CONVOLUTIONAL NEURAL NETWORK

한재근 과장 | jahan@nvidia.com
유현곤 부장 | hryu@nvidia.com / 양한별 과장 | hanbyuly@nvidia.com

NVIDIA.

AGENDA

CNN Models

Visualization

Localization and Detection

Segmentation

Transfer Learning

Deep Dream and Neural Style

Computer Vision Tasks

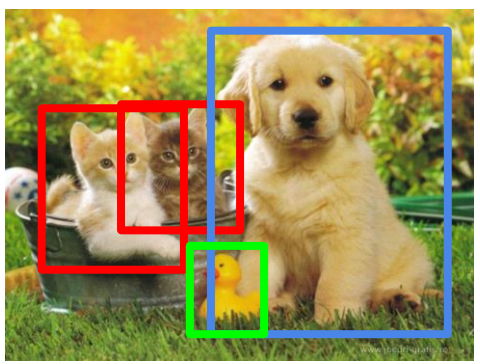
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

Visualization
[understanding CNN]

Deep Visualization

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis


Jason Yosinski


Jeff Clune


Anh Nguyen


Thomas Fuchs


Hod Lipson

Cornell University

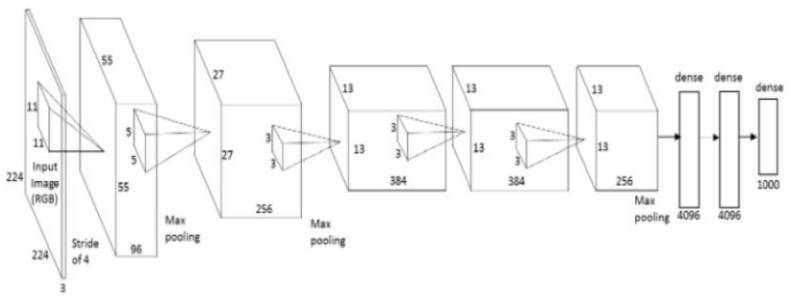
University of Wyoming

NASA Jet Propulsion Laboratory
California Institute of Technology

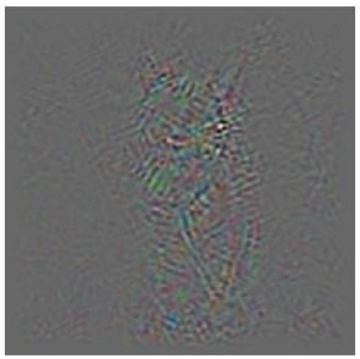
https://youtu.be/AgkfIQ4IGaM

Deconv Approach

1. Feed image into net



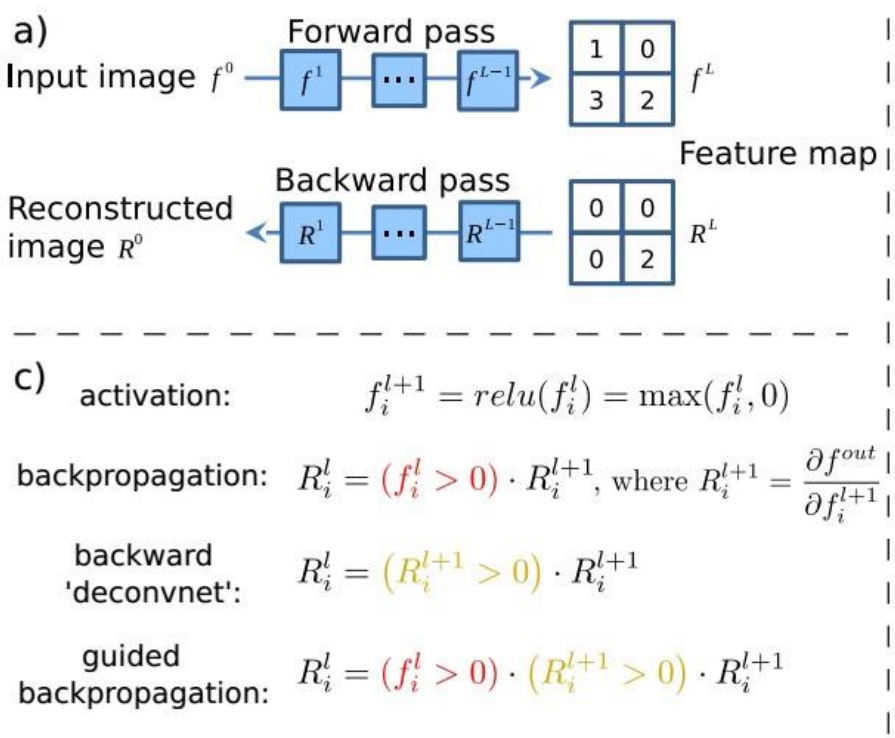
2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:

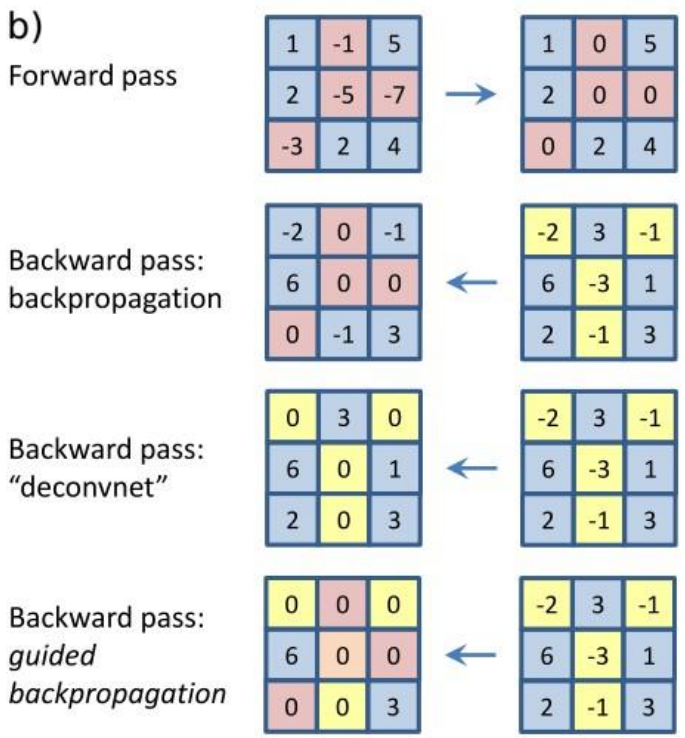


"Guided backpropagation:" instead

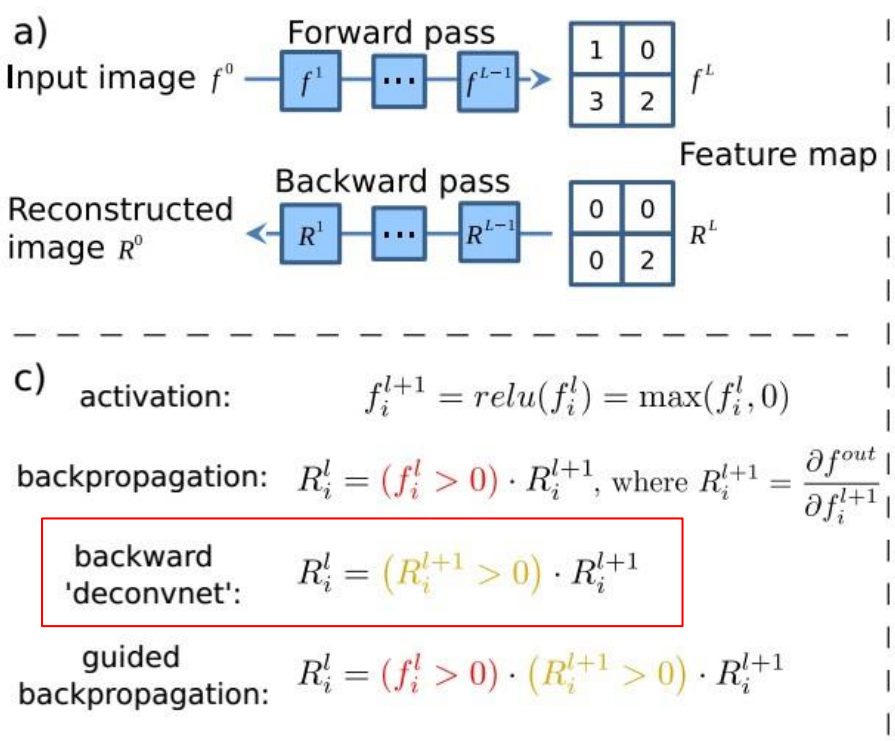
Guided backpropagation

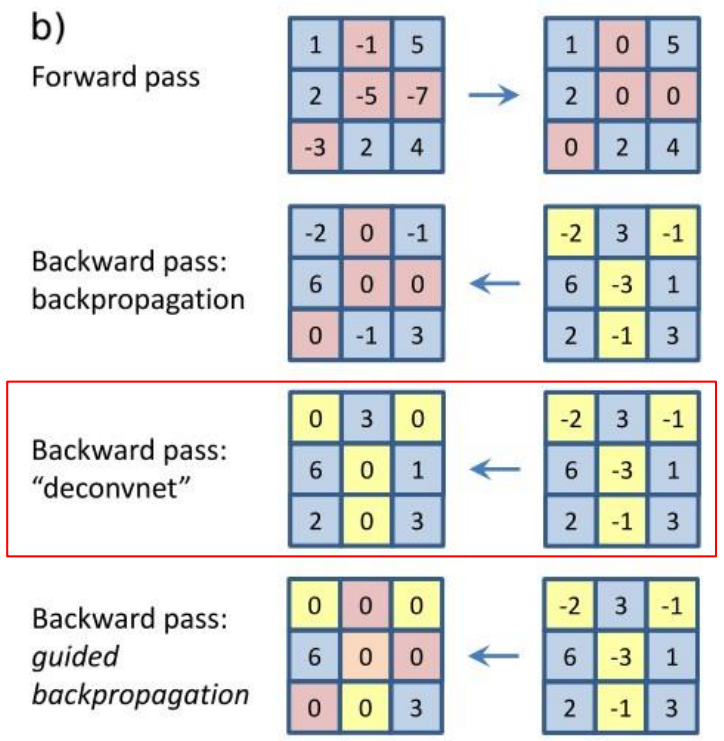
[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]





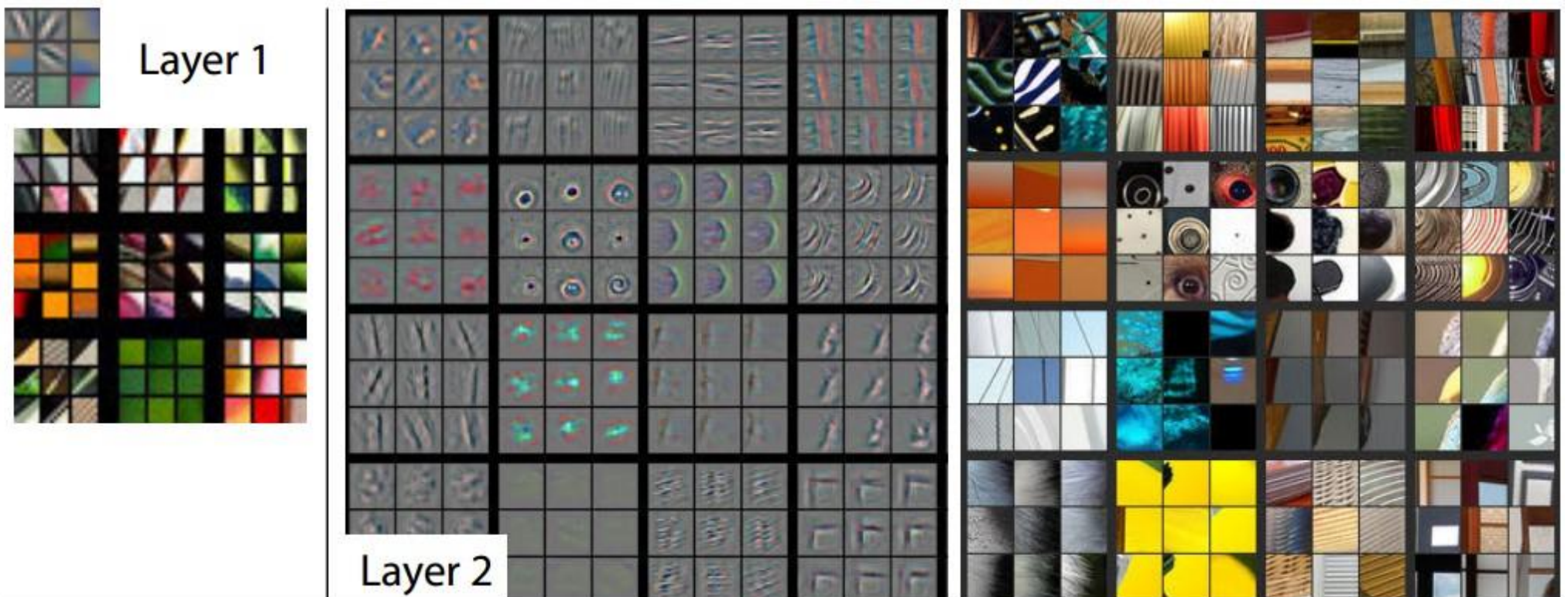
Deconv approaches



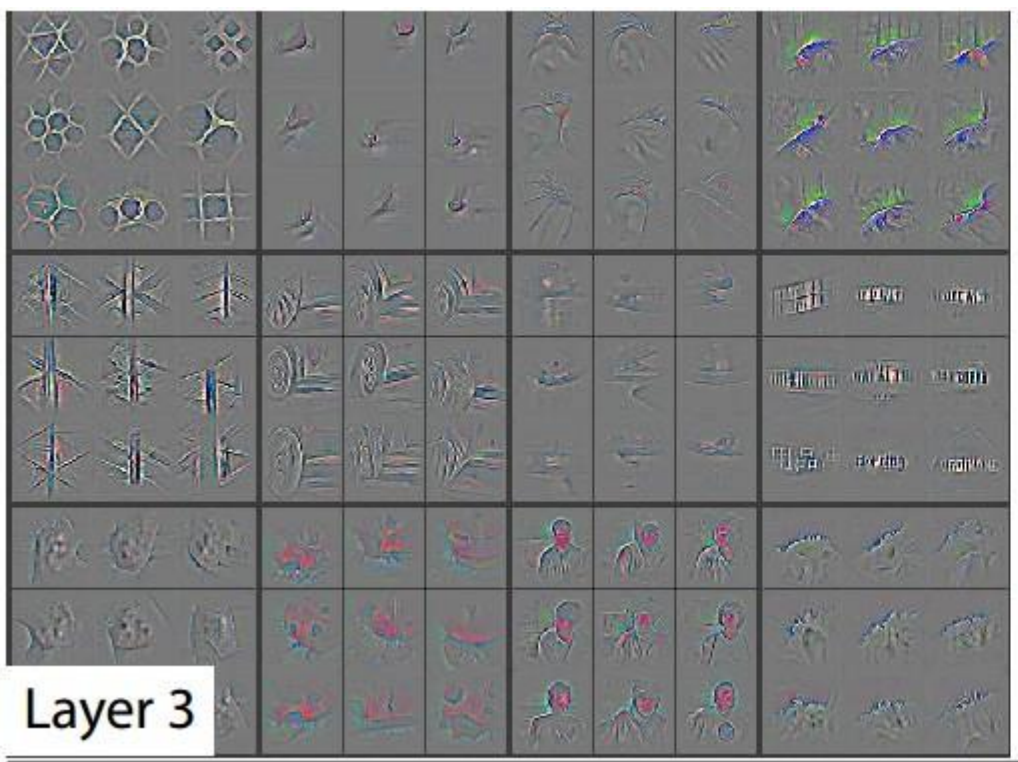


bit weird

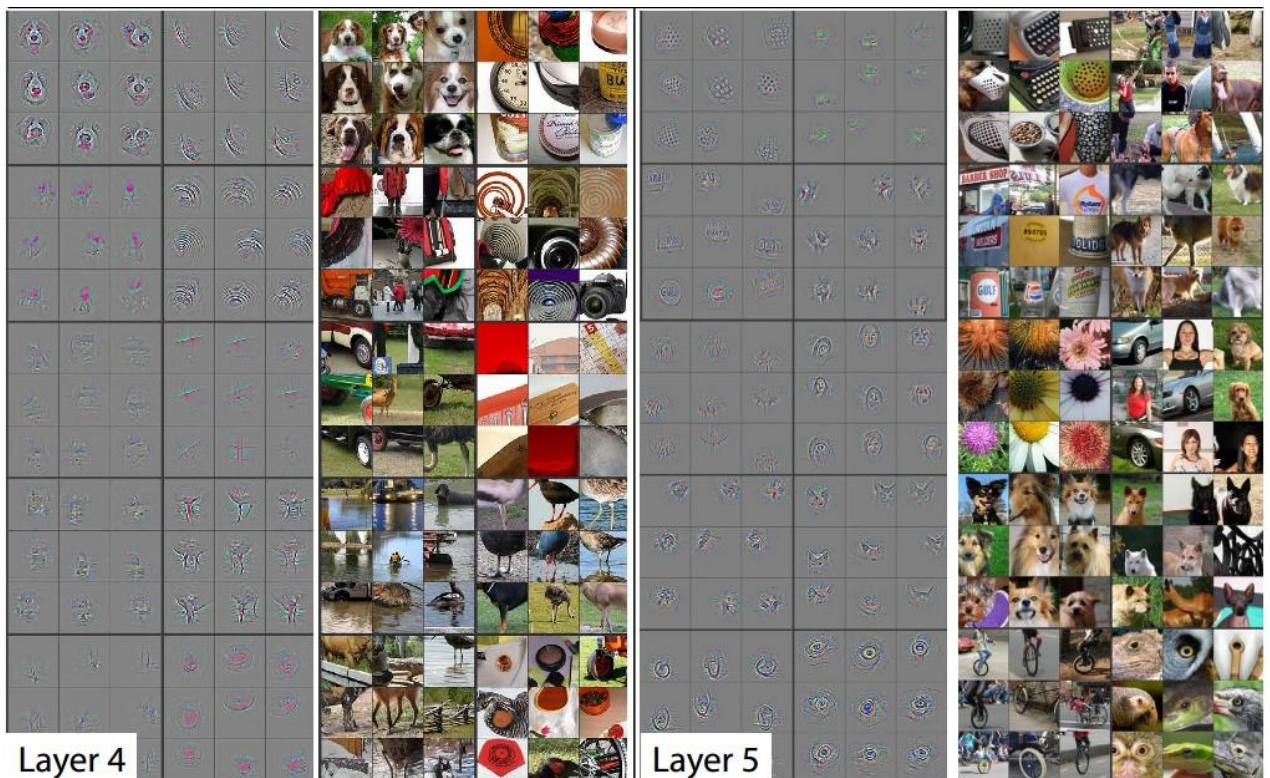
Visualization arbitrary neurons



Visualization arbitrary neurons

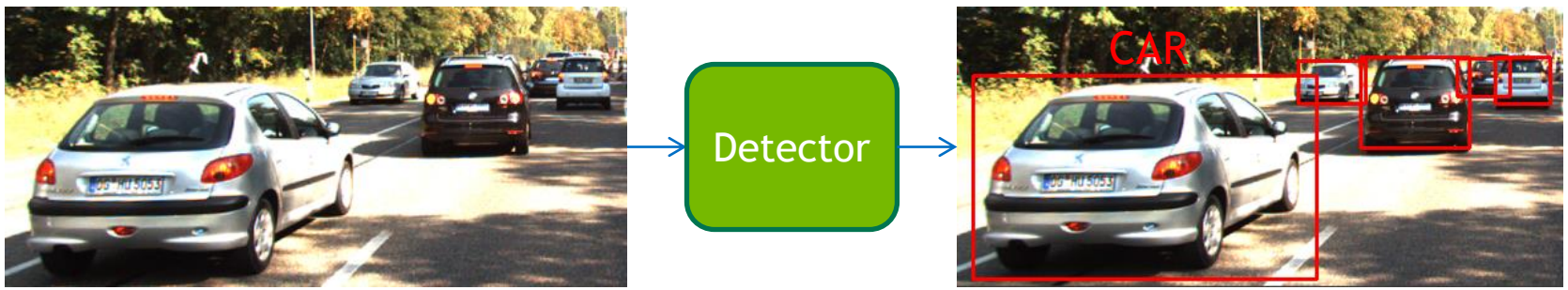


Visualization arbitrary neurons



Localization and Detection

Object localization goals

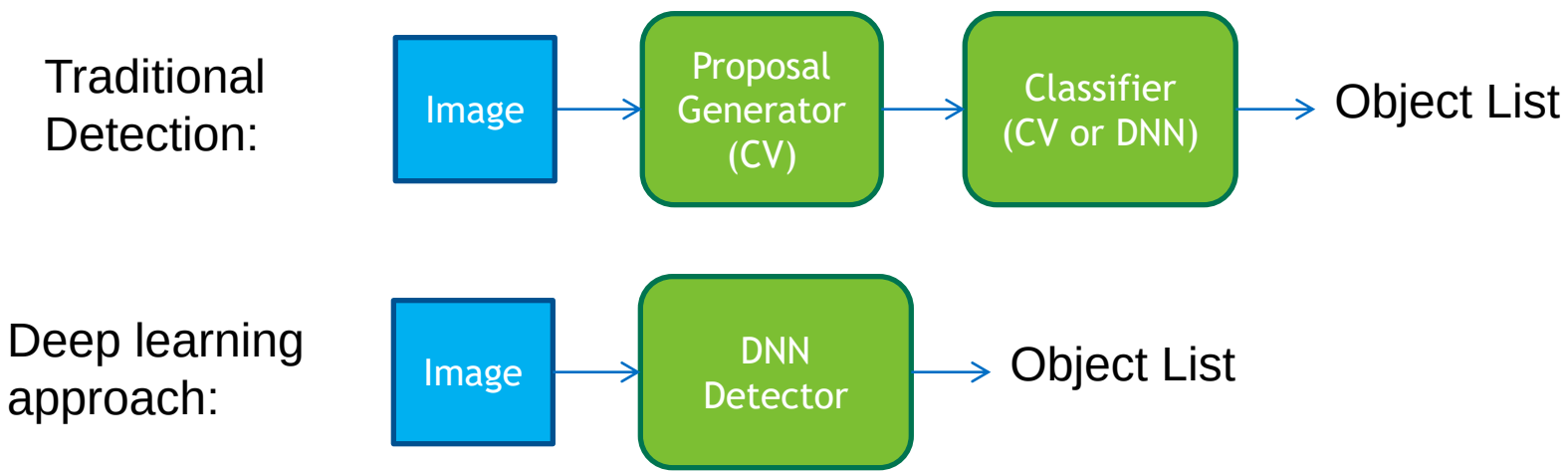


Given an image output two things:

object class (integer)

x_1, y_1, x_2, y_2 bounding box coordinates

Localization pipeline

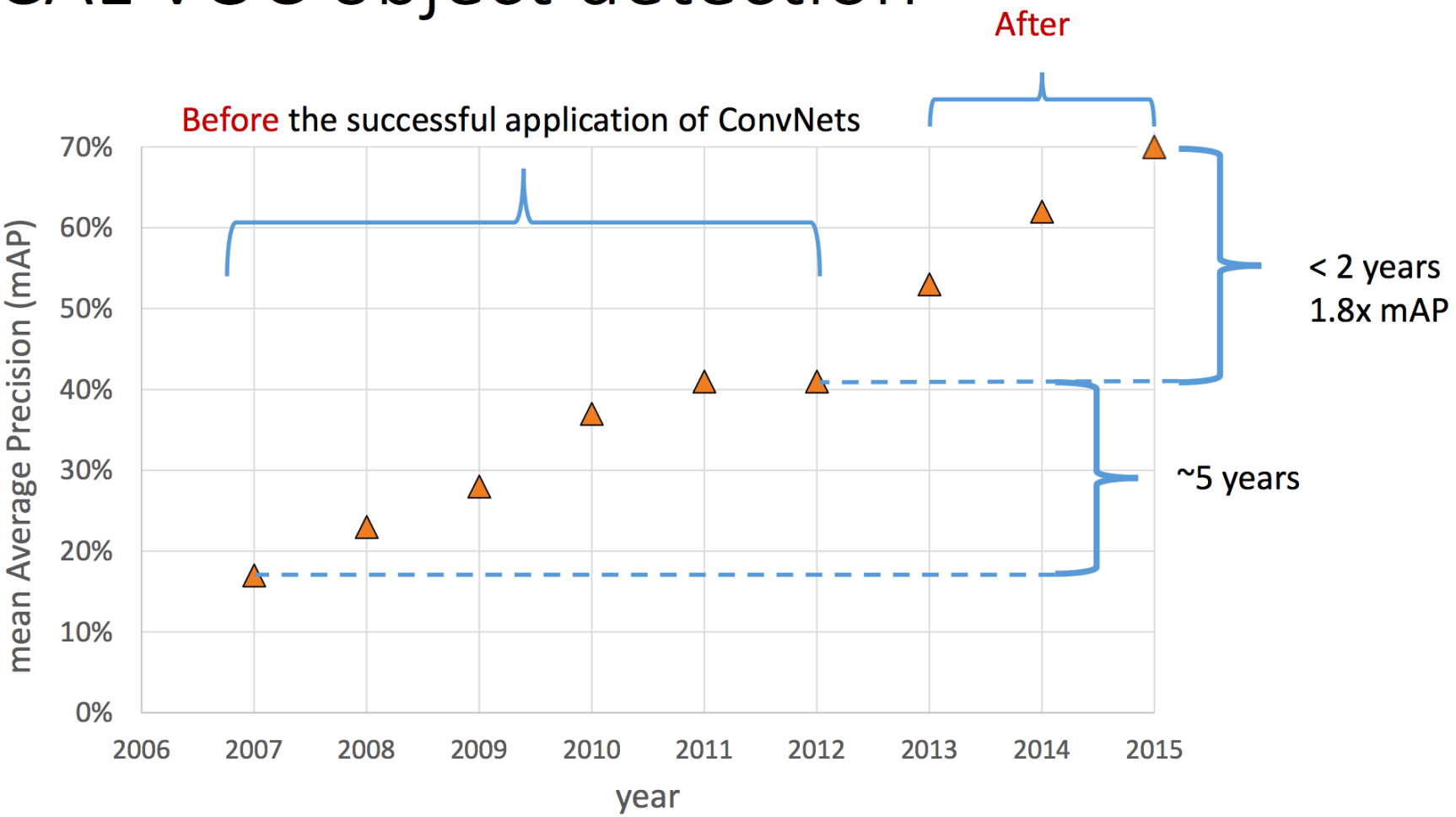


In one shot, get an object list

Goal is real-time detection rates

Traditional detection pipelines can generate 1000's of proposals, which may get in the way of realtime

PASCAL VOC object detection



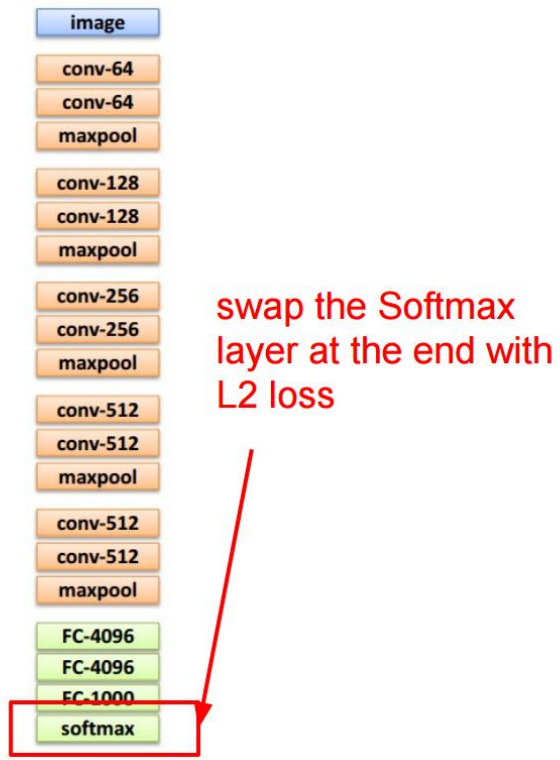
Slide credit: Ross Girshick, Microsoft Research

Idea: train a localization network

Very Deep Convolutional Networks for Large-Scale Image Recognition, Simonyan et al., 2014

OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, Sermanet et al., 2014

- Take out Softmax loss, swap in L2 (regression) loss, fine-tune the classification network.



Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Idea: train a localization network

Very Deep Convolutional Networks for Large-Scale Image Recognition, Simonyan et al., 2014

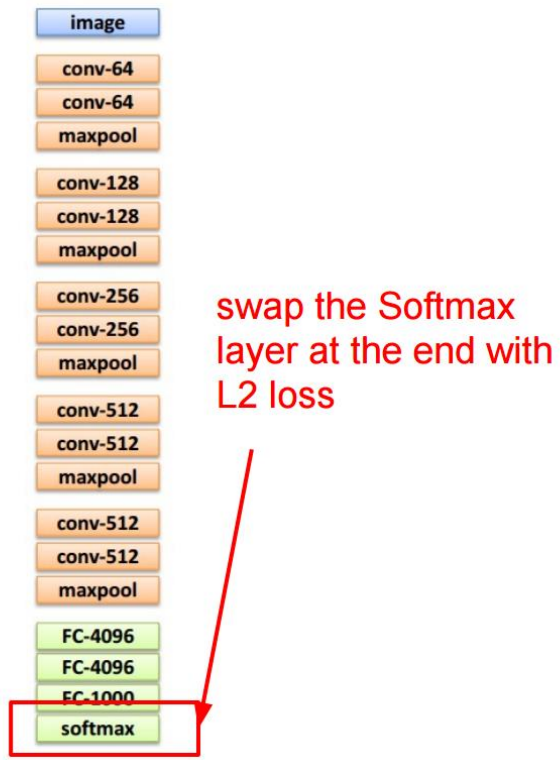
OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, Sermanet et al., 2014

- Take out Softmax loss, swap in L2 (regression) loss, fine-tune the classification network.

predictions: instead of class scores, now interpreted as the 4 bounding box coords **(also 4D vector from net)**

targets: true bounding box **4D vector of [x1,y1,x2,y2]**

$$L_i = \|f - y_i\|_2^2$$



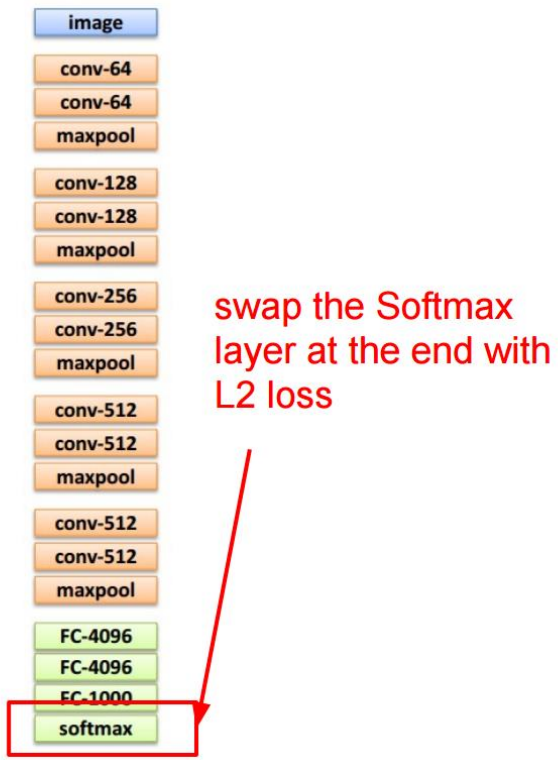
Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Idea: train a localization network

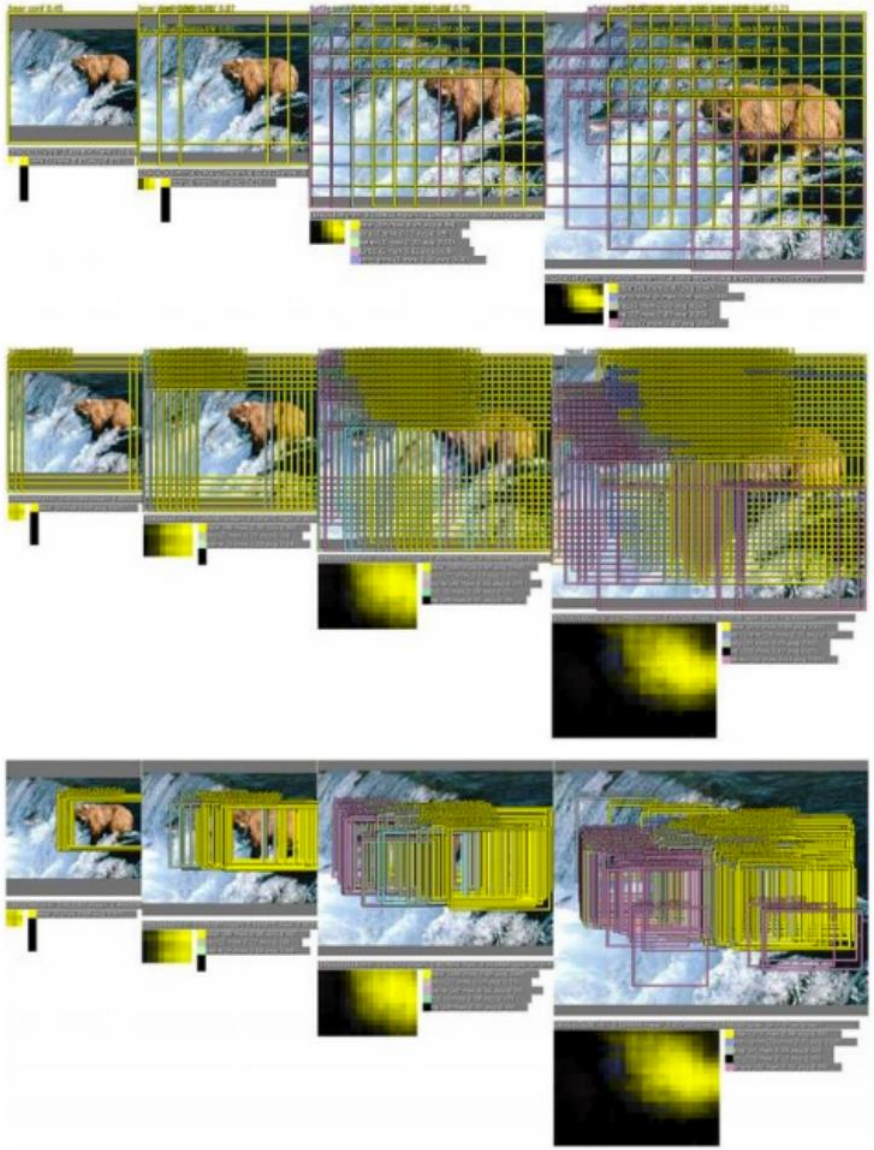
Very Deep Convolutional Networks for Large-Scale Image Recognition, Simonyan et al., 2014

OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, Sermanet et al., 2014

- **In practice:**
  - It works better to predict a **4D vector for every class** (e.g. 4000D vector for 1000 ImageNet classes). During training only backprop the loss for the correct class
  - apply at **multiple locations and scales**

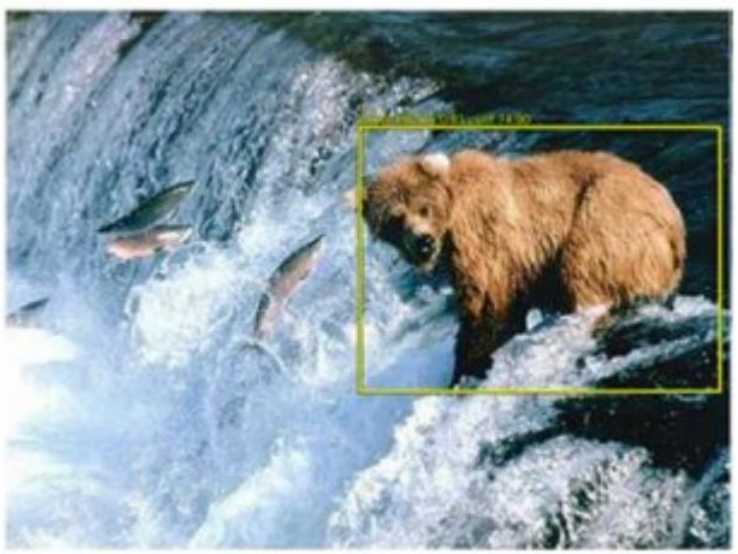


Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n



OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,
Sermanet et al., 2014

greedy merging procedure



Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Tip: Note that we can apply ConvNets over all positions in an image very efficiently

Normal ConvNet:

[224x224x3] -> … -> [7x7x512] -> [1x1x4096] -> … -> [1x1x1000]

last volume first FC layer class scores

Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Tip: Note that we can apply ConvNets over all positions in an image very efficiently

Normal ConvNet:

[224x224x3] -> … -> [7x7x512] -> [1x1x4096] -> … -> [1x1x1000]

last volume first FC layer class scores

Convert the first FC layer into a CONV layer:

Note: This FC layer is equivalent to CONV layer with:

receptive field size of 7x7, pad 0, stride 1, and 4096 neurons

Convert later FC layers: receptive field sizes 1x1, pad 0, stride 1

Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Tip: Note that we can apply ConvNets over all positions in an image very efficiently

Normal ConvNet:

[224x224x3] -> … -> [7x7x512] -> [1x1x4096] -> … -> [1x1x1000]

last volume first FC layer class scores

Convert the first FC layer into a CONV layer:

Note: This FC layer is equivalent to CONV layer with:

receptive field size of 7x7, pad 0, stride 1, and 4096 neurons

Convert later FC layers: receptive field sizes 1x1, pad 0, stride 1

Modified ConvNet:

[384x384x3] -> … -> [**12**x**12**x512] -> [**6**x**6**x4096] -> … -> [**6**x**6**x1000]

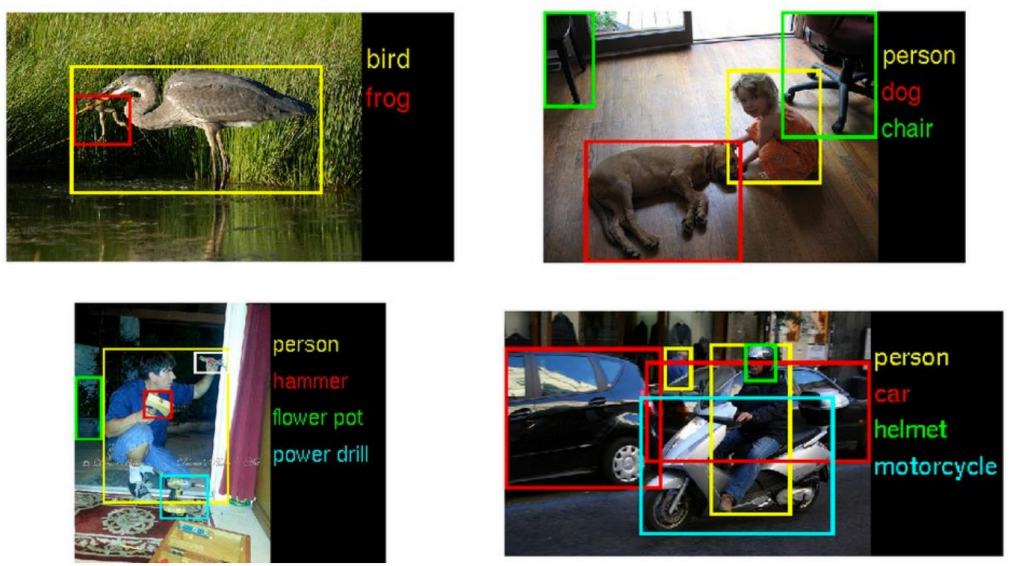
last volume 7x7 CONV 1x1 CONV

class scores volume!

This is very efficient, can be seen as evaluating the FC layer in parallel on many locations in the image. This is a common trick. Same trick used in localization.

Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Detection



- Given an image, output 3 things:
  - Confidence
  - Class (integer)
  - x_1, y_1, x_2, y_2 bounding box coordinates

Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Rich feature hierarchies for accurate object detection and semantic segmentation
[Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik]

***Idea**: Turn a Detection Problem into an Image Classification problem (but over image regions).*

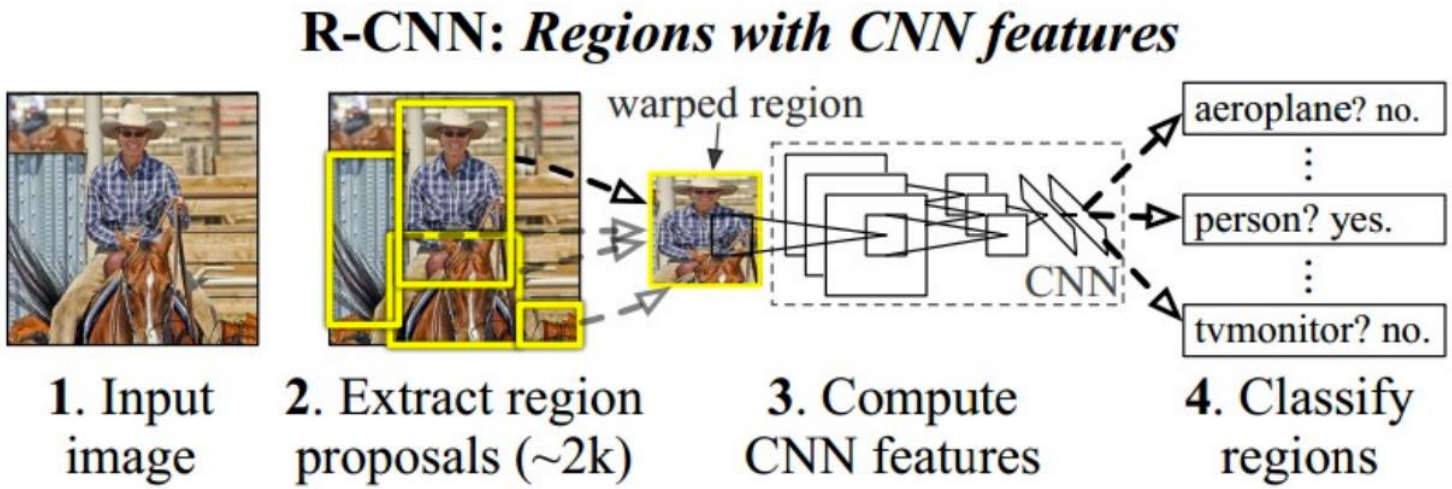
Content of every labeled bounding box for is a positive example for a class.

Every other bounding box in the image is a special **negative class**.

Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Rich feature hierarchies for accurate object detection and semantic segmentation
[Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik]

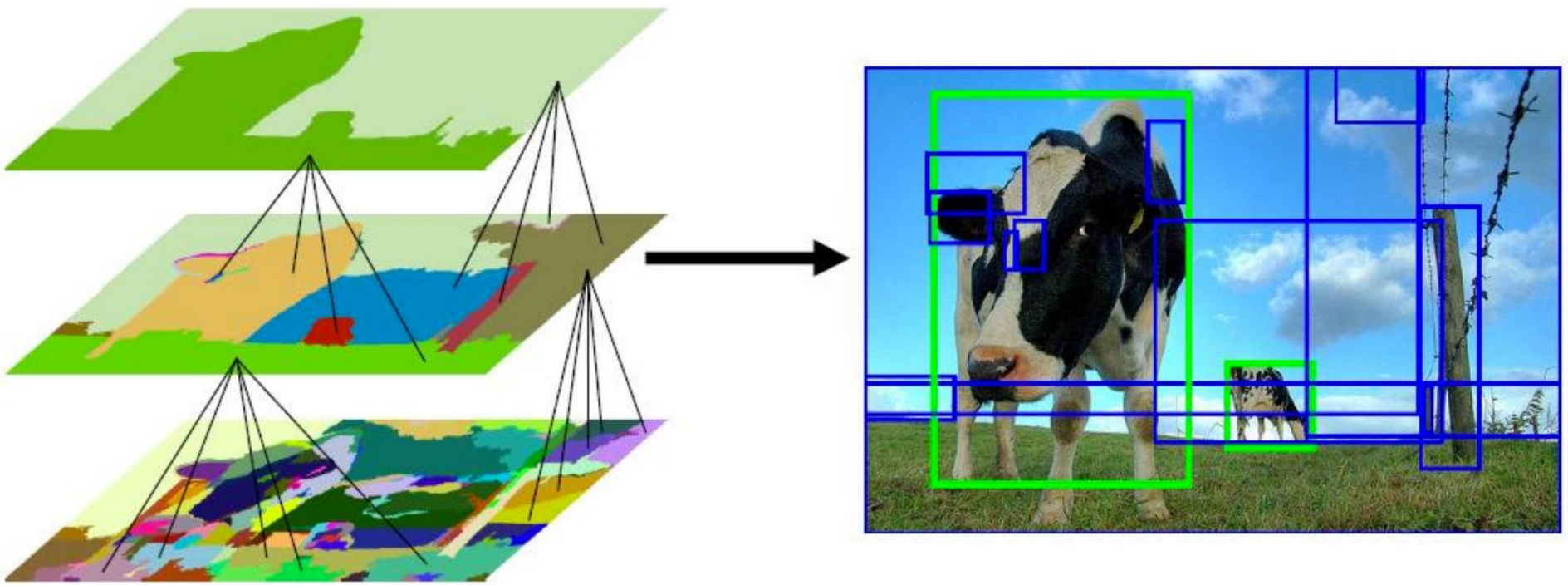
***Idea**: Turn a Detection Problem into an Image Classification problem (but over image regions).*



Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Selective Search for Object Recognition

[J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders]



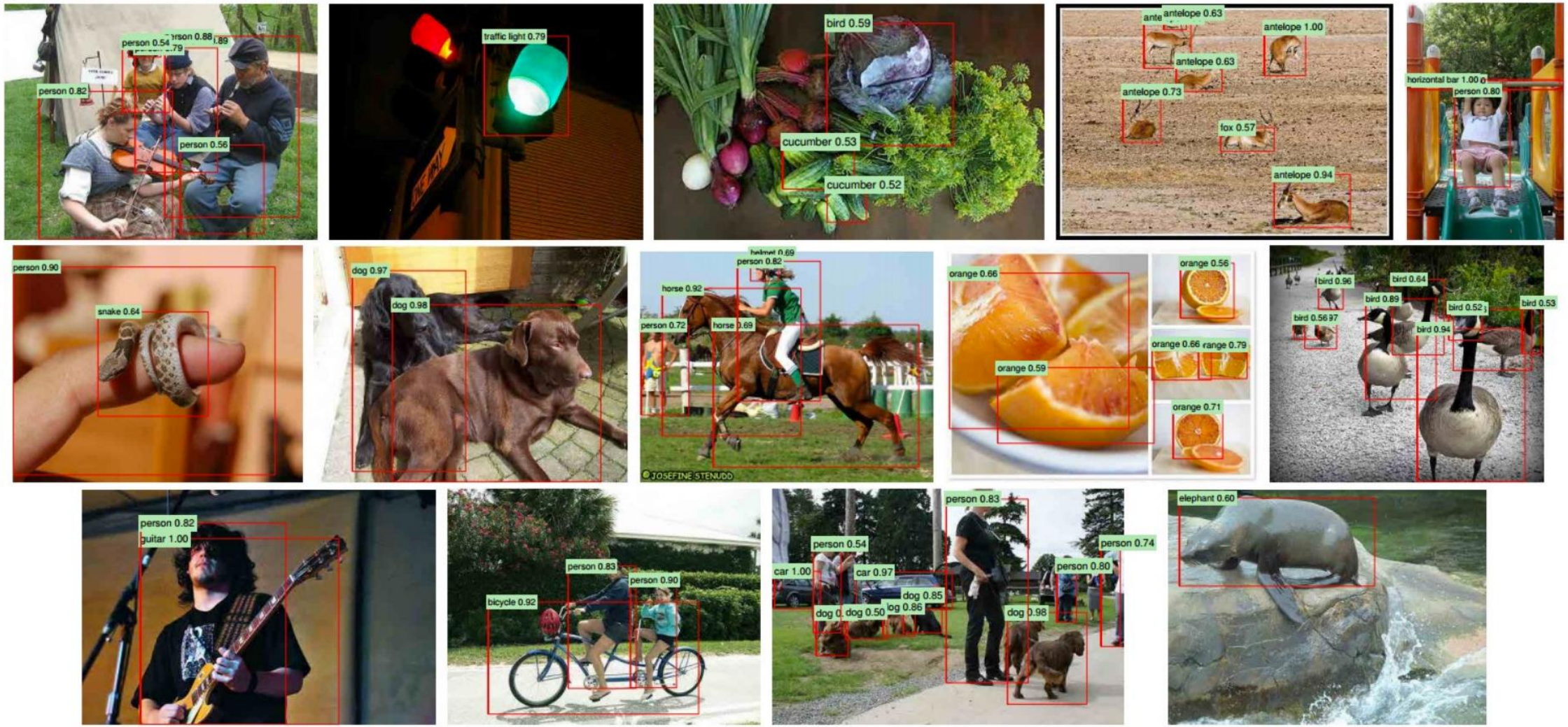
Gives on average ~2,000 candidate region proposals per image.

(This paradigm currently outperform the "sliding window" approach)

Slide credit: Fei-Fei Li & Andrej Karpathy, Stanford cs231n

Rich feature hierarchies for accurate object detection and semantic segmentation

[Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik]



Detection software

Compiled C++ binaries, Torch and Python APIs for Overfeat: http://cilvr.nyu.edu/doku.php?id=software:overfeat:start

Caffe fork implementing R-CNN (requires MATLAB): https://github.com/rbgirshick/rcnn

Caffe fork implementing Fast R-CNN: https://github.com/rbgirshick/fast-rcnn

Theano based Python library with Overfeat wrapper: http://sklearn-theano.github.io/

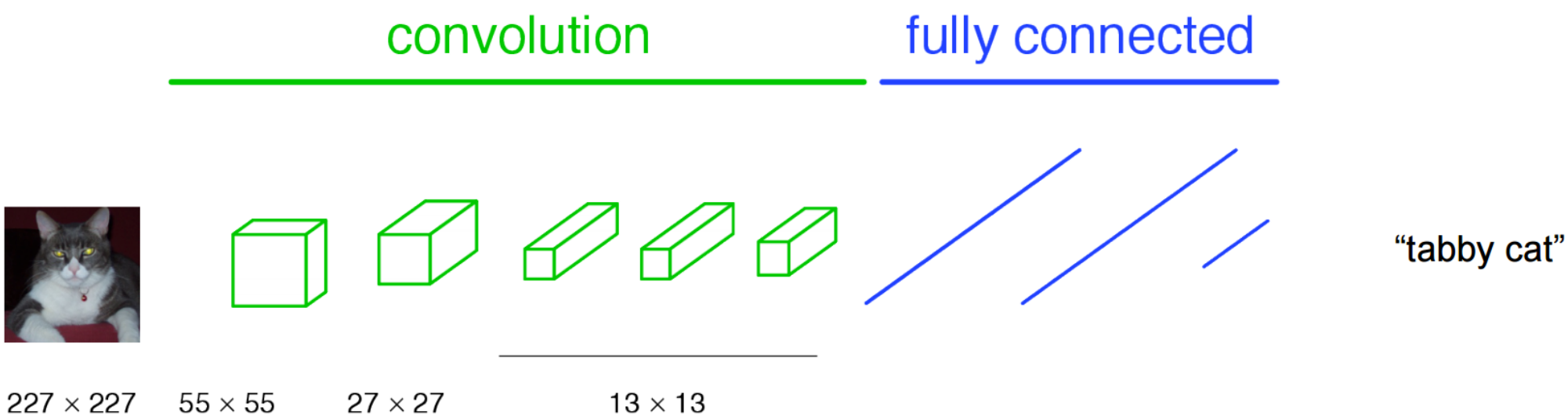
Torch based R-CNN implementation: https://github.com/fmassa/object-detection.torch

Practice

Object Detection for
PASCAL VOC
KITTI Dataset

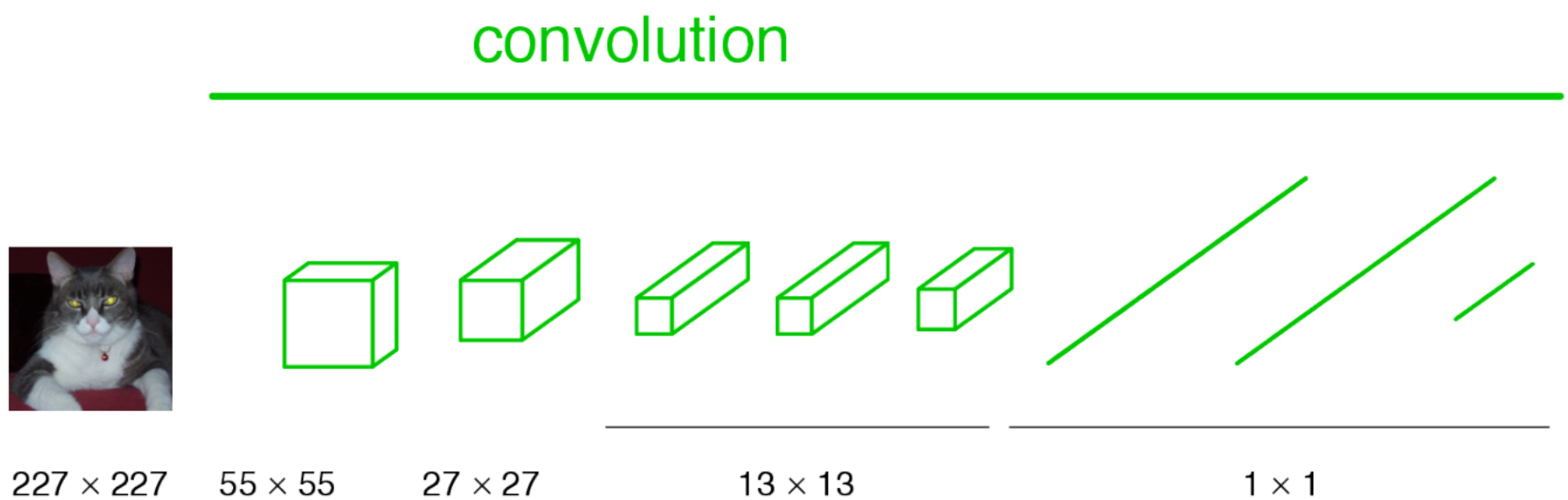
Segmentation

A classification network



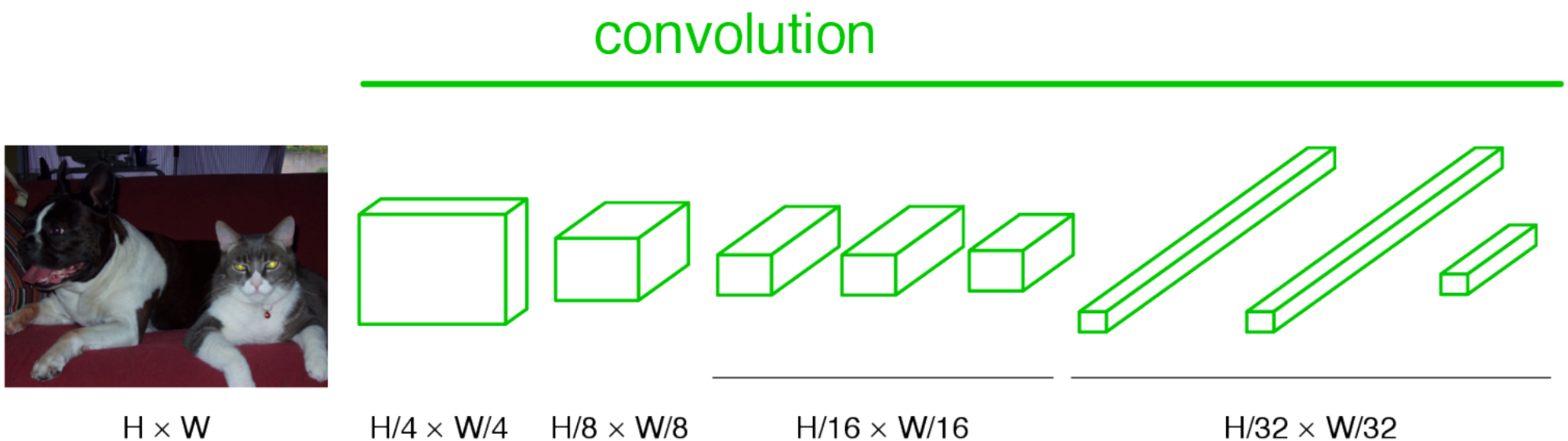
Slide credit: Jon Long and Evan Shelhamer, CVPR 2015

Becoming fully convolutional



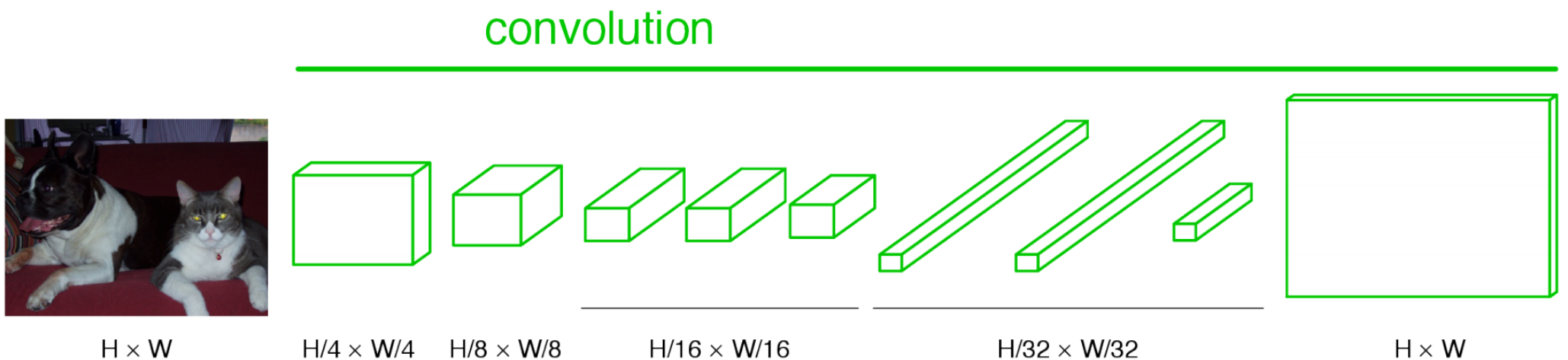
Slide credit: Jon Long and Evan Shelhamer, CVPR 2015

Becoming fully convolutional



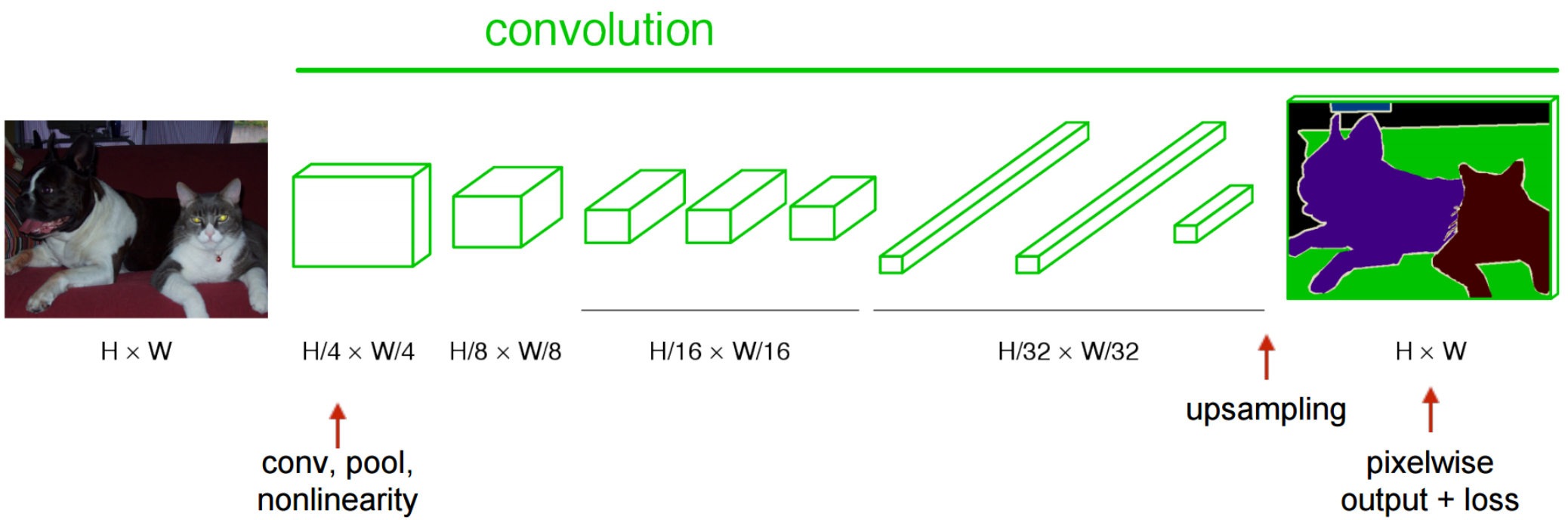
Slide credit: Jon Long and Evan Shelhamer, CVPR 2015

Upsampling output



Slide credit: Jon Long and Evan Shelhamer, CVPR 2015

End-to-end pixels-to-pixels network

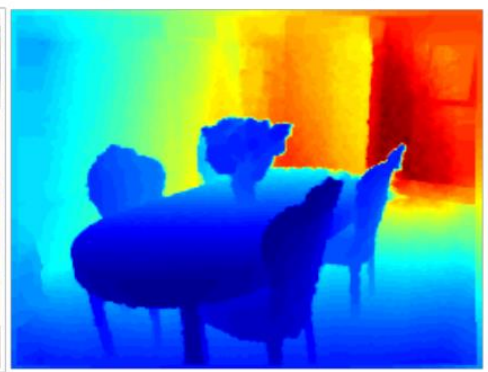


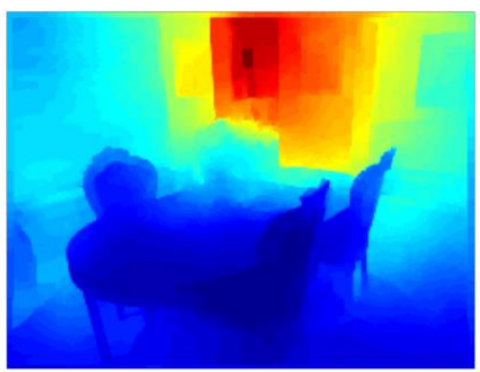
Slide credit: Jon Long and Evan Shelhamer, CVPR 2015

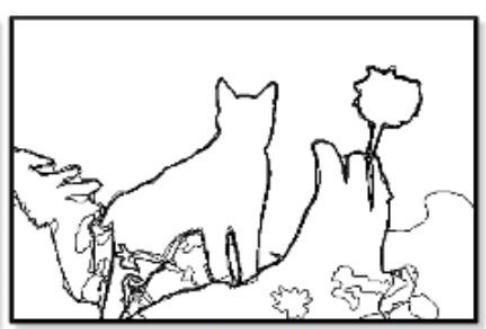
Applications

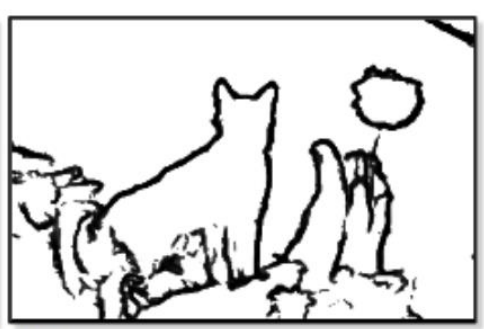
semantic segmentation

monocular depth estimation (Liu et al. 2015)



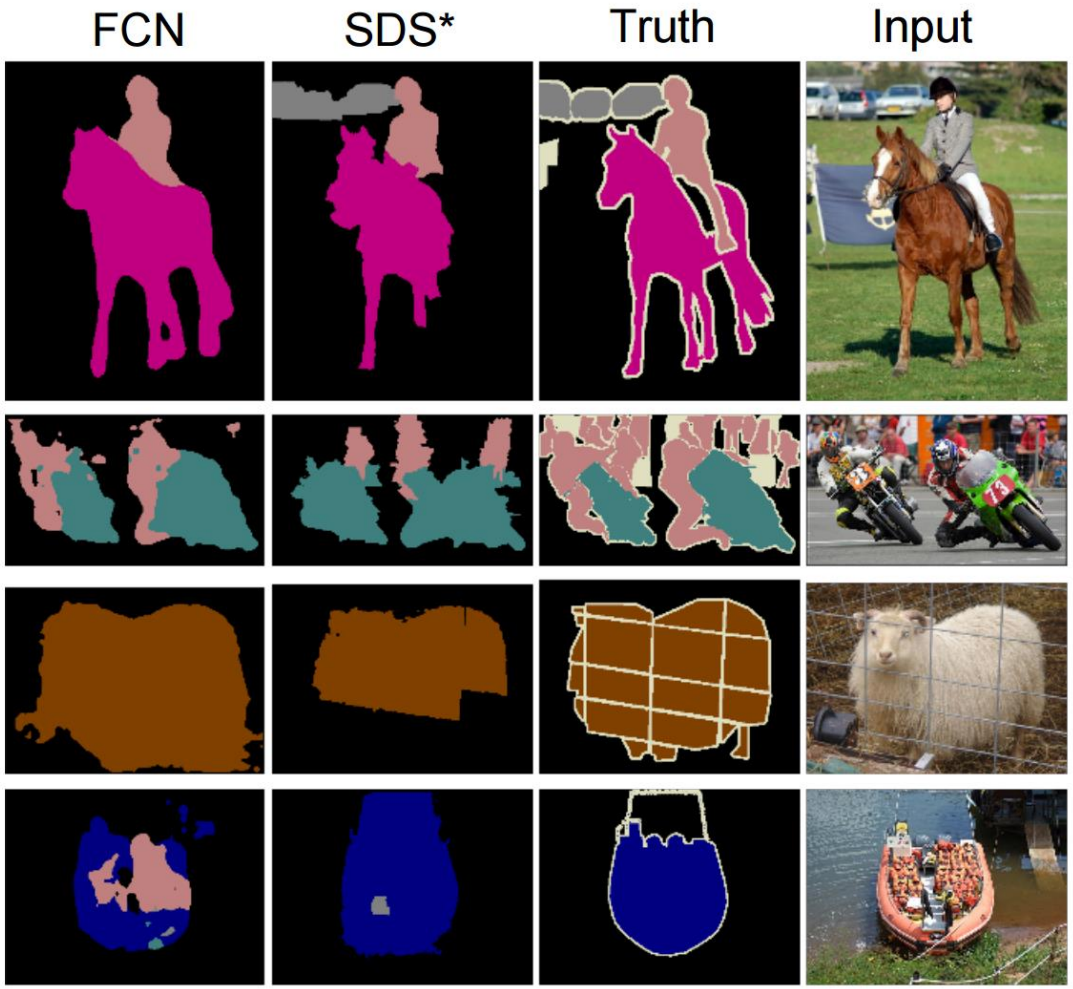






boundary prediction (Xie & Tu 2015)

Segmentation performance



Relative to prior state-of-the-art SDS:

- 20% improvement for mean IoU
- 286× faster

*Simultaneous Detection and Segmentation Hariharan et al. ECCV14

Software

Caffe branch implementing FCN: fcn.berkeleyvision.org

Practice

Image Segmentation
PASCAL VOC

Neural Style


SUNDRYMULTIFORM

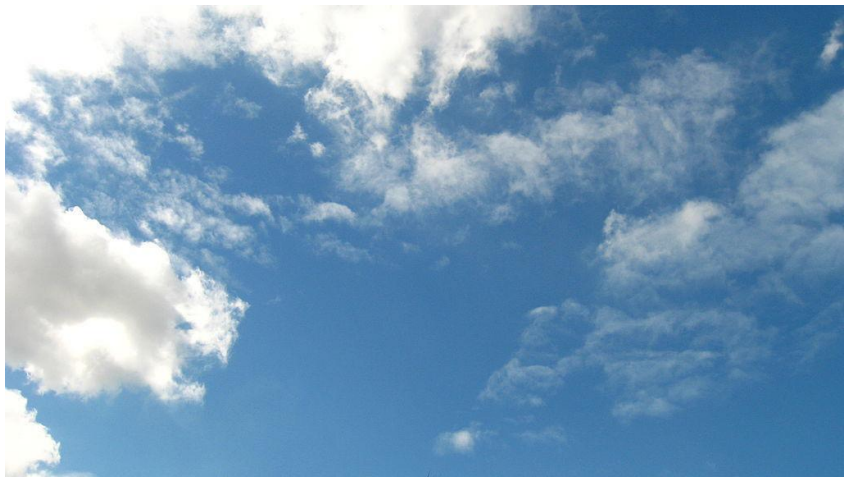
```python
def objective_L2(dst):
    dst.diff[:] = dst.data

def make_step(net, step_size=1.5, end='inception_4c/output',
              jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst)  # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)
```

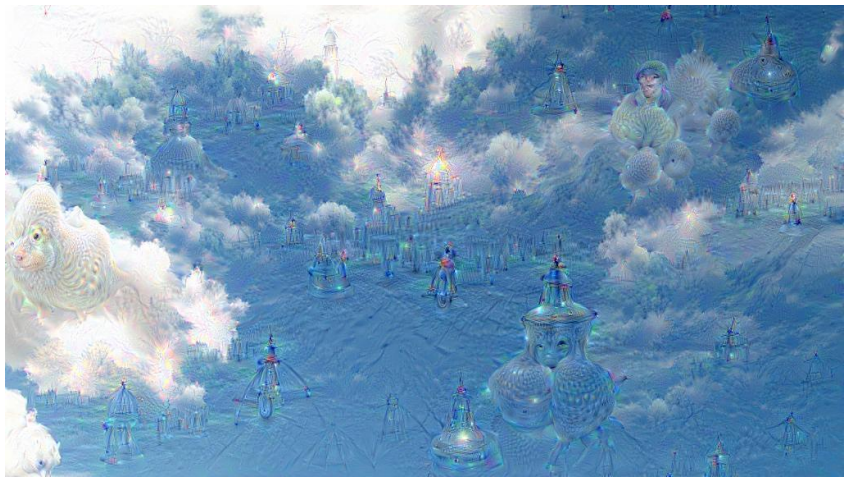
```python
def objective_L2(dst):
    dst.diff[:] = dst.data

def make_step(net, step_size=1.5, end='inception_4c/output',
              jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst)  # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)
```

Jitter regularizer

"image update"

inception_4c/output

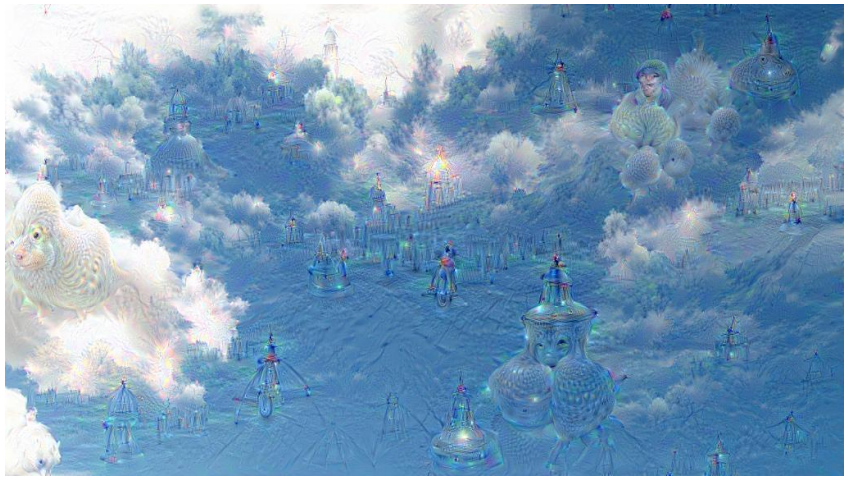




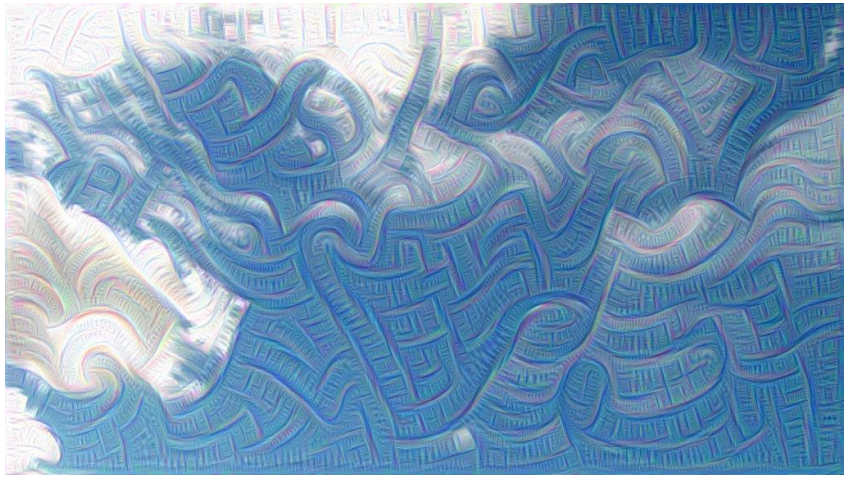
DeepDream modifies the image in a way that “boosts” all activations, at any layer

this creates a feedback loop: e.g. any slightly detected dog face will be made more and more dog like over time

inception_4c/output

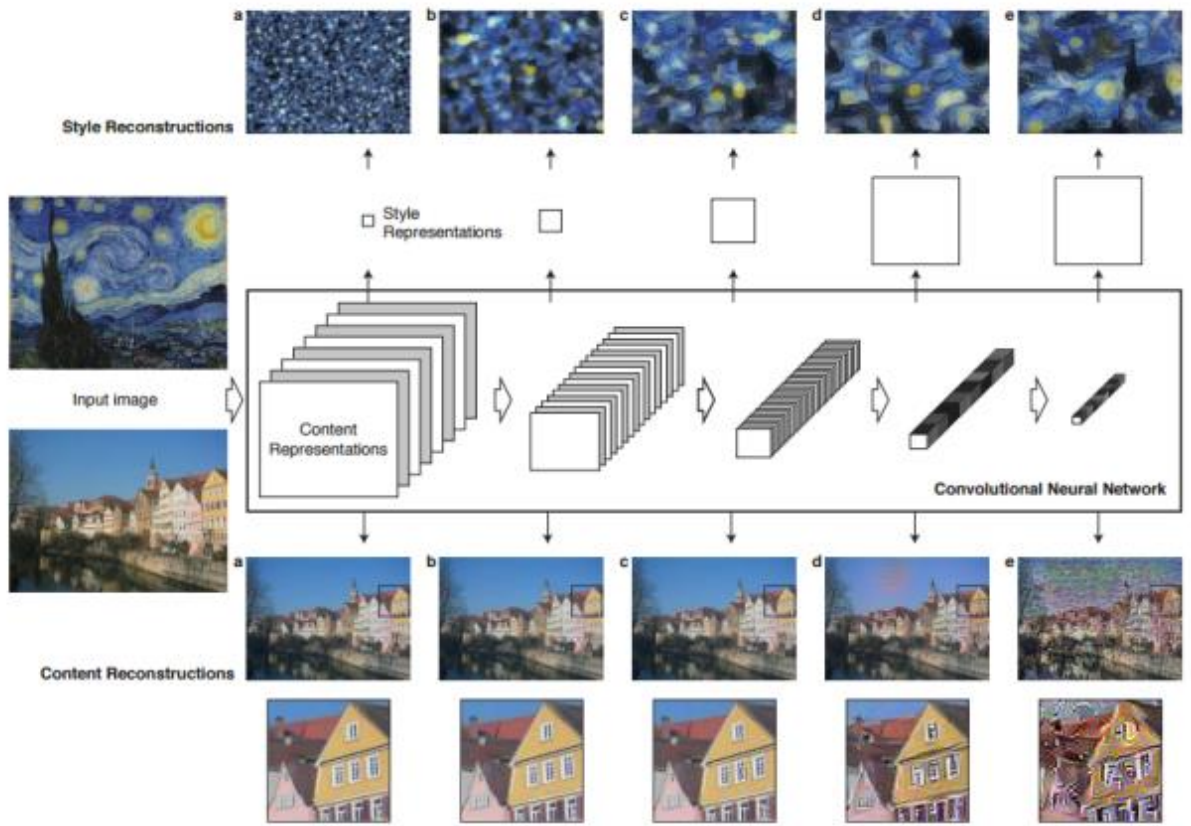


inception_3b/5x5_reduce



DeepDream modifies the image in a way that “boosts” all activations, at any layer

Neural Style



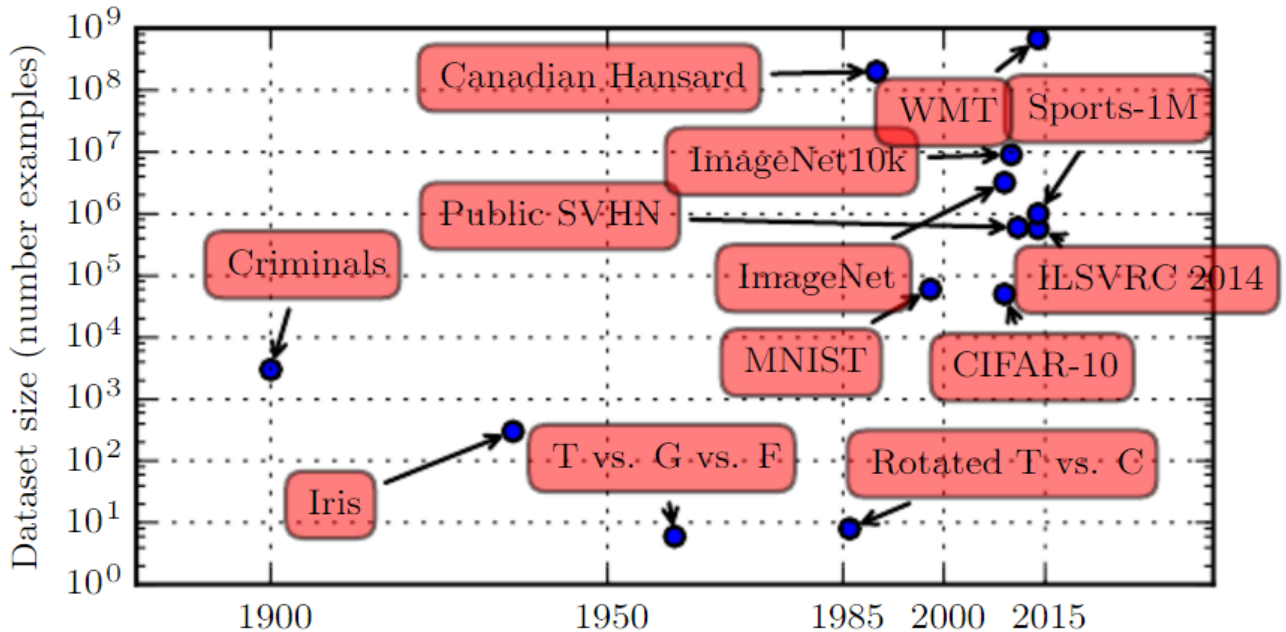
Practice

Google Deep Dream

Transfer Learning

Should I training all this?

- Dataset sizes have increased greatly over time
- What about your problem?



The deep learning book, Ian Goodfellow and Yoshua Bengio and Aaron Courville

Transfer learning

Instead of initializing network with random weights we use the weights that have been learned on a different dataset

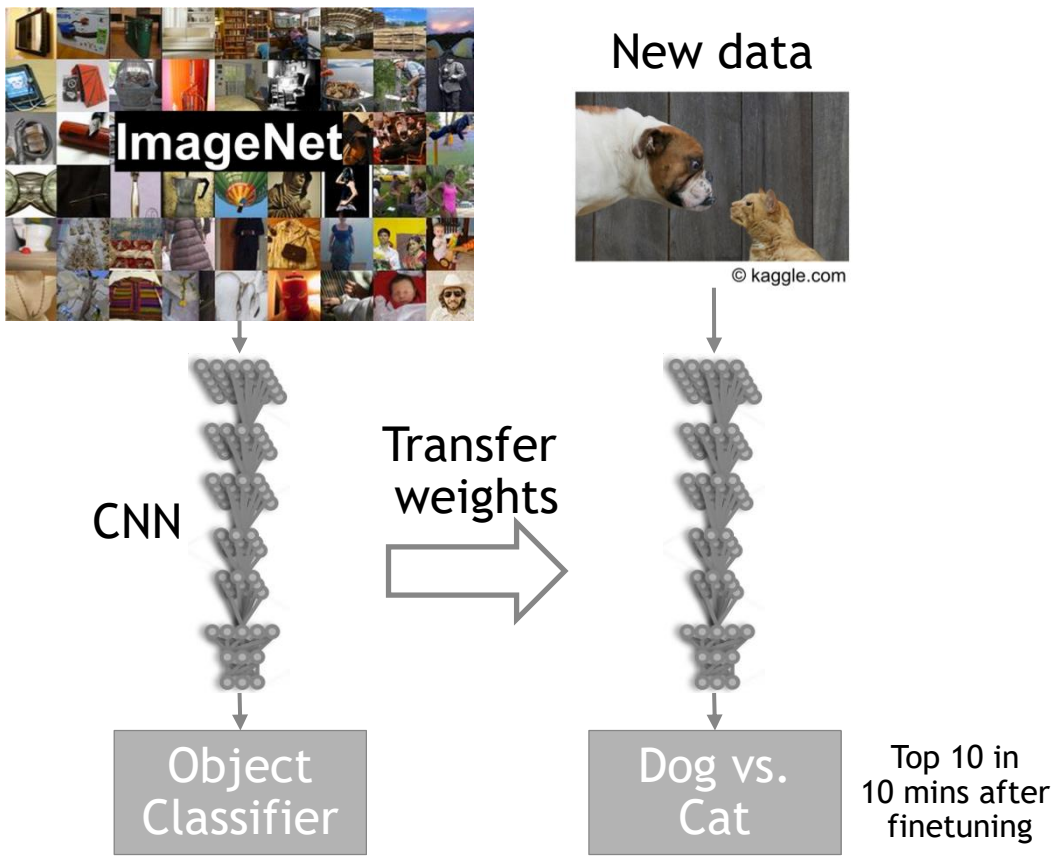
We then continue to train the network (or part of it) – called fine-tuning – using our true dataset of interest

Three modes:

- If new dataset is small and similar to original, just train a linear classifier on high-level features output by original network. Fine-tuning whole network would overfit.
- If new dataset is large and similar, fine-tune the whole network
- If new dataset is small and different, train linear classifier on mid-level features from original network
- If new dataset is large and different, still fine-tune the whole network as convergence is often still faster than starting from scratch

Transfer learning

What is it?



Myth: You need a lot of data to train deep neural networks

Transfer learning

What is it?

New data

CNN

Transfer weights

Object Classifier

Dog vs. Cat

Top 10 in 10 mins after finetuning

Myth: You need a lot of data to train deep neural networks

BUSTED

Pretrained model

Starting with other's effort

Deep learning software by name [edit]

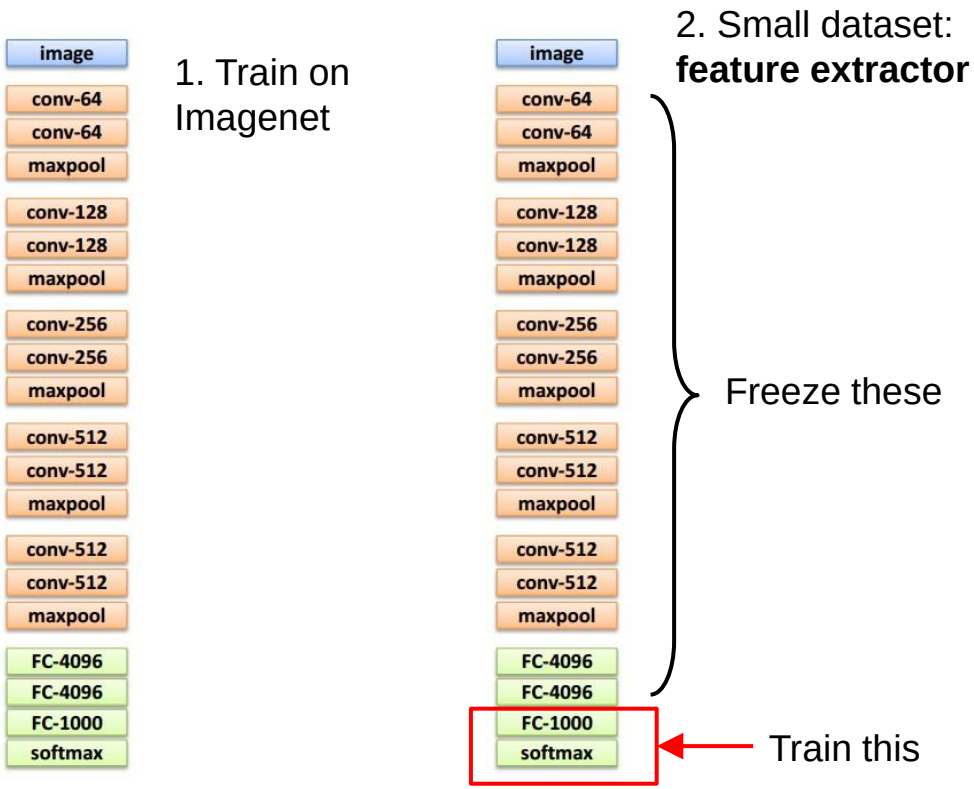
This list is incomplete; you can help by expanding it.

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Apache Singa	Apache Incubator	Apache 2.0	Yes	Linux, Mac OS X, Windows	C++	Python, C++, Java	No	Yes	Yes	?	No	Yes	Yes	Yes	Yes
Caffe[2]	Berkeley Vision and Learning Center, community contributors	BSD 2-Clause License	Yes	Linux, Mac OS X, AWS,[3] unofficial Android port,[4] Windows support by Microsoft Research,[5] unofficial Windows port[6]	C++	C++, command line, Python, MATLAB[7]	No	Branch,[8] pull request,[9] third party implementation[10]	Yes	?	Yes[11]	Yes	Yes	No[12]	Yes[13]
Chainer[14]	Preferred Networks, inc., Preferred Infrastructure, inc.[14]	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	No	On roadmap[15]	Yes	Yes	Through Caffe's model zoo[16]	Yes	Yes	Yes	Yes[17]
cnn[18]	Carnegie Mellon School of Computer Science	Apache 2.0	Yes	Linux	C++	C++, Python	No	No	Yes	Yes	No	Yes	Yes	No	No
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, Mac OS X, Windows, Android (Cross-platform)	C, C++	Java, Scala, Clojure	Yes	On roadmap[19]	Yes[20]	Computational Graph	Yes[21]	Yes	Yes	Yes	Yes[22]
Keras	François Chollet	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	Only if using Theano as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[23]	Yes	Yes	Yes	Yes[24]
Knet.jl[25]	Deniz Yuret	MIT license	Yes	Linux, Mac OS X	Julia	Julia	?	?	Yes	Yes	?	Yes	Yes	?	?
Microsoft Cognitive Toolkit - CNTK	Microsoft Research	MIT license[26]	Yes	Windows, Linux[27] (OSX via Docker on roadmap)	C++	Python, C++, Command line,[28] BrainScript[29] (.NET on roadmap[30])	Yes[31]	No	Yes	Yes	Yes[32]	Yes[33]	Yes[33]	No[34]	Yes[35]
MXNet[36]	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Linux, Mac OS X, Windows,[37][38] AWS, Android,[39] iOS, JavaScript[40]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala	Yes	On roadmap[41]	Yes	Yes[42]	Yes[43]	Yes	Yes	Yes	Yes[44]
Neural Designer	Artelnics	Proprietary	No	Linux, Mac OS X, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	No	?	?	No	No	No	?

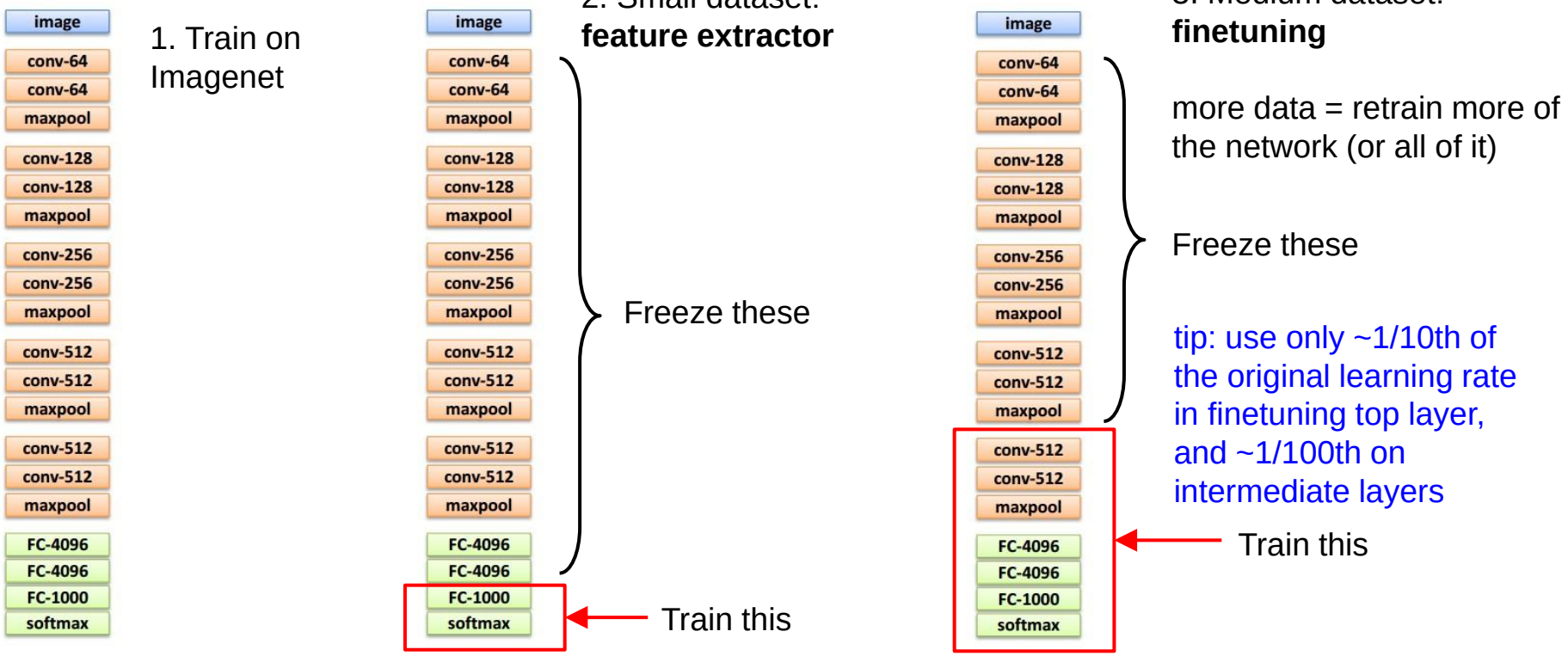
Transfer learning with CNNs

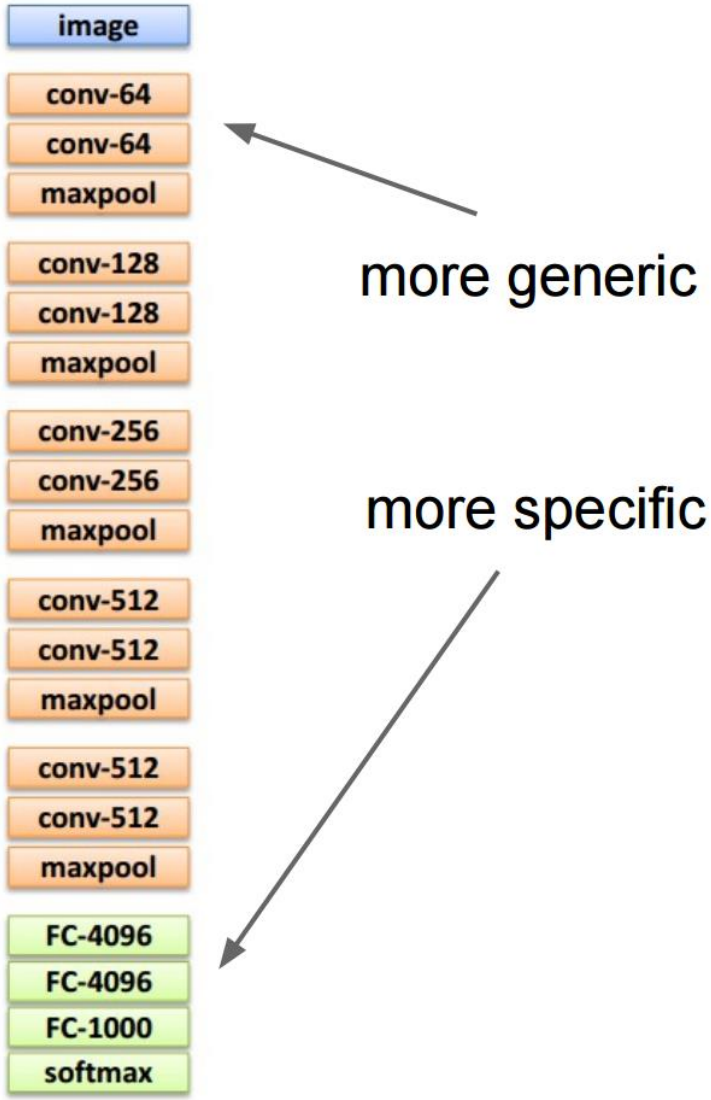
1. Train on Imagenet

Transfer learning with CNNs



Transfer learning with CNNs





	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Pretrained model

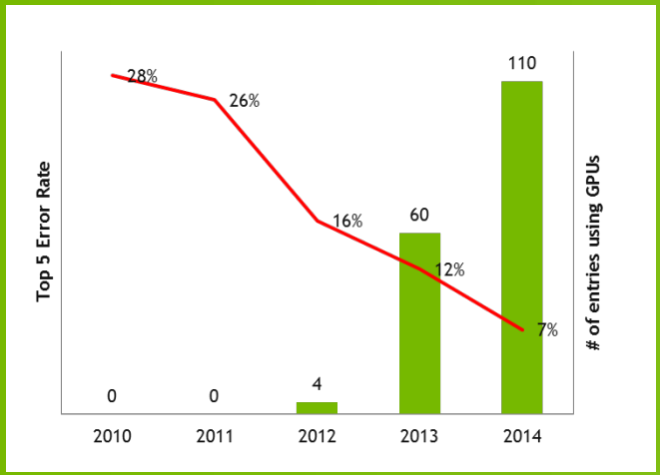
Starting with other's effort

Deep learning software by name [edit]

This list is incomplete; you can help by expanding it.

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Apache Singa	Apache Incubator	Apache 2.0	Yes	Linux, Mac OS X, Windows	C++	Python, C++, Java	No	Yes	Yes	?	No	Yes	Yes	Yes	Yes
Caffe[2]	Berkeley Vision and Learning Center, community contributors	BSD 2-Clause License	Yes	Linux, Mac OS X, AWS,[3] unofficial Android port,[4] Windows support by Microsoft Research,[5] unofficial Windows port[6]	C++	C++, command line, Python, MATLAB[7]	No	Branch,[8] pull request,[9] third party implementation[10]	Yes	?	Yes[11]	Yes	Yes	No[12]	Yes[13]
Chainer[14]	Preferred Networks, inc., Preferred Infrastructure, inc.[14]	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	No	On roadmap[15]	Yes	Yes	Through Caffe's model zoo[16]	Yes	Yes	Yes	Yes[17]
cnn[18]	Carnegie Mellon School of Computer Science	Apache 2.0	Yes	Linux	C++	C++, Python	No	No	Yes	Yes	No	Yes	Yes	No	No
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, Mac OS X, Windows, Android (Cross-platform)	C, C++	Java, Scala, Clojure	Yes	On roadmap[19]	Yes[20]	Computational Graph	Yes[21]	Yes	Yes	Yes	Yes[22]
Keras	François Chollet	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	Only if using Theano as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[23]	Yes	Yes	Yes	Yes[24]
Knet.jl[25]	Deniz Yuret	MIT license	Yes	Linux, Mac OS X	Julia	Julia	?	?	Yes	Yes	?	Yes	Yes	?	?
Microsoft Cognitive Toolkit - CNTK	Microsoft Research	MIT license[26]	Yes	Windows, Linux[27] (OSX via Docker on roadmap)	C++	Python, C++, Command line,[28] BrainScript[29] (.NET on roadmap[30])	Yes[31]	No	Yes	Yes	Yes[32]	Yes[33]	Yes[33]	No[34]	Yes[35]
MXNet[36]	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Linux, Mac OS X, Windows,[37][38] AWS, Android,[39] iOS, JavaScript[40]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala	Yes	On roadmap[41]	Yes	Yes[42]	Yes[43]	Yes	Yes	Yes	Yes[44]
Neural Designer	Artelnics	Proprietary	No	Linux, Mac OS X, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	No	?	?	No	No	No	?

“Using 4 NVIDIA Kepler GPUs and optimizations described below, training took from 3.5 days for the 18-layer model to 14 days for the 101-layer model.”

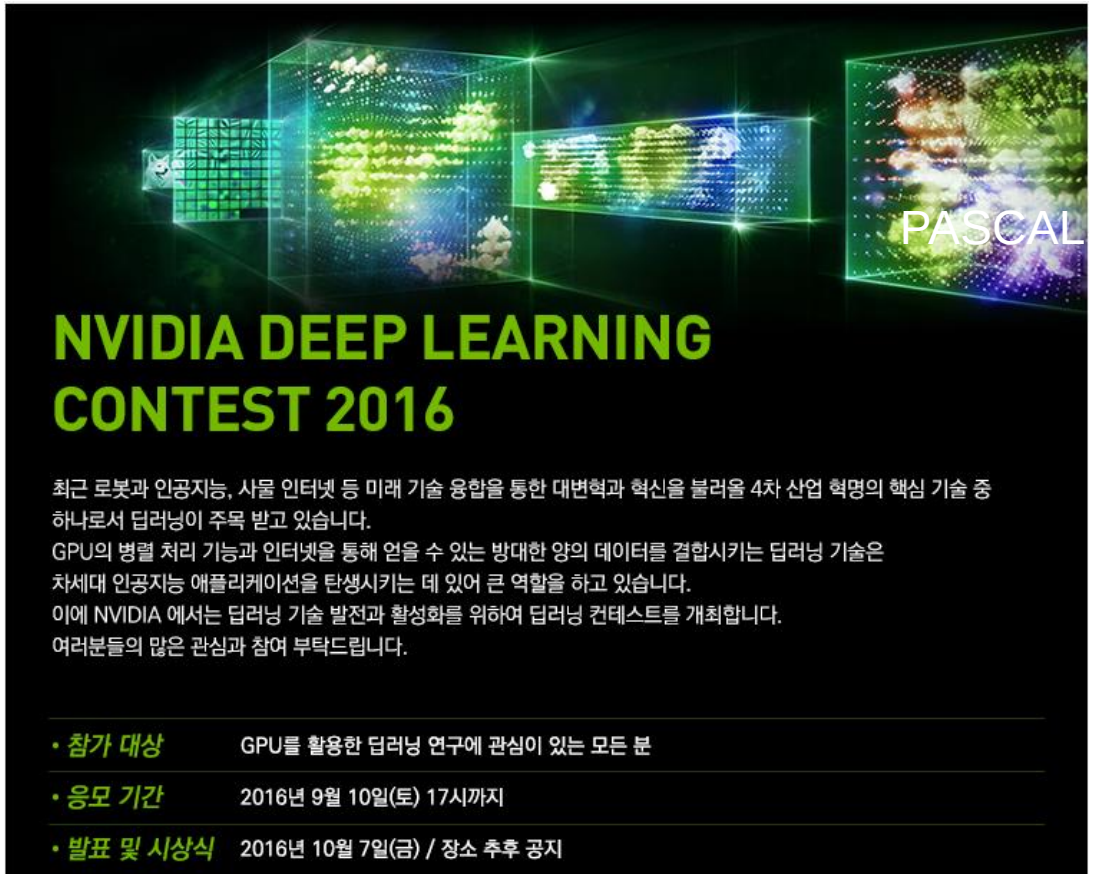


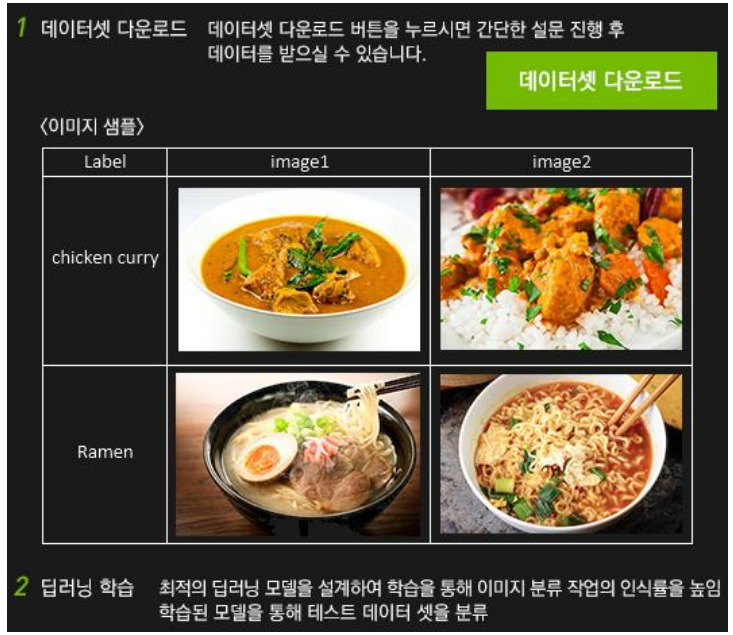
- Torch & Facebook tech blog

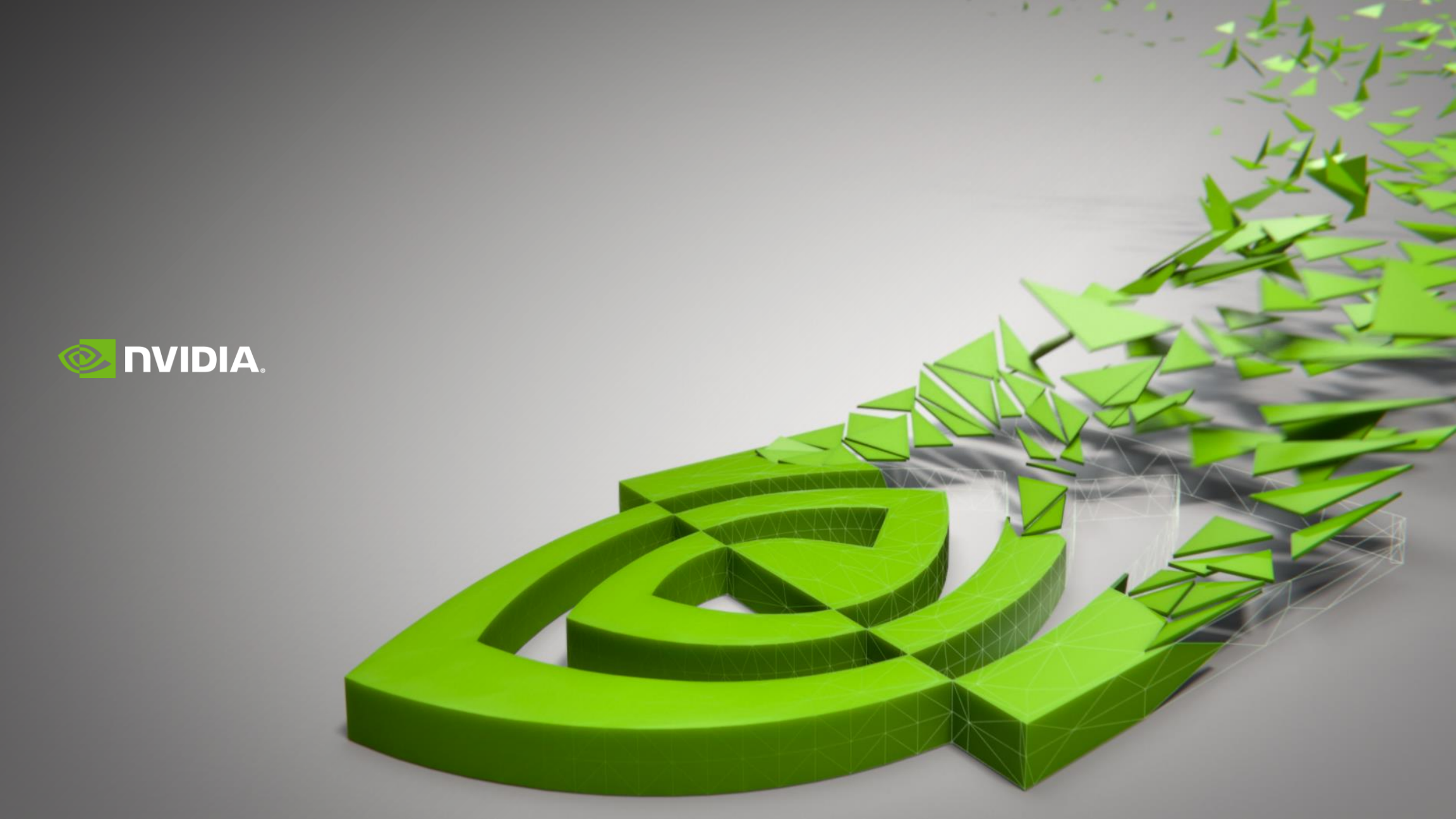
http://torch.ch/blog/2016/02/04/resnets.html

Practice

Food dataset-101 + k-foods






NVIDIA