# 8. More Tasks in Computer Vision

CS 519 Deep Learning, Winter 2018

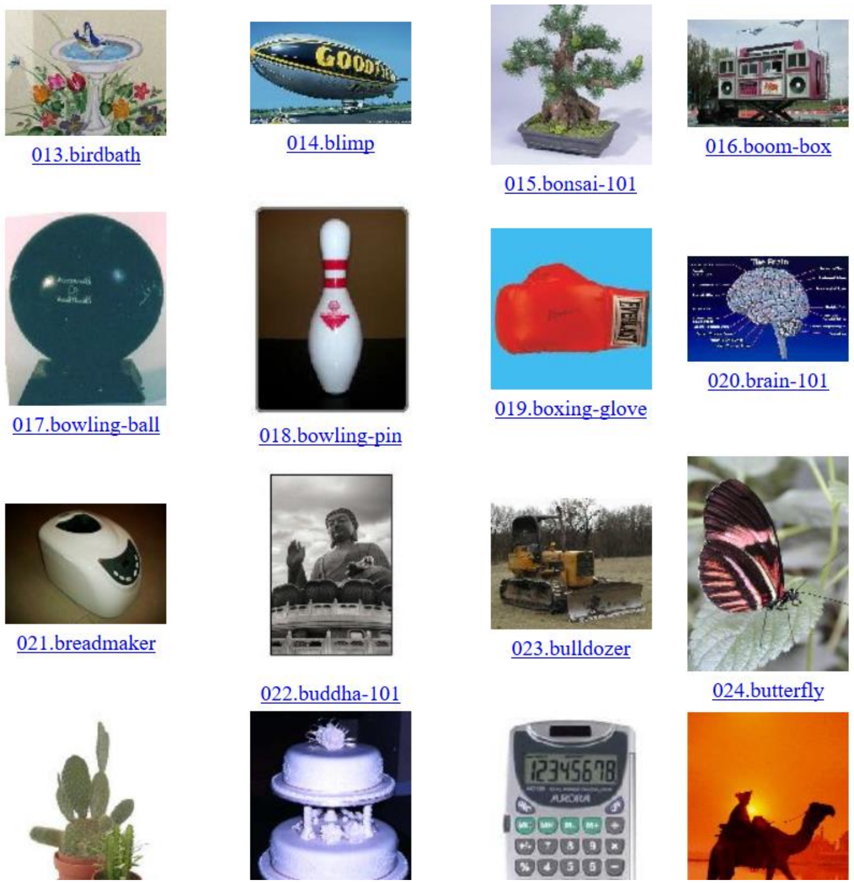
Fuxin Li

*With materials from Zsolt Kira, Roger Grosse, Nitish Srivastava*

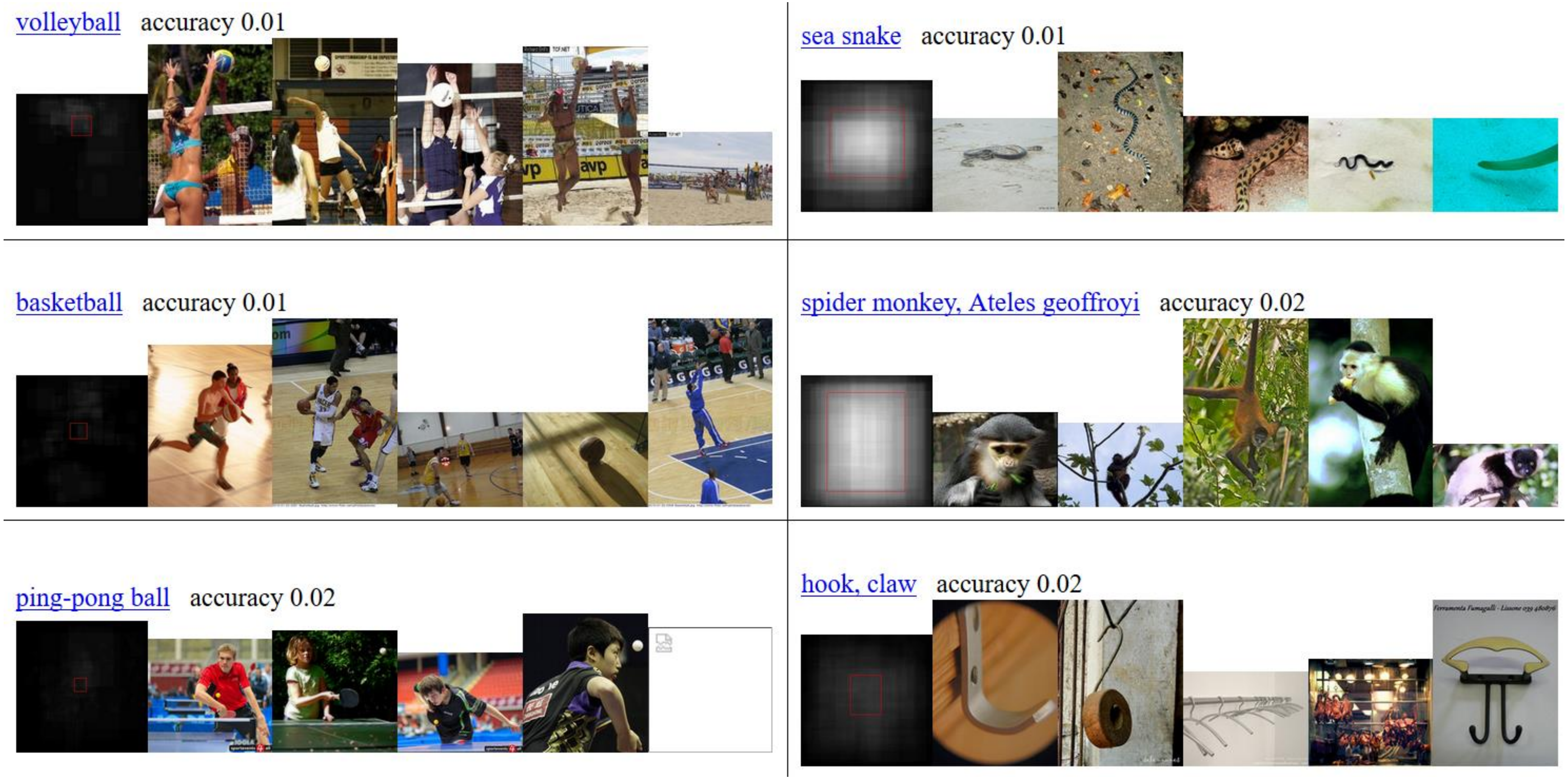
# Image Classification History

- Caltech datasets:
  - Caltech-101: 3,030 images in 101 categories
  - Caltech-256: 30,607 images in 256 categories
- ImageNet
  - Full set: more than 10 million images
  - WordNet taxonomy
  - Challenge: 1.2 million images in 1,000 categories
  - Dog breeds

Caltech-256

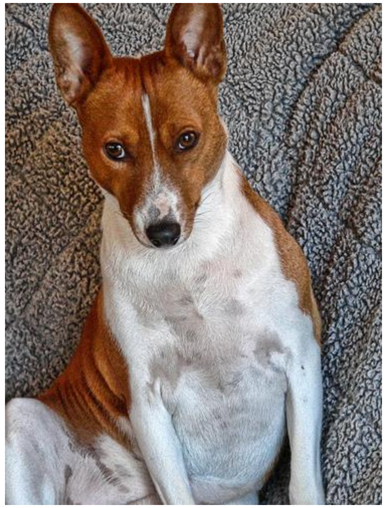
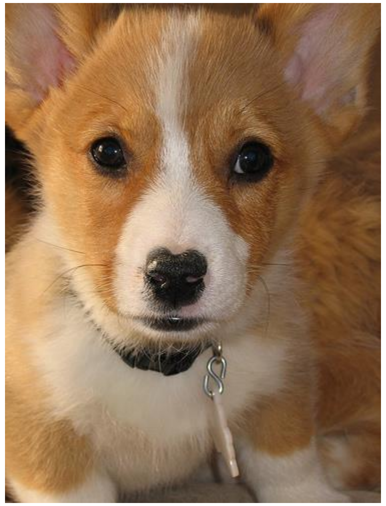


# ImageNet


volleyball accuracy 0.01
sea snake accuracy 0.01
basketball accuracy 0.01
spider monkey, Ateles geoffroyi accuracy 0.02
ping-pong ball accuracy 0.02
hook, claw accuracy 0.02

# Dog breeds

- More than 120 different dog breeds in the dataset
- Hard for human to discriminate

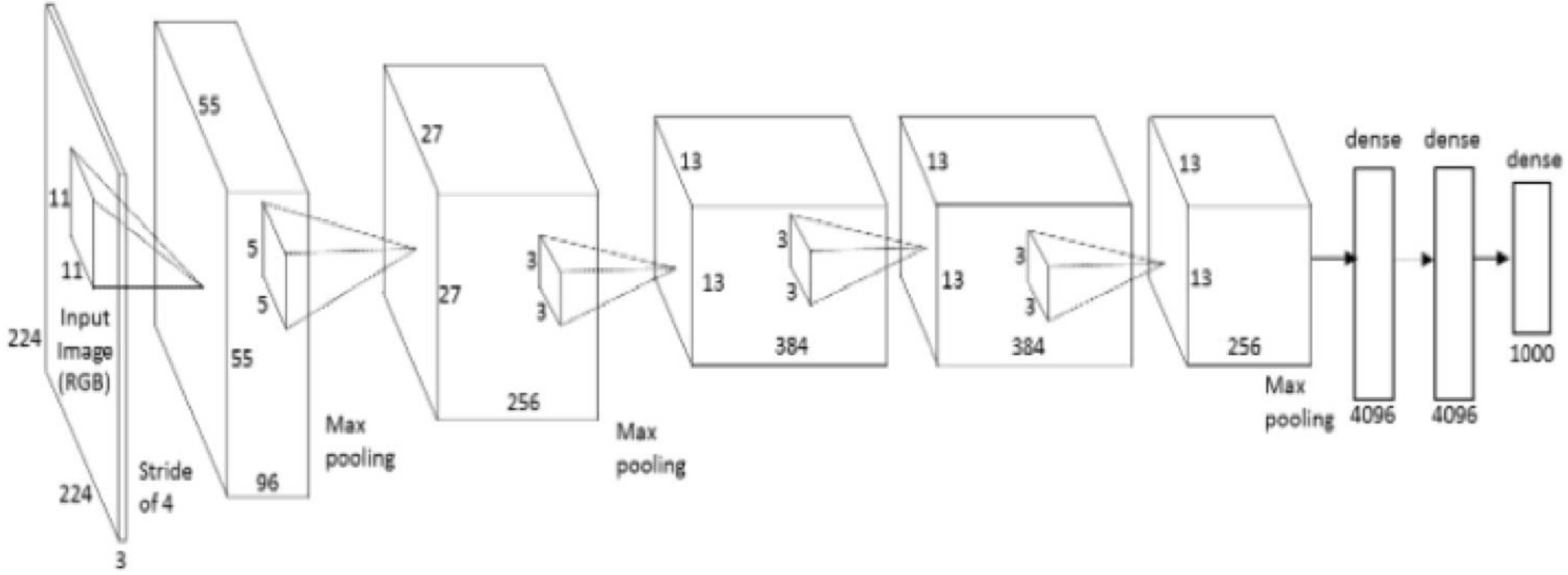
# ILSVRC

Task : Given an image and a predefined set of categories, find out which category the image belongs to.
There is an annual competition ILSVRC (ImageNet Large Scale Visual Recognition Challenge).

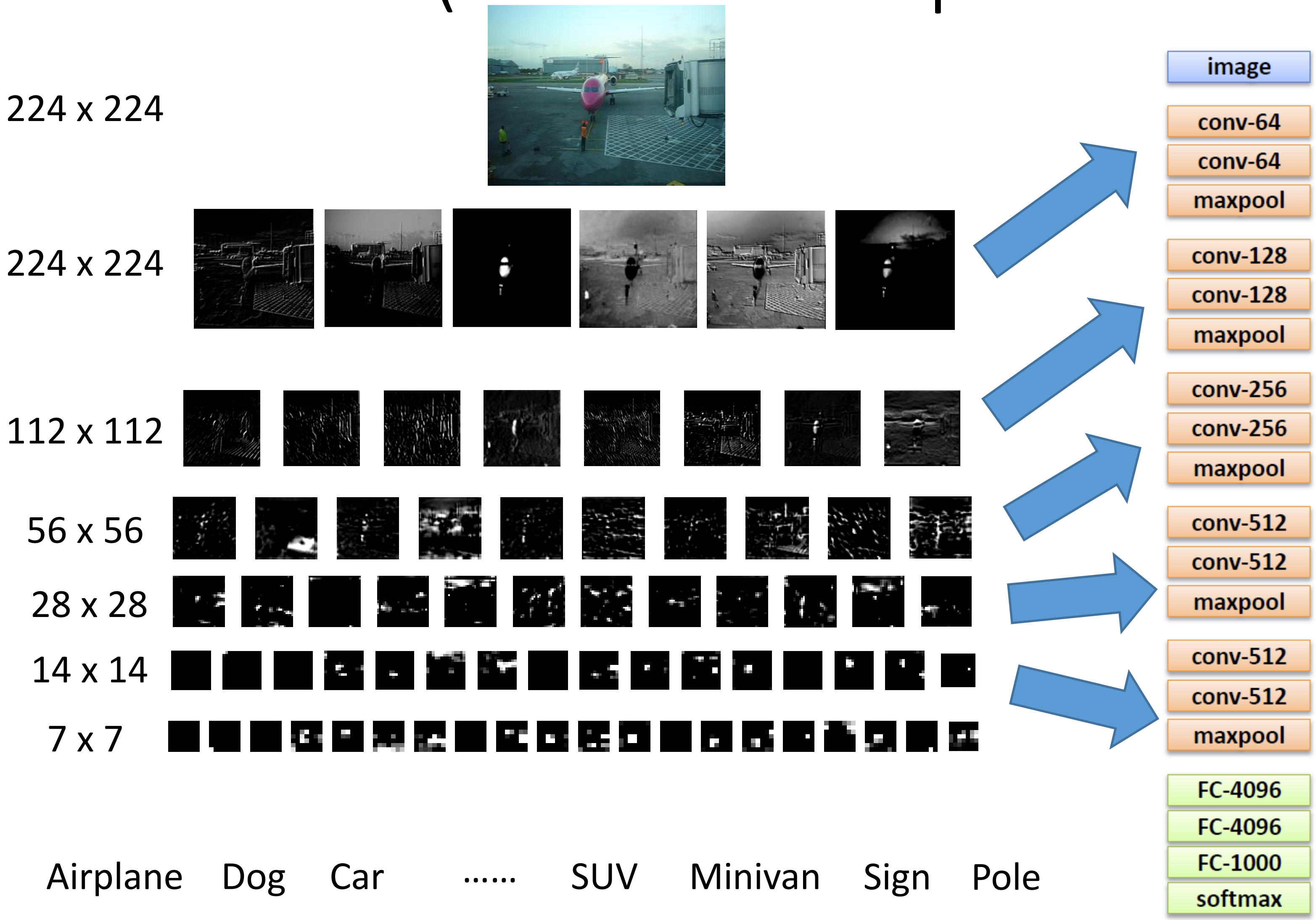
| **Year** | **Model** | **Best Result (Error %)** |
|---|---|---|
| 2010 | Hand-designed descriptors + SVM | 28.2 % |
| 2011 | Compressed Fisher Vectors + SVM | 25.8 % |
| 2012 | Deep Conv Net (AlexNet) | 16.4 % |
| 2013 | Deeper Conv Net | 11.7 % |
| 2014 | Even Deeper Conv Net (VGG) | 6.6 % |
| 2015 | 152-level Conv Net (covered later) | 3.6% |

Human-performance is around 5.1%.

# AlexNet (60 million parameters)



# The VGG Network (138 Million parameters)




*(Simonyan and Zisserman 2014)*

# Softmax Cross-Entropy

- Softmax layer in multi-class

$$P(y=j|\mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}$$

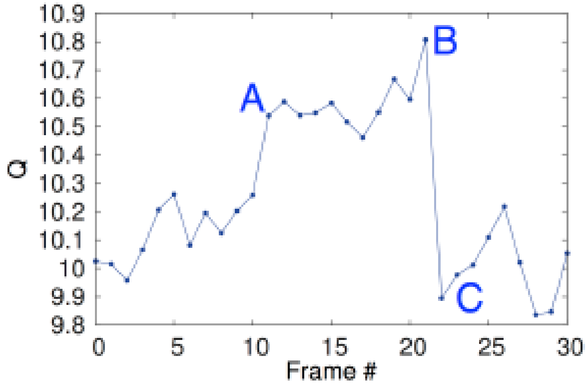
- Log-likelihood:
    - Loss function is minus log-likelihood

$$-\log P(y=j|x) = -\boldsymbol{x}^\top \boldsymbol{w}_j + \log \sum_k e^{\boldsymbol{x}^\top \boldsymbol{w}_k}$$

- Total energy: $\min_{\mathbf{W}} \sum_i -\log P(y=y_i|x_i)$

# Reinforcement Learning: Atari games

- Predict the Q-function (value function) of each move using the current scene as input
- Use normal MDP value iterations to decide the best current move





Mnih et al. Playing Atari with Deep Reinforcement Learning

# Reinforcement Learning: Playing go

- Predict next 3 moves using CNN
- Combine with Monte Carlo Tree Search to obtain state-of-the-art go-playing system

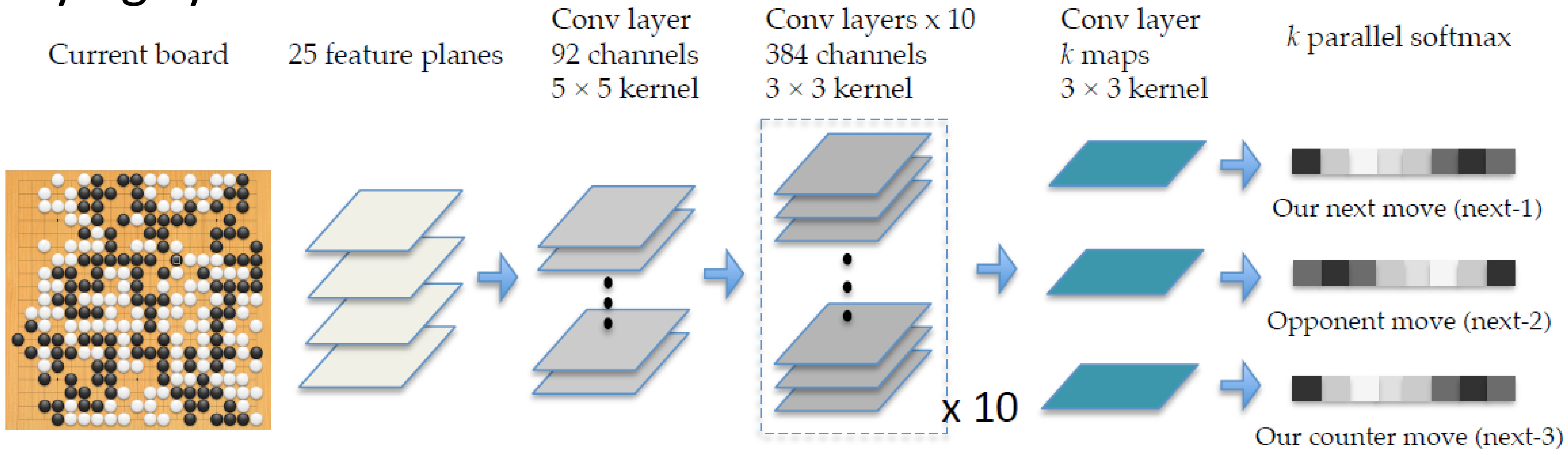



Figure 3: Our network structure ($d = 12$, $w = 384$). The input is the current board situation (with history information), the output is to predict next $k$ moves.

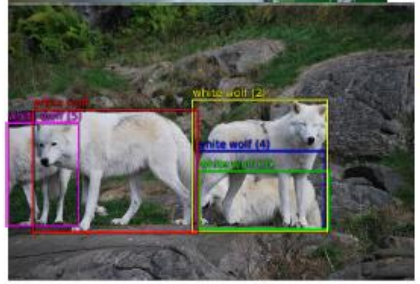
Tian and Zhu. arXiv 0511:06410

- classification


**Top 5:**
pencil sharpener
pool table
hand blower
oil filter
packet

**Groundtruth:**
pencil sharpener

- localization

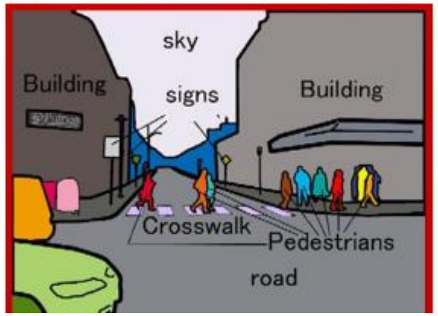


**Groundtruth:**
white wolf
white wolf (2)
white wolf (3)
white wolf (4)
white wolf (5)

- detection


**Groundtruth:**
tv or monitor
tv or monitor (2)
tv or monitor (3)
person
remote control
remote control (2)

- segmentation




difficulty

# Object Detection

- Faster/Mask R-CNN:
  - Deep network on object proposals
  - Jointly train network to propose boxes inside the image and classification in the box

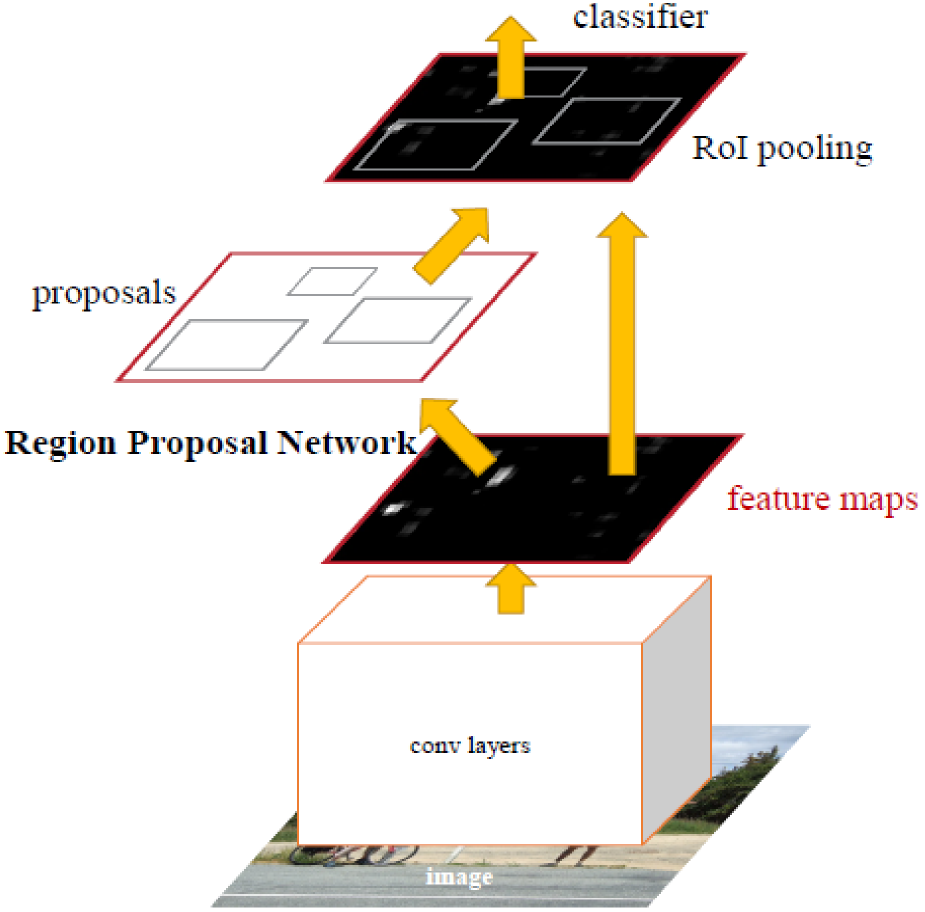



Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

# Predicting Regions

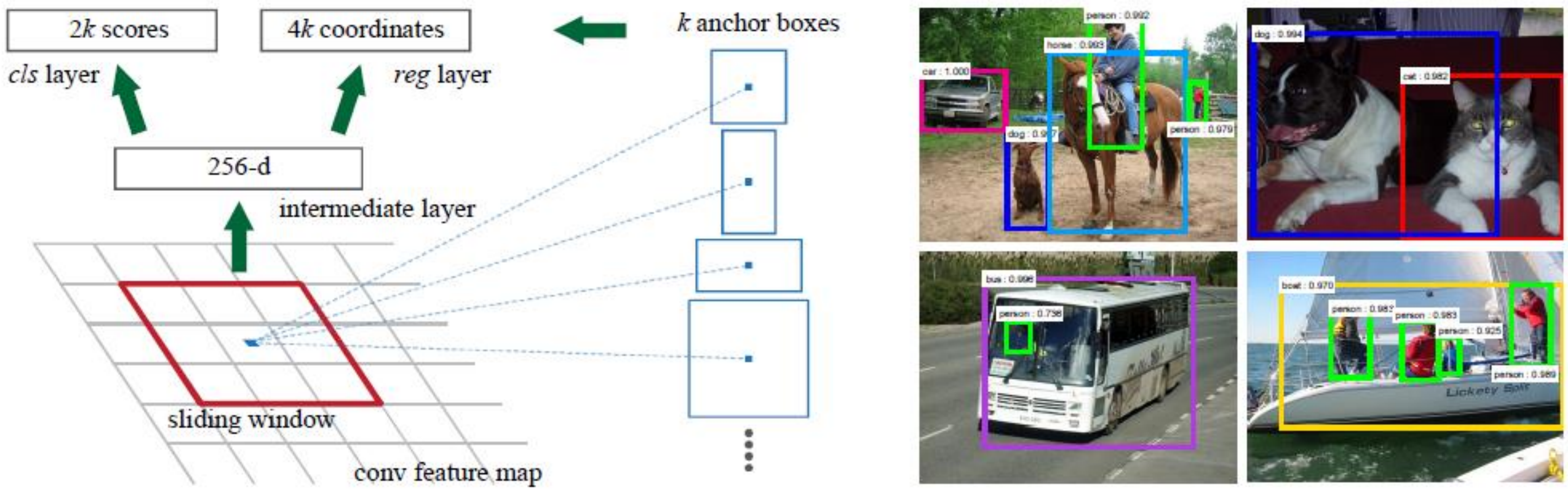


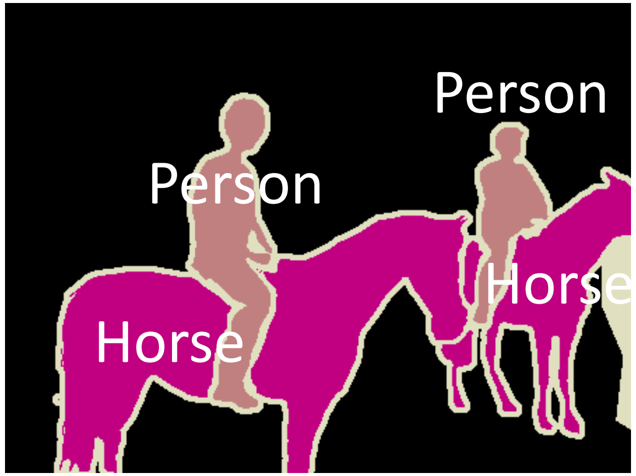

Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

# Semantic Segmentation

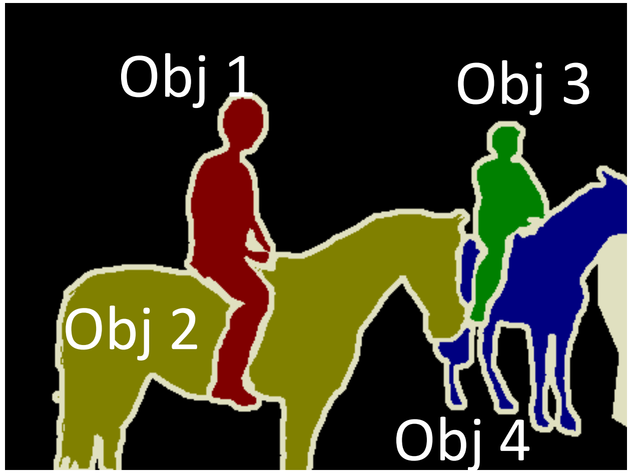
- Given an image, identify the category and spatial extent of all relevant objects

Image

Category Label

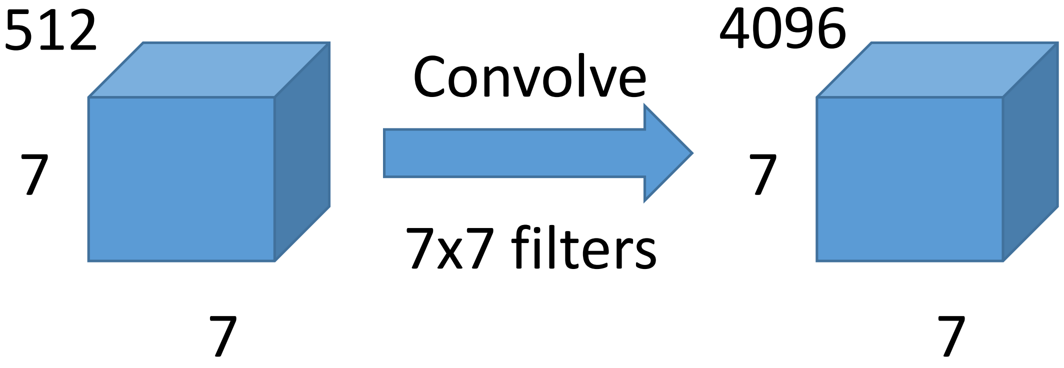


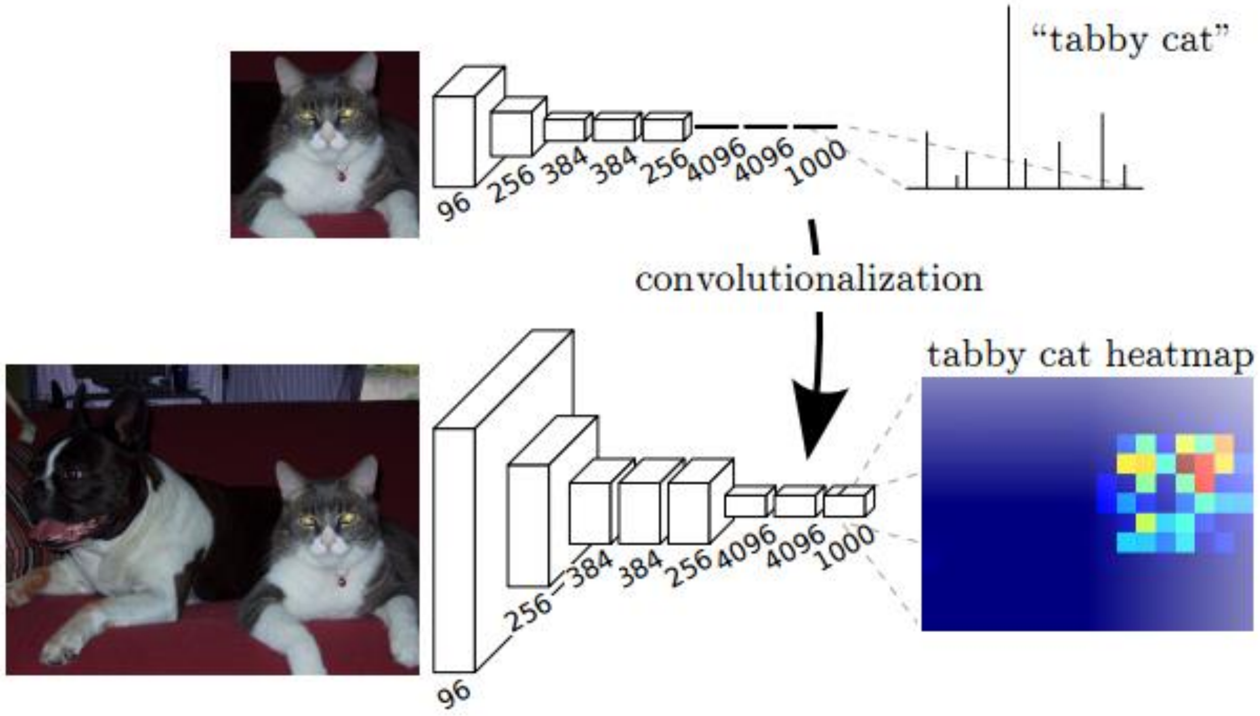

Object Label



# Fully Convolutional Network

- Idea – Fully connected can be turned into fully-convolutional
- Zero-padding can help outputting more numbers!

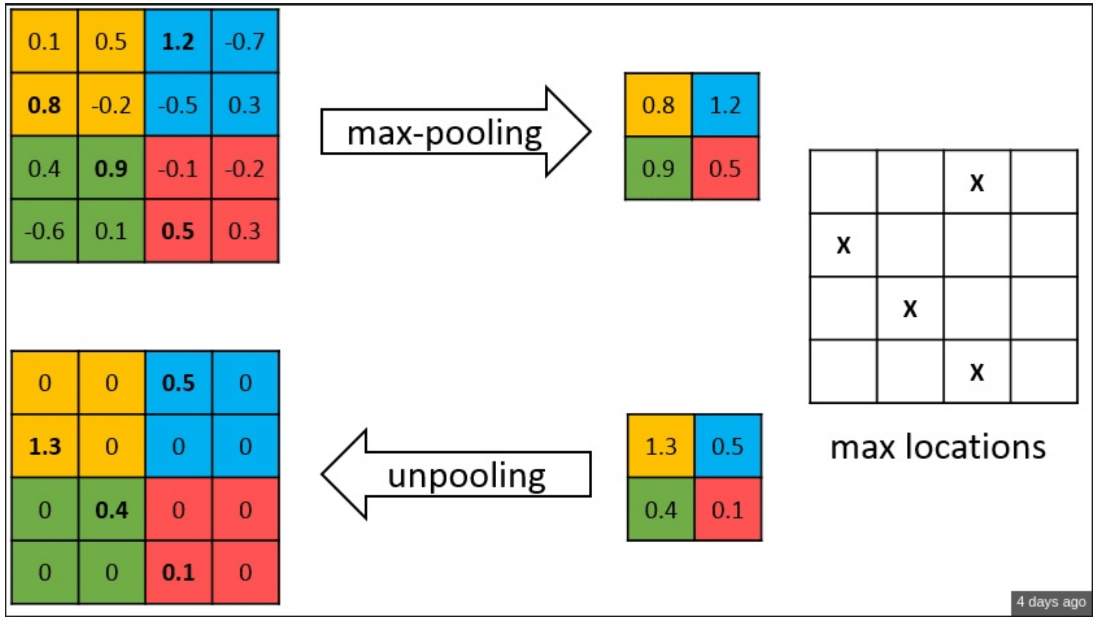


# Decoding Step (Deconvolution)

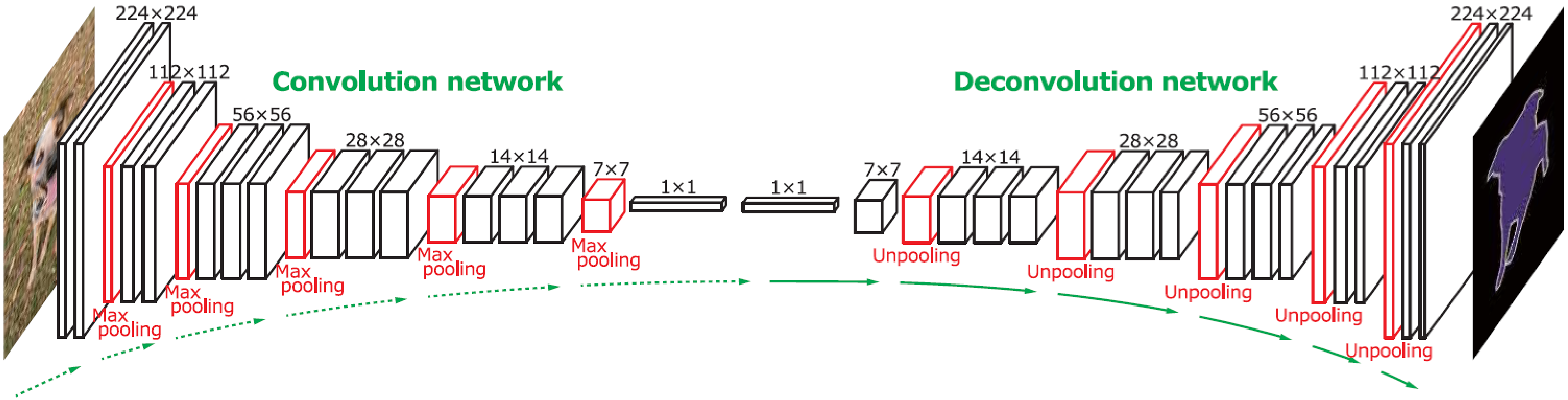
- Can also train network to "decode"
  - Suppose CNN is an "encoding" process
  - One could train a "decoder" to retain the full image resolution
  - Decoder is another CNN, with filter weights tied/not tied to the filter weights in the "encoder" CNN

  - One could use "Un-max-pooling" to increase resolution



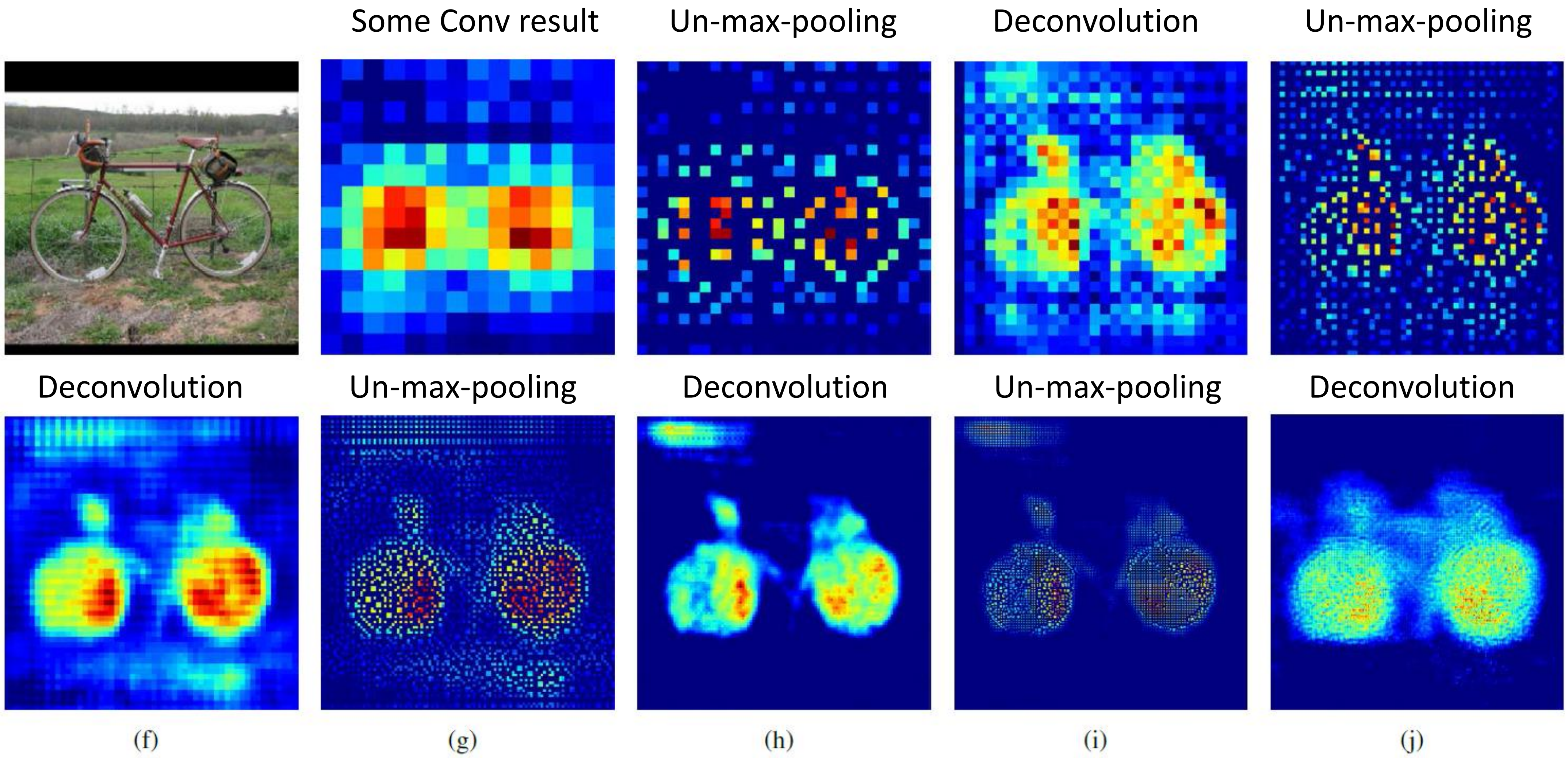
# Deconvolution used for finer details

- Same convolutional networks – deconvolute all the way




H. Noh, S. Hong, B. Han. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015

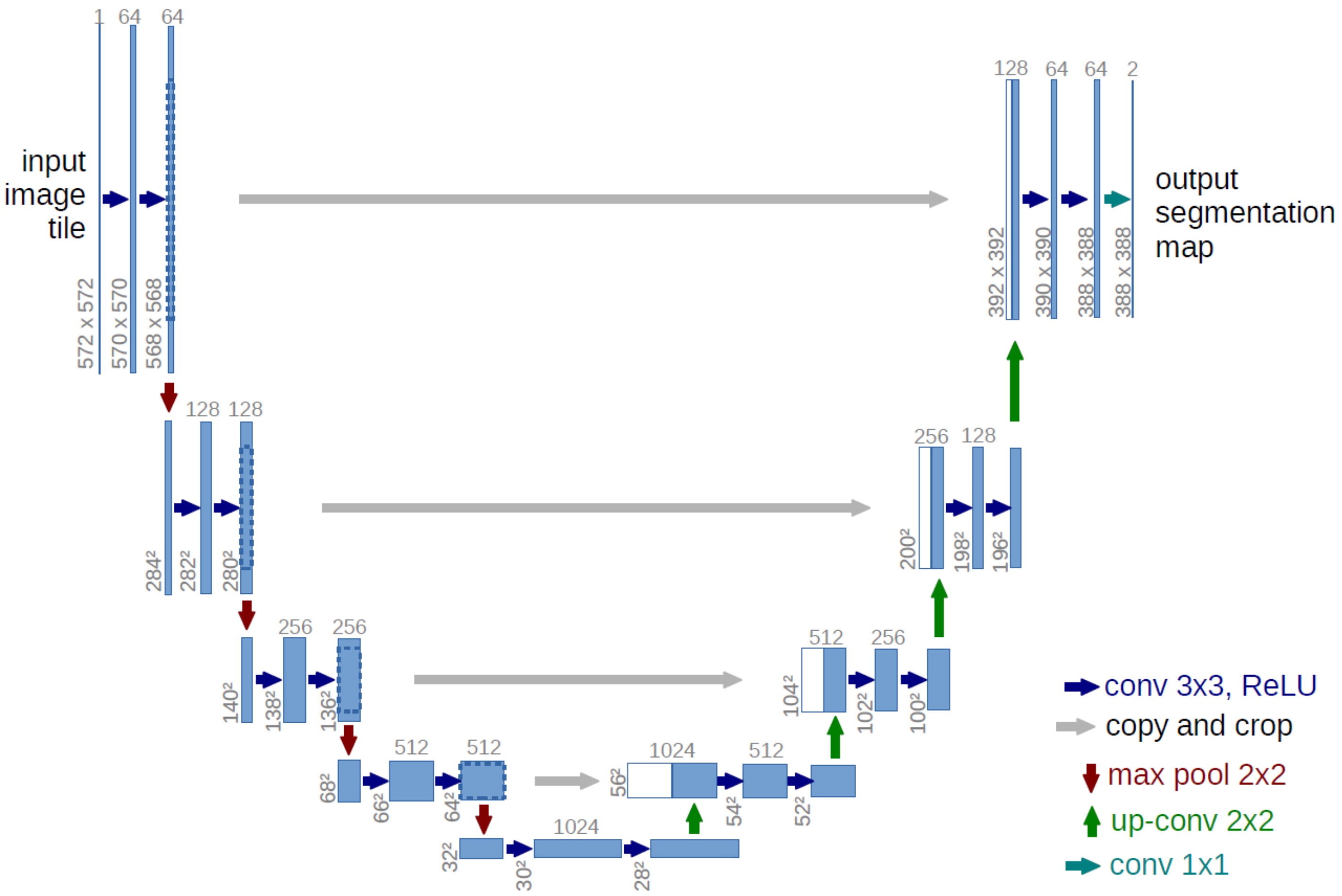
# Deconvolution



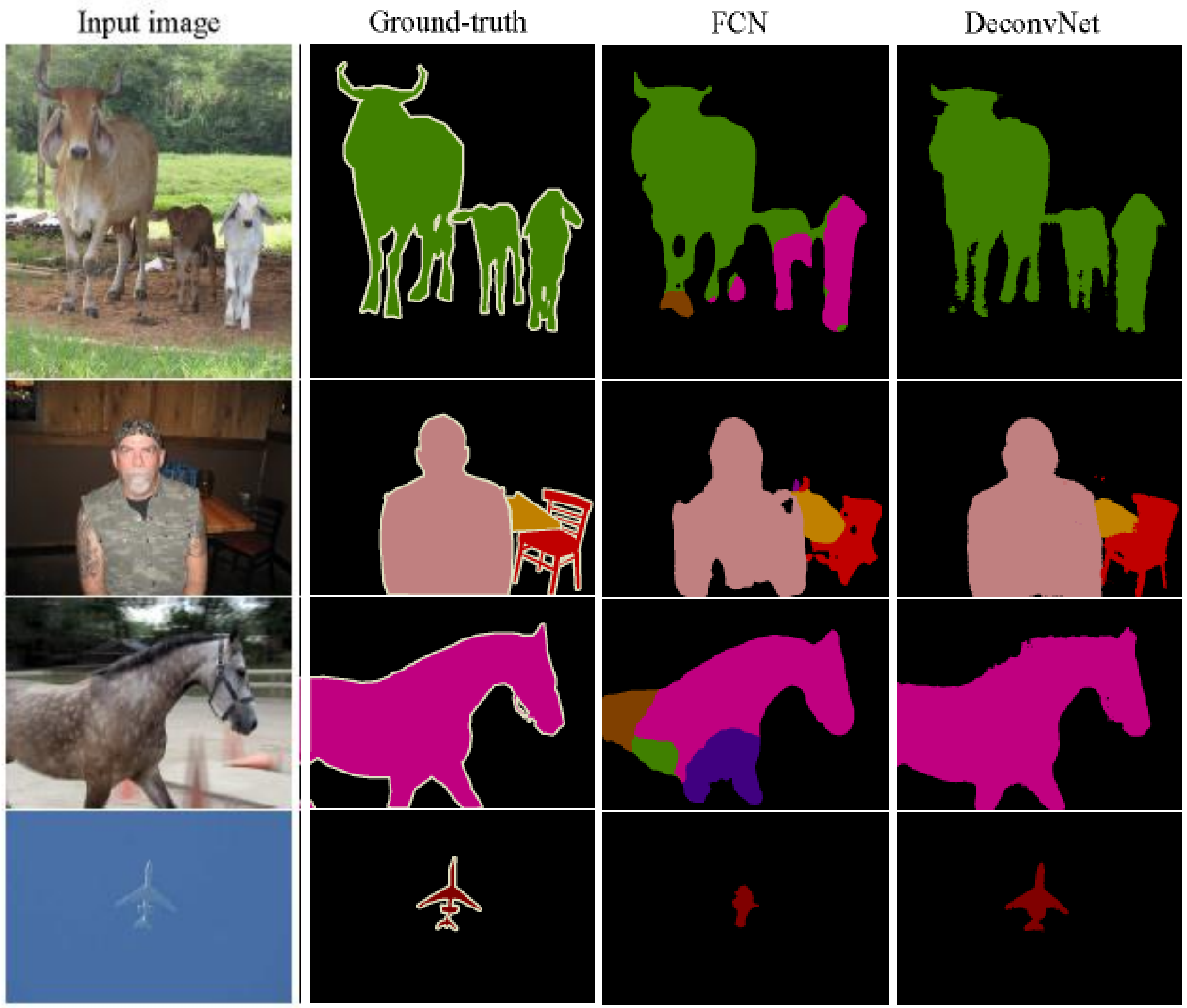
# U-Net: Add linkage

- Add linkage between conv layers and deconv layers with the same resolution

- Improve spatial precision and helps at boundaries (low-level information)

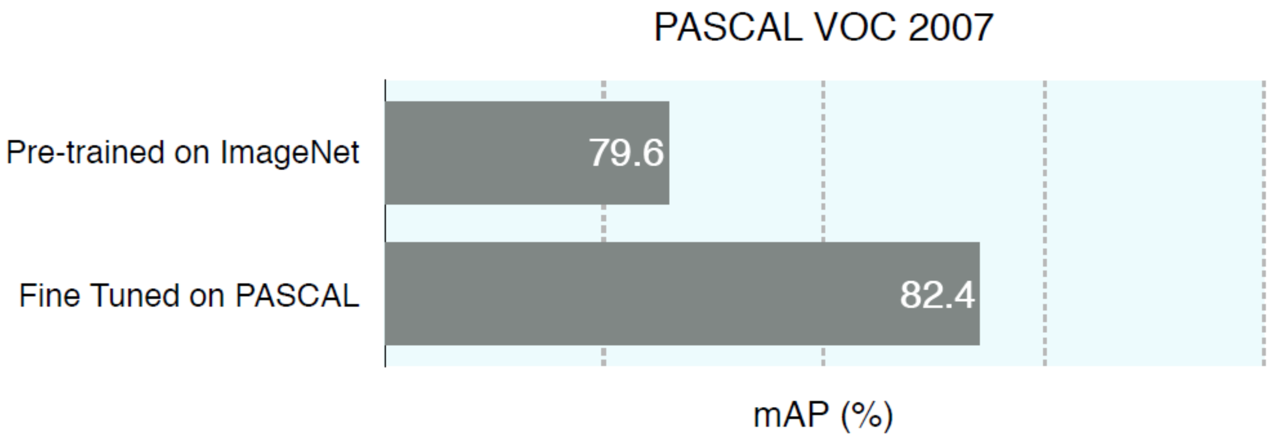


# Sample results for deconvolution-based semantic segmentation



# Other trivia: Fine-Tuning

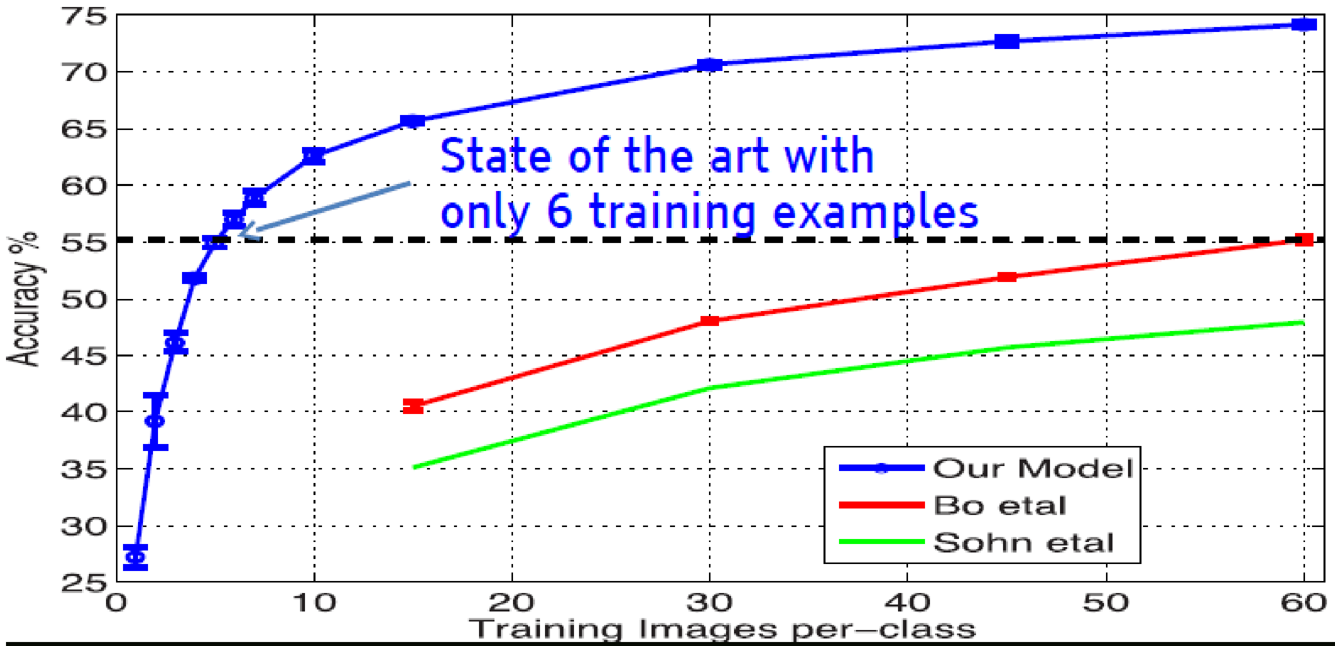
- Take pre-trained network
- Remove last layer
- Add your new layer
  - Say with 10 classes
- Best results
  - Train last layer
  - Retrain entire network




- Pre-trained CNNs can be **tuned on target dataset**
  - Use target data to provide more training images
  - Remark: tuning in PASCAL requires a multi-class loss
- Often (but not always) yields a nice improvement

# Fine-tuning

- Network first trained on ImageNet.
- Last layer chopped off
- Last layer trained on Caltech 256,
- first layers N-1 kept fixed.
- State of the art accuracy with only 6 training samples/class




| # Train | Acc % 15/class | Acc % 30/class | Acc % 45/class | Acc % 60/class |
|---|---|---|---|---|
| Sohn *et al.* [16] | 35.1 | 42.1 | 45.7 | 47.9 |
| Bo *et al.* [3] | $40.5 \pm 0.4$ | $48.0 \pm 0.2$ | $51.9 \pm 0.2$ | $55.2 \pm 0.3$ |
| Non-pretr. | $9.0 \pm 1.4$ | $22.5 \pm 0.7$ | $31.2 \pm 0.5$ | $38.8 \pm 1.4$ |
| ImageNet-pretr. | **$65.7 \pm 0.2$** | **$70.6 \pm 0.2$** | **$72.7 \pm 0.4$** | **$74.2 \pm 0.3$** |

3: [Bo, Ren, Fox. CVPR, 2013] 16: [Sohn, Jung, Lee, Hero ICCV 2011]