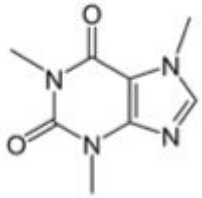
The Caffe Framework: DIY Deep Learning

Maximally accurate	Maximally specific
espresso	2.23192
coffee	2.19914
beverage	1.93214
liquid	1.89367
fluid	1.85519



caffe.berkeleyvision.org

github.com/BVLC/caffe

Evan Shelhamer, Jeff Donahue, Jon Long

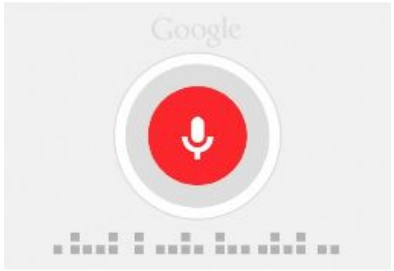
from the tutorial by

Evan Shelhamer, Jeff Donahue, Jon Long,
Yangqing Jia, and Ross Girshick

Why Deep Learning?

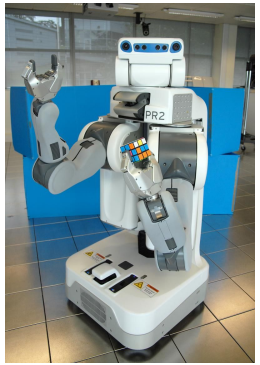
End-to-End Learning for Many Tasks

vision



speech

text



control

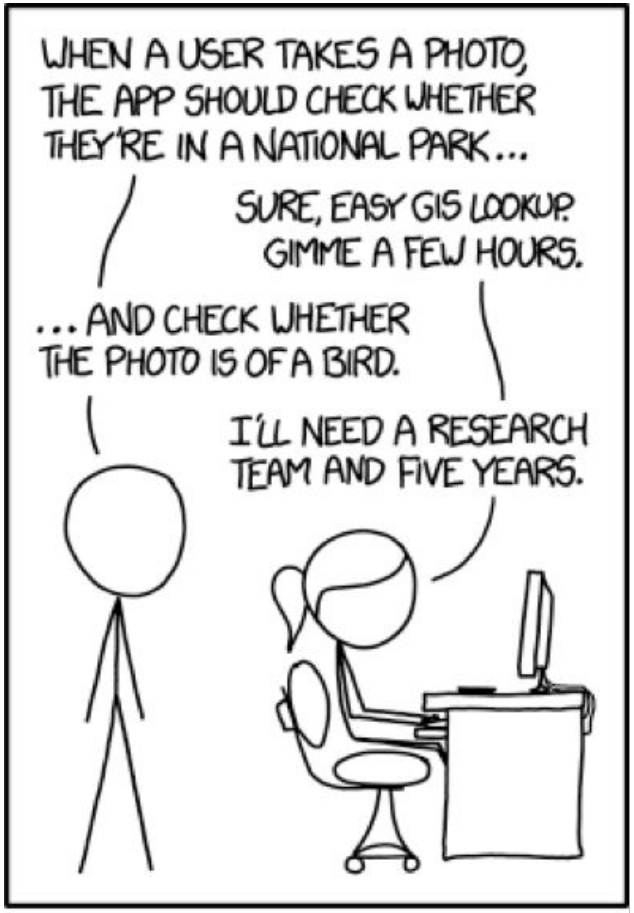
What is Deep Learning?

Deep Learning is

Stacking **Layers**

and

Learning **End-to-End**



xkcd: Tasks

"The Virtually Impossible"

EXAMPLE PHOTOS



Photo credits

PARK or BIRD

Want to know if your photo is from a U.S. national park? Want to know if it contains a bird? Just drag it into the box to the left, and we'll tell you. We'll use the GPS embedded in your photo (if it's there) to see whether it's from a park, and we'll use our super-cool computer vision skills to try to see whether it's a bird (which is a hard problem, but we do a pretty good job at it).

To try it out, just drag any photo from your desktop into the upload box, or try dragging any of our example images. We'll give you your answers below!

Want to know more about PARK or BIRD, including why the heck we did this? Just click here for more info →

PARK?

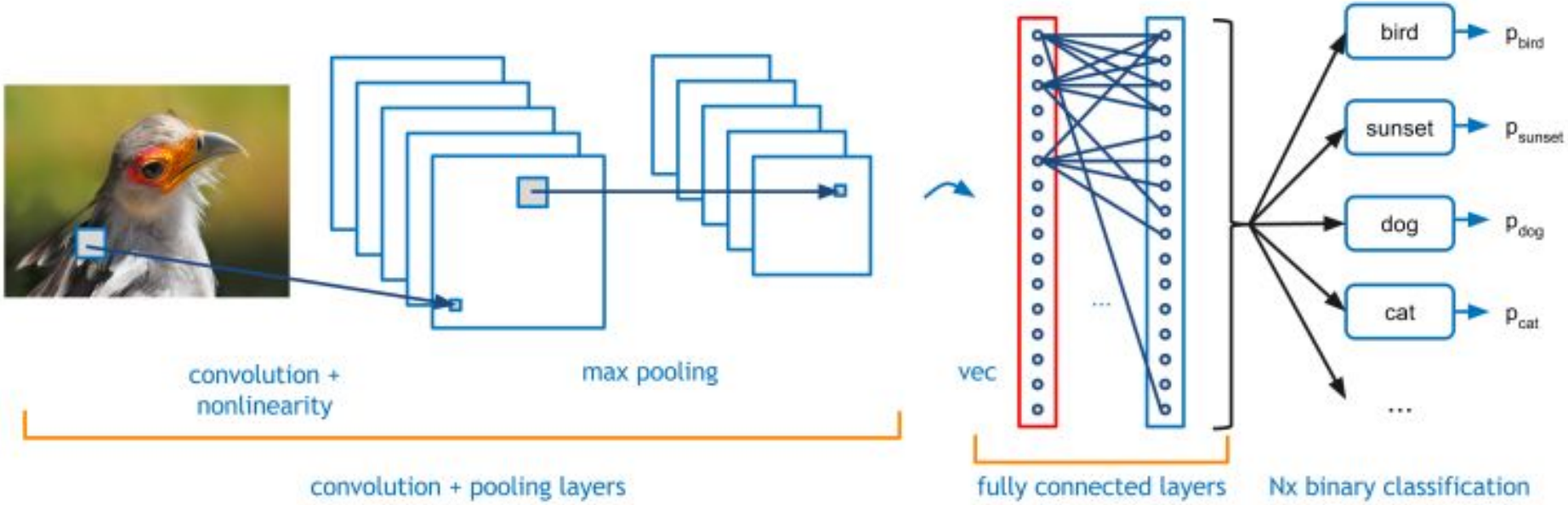
YES

Ah yes, Everglades is truly beautiful.

BIRD?

YES

Dude, that is such a bird.



All in a day's work with Caffe

http://code.flickr.net/2014/10/20/introducing-flickr-park-or-bird/

Visual Recognition Tasks: Classification

Classification

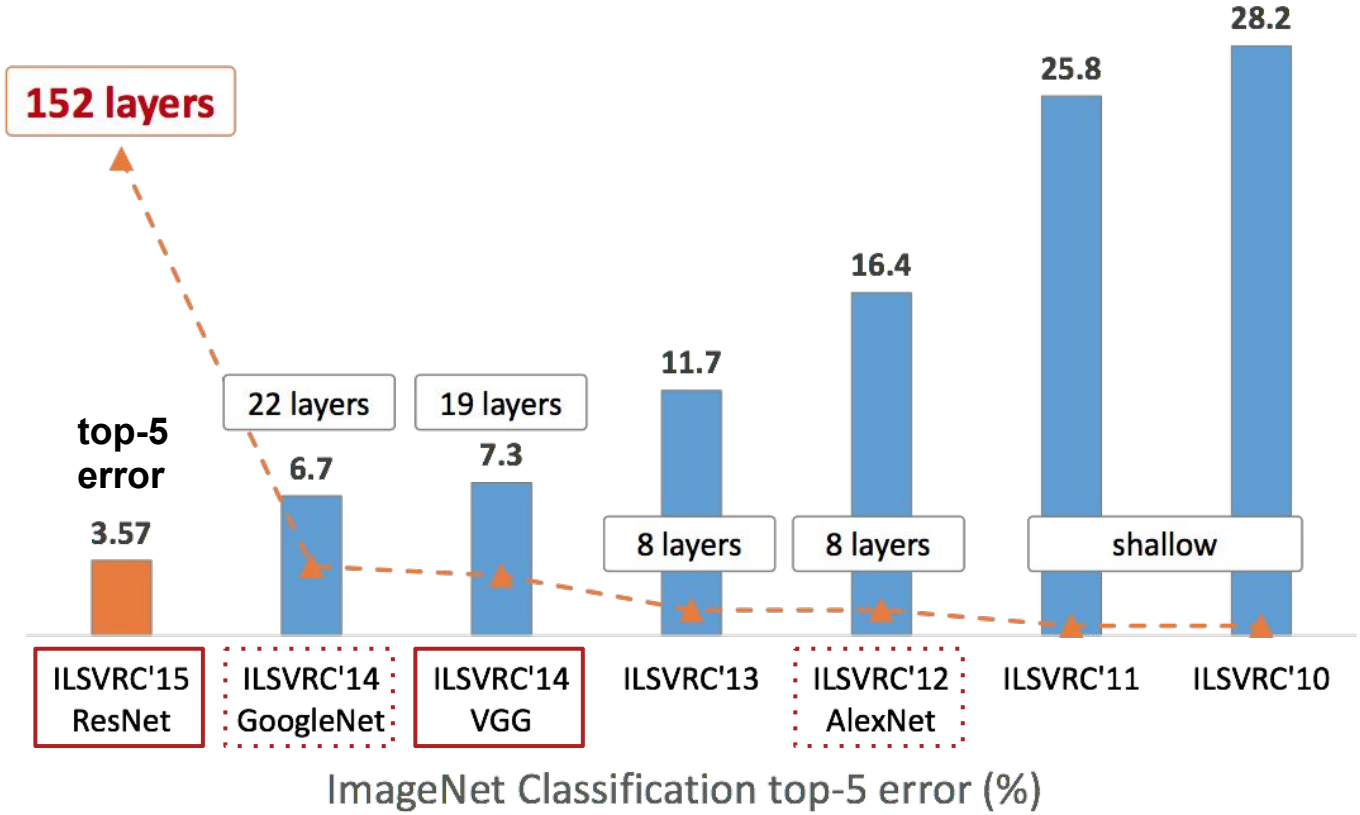
- what kind of image?
- which kind(s) of objects?

Challenges

- appearance varies by lighting, pose, context, ...
- clutter
- fine-grained categorization (horse or exact species)

- ❑ dog
- ☑ car
- ☑ horse
- ❑ bike
- ❑ cat
- ❑ bottle
- ☑ person

Image Classification: ILSVRC 2010-2015



- ❑ dog
- ☑ car
- ☑ horse
- ❑ bike
- ❑ cat
- ❑ bottle
- ☑ person

[graph credit K. He]

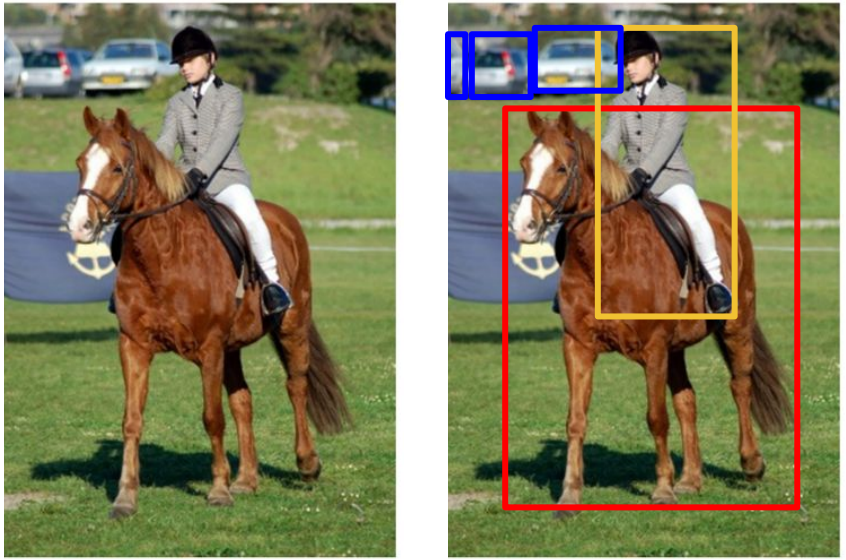
Visual Recognition Tasks: Detection

Detection

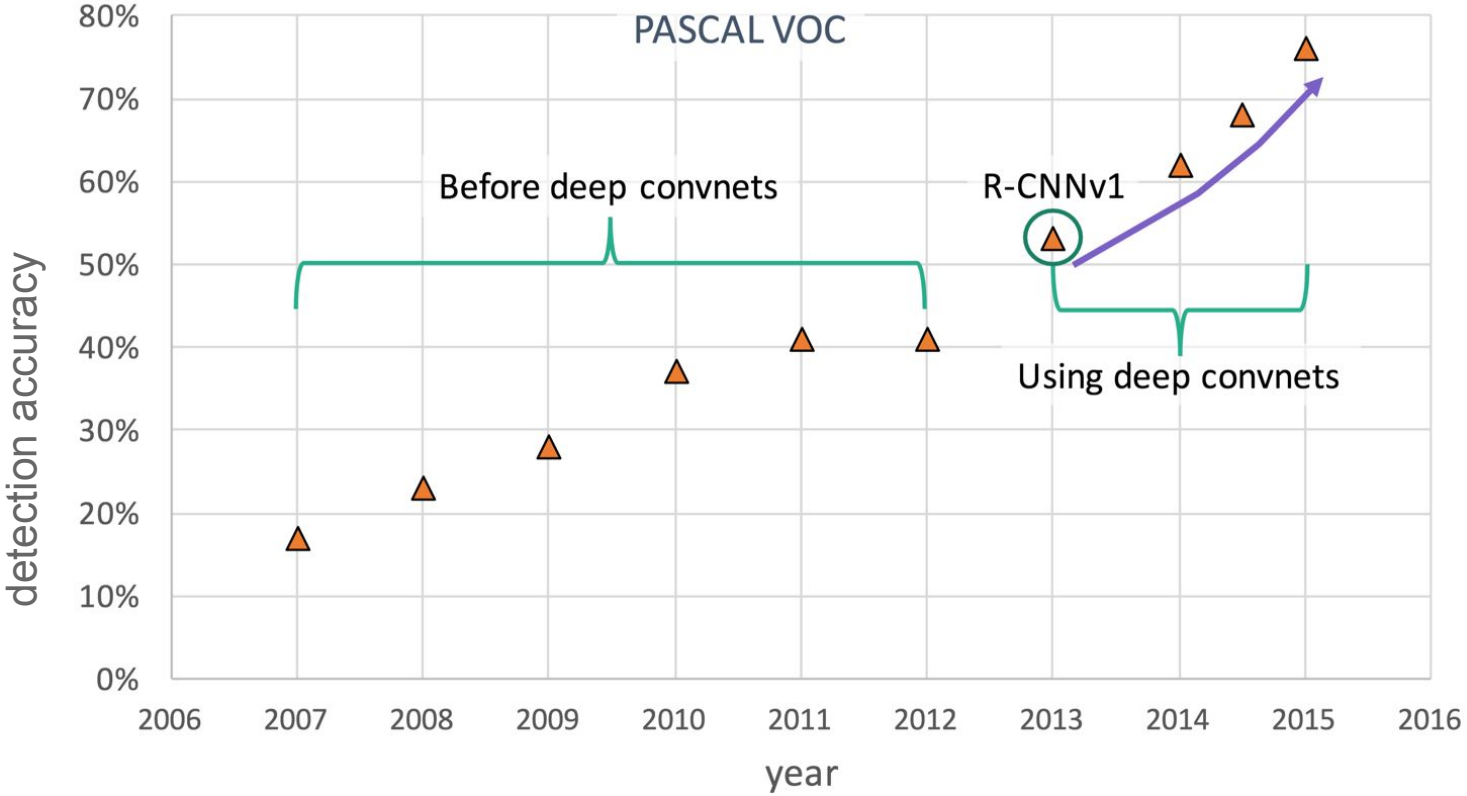
- what objects are there?
- where are the objects?

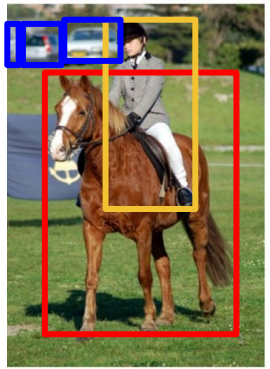
Challenges

- localization
- multiple instances
- small objects



Detection: PASCAL VOC





R-CNN:
regions + convnets

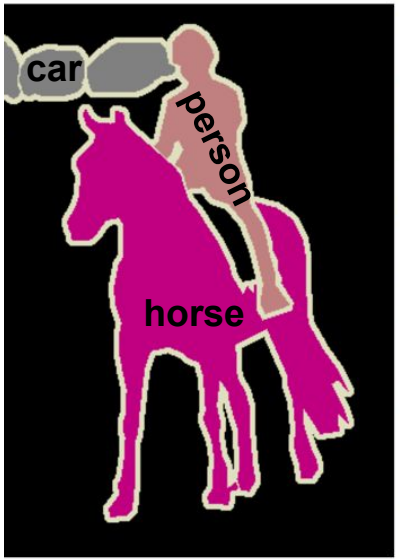
state-of-the-art, in Caffe

[graph credit R. Girshick]

Visual Recognition Tasks: Segmentation

Semantic Segmentation

- what kind of thing
  is each pixel part of?
- what kind of stuff
  is each pixel?



Challenges

- tension between
  recognition and localization
- amount of computation

Segmentation: PASCAL VOC

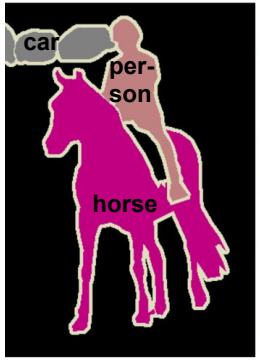
Leaderboard

MSRA_BoxSup [?]	75.2
Oxford_TVG_CRF_RNN_COCO [?]	74.7
DeepLab-MSc-CRF-LargeFOV-COCO-CrossJoint [?]	73.9
Adelaide_Context_CNN_CRF_VOC [?]	72.9
DeepLab-CRF-COCO-LargeFOV [?]	72.7
POSTECH_EDeconvNet_CRF_VOC [?]	72.5
Oxford_TVG_CRF_RNN_VOC [?]	72.0
DeepLab-MSc-CRF-LargeFOV [?]	71.6
MSRA_BoxSup [?]	71.0
DeepLab-CRF-COCO-Strong [?]	70.4
DeepLab-CRF-LargeFOV [?]	70.3
TTI_zoomout_v2 [?]	69.6
DeepLab-CRF-MSc [?]	67.1
DeepLab-CRF [?]	66.4
CRF_RNN [?]	65.2
TTI_zoomout_16 [?]	64.4
Hypercolumn [?]	62.6
FCN-8s [?]	62.2
MSRA_CFM [?]	61.8
TTI_zoomout [?]	58.4
SDS [?]	51.6
NUS_UDS [?]	50.0
TTIC-divmbest-rerank [?]	48.1
BONN_O2PCPMC_FGT_SEGM [?]	47.8
BONN_O2PCPMC_FGT_SEGM [?]	47.5
BONNGC_O2P_CPMC_CSI [?]	46.8
BONN_CMBR_O2P_CPMC_LIN [?]	46.7

deep learning with Caffe

end-to-end networks lead to
25 points absolute or 50% relative improvement
and >100x speedup in 1 year!

(papers published for +1 or +2 points)



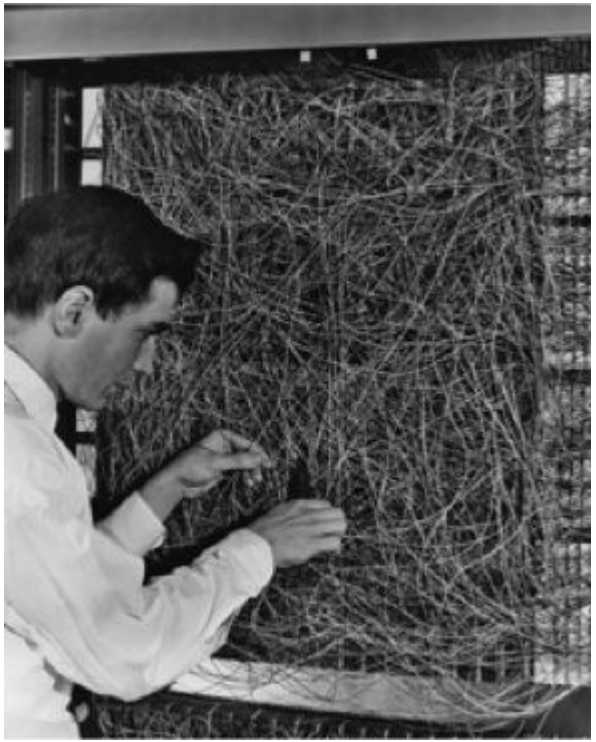
FCN:
pixelwise
convnet

state-of-the-art,
in Caffe

Deep History

1958 Rosenblatt proposed perceptrons
1980 Neocognitron (Fukushima, 1980)
1982 Hopfield network, SOM (Kohonen, 1982), Neural PCA (Oja, 1982)
1985 Boltzmann machines (Ackley et al., 1985)
1986 Multilayer perceptrons and backpropagation (Rumelhart et al., 1986)
1988 RBF networks (Broomhead&Lowe, 1988)
1989 Autoencoders (Baldi&Hornik, 1989), Convolutional network (LeCun, 1989)
1992 Sigmoid belief network (Neal, 1992)
1993 Sparse coding (Field, 1993)
2000s Sparse, Probabilistic, and Layer-wise models (Hinton, Bengio, Ng)
2012 DL popularized in vision by contest victory (Krizhevsky et al. 2012)

Is deep learning 4, 20, or 50 years old? What's changed?



Rosenblatt's Perceptron

Why Now?

1. Data

   ImageNet et al.: millions of *labeled* (crowdsourced) images
2. Compute

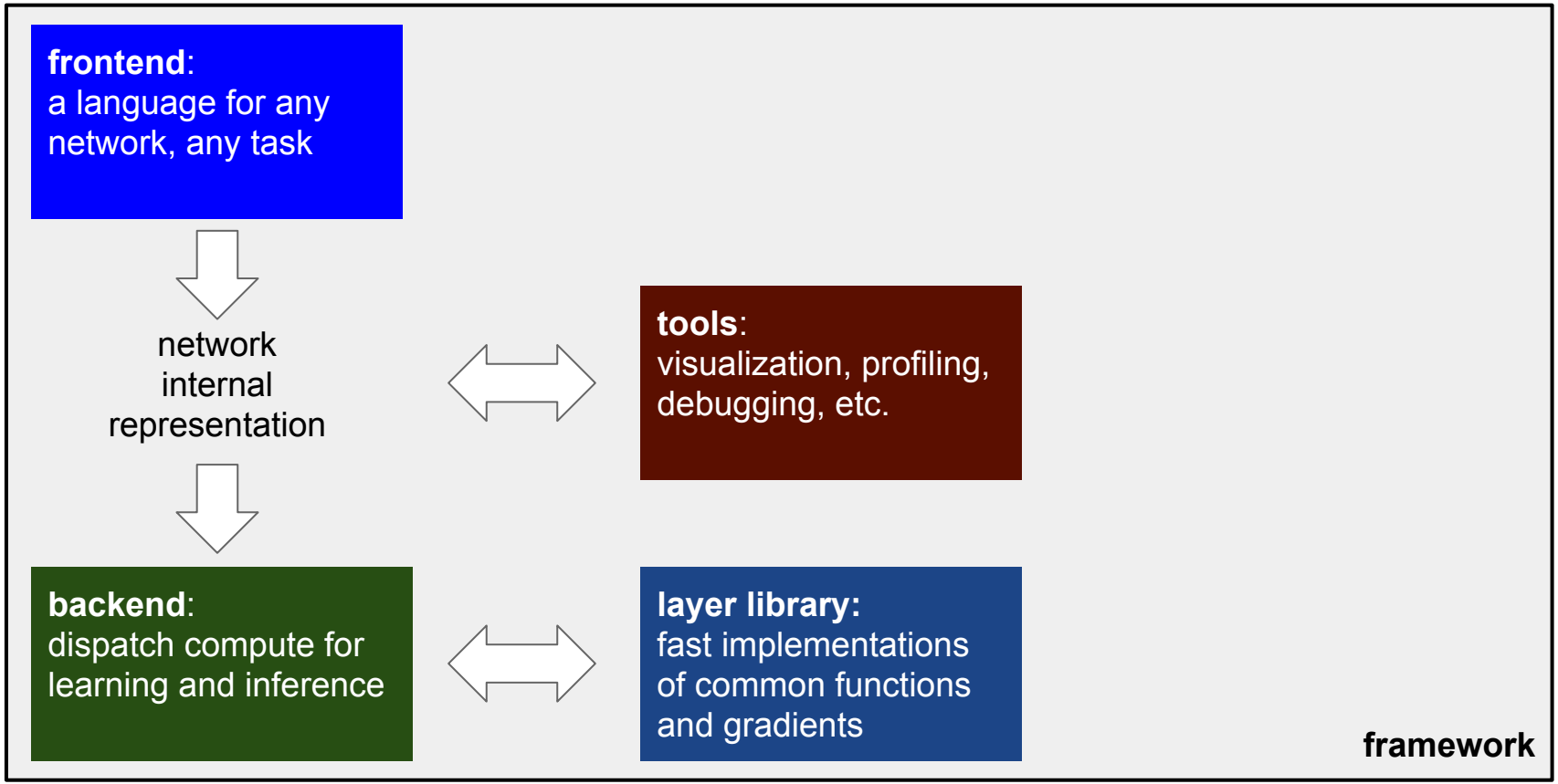
   GPUs: terabytes/s memory bandwidth, teraflops compute
3. Technique

   new optimization know-how,

   new variants on old architectures,

   new tools for rapid experiments and deployments

Why Now? Deep Learning Frameworks



we like to brew our networks with **Caffe**

What is Caffe?

Open framework, models, and worked examples
for deep learning

- 2 years old
- 1,000+ citations, 150+ contributors, 9,000+ stars
- 5,000+ forks, >1 pull request / day average
- focus has been vision, but branching out: sequences, reinforcement learning, speech + text

Prototype

Train

Deploy

What is Caffe?

Open framework, models, and worked examples
for deep learning

- Pure C++ / CUDA library for deep learning
- Command line, Python, MATLAB interfaces
- Fast, well-tested code
- Tools, reference models, demos, and recipes
- Seamless switch between CPU and GPU

Prototype

Train

Deploy

Caffe is a Community

[project pulse]

BVLC / caffe

Unwatch 1,205 | Unstar 8,498 | Fork 4,821

January 19, 2016 – February 19, 2016

Period: **1 month**

Overview

45 Active Pull Requests

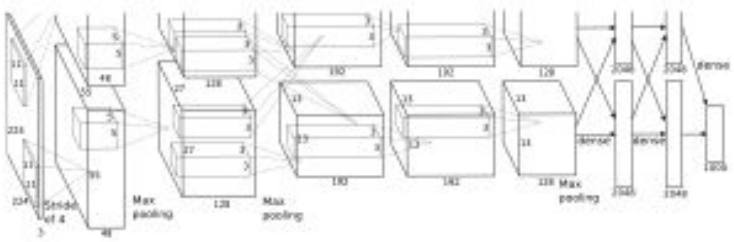
90 Active Issues

22 Merged Pull Requests	**23** Proposed Pull Requests	**52** Closed Issues	**38** New Issues

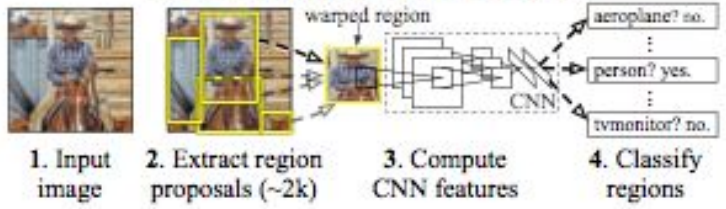
Excluding merges, **20 authors** have pushed **19 commits** to master and **53 commits** to all branches. On master, **44 files** have changed and there have been **2,268 additions** and **162 deletions**.

Reference Models

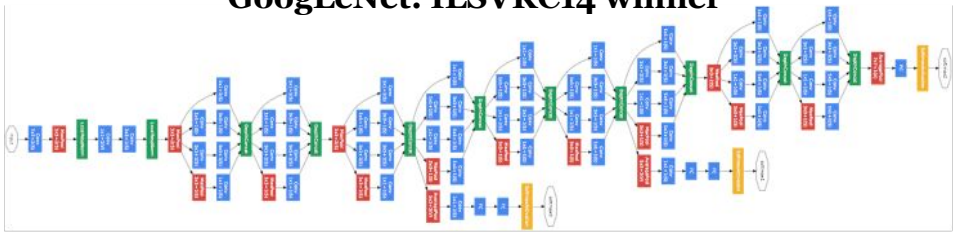
AlexNet: ImageNet Classification



R-CNN: *Regions with CNN features*



GoogLeNet: ILSVRC14 winner



Caffe offers the
- model definitions
- optimization settings
- pre-trained weights

so you can start right away

The BVLC models are licensed for unrestricted use

The community shares models in our Model Zoo

Open Model Collection

The Caffe Model Zoo open collection of deep models to share innovation

- MSRA ResNet ILSVRC15 winner in the zoo
- VGG ILSVRC14 + Devil models in the zoo
- MIT Places scene recognition model in the zoo
- Network-in-Network / CCCP model in the zoo

helps disseminate and reproduce research

bundled tools for loading and publishing models

Share Your Models! with your citation + license of course

Brewing by the Numbers...

Speed with Krizhevsky's 2012 model:

- 2 ms/image on K40 GPU
- <1 ms inference with Caffe + cuDNN v4 on Titan X
- 72 million images/day with batched IO
- 8-core CPU: ~20 ms/image Intel optimization in progress

9k lines of C++ code (20k with tests)

Sharing a Sip of Brewed Models

demo.caffe.berkeleyvision.org

demo code open-source and bundled

Maximally accurate | Maximally specific

cat	1.80727
domestic cat	1.74727
feline	1.72787
tabby	0.99133
domestic animal	0.78542

Scene Recognition http://places.csail.mit.edu/

B. Zhou et al. NIPS 14

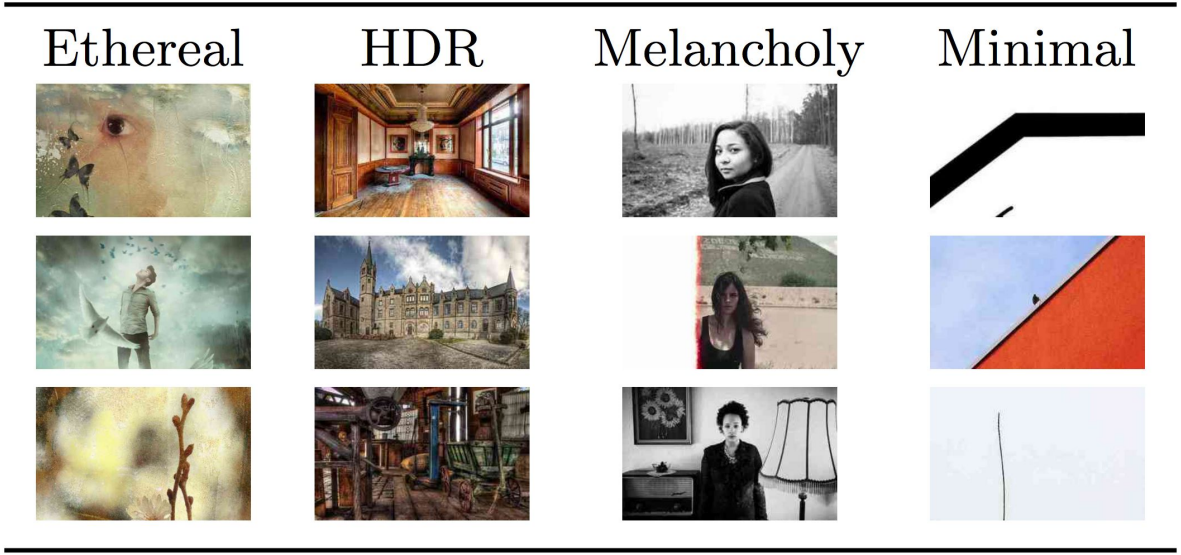
Predictions:

- **Type of environment:** outdoor
- **Semantic categories:** skyscraper:0.69, tower:0.16, office_building:0.11,
- **SUN scene attributes:** man-made, vertical components, natural light, open area, nohorizon, glossy, metal, wire, clouds, far-away horizon

Visual Style Recognition

Karayev et al. *Recognizing Image Style*. BMVC14. Caffe fine-tuning example.

Demo online at http://demo.vislab.berkeleyvision.org/ (see Results Explorer).



Other Styles:

Vintage
Long Exposure
Noir
Pastel
Macro
… and so on.

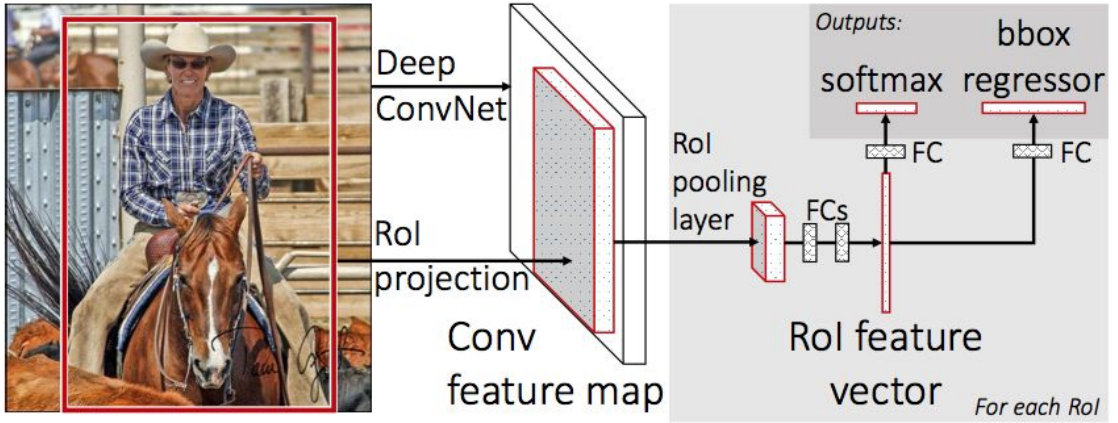
[Karayev14]

Object Detection

R-CNNs: Region-based Convolutional Networks

[Fast R-CNN]

- convolve once
- project + detect



[*Faster* R-CNN]

- end-to-end proposals and detection
- image inference in 200 ms
- Region Proposal Net + Fast R-CNN

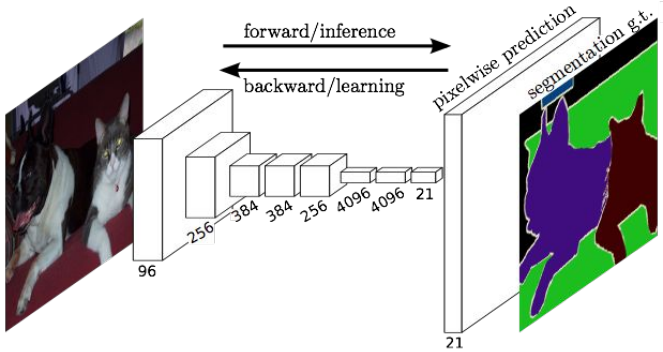
papers + code online

Ross Girshick, Shaoqing Ren, Kaiming He, Jian Sun

Pixelwise Prediction

Fully convolutional networks for pixel prediction in particular semantic segmentation

- end-to-end learning
- efficient inference and learning
  100 ms per-image prediction
- multi-modal, multi-task



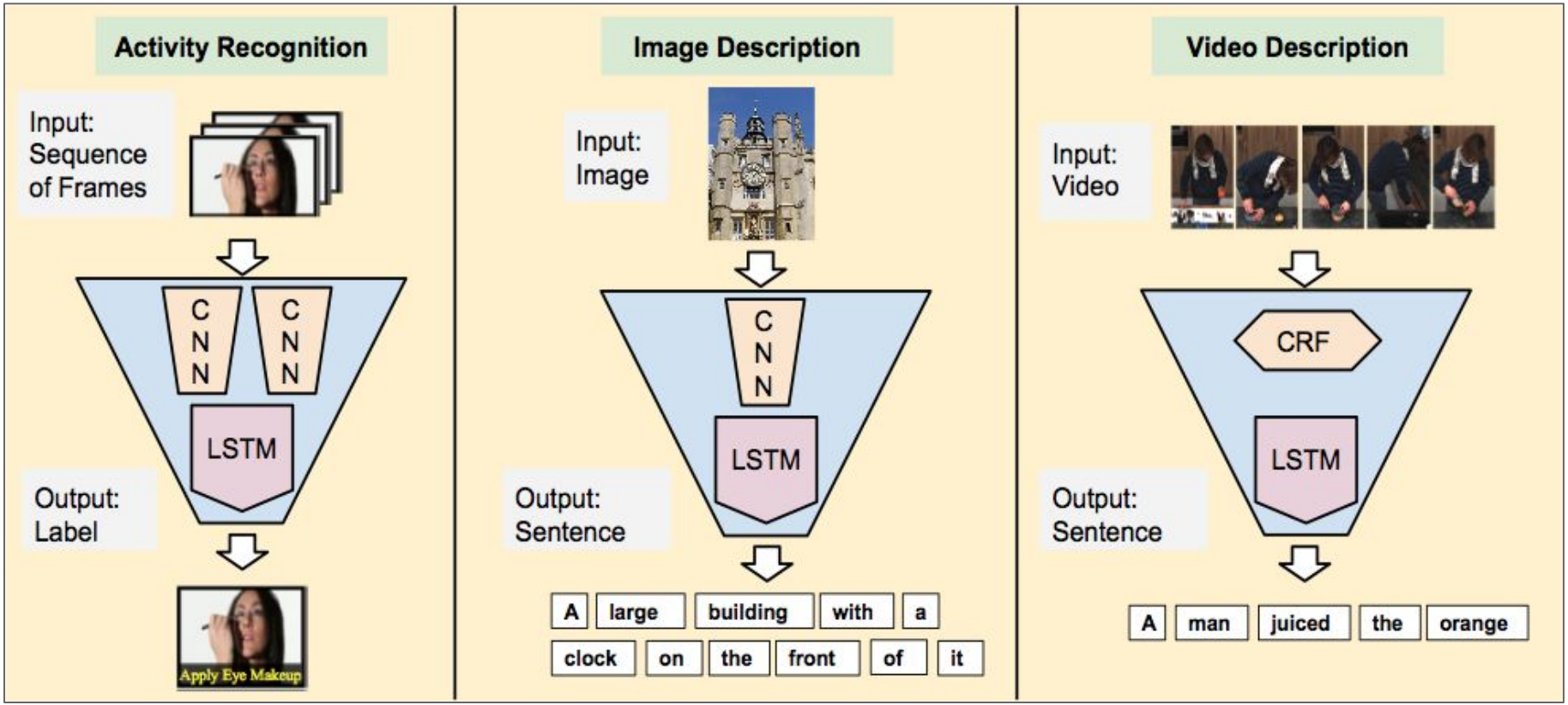
Applications

- semantic segmentation
- denoising
- depth estimation
- optical flow

Jon Long & Evan Shelhamer*, Trevor Darrell. CVPR'15*

CVPR'15 paper and code + models

Visual Sequence Tasks

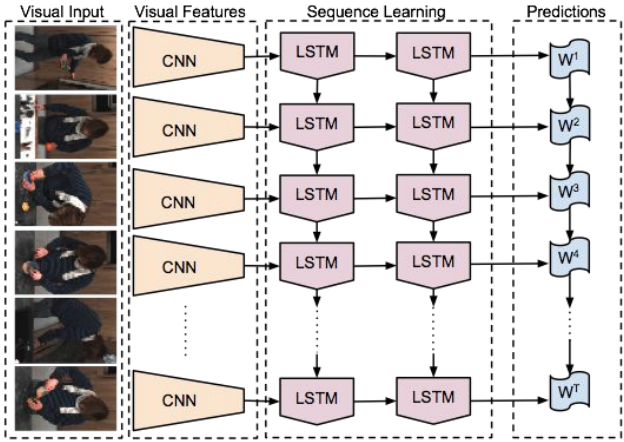


Recurrent Networks for Sequences

Recurrent Nets and Long Short Term Memories (LSTM) are sequential models

- video
- language
- dynamics

learned by backpropagation through time



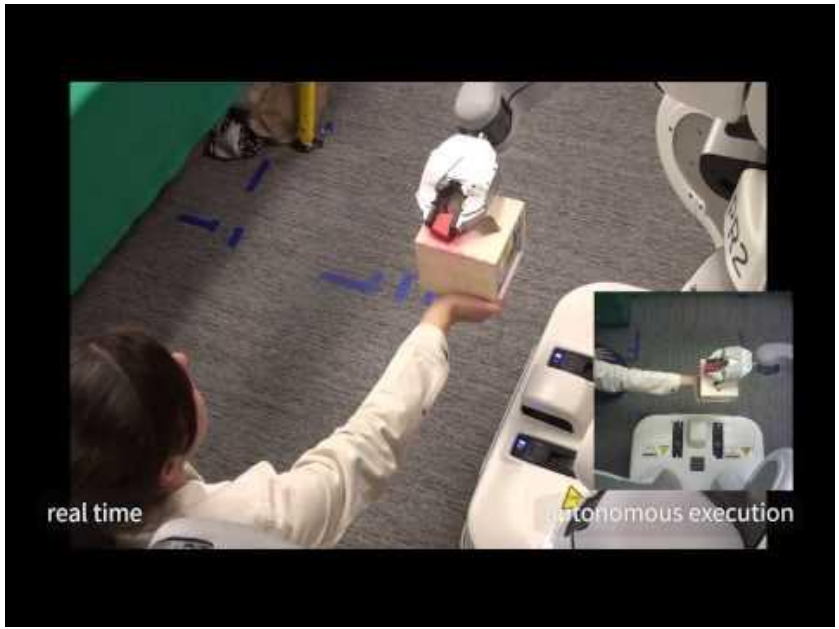
LRCN: Long-term Recurrent Convolutional Network

- activity recognition (sequence-in)
- image captioning (sequence-out)
- video captioning (sequence-to-sequence)

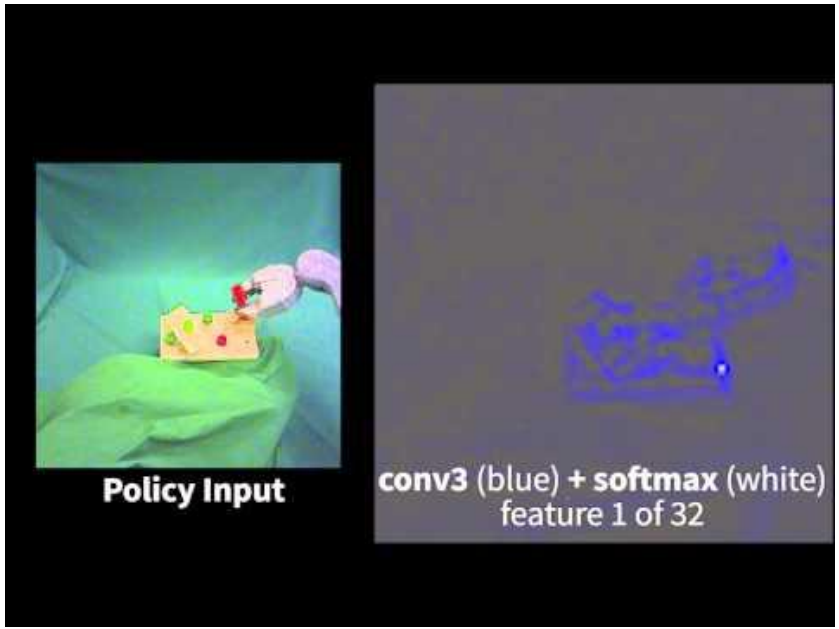
LRCN:
recurrent + convolutional
for visual sequences

CVPR'15 paper and code + models

Deep Visuomotor Control



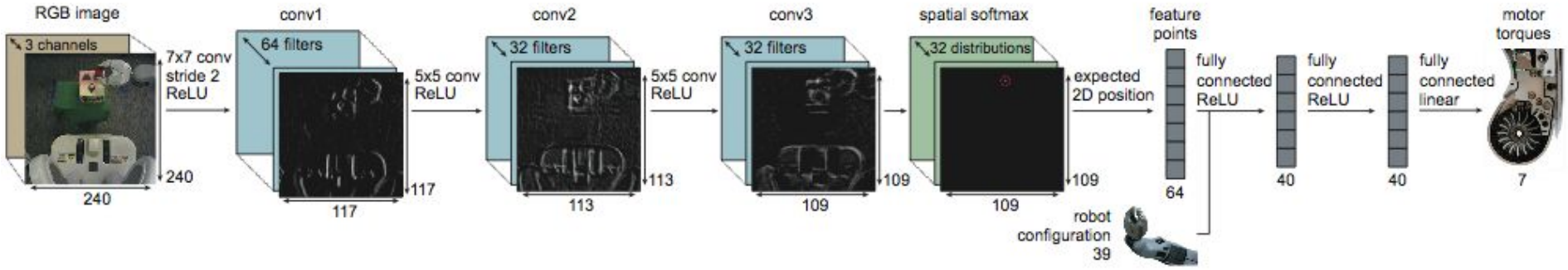
example experiments



feature visualization

Sergey Levine & Chelsea Finn*,*
Trevor Darrell, and Pieter Abbeel

Deep Visuomotor Control Architecture



- multimodal (images & robot configuration)
- runs at 20 Hz - mixed GPU & CPU
  for real-time control

paper + code for guided policy search

Sergey Levine & Chelsea Finn*,*
Trevor Darrell, and Pieter Abbeel

Embedded Caffe

Caffe runs on embedded CUDA hardware and mobile devices

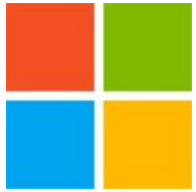
- same model weights,
  same framework interface
- out-of-the-box on
  CUDA platforms
- in-progress OpenCL port
  thanks Fabian Tschopp!
  + AMD, Intel, and the community
- community Android port
  thanks sh1r0!

CUDA Jetson TX1, TK1

OpenCL branch

Android lib, demo

Caffeinated Companies



… startups, big companies, more ...

Caffe at Facebook

- in production for vision at scale: uploaded photos run through Caffe
- Automatic Alt Text for the blind
- On This Day for surfacing memories
- objectionable content detection
- contributing back to the community: inference tuning, tools, code review

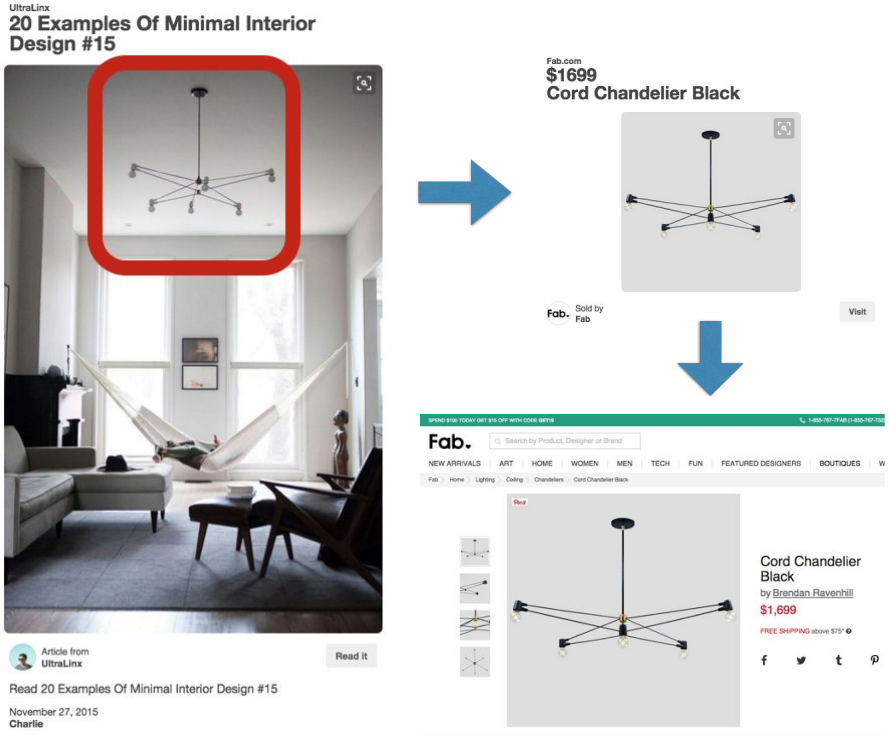
On This Day highlight content

Automatic Alt Text recognize photo content for accessibility

[example credit Facebook]

Caffe at Pinterest

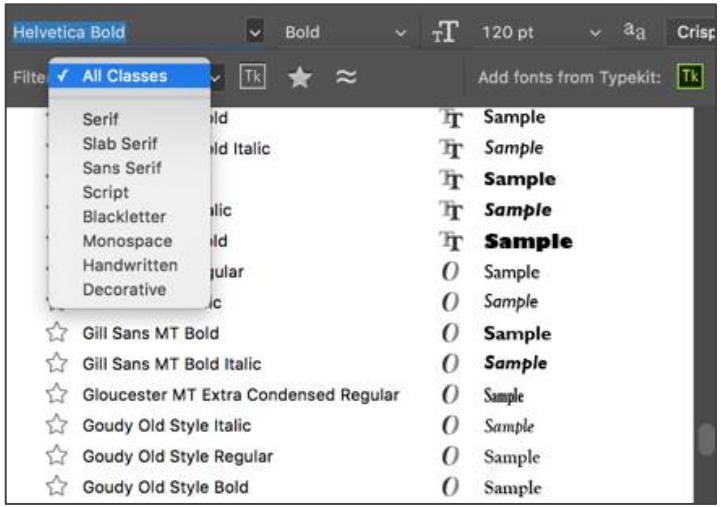
- in production for vision at scale: uploaded photos run through Caffe
- deep learning for visual search: retrieval over billions of images in <250 ms
- ~4 million requests/day
- built on an open platform of Caffe, FLANN, Thrift, ...



[example credit Andrew Zhai, Pinterest]

Caffe at Adobe

- training networks for research in vision and graphics
- custom inference in products, including Photoshop



Photoshop Type Similarity
catalogue typefaces automatically

Caffe at Yahoo! Japan

- personalize news and content, and de-duplicate suggestions
- curate news and restaurant photos for recommendation
- arrange user photo albums

News Image Recommendation
select and crop images for news

Deep Learning, as it is executed...

What does the Caffe framework handle?

Compositional Models

Decompose the problem and code!

End-to-End Learning

Configure and solve!

Many Architectures and Tasks

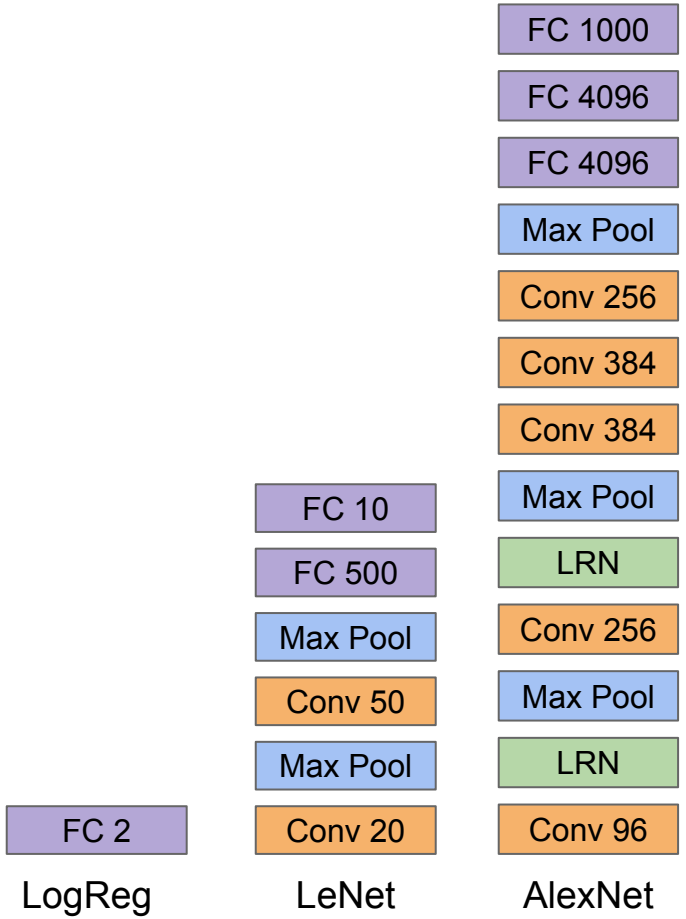
Define, experiment, and extend!

Net

A network is a set of layers and their connections:

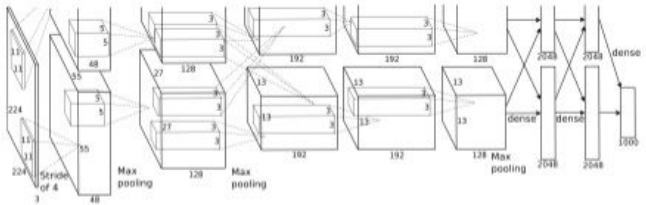
```
name: "dummy-net"
layer { name: "data" …}
layer { name: "conv" …}
layer { name: "pool" …}
    … more layers …
layer { name: "loss" …}
```

- Caffe creates and checks the net from the definition
- Data and derivatives flow through the net



Forward / Backward The Essential Net Computations

Forward: inference $f_W(x)$



"espresso" + loss

$\nabla f_W(x)$ Backward: learning

Caffe models are complete machine learning systems for inference and learning
The computation follows from the model definition: define the model and run

Layer Protocol

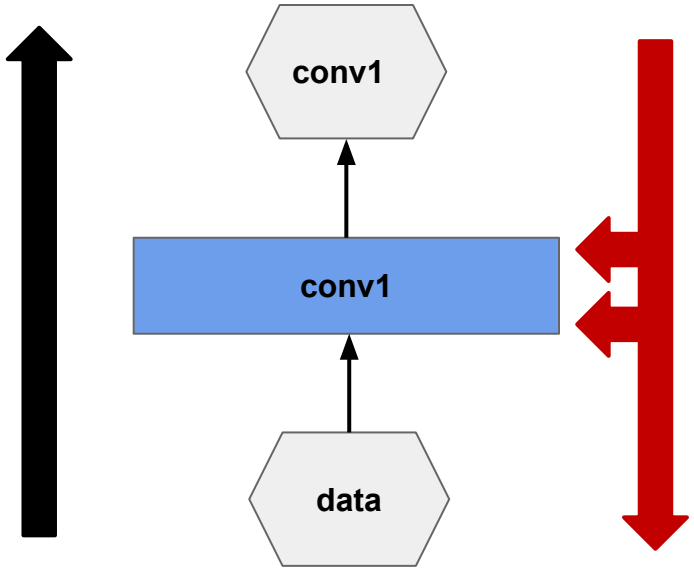
Forward: make output given input.

Backward: make gradient of output
- w.r.t. bottom
- w.r.t. parameters (if needed)

Setup: run once for initialization.

Reshape: set dimensions.

Compositional Modeling
The Net's forward and backward passes are composed of the layers' steps



Layer Development Checklist

```python
import caffe
import numpy as np

class EuclideanLoss(caffe.Layer):

    def setup(self, bottom, top):
        # check input pair
        if len(bottom) != 2:
            raise Exception("Need two inputs to compute distance.")

    def reshape(self, bottom, top):
        # check input dimensions match
        if bottom[0].count != bottom[1].count:
            raise Exception("Inputs must have the same dimension.")
        # difference is shape of inputs
        self.diff = np.zeros_like(bottom[0].data, dtype=np.float32)
        # loss output is scalar
        top[0].reshape(1)

    def forward(self, bottom, top):
        self.diff[...] = bottom[0].data - bottom[1].data
        top[0].data[...] = np.sum(self.diff**2) / bottom[0].num / 2.

    def backward(self, top, propagate_down, bottom):
        for i in range(2):
            if not propagate_down[i]:
                continue
            if i == 0:
                sign = 1
            else:
                sign = -1
            bottom[i].diff[...] = sign * self.diff / bottom[i].num
```

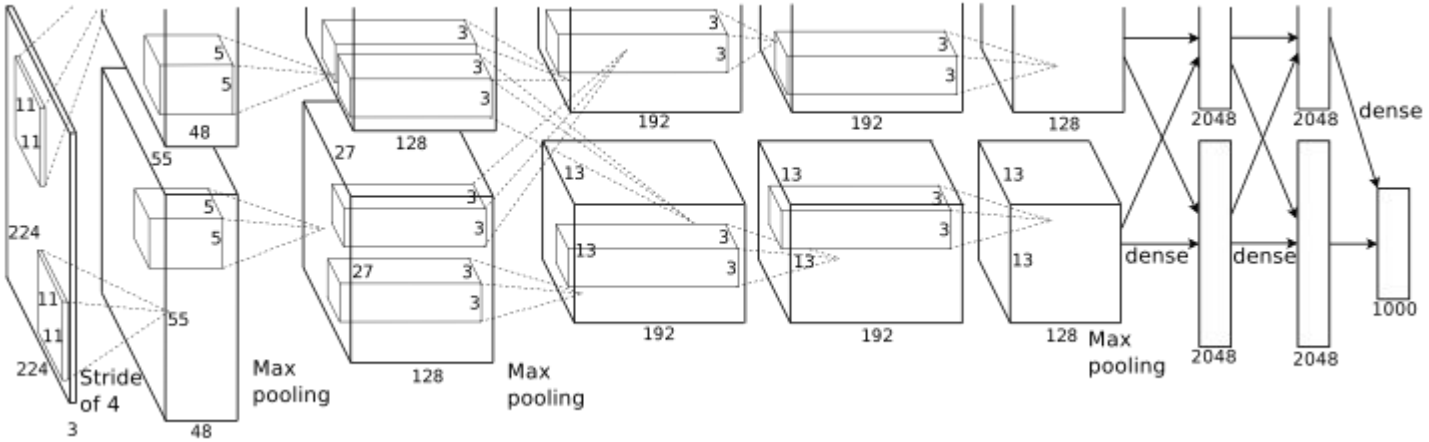
Layer Protocol == Class Interface

Define a class in C++ or Python to extend `Layer`

Include your new layer type in a network and keep brewing

```
layer {
type: "Python"
python_param {
  module: "layers"
  layer: "EuclideanLoss"
} }
```

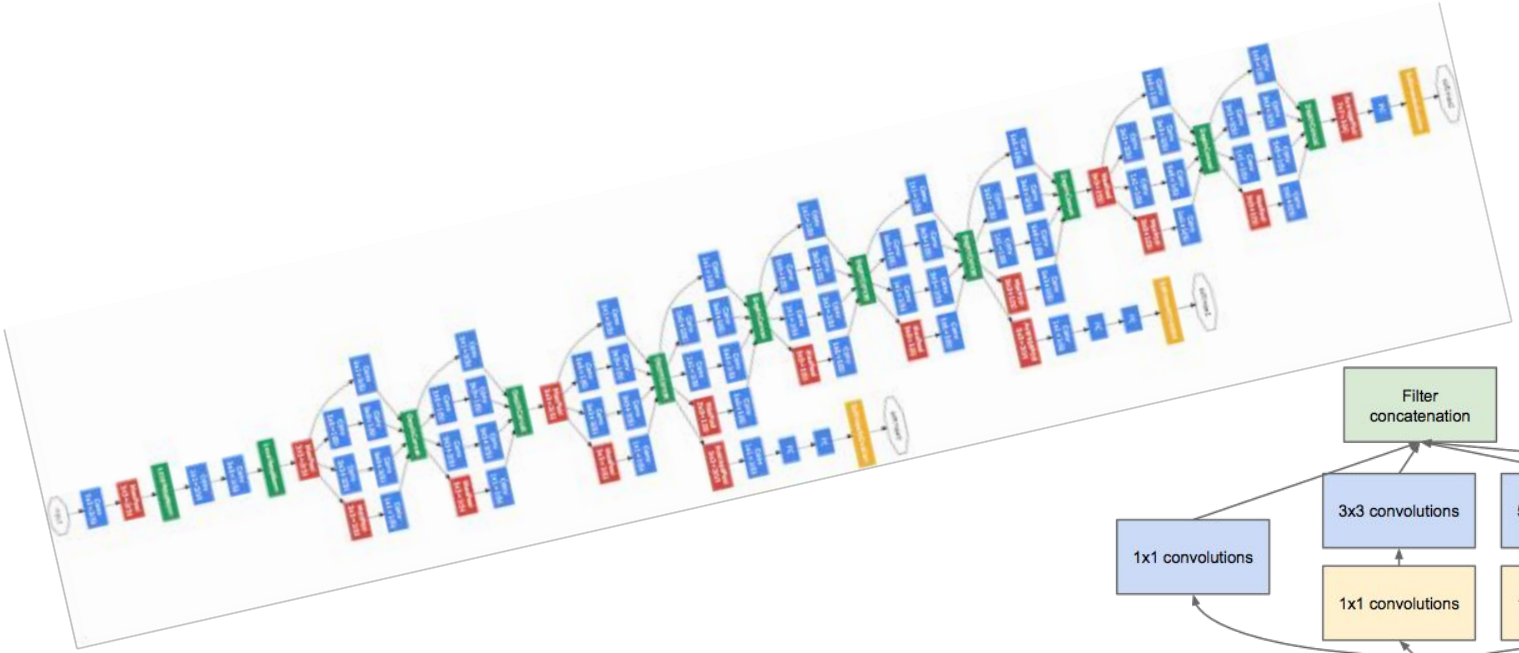
Convolutional Networks: 2012

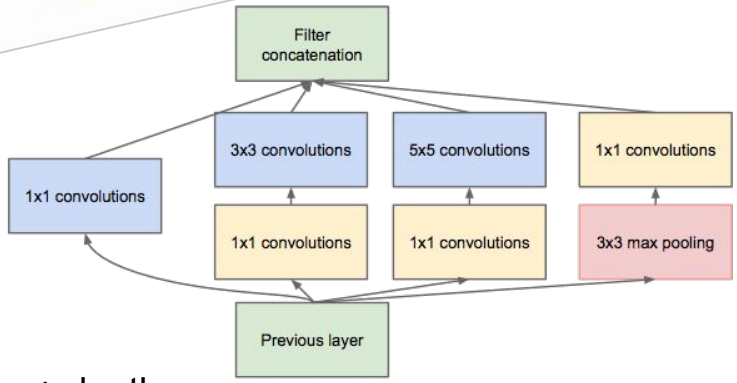


AlexNet: a layered model composed of convolution, pooling, and further operations followed by a holistic representation. A landmark classifier on ILSVRC12 [AlexNet]

+ data
+ gpu
+ non-saturating non-linearity
+ regularization

Convolutional Nets: 2014





GoogLeNet ILSVRC14 Winner: ~6.6% Top-5 error

- composition of multi-scale dimension-reduced "Inception" modules
- no FC layers and only 5 million parameters

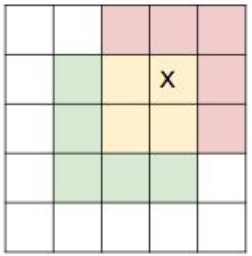
+ depth
+ dimensionality reduction
+ auxiliary classifiers

[Szegedy15]

Convolutional Nets: 2014

A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

stack 2
3x3 conv



for a 5x5
receptive field

*[figure credit
A. Karpathy]*

VGG16 ILSVRC14 Runner-up: ~7.3% Top-5 error

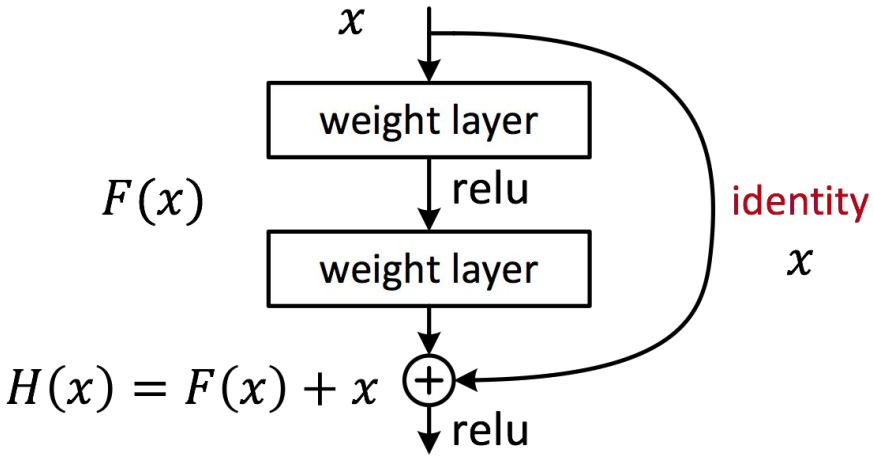
- simple architecture, *good for transfer learning*
- 155 million params and more expensive to compute

+ depth
+ stacking small filters
+ fine-tuning deeper and deeper

[Simonyan15]

Convolutional Nets: 2015

Learn **residual** mapping w.r.t. **identity**



ILSVRC15 and COCO15 Winner: **MSRA ResNet**
- classification
- detection
- segmentation

- very deep 100+ layer nets
- skip connections across layers
- normalization to help propagation

Kaiming He, et al.
Deep Residual Learning for Image Recognition
arXiv 1512.03385. Dec. 2015.

[He15]

Model Format

- Protobuf serialization
- Auto-generates code
- Developed by Google
- Defines Net / Layer / Solver schemas in **caffe.proto**

```
message ConvolutionParameter {
 // The number of outputs for the layer
 optional uint32 num_output = 1;
 // whether to have bias terms
 optional bool bias_term = 2 [default = true];
}
```

```
layer {
name: "conv1"
type: "Convolution"
bottom: "data"
top: "conv1"
convolution_param {
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
        type: "xavier"
    }
}}
```

Net Specification

```
from caffe import layers as L
from caffe import params as P

data = L.data(batch_size=64, backend=P.data.LMDB, source='examples/mnist/mnist_train_lmdb')
conv1 = L.convolution(data, kernel_size=5, num_output=20)
pool1 = L.pooling(conv1, kernel_size=2, stride=2, pool=P.pooling.MAX)
conv2 = L.convolution(pool1, kernel_size=5, num_output=50)
pool2 = L.pooling(conv2, kernel_size=2, stride=2, pool=P.pooling.MAX)
ip1 = L.relu(L.inner_product(pool2, num_output=500))
prob = L.softmax(L.inner_product(ip1, num_output=10))
```

Model Zoo Format

readme.md Raw

name: FCN-32s Fully Convolutional Semantic Segmentation on PASCAL-Context caffemodel: fcn-32s-pascalcontext.caffemodel caffemodel_url: http://dl.caffe.berkeleyvision.org/fcn-32s-pascalcontext.caffemodel sha1: adbbd504c280e2b8966fc32e32ada2a2ecf13603

gist_id: 80667189b218ad570e82

This is a model from the paper:

```
Fully Convolutional Networks for Semantic Segmentation
Jonathan Long, Evan Shelhamer, Trevor Darrell
arXiv:1411.4038
```

Gists on github hold model definition, license, url for weights, and hash of Caffe commit that guarantees compatibility

Solving: Training a Net

Optimization like model definition is configuration

```
train_net: "lenet_train.prototxt"
type: SGD
base_lr: 0.01
momentum: 0.9
weight_decay: 0.0005
max_iter: 10000
snapshot_prefix: "lenet_snapshot"
```

All you need to run things on the GPU

```
> caffe train -solver lenet_solver.prototxt -gpu 0
```

SGD + momentum **`SGD`** · Nesterov's Accelerated Gradient `Nesterov`

Adaptive Solvers **`Adam`** · `RMSProp` · `AdaDelta` · `AdaGrad`

Recipe for Brewing

- Convert the data to Caffe-format

  python layer, lmdb, leveldb, hdf5 / .mat, list of images, etc.
- Define the Net
- Configure the Solver
- caffe train -solver solver.prototxt -gpu 0

  or interface with Python or MATLAB

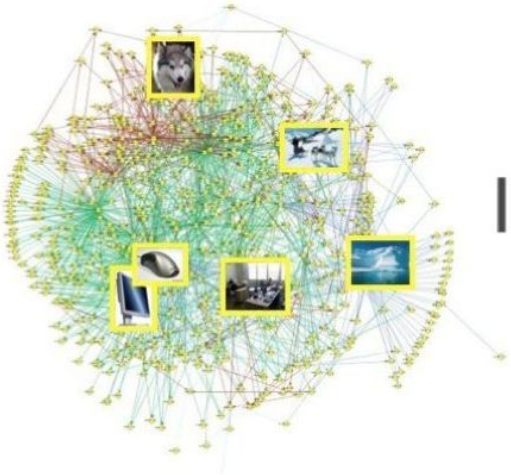
- Examples are your friends

  `caffe/examples/*.ipynb`

  `caffe/models/*`

  `caffe/examples/mnist,cifar10,imagenet`

Transfer Learning and Fine-Tuning



IMAGENET

pre-training data

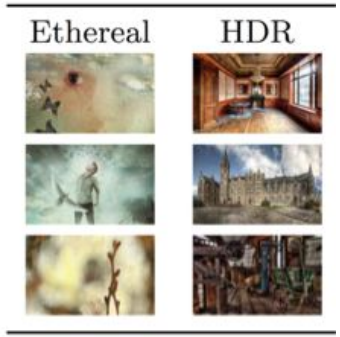
general, tuneable features

weights are a way to cache computation and transfer learning

reference models + the model zoo help exchange weights and ideas

Take a Pre-trained Model and Fine-tune to New Datasets...

Lots of Data



Style Recognition

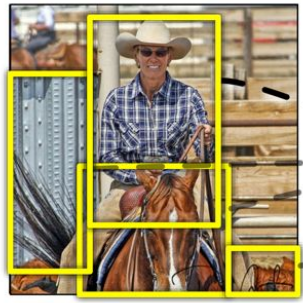
© kaggle.com

Dogs vs. Cats
top 10 in 10 minutes

Your Data

Take a Pre-trained Model and Fine-tune to New Tasks...

Lots of Data



Detection

Segmentation

Your Task

Take a Pre-trained Model and Fine-tune to New Modalities...

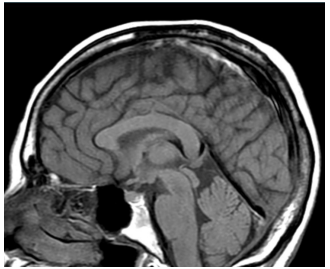
Lots of Data

IMAGENET

Your Modality

Depth/ Range

Remote Sensing



Medical Imaging

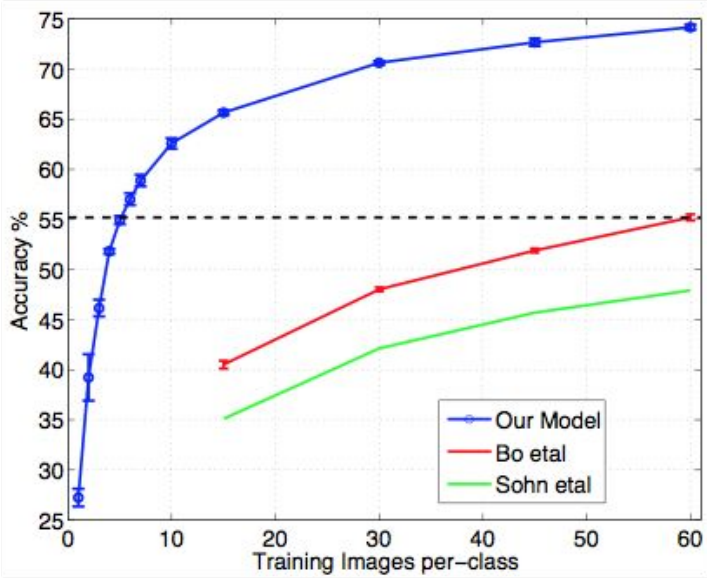
When to Fine-tune?

Almost always

- Robust initialization
- Needs less data
- Faster learning

State-of-the-art results in

- classification
- detection
- segmentation
- more



high accuracy with few examples through **fine-tuning**

[Zeiler-Fergus]

How to Fine-Tune? (1/2)

Simply change a few lines in the model definition

```
layer {
  name: "data"
  type: "Data"
  data_param {
    source: "ilsvrc12_train_lmdb"
    mean_file: "../../data/ilsvrc12"
    ...
  }
  ...
}
...
layer {
  name: "fc8"
  type: "InnerProduct"
  inner_product_param {
    num_output: 1000
    ...
  }
}
```

```
layer {
  name: "data"
  type: "Data"
  data_param {
    source: "style_train_lmdb"
    mean_file: "../../data/ilsvrc12"
    ...
  }
  ...
}
...
layer {
  name: "fc8-style"
  type: "InnerProduct"
  inner_product_param {
    num_output: 20
    ...
  }
}
```

new name =
new params

Input:
A different source

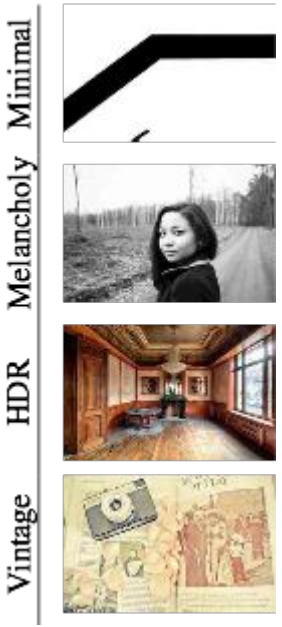
Last Layer:
A different classifier

How to Fine-Tune? (2/2)

```
> caffe train -solver models/finetune_flickr_style/solver.prototxt
              -weights bvlc_reference_caffenet.caffemodel
```

Step-by-step in pycaffe:

```
pretrained_net = caffe.Net(
  "net.prototxt", "net.caffemodel")
solver = caffe.SGDSolver("solver.prototxt")
solver.net.copy_from(pretrained_net)
solver.solve()
```



Framework Future

1.0 is coming stability, documentation, packaging

Performance Tuning for GPU (cuDNN v5) and CPU (nnpack)

In-progress Ports for OpenCL and Windows

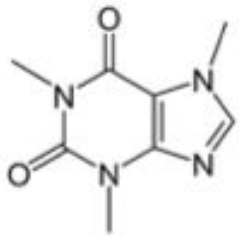
Halide interface for prototyping and experimenting

Widening the Circle continued and closer collaborative development

Next Steps

Today you've seen the progress made with
DIY deep learning and the democratization of models

Next Up:



caffe.berkeleyvision.org

github.com/BVLC/caffe

Check out Caffe on github

Run Caffe through Docker and NVIDIA Docker for GPU

Come to our hands-on lab at GTC!
Join the caffe-users mailing list

Caffe at the Embedded Vision Summit

The event for vision product developers—May 2-4, Santa Clara

Come to the Embedded Vision Summit for a full-day tutorial on convolutional networks and Caffe:

- In-depth, practical training on convnets for vision applications
- Hands-on labs using Caffe to create, train, and evaluate convnets

The Embedded Vision Summit includes:

- 3-day, multi-track program on computer vision product development techniques and markets
- Demos, talks and workshops on the latest processors, tools, APIs, and more

For details and to register: www.EmbeddedVisionSummit.com

Thanks to the whole Caffe Crew

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Jonathan Long,
Sergey Karayev, Ross Girshick, Sergio Guadarrama, Ronghang Hu, Trevor Darrell

and our open source contributors!

Acknowledgements

Thank you to the Berkeley Vision and Learning Center and its Sponsors

Thank you to NVIDIA
for GPUs, cuDNN collaboration,
and hands-on cloud instances

Thank you to A9 and AWS
for a research grant for Caffe dev
and reproducible research

Thank you to our 150+
open source contributors
and vibrant community!

References

[DeCAF] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. ICML, 2014.

[R-CNN] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

[Zeiler-Fergus] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. ECCV, 2014.

[LeNet] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. IEEE, 1998.

[AlexNet] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. NIPS, 2012.

[OverFeat] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. ICLR, 2014.

[Image-Style] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, H. Winnemoeller. Recognizing Image Style. BMVC, 2014.

[Karpathy14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. CVPR, 2014.

[Sutskever13] I. Sutskever. Training Recurrent Neural Networks.
PhD thesis, University of Toronto, 2013.

[Chopra05] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. CVPR, 2005.

END