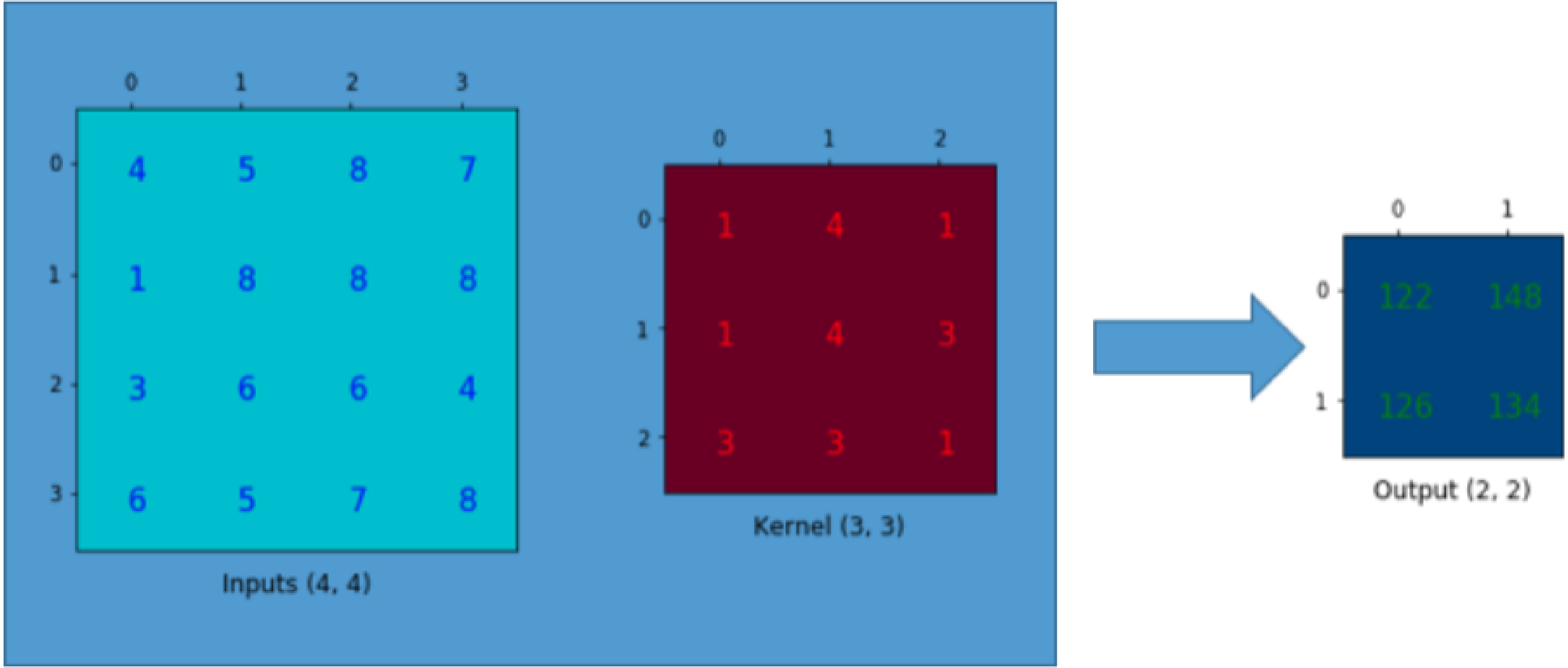


Q: What is “up-convolution”?

- Up-sampling (also called un-pooling)
- Followed by 2x2 conv, halving feature maps

Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. MICCAI (pp. 234-241). Springer, Cham.

A Short Aside: Transpose Convolution for Upsampling



Based on this blog post and this one, too. This also contains references to scientific papers that provide further insight.

A Short Aside: Transpose Convolution for Upsampling

Towards transposed convolutions: Re-arranging the kernel matrix

Convolution is linear, so can be represented by matrix multiplication!

	0	1	2
0	1	4	1
1	1	4	3
2	3	3	1

Kernel (3, 3)

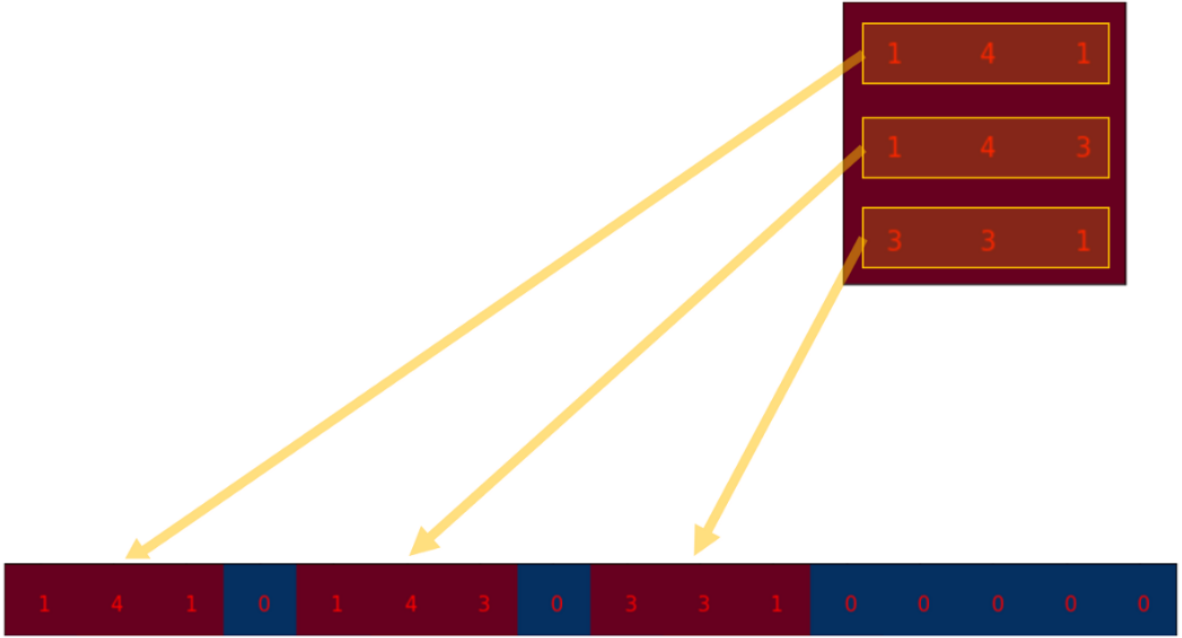
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0	0
1	0	1	4	1	0	1	4	3	0	3	3	1	0	0	0	0
2	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1	0
3	0	0	0	0	0	1	4	1	0	1	4	3	0	3	3	1

Convolution Matrix (4, 16)

Based on this blog post and this one, too. This also contains references to scientific papers that provide further insight.

A Short Aside: Transpose Convolution for Upsampling

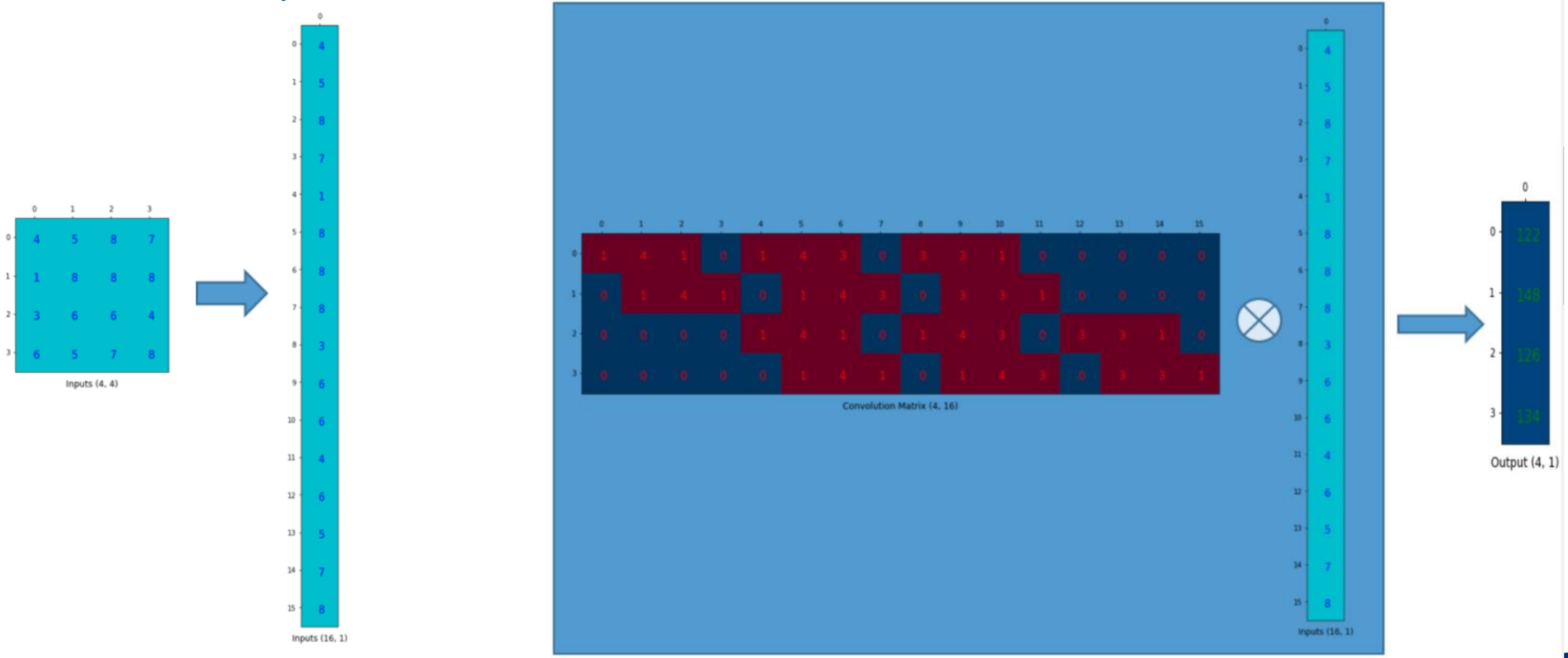
Linearization process in greater detail



Based on this blog post and this one, too. This also contains references to scientific papers that provide further insight.

A Short Aside: Transpose Convolution for Upsampling

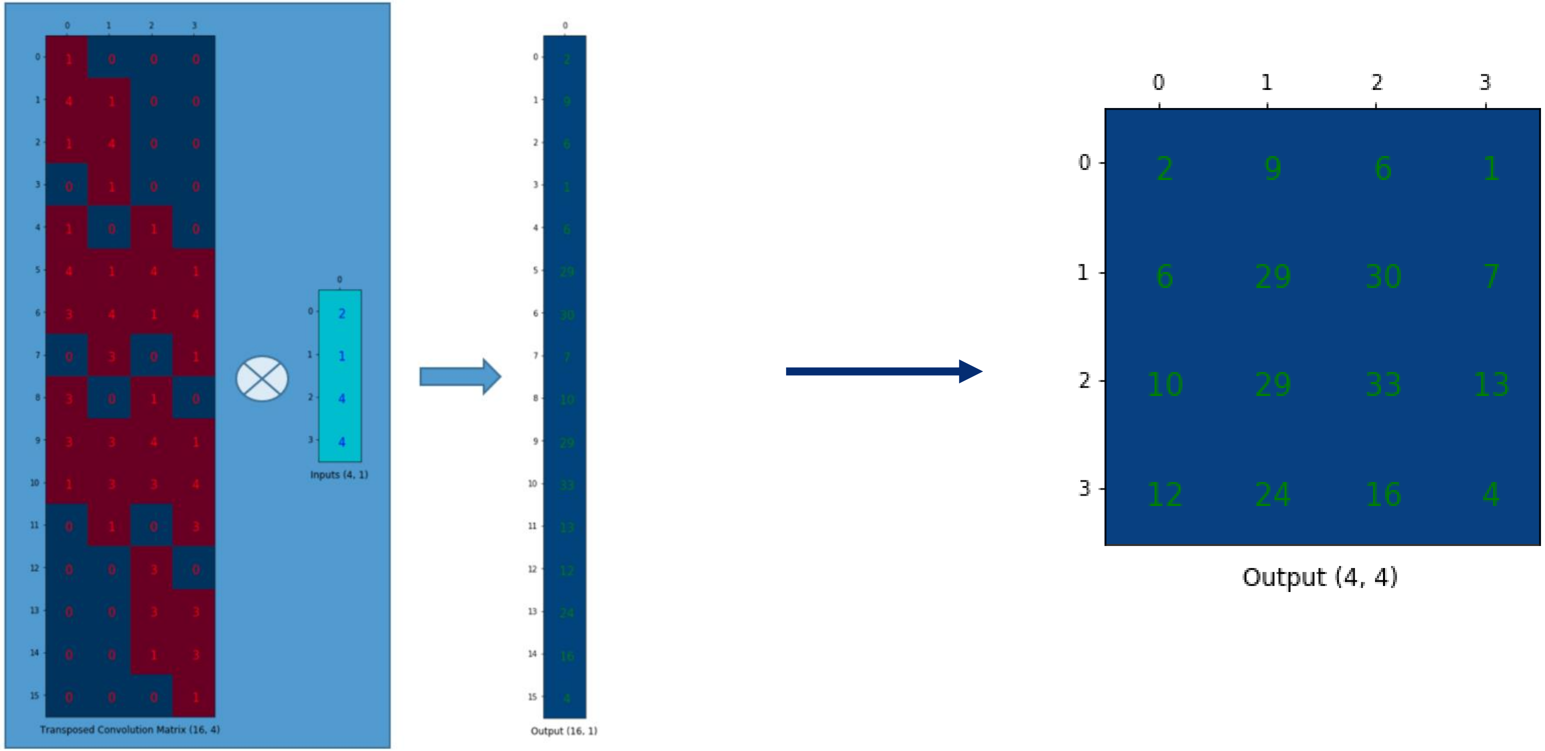
Linearized operation is now applied to the full flattened input!



Based on this blog post and this one, too. This also contains references to scientific papers that provide further insight.

A Short Aside: Transpose Convolution for Upsampling

Linearized operations can be transposed!



Based on this blog post and this one, too. This also contains references to scientific papers that provide further insight.

A Short Aside: Transpose Convolution for Upsampling

This process can generate artifact (see second blog post).

→ Separating upsampling and convolution (as in U-Net) can be a good idea.

Using deconvolution. *Heavy checkerboard artifacts.*

Using resize-convolution. *No checkerboard artifacts.*

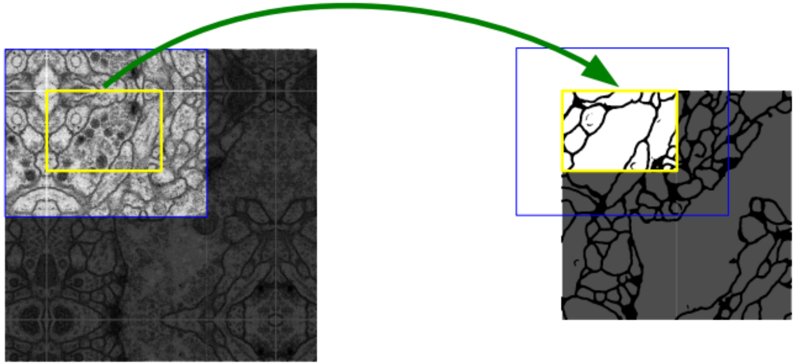
Based on [this blog post] and [this one, too]. This also contains references to scientific papers that provide further insight.

Fully Convolutional Neural Networks and the U-Net

Observations

- Fully convolutional neural network: No fully connected layers

- However, input size still important due to “cropping”
  - Skip-ahead connections must crop to central region
  - Border regions in image are processed via “mirroring”



Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. MICCAI (pp. 234-241). Springer, Cham.

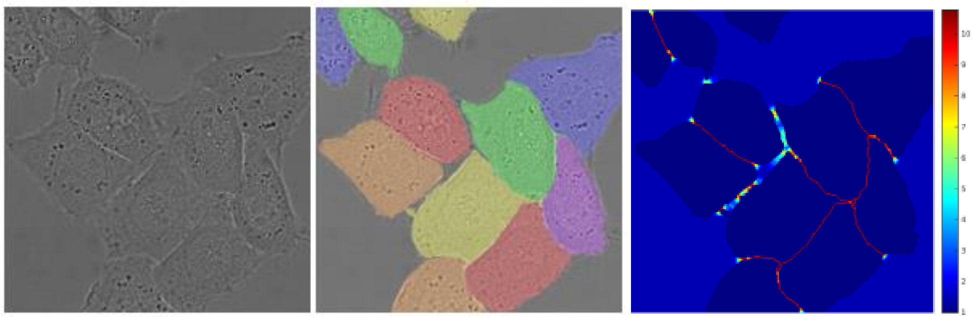
Fully Convolutional Neural Networks and the U-Net

Observations

- Fully convolutional neural network: No fully connected layers

- However, input size still important due to "cropping"
  - Skip-ahead connections must crop to central region
  - Border regions in image are processed via "mirroring"

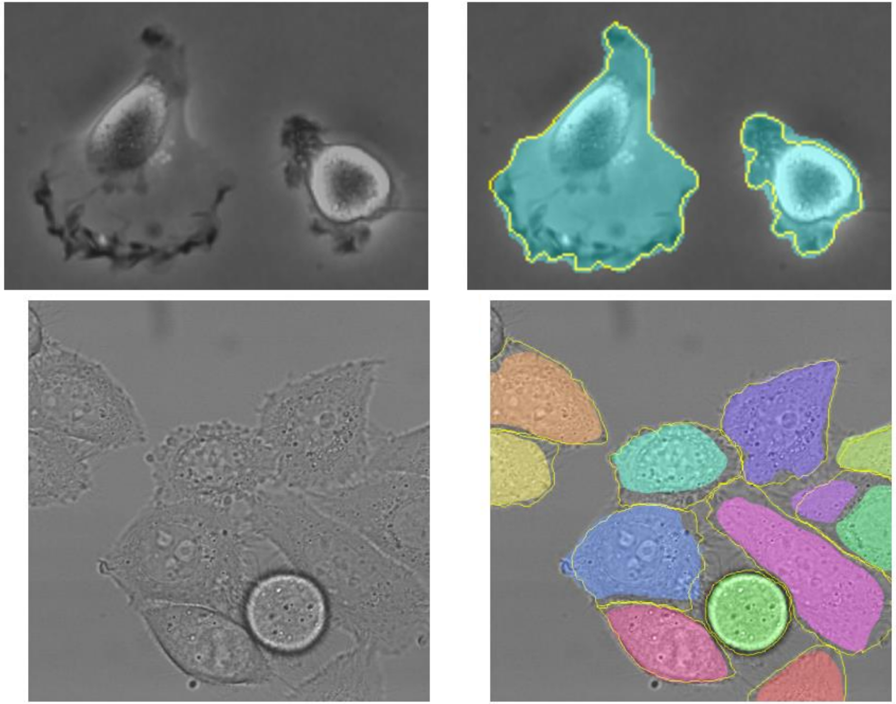
- End-to-end training
  - Penalized cross-entropy loss
  - Weights to account for small/narrow structures and class imbalance



Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. MICCAI (pp. 234-241). Springer, Cham.

The U-Net: Why all the Fuzz?

On multiple challenges (some shown here) U-Net outperformed all previous methods by a large margin!



Why?

- Improvements of CNN architecture
- Meaningful, massive data augmentation

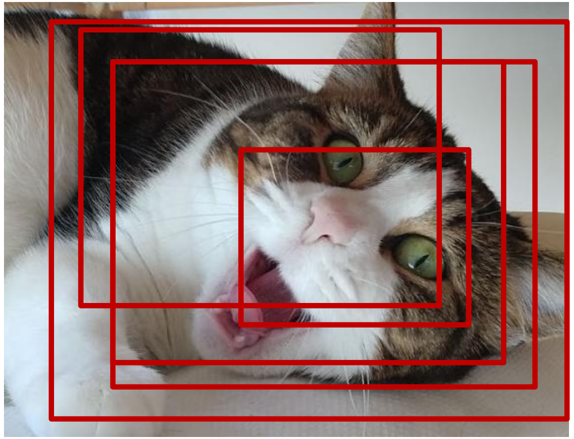
Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

Image Transformations to Use for Augmentation

Rule of thumb
Every transformation that yields a **valid** image.

Examples: All these are random (within reasonable ranges)

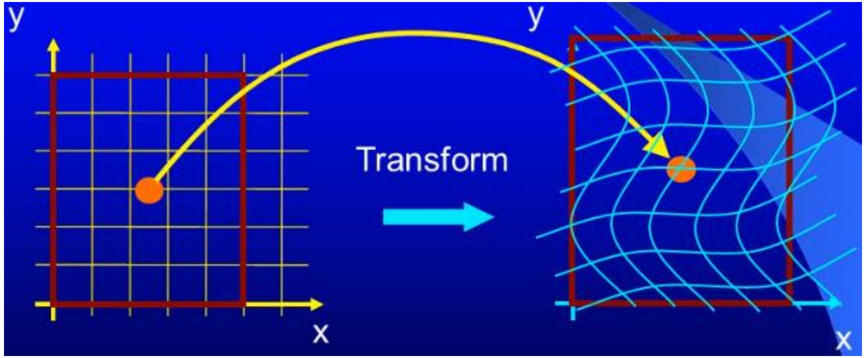
- Horizontal / vertical flips
- Rotations and translations
- Noise (!)
- Scaling
- Cropping
- Color variations
- Distortions



→ We will see an interesting example of this soon!

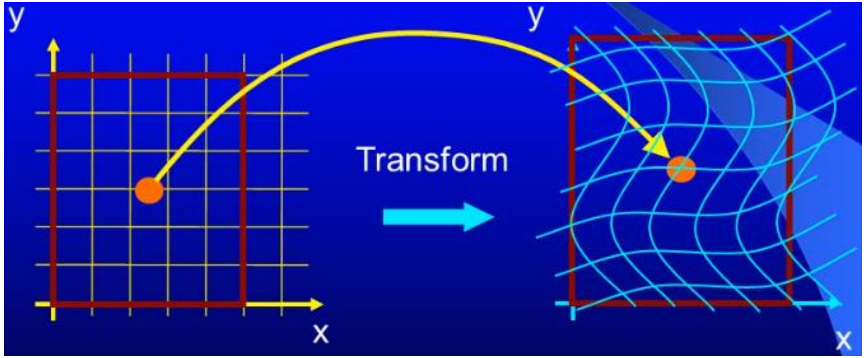
State-of-the-art Performance via Data Augmentation

Remember: All transformations that yield a **valid** image



- Horizontal / vertical flips
- Rotations and translations
- Noise (!)
- Scaling
- Cropping
- Color variations
- **Distortions**
  - **B-spline transformations**

State-of-the-art Performance via Data Augmentation



B-spline transformations

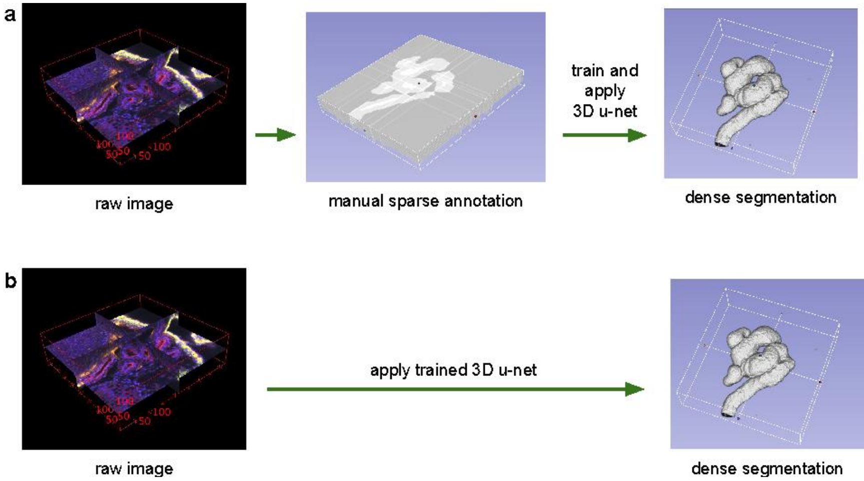
- Random elastic transformations
- Coarse 3x3 grid centered on image
- Random displacements of control points
  Sampled from zero mean Gaussian distribution with 10px standard deviation
- Pixel-level displacements then via bicubic interpolation

Beyond the Initial U-Net

- Fully convolutional networks (particularly with skip connections) define the state-of-the-art in segmentation

- Probabilistic approaches for ambiguous images

- 3D approaches exist
  - 3D U-Net (sparse annotations)
  - V-net



Network Architectures

Questions?