

# Image Classification

# Convolutional networks - Why

- Convolutions
  - Reduce parameters
  - Capture shift-invariance: location of patch in image should not matter
- Subsampling
  - Allows greater invariance to deformations
  - Allows the capture of large patterns with small filters


# How to do machine learning

- Create training / validation sets
- Identify loss functions
- Choose hypothesis class
- Find best hypothesis by minimizing training loss



# How to do machine learning

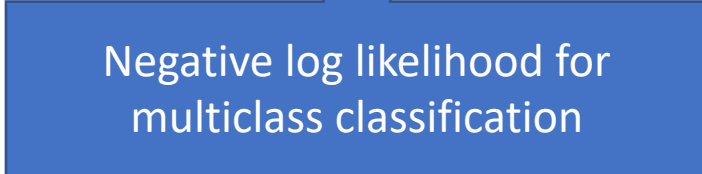

- Create training / validation sets
- Identify loss functions
- Choose hypothesis class
- Find best hypothesis by minimizing training loss



Multiclass  
classification  
!!

$$h(x) = \mathbf{s} \quad \hat{p}(y = k|x) \propto e^{s_k} \quad \hat{p}(y = k|x) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

$$L(h(x), y) = -\log \hat{p}(y|x)$$



Negative log likelihood for  
multiclass classification

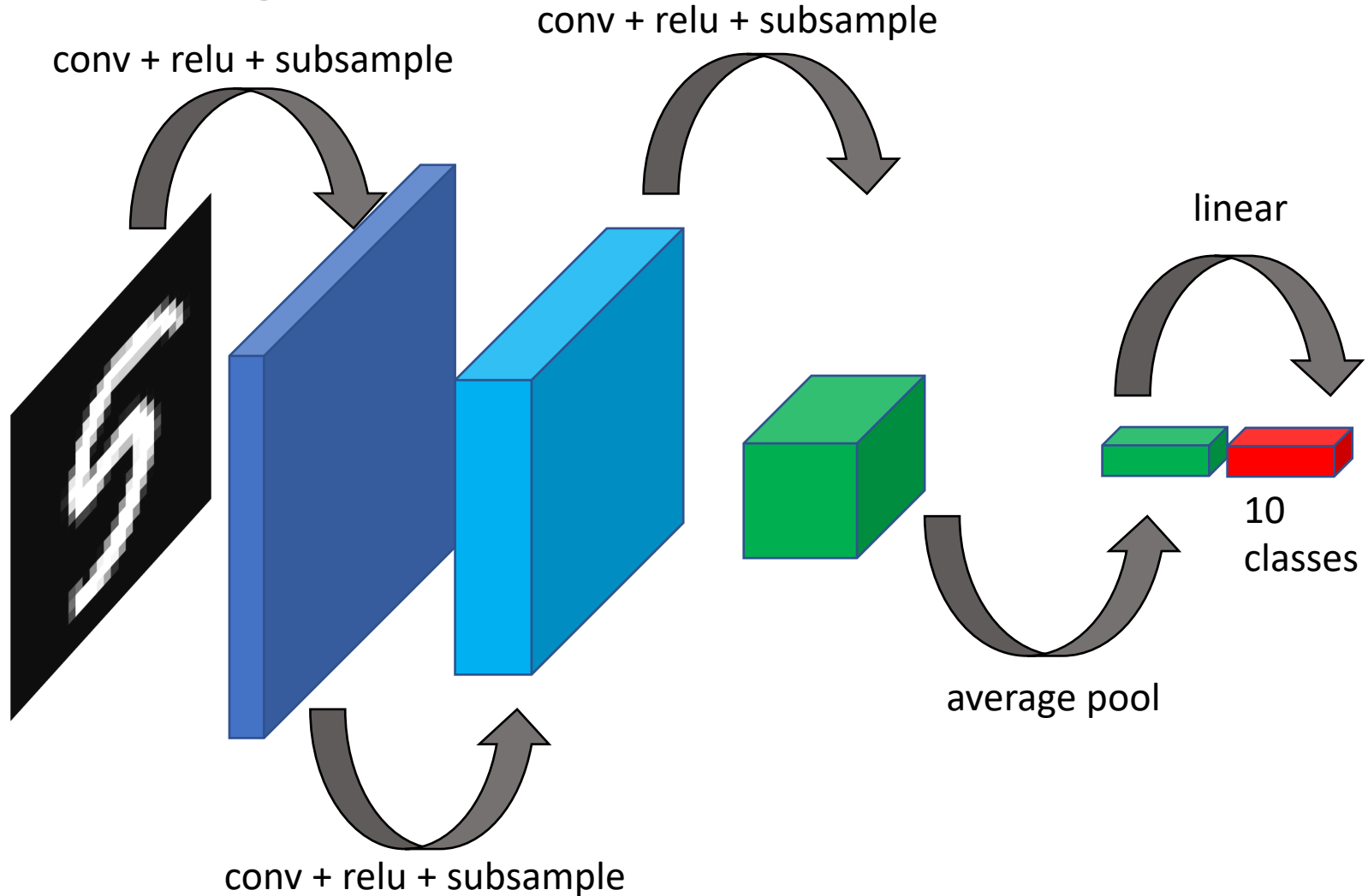
# Negative log likelihood for multiclass classification

$$L(h(x), y) = -\log \hat{p}(y|x)$$

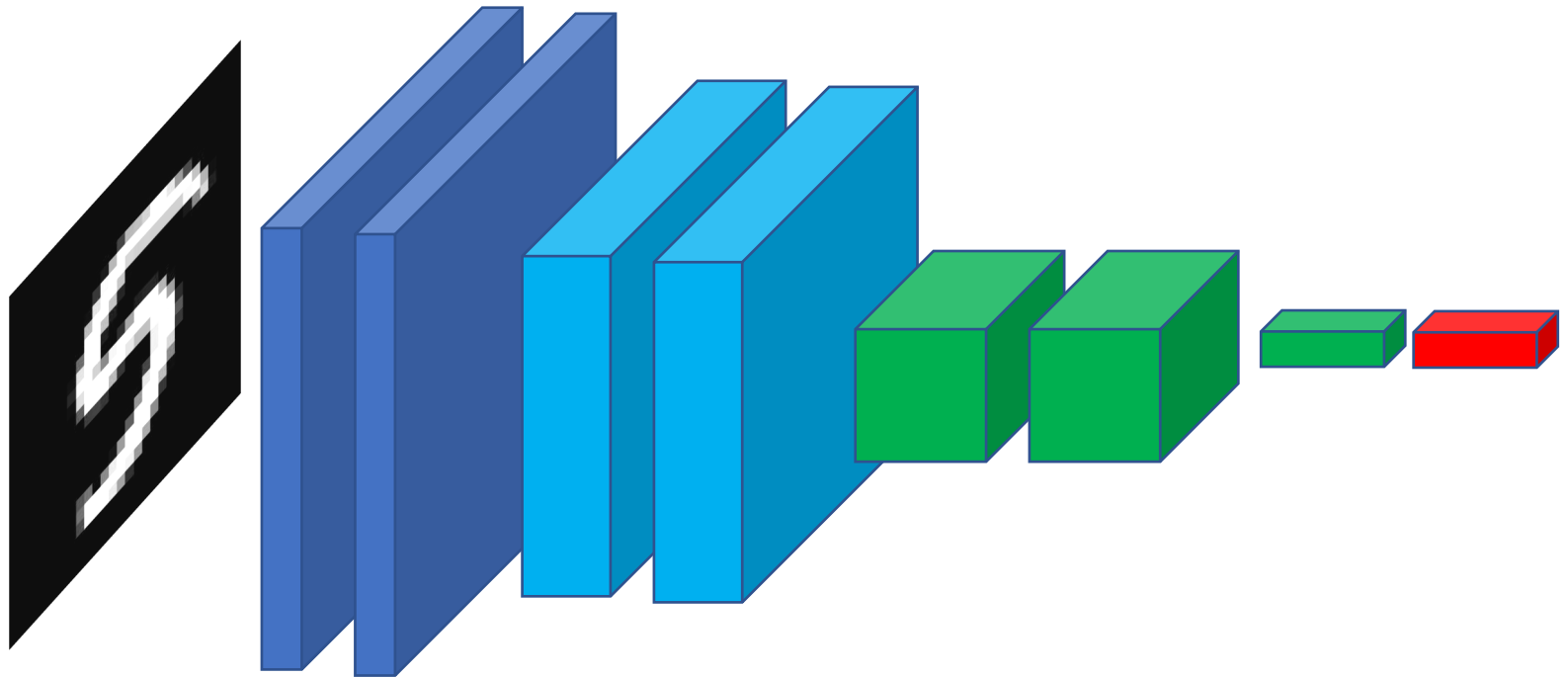
- Often represent label as a “one-hot” vector  $\mathbf{y}$ 
  - $\mathbf{y} = [0, 0, \dots, 1, \dots, 0]$
  - $y_k = 1$  if label is  $k$ , 0 otherwise

$$L(h(x), \mathbf{y}) = -\sum_k y_k \log \hat{p}(y = k|x)$$

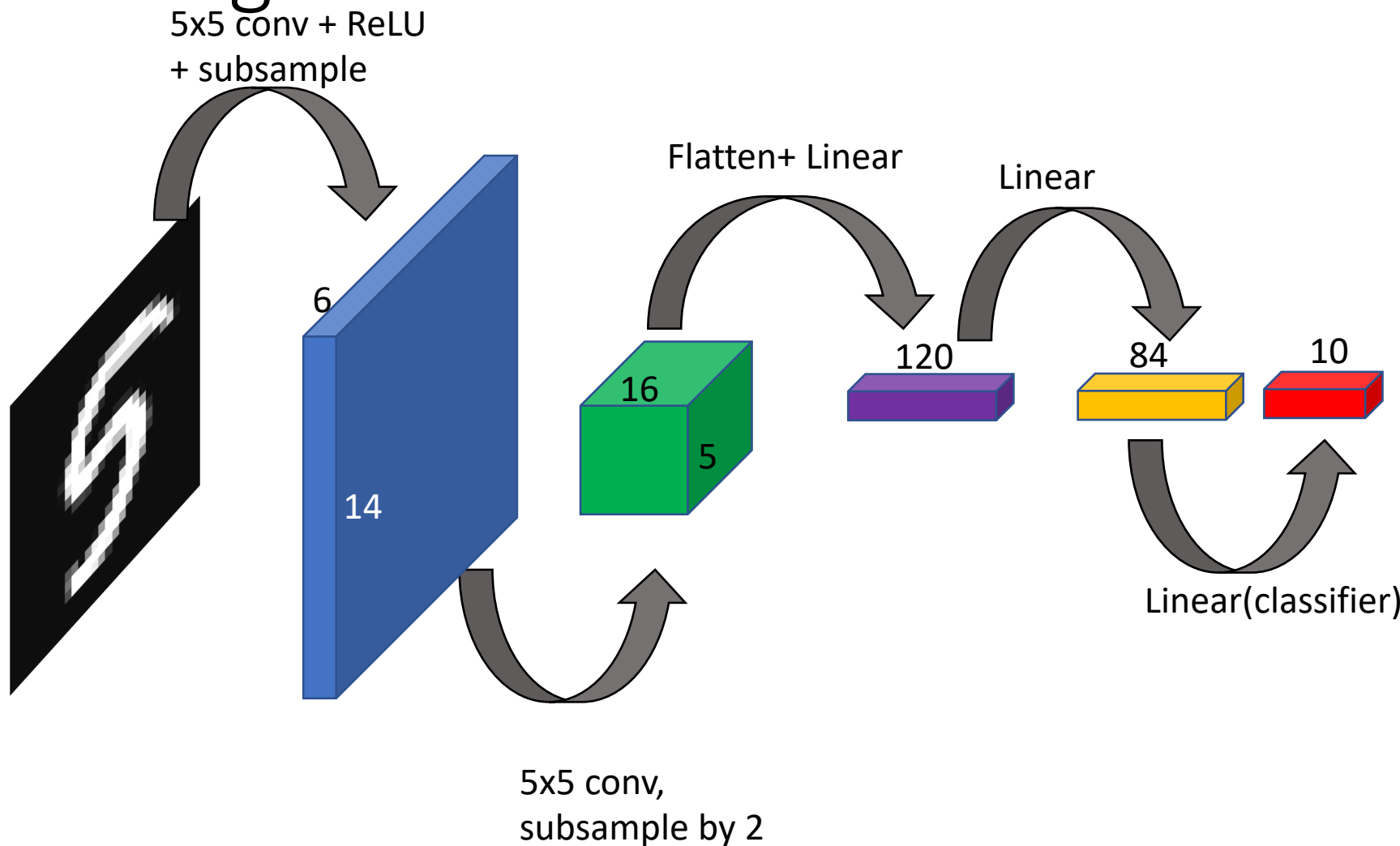
# Building a convolutional network



# Building a convolutional network

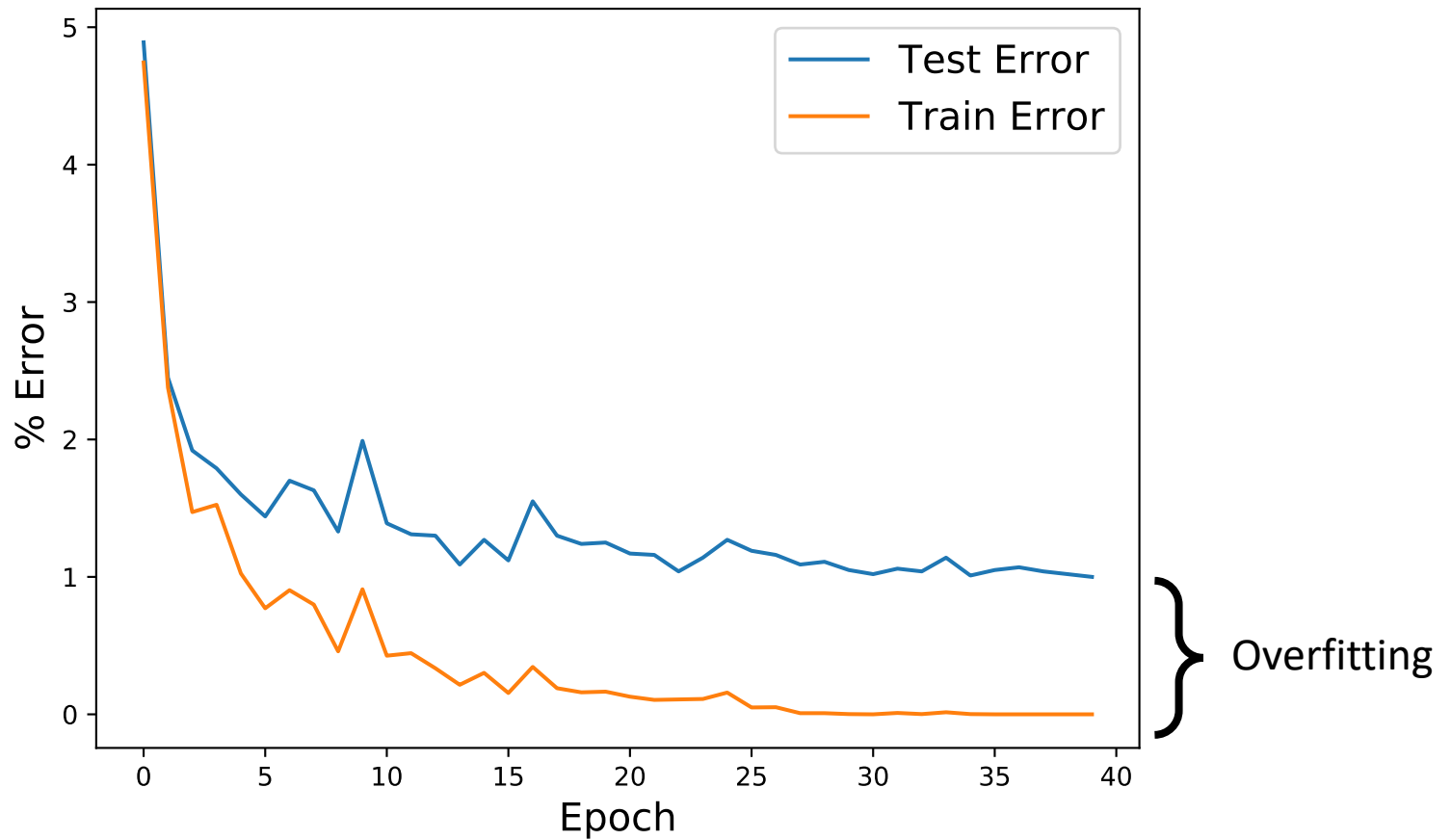


# Building a convolutional network





# Training the network



# Controlling overfitting in convolutional networks

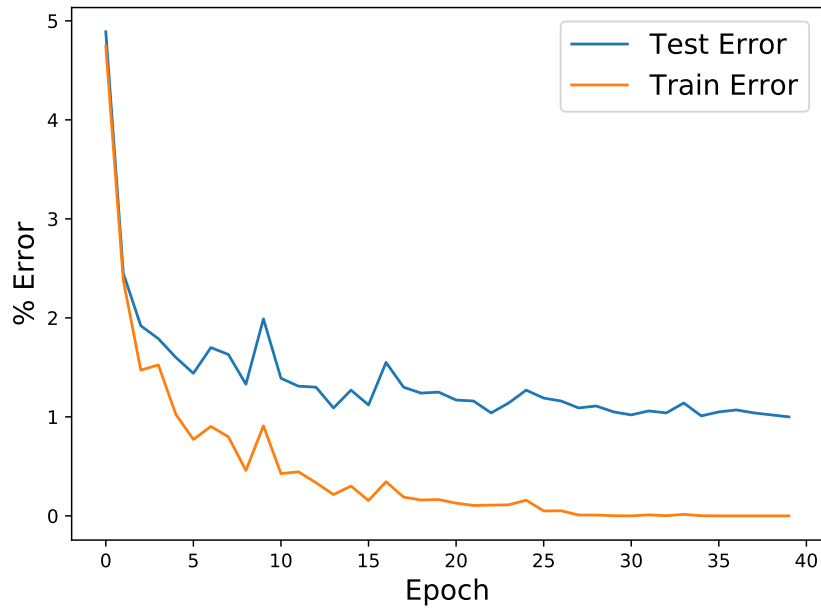
- Reduce parameters?
- Increase dataset size?
  - Automatically by jittering examples - “Data augmentation”



# Controlling overfitting in convolutional networks

- Dropout: Internally create data augmentations
  - Randomly zero out some fraction of values before a layer
  - Can be thought of as per-layer data augmentation
  - Typically applied on inputs to linear layers (since linear layers have tons of parameters)

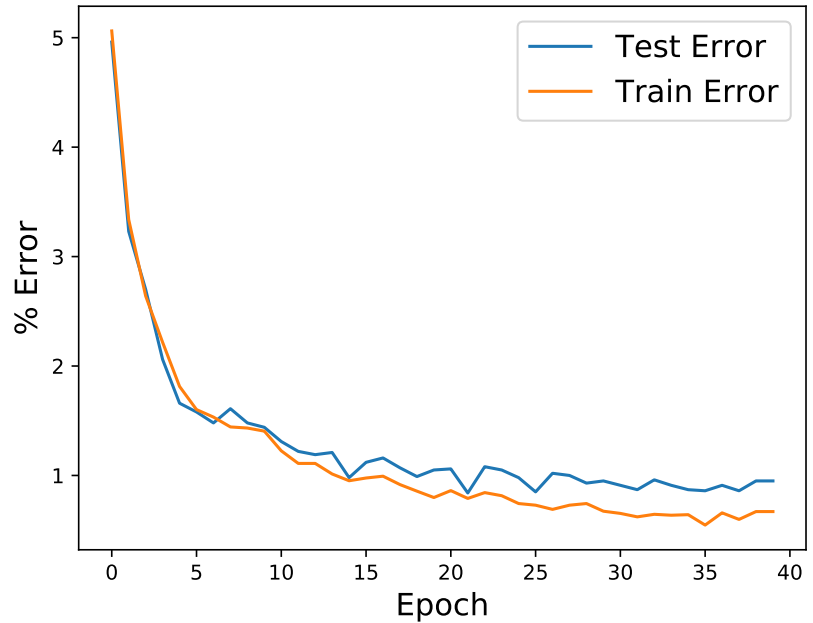
# Dropout



Without dropout

Train error: 0%

Test error: 1%



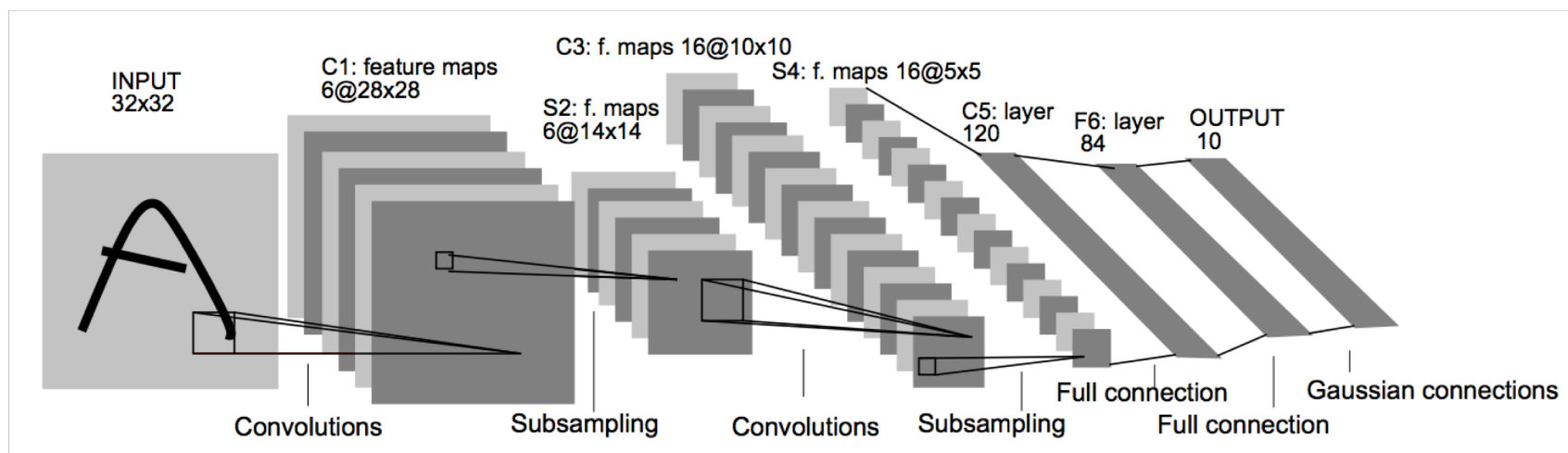
With dropout

Train error: 0.7%

Test error: 0.85%

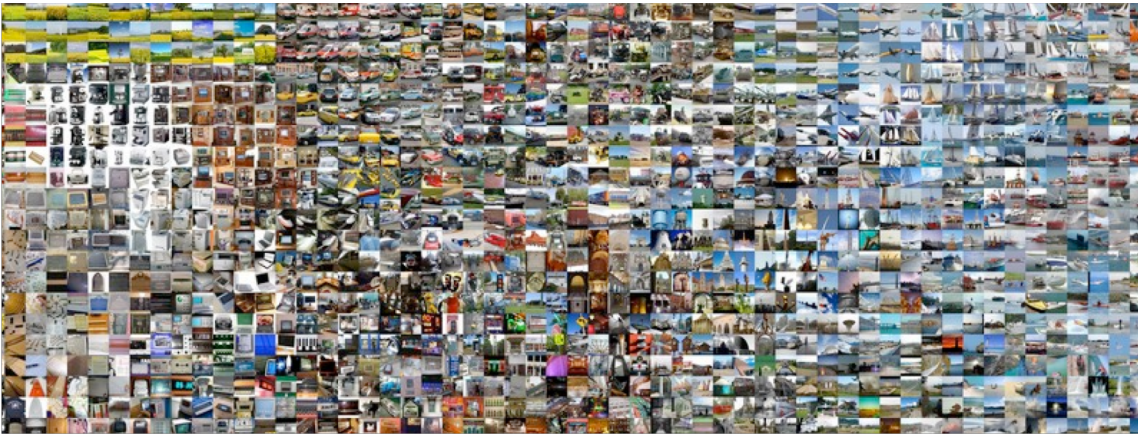
# MNIST Classification

Method	Error rate (%)
Linear classifier over pixels	12
Non-linear classifier over pixels	1.41
Linear classifier over HOG	1.44
Kernel SVM over HOG	0.79
Convolutional Network	0.95



# ImageNet

- 1000 categories
- ~1000 instances per category

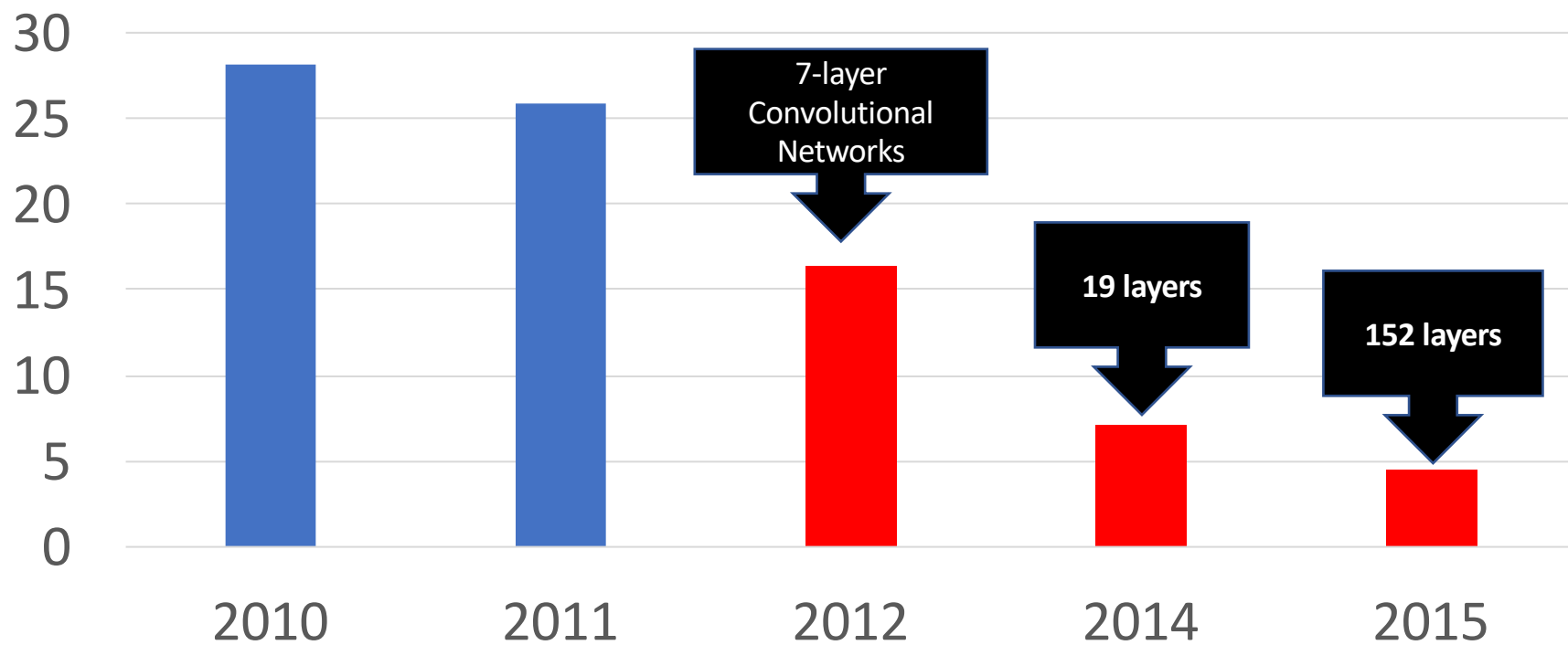


Olga Russakovsky\*, Jia Deng\*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (\* = equal contribution) **ImageNet Large Scale Visual Recognition Challenge**. *International Journal of Computer Vision*, 2015.

# ImageNet

- Top-5 error: algorithm makes 5 predictions, true label must be in top 5
- Useful for incomplete labelings

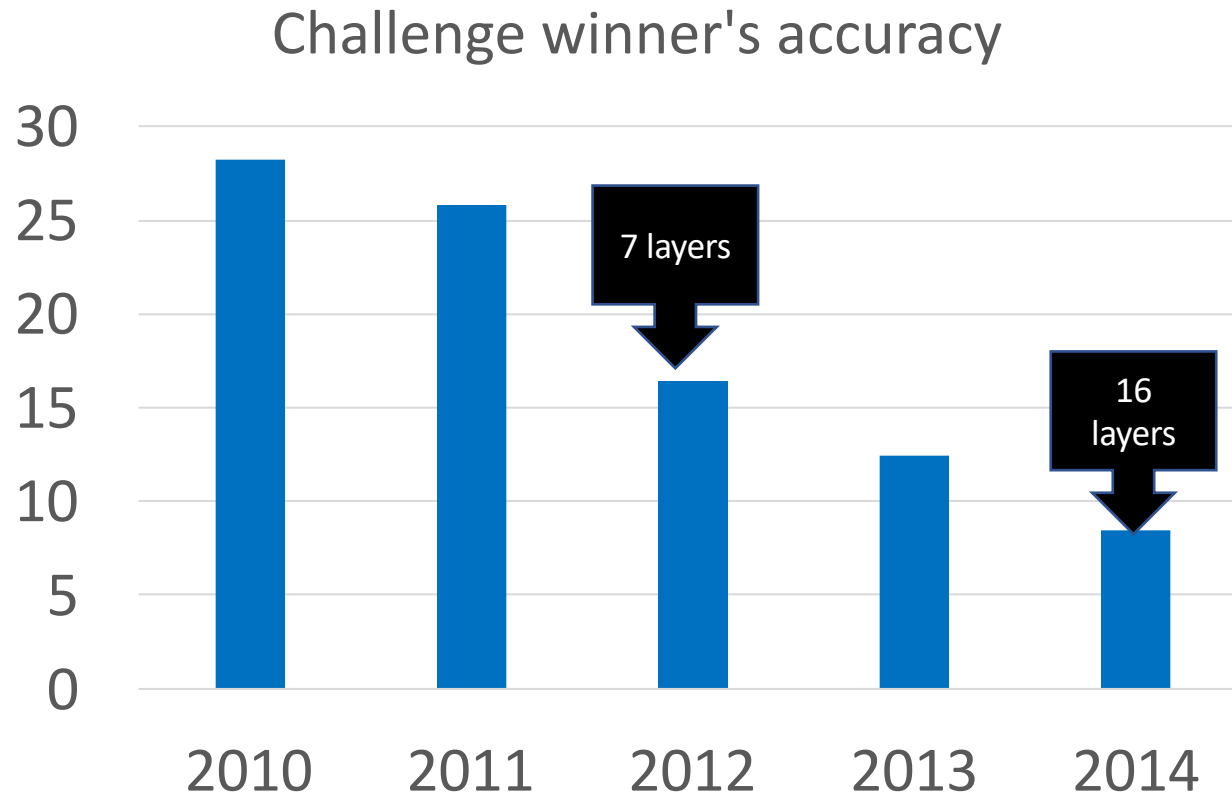
## Challenge winner's accuracy



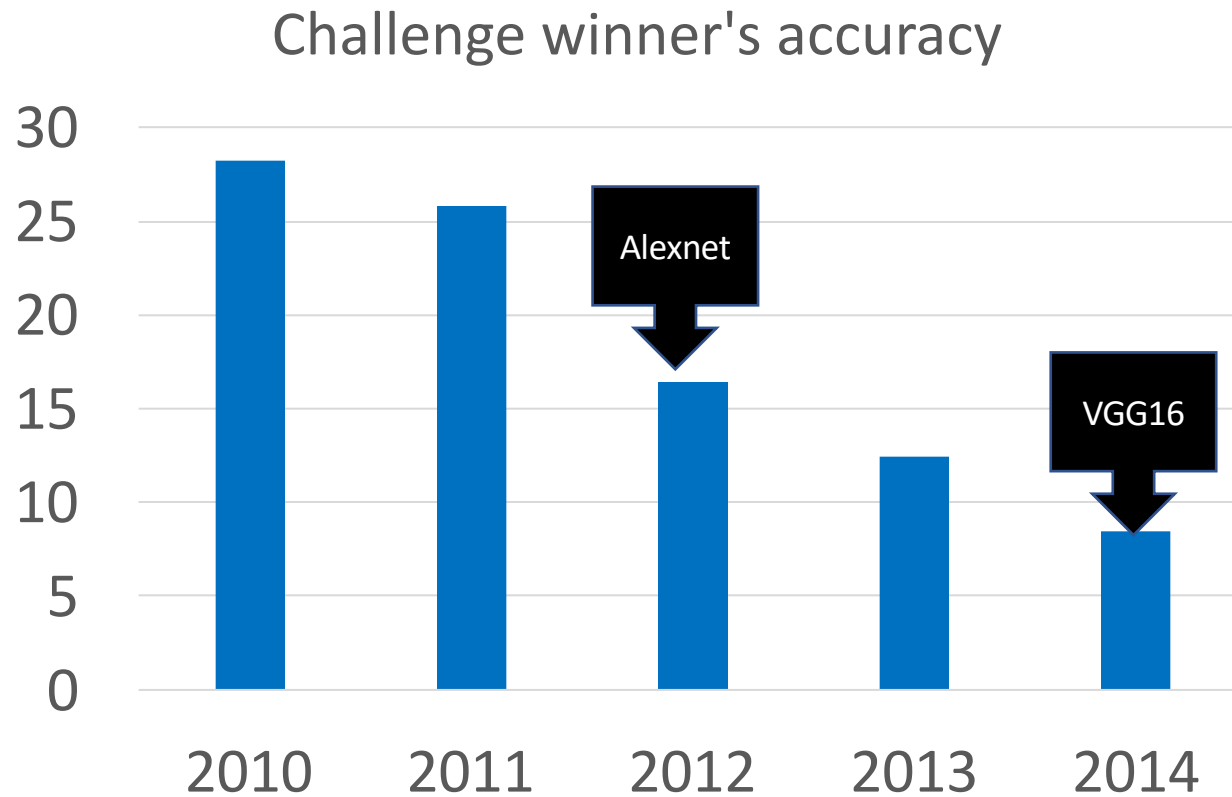


# Exploring convnet architectures

# Deeper is better



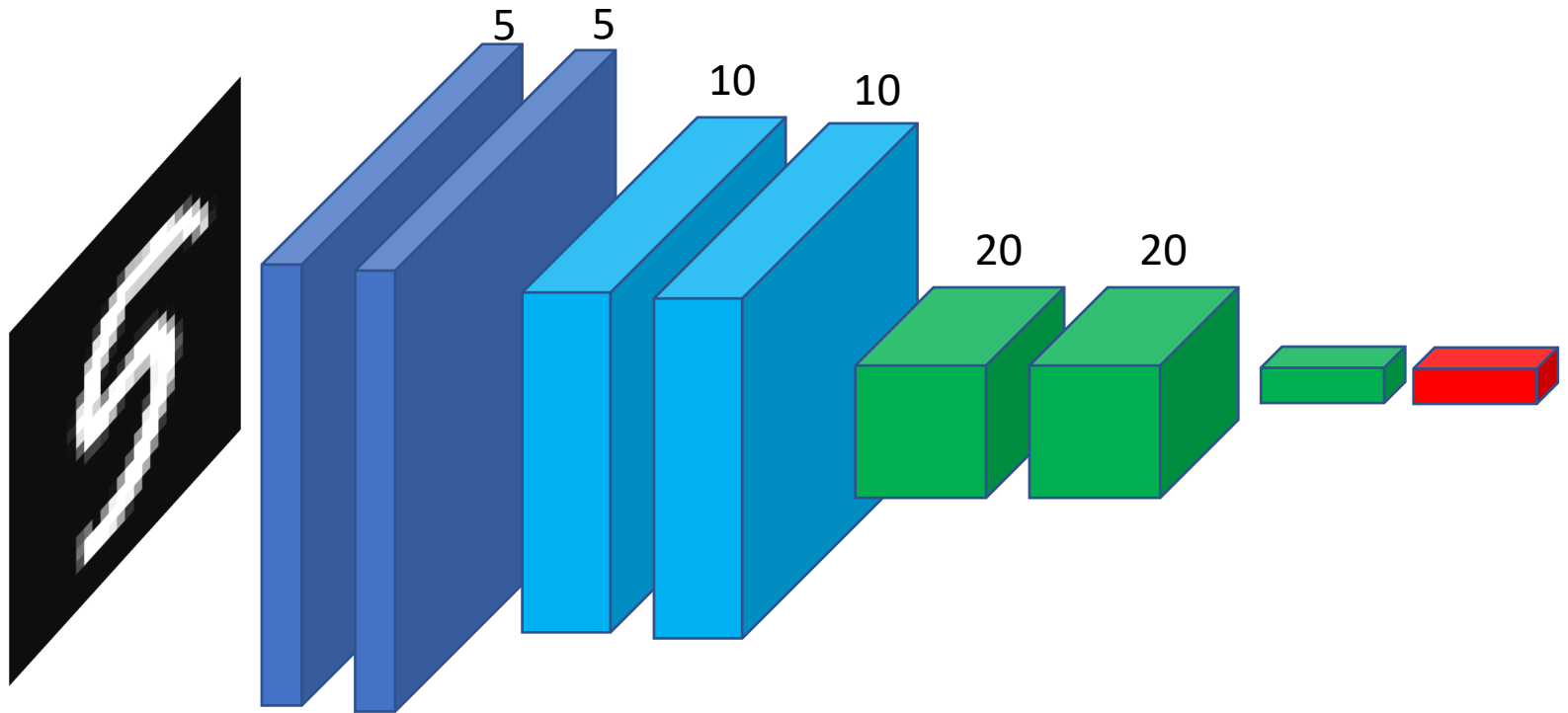
# Deeper is better



# The VGG pattern

- Every convolution is 3x3, padded by 1
- Every convolution followed by ReLU
- ConvNet is divided into “stages”
  - Layers within a stage: no subsampling
  - Subsampling by 2 at the end of each stage
- Layers within stage have same number of channels
- Every subsampling → double the number of channels

# Example network



# Challenges in training: exploding / vanishing gradients

- Vanishing / exploding gradients

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}$$

- If each term is (much) greater than 1  $\rightarrow$  *explosion of gradients*
- If each term is (much) less than 1  $\rightarrow$  *vanishing gradients*

# Residual connections

- In general, gradients tend to vanish
- Key idea: allow gradients to flow unimpeded

$$z_{i+1} = f_{i+1}(z_i, w_{i+1}) \quad \frac{\partial z_{i+1}}{\partial z_i} = \frac{\partial f_{i+1}(z_i, w_{i+1})}{\partial z_i}$$

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}$$

# Residual connections

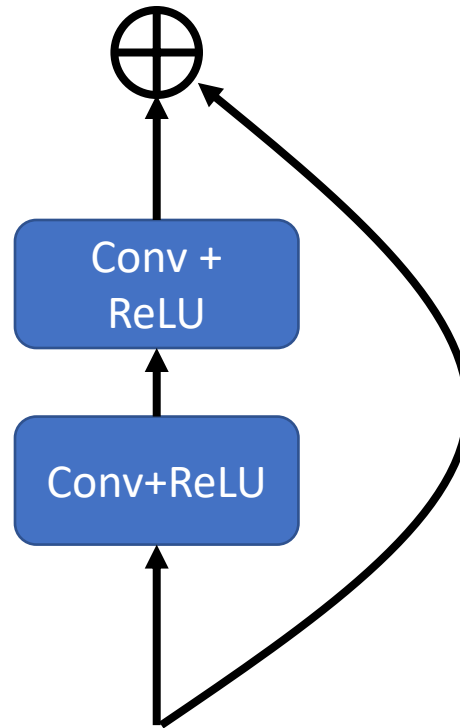
- In general, gradients tend to vanish
- Key idea: allow gradients to flow unimpeded

$$z_{i+1} = g_{i+1}(z_i, w_{i+1}) + z_i \quad \frac{\partial z_{i+1}}{\partial z_i} = \frac{\partial g_{i+1}(z_i, w_{i+1})}{\partial z_i} + I$$

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_{i+1}}{\partial z_i}$$



# Residual block



# Residual connections

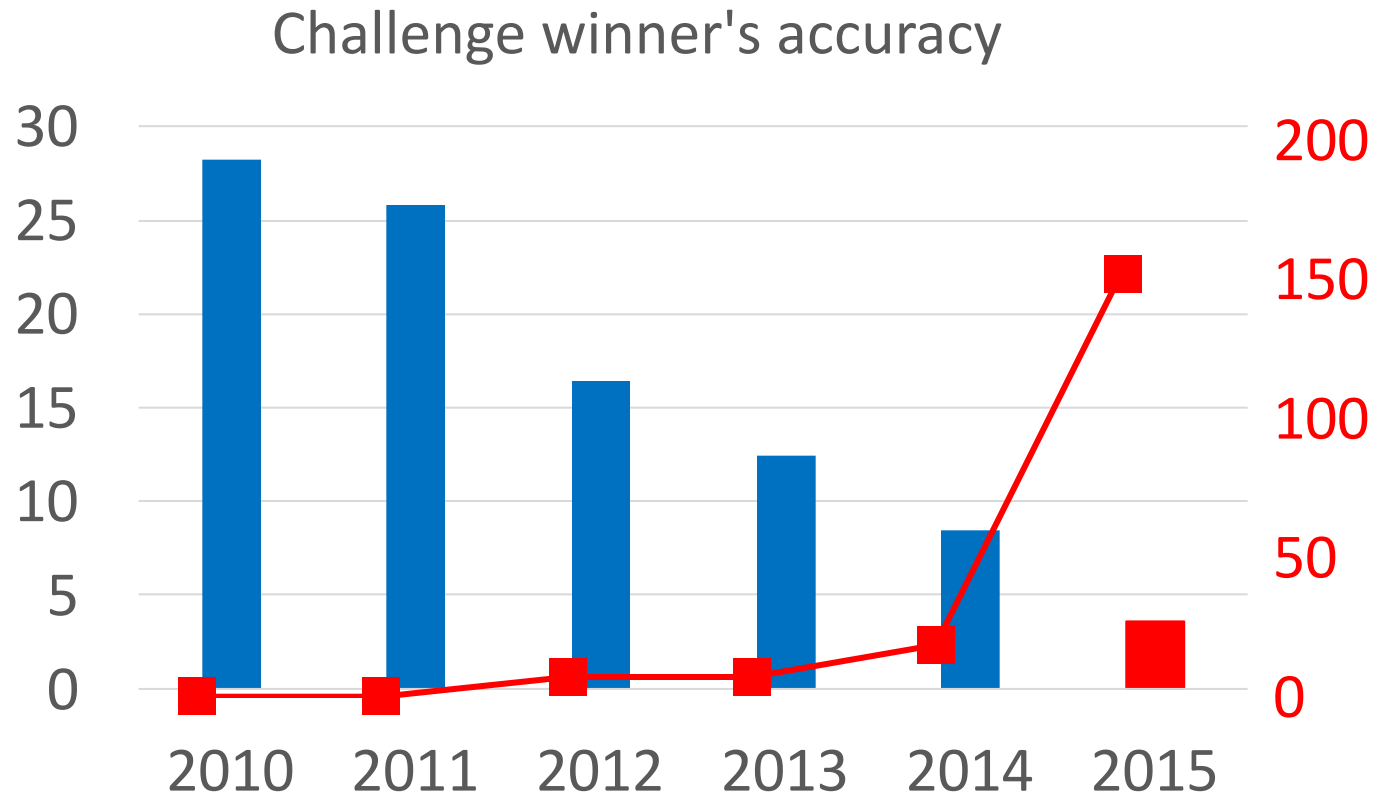
- Assumes all  $z_i$  have the same size
- True within a stage
- Across stages?
  - Doubling of feature channels
  - Subsampling
- Increase channels by 1x1 convolution
- Decrease spatial resolution by subsampling

$$z_{i+1} = g_{i+1}(z_i, w_{i+1}) + \text{subsample}(W z_i)$$

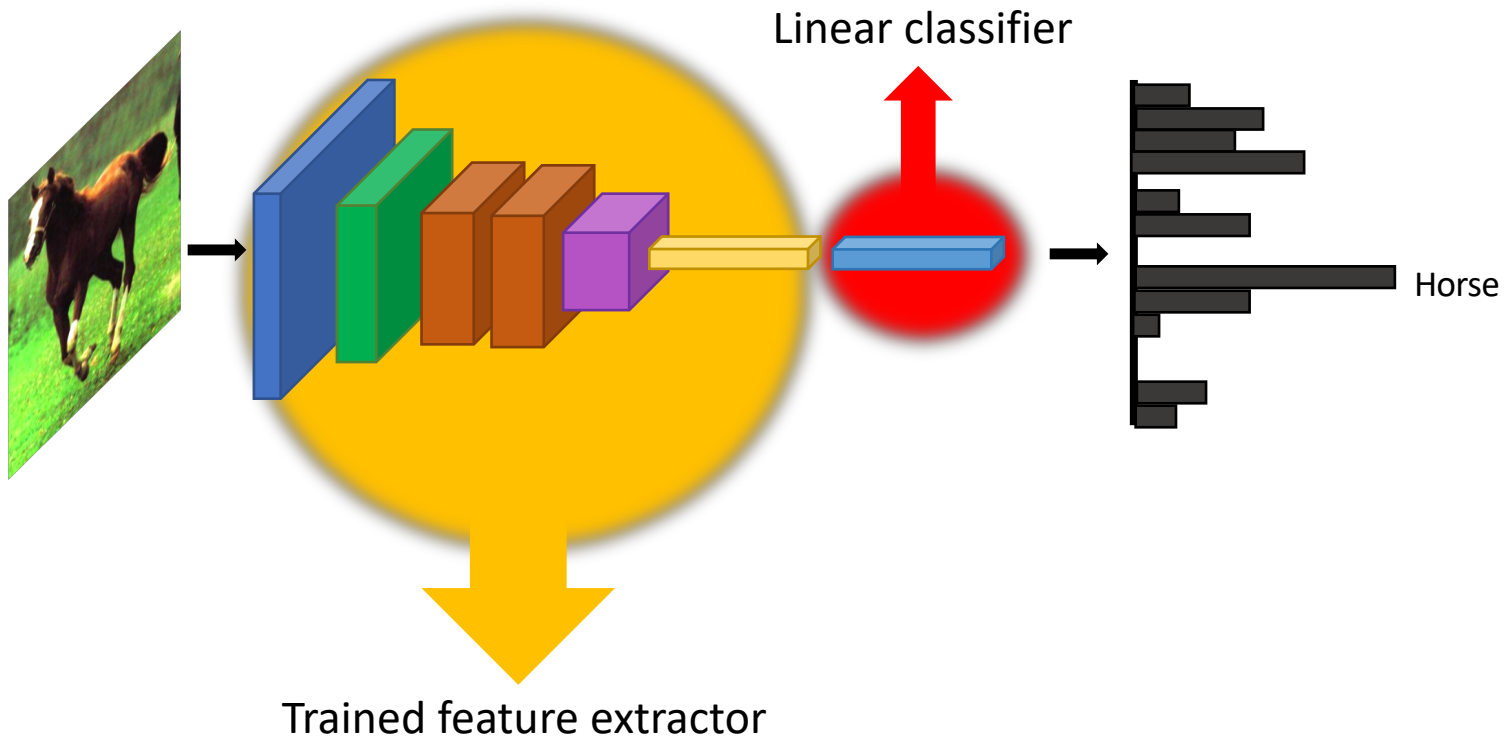
# The ResNet pattern

- Decrease resolution substantially in first layer
  - Reduces memory consumption due to intermediate outputs
- Divide into stages
  - maintain resolution, channels in each stage
  - halve resolution, double channels between stages
- Divide each stage into residual blocks
- At the end, compute average value of each channel to feed linear classifier

# Putting it all together - Residual networks

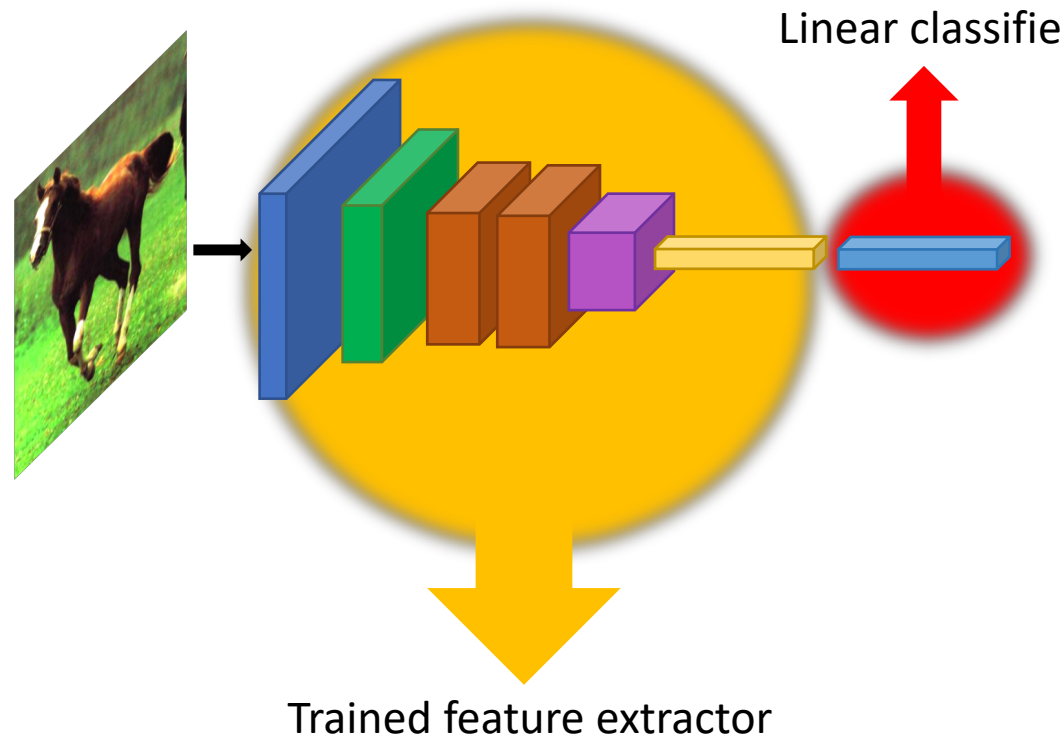


# Transfer learning with convolutional networks



# Transfer learning with convolutional networks

- What do we do for a new image classification problem?
- Key idea:
  - *Freeze* parameters in feature extractor
  - *Retrain* classifier



# Transfer learning with convolutional networks

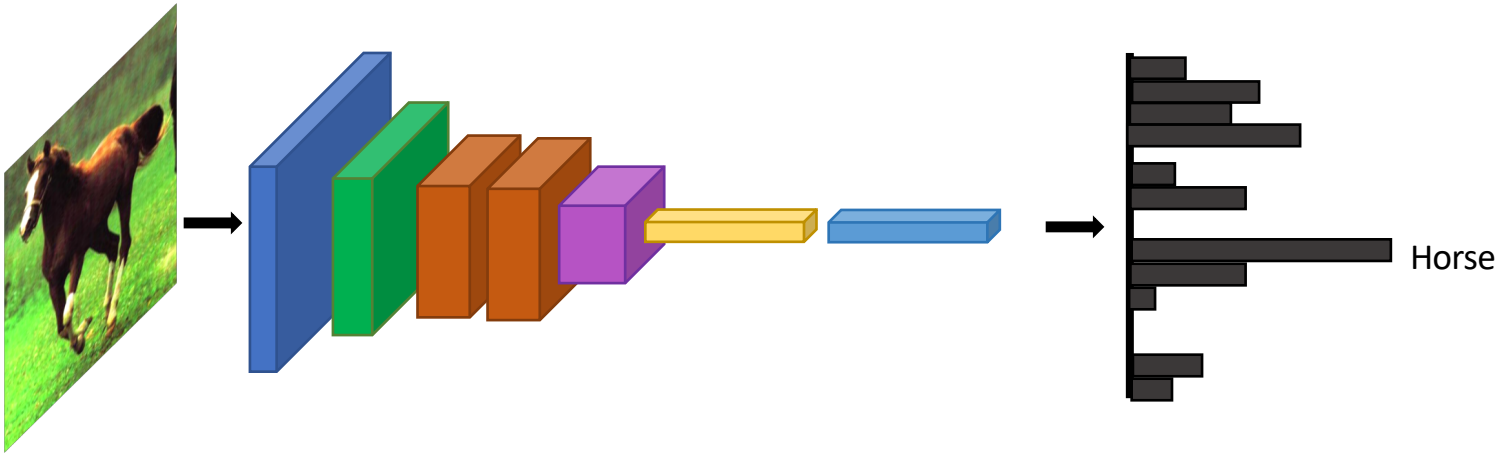
Dataset	Best Non-Convnet perf	Pretrained convnet + classifier	Improvement
Caltech 101	84.3	87.7	+3.4
VOC 2007	61.7	79.7	+18
CUB 200	18.8	61.0	+42.2
Aircraft	61.0	45.0	-16
Cars	59.2	36.5	-22.7

# Why transfer learning?

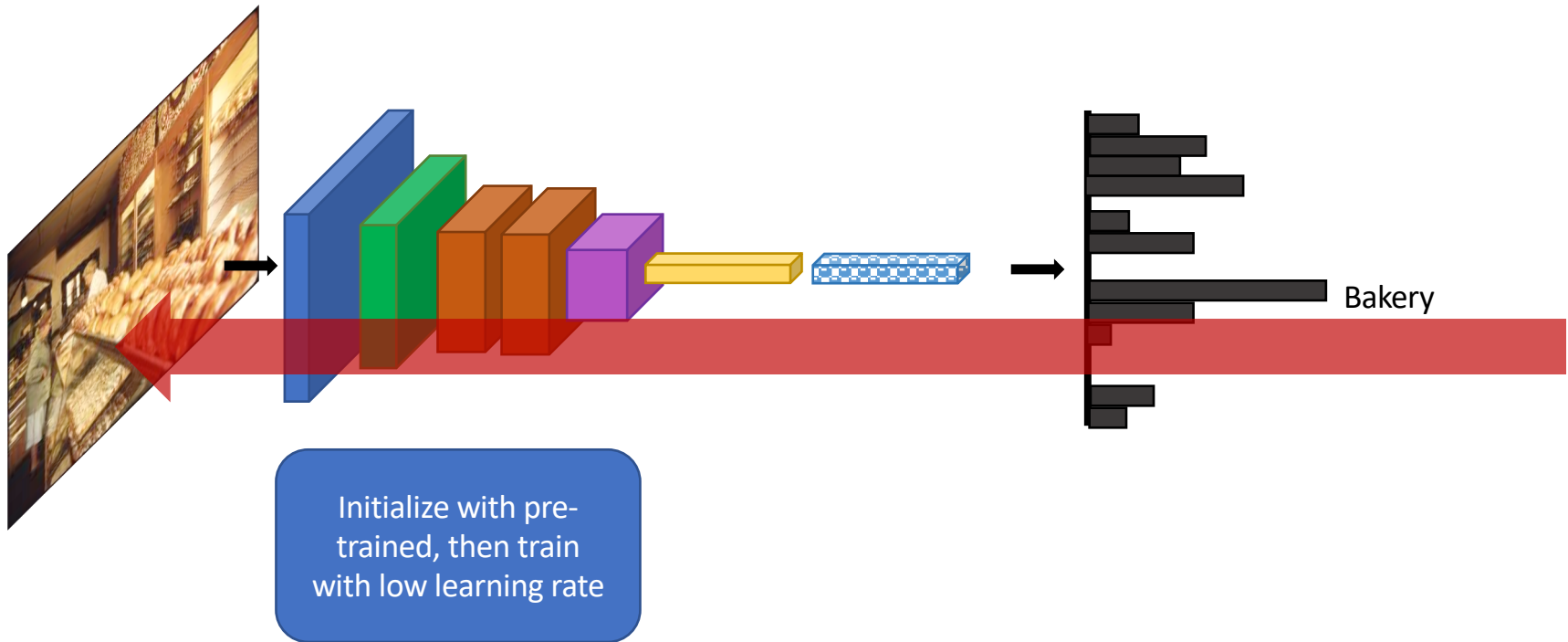
- Availability of training data
- Computational cost
- Ability to pre-compute feature vectors and use for multiple tasks
- *Con: NO end-to-end learning*



# Finetuning



# Finetuning



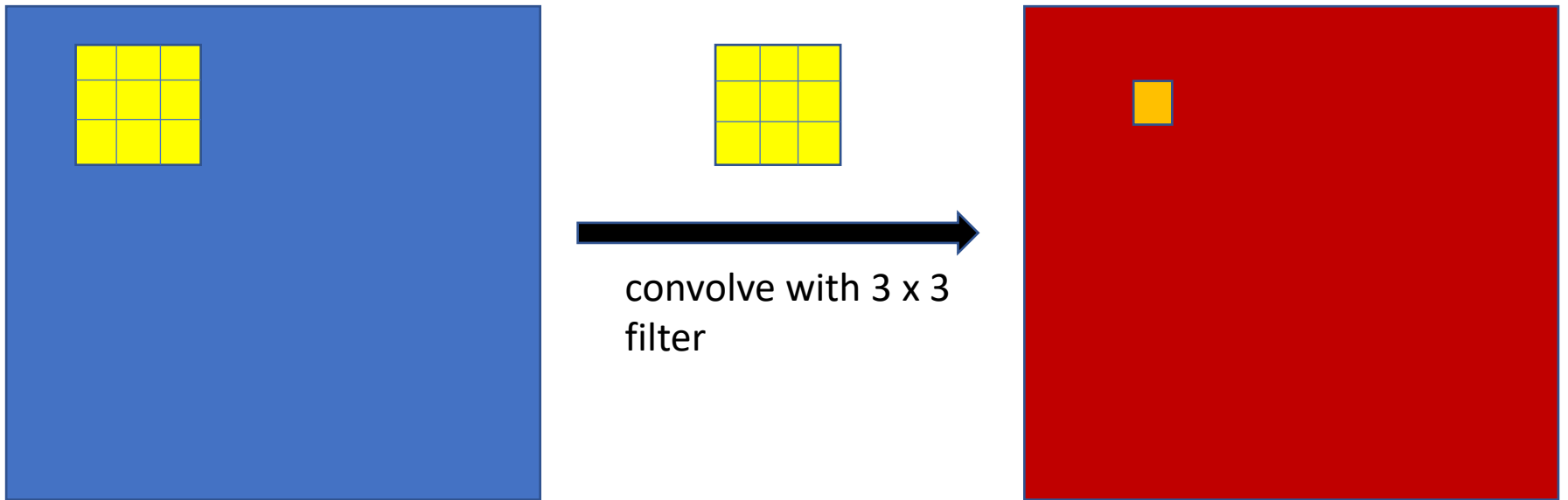
# Finetuning

Dataset	Best Non-Convnet perf	Pretrained convnet + classifier	Finetuned convnet	Improvement
Caltech 101	84.3	87.7	88.4	+4.1
VOC 2007	61.7	79.7	82.4	+20.7
CUB 200	18.8	61.0	70.4	+51.6
Aircraft	61.0	45.0	74.1	+13.1
Cars	59.2	36.5	79.8	+20.6

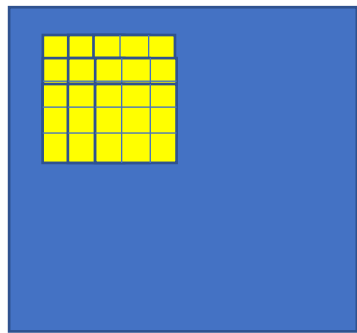
# Visualizing convolutional networks

# Receptive field

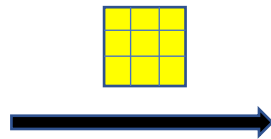
- Which input pixels does a particular unit in a feature map depends on



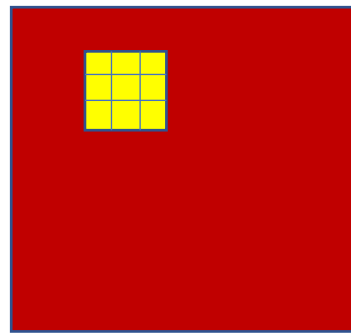
# Receptive field



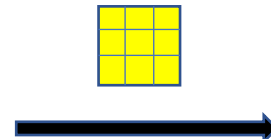
5x5 receptive field



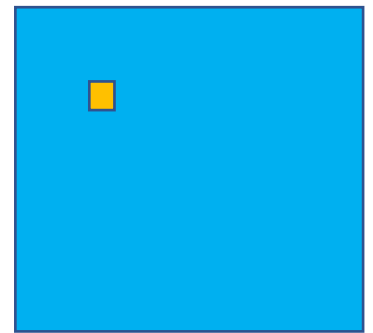
convolve  
with 3 x 3  
filter



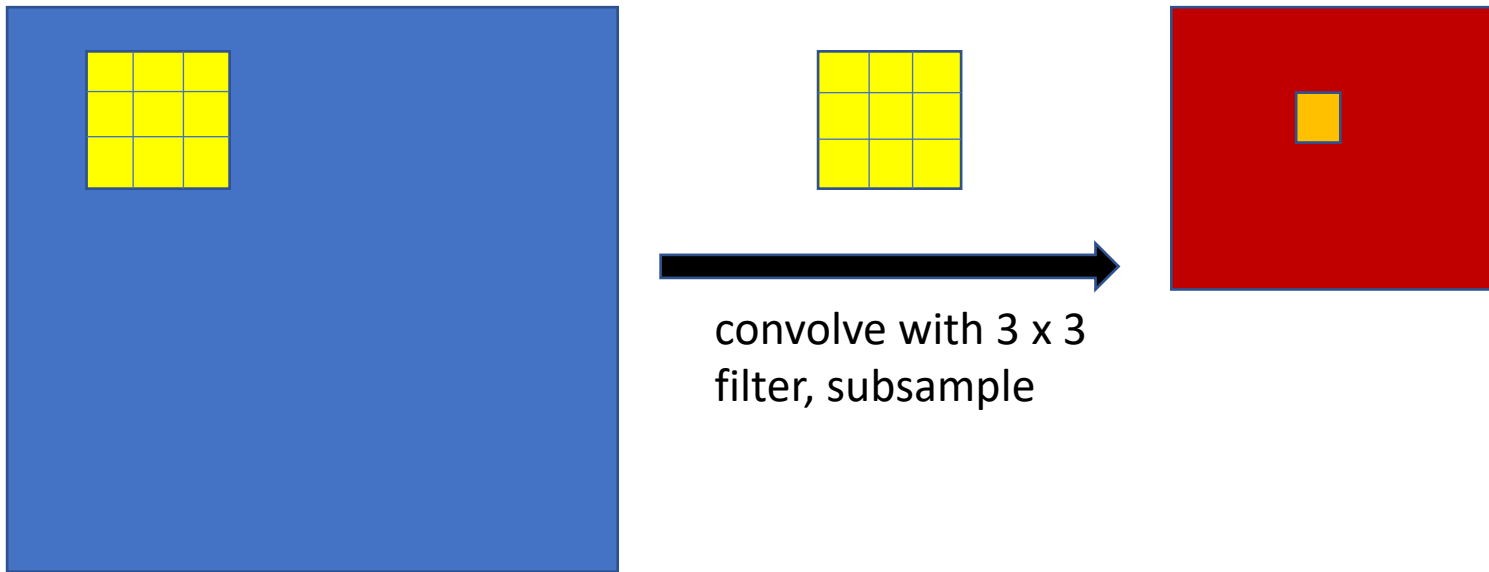
3x3 receptive field



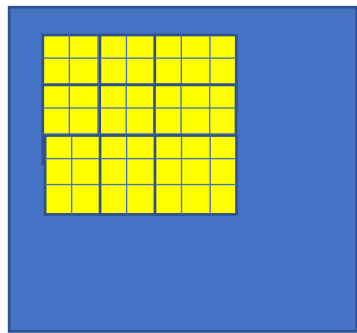
convolve  
with 3 x 3  
filter



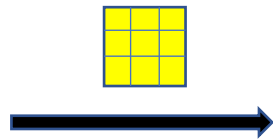
# Receptive field



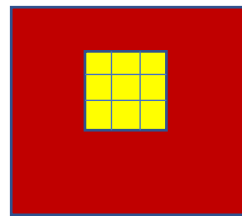
# Receptive field



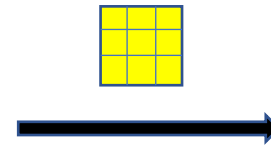
7x7 receptive field: union of 9 3x3 fields with stride of 2



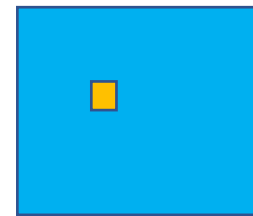
convolve with 3 x 3 filter, subsample by factor 2



3x3 receptive field



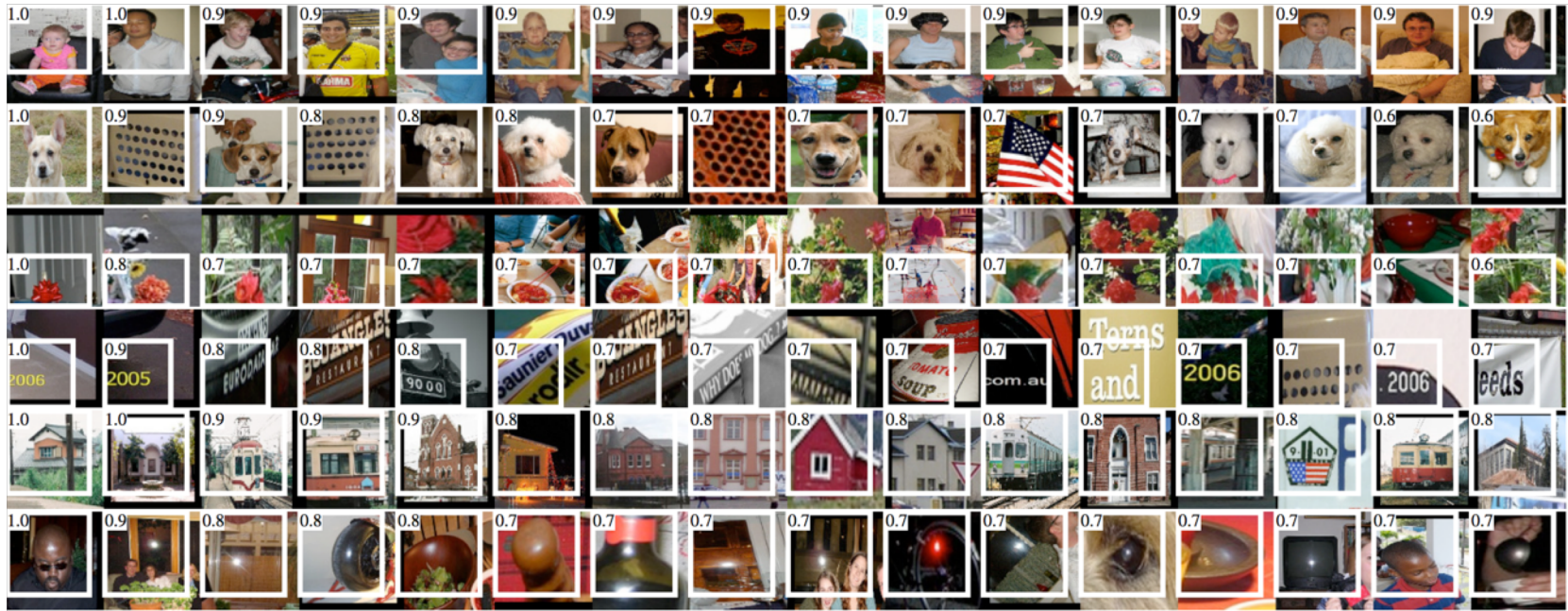
convolve with 3 x 3 filter





# Visualizing convolutional networks

- Take images for which a given unit in a feature map scores high
- Identify the receptive field for each.



Rich feature hierarchies for accurate object detection and semantic segmentation. R. Girshick, J. Donahue, T. Darrell, J. Malik. In *CVPR*, 2014.

# Visualizing convolutional networks II

- Block regions of the image and classify



Visualizing and Understanding Convolutional Networks. M. Zeiler and R. Fergus. In *ECCV 2014*.

# Visualizing convolutional networks II

- Image pixels important for classification = pixels when blocked cause misclassification

