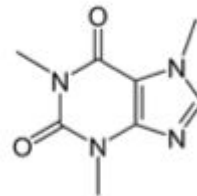


# DIY Deep Learning for Vision: a Hands-On Tutorial with Caffe



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



[caffe.berkeleyvision.org](http://caffe.berkeleyvision.org)



[github.com/BVLC/caffe](https://github.com/BVLC/caffe)

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



Look for further  
details in the  
outline notes



# Tutorial Schedule

Caffe tour and latest roast

## Caffe Tour

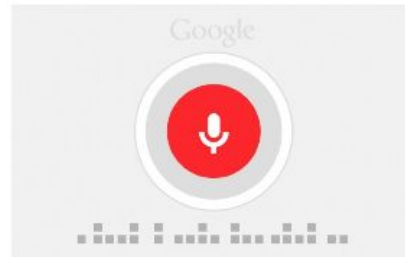
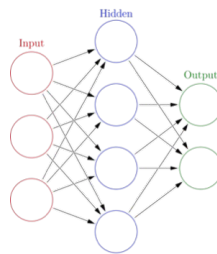
- the why and how of Caffe
- highlight reel of examples + applications
- do-it-yourself notebooks

## Latest Roast

- [detection](#) *Ross Girshick*
- [sequences and vision](#) + language *Jeff Donahue*
- [pixelwise prediction](#) *Jon Long and Evan Shelhamer*
- [framework future](#) *Yangqing Jia*

# Why Deep Learning?

## End-to-End Learning for Many Tasks



# What is Deep Learning?

## Compositional Models Learned End-to-End

### Hierarchy of Representations

- vision: pixel, motif, part, object
- text: character, word, clause, sentence
- speech: audio, band, phone, word

concrete



abstract

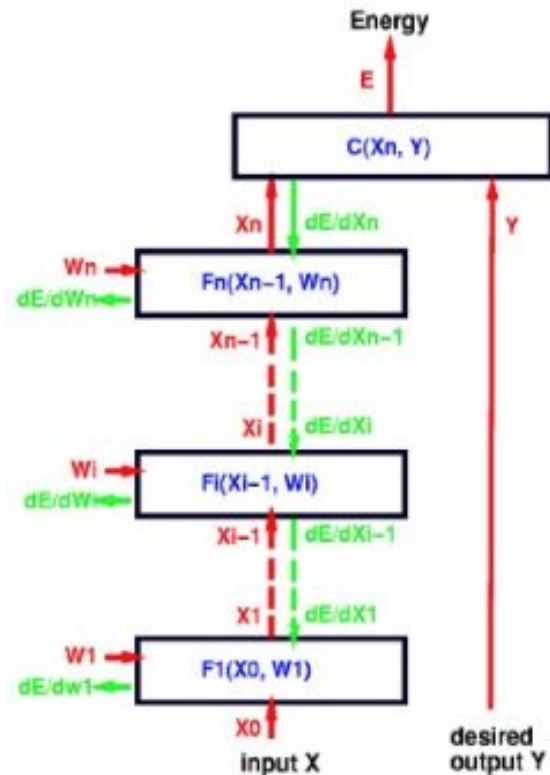


figure credit Yann LeCun, ICML '13 tutorial

# What is Deep Learning?

## Compositional Models Learned End-to-End

**Back-propagation** jointly learns all of the model parameters to optimize the output for the task.

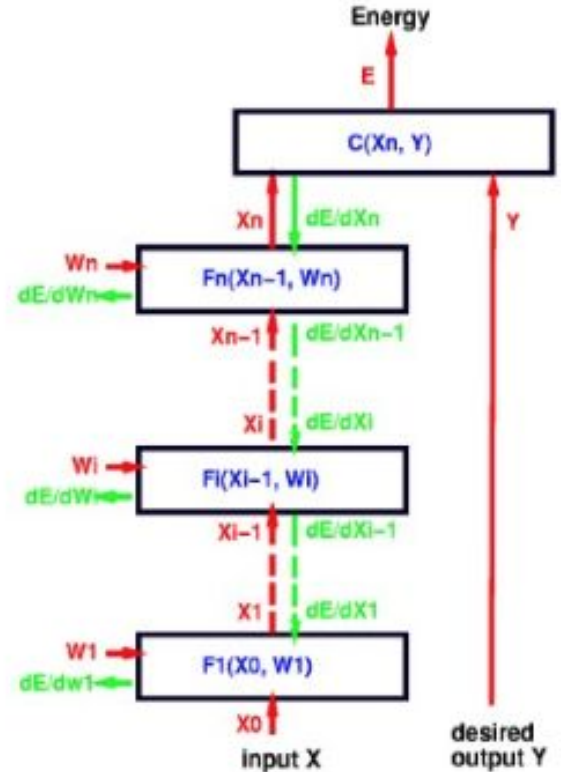
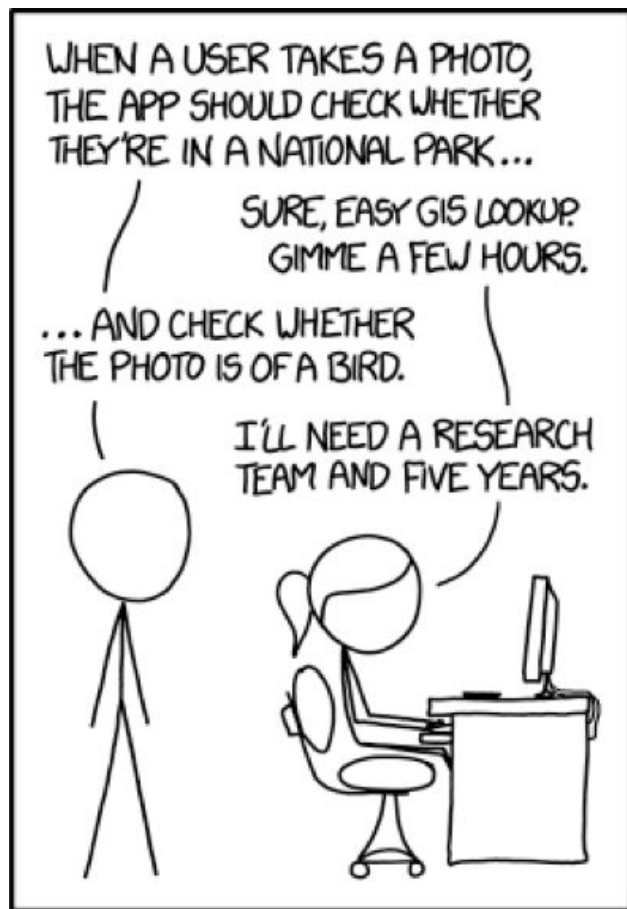


figure credit Yann LeCun, ICML '13 tutorial



IN CS, IT CAN BE HARD TO EXPLAIN  
THE DIFFERENCE BETWEEN THE EASY  
AND THE VIRTUALLY IMPOSSIBLE.

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 → [i](#)

PARK?

**YES**

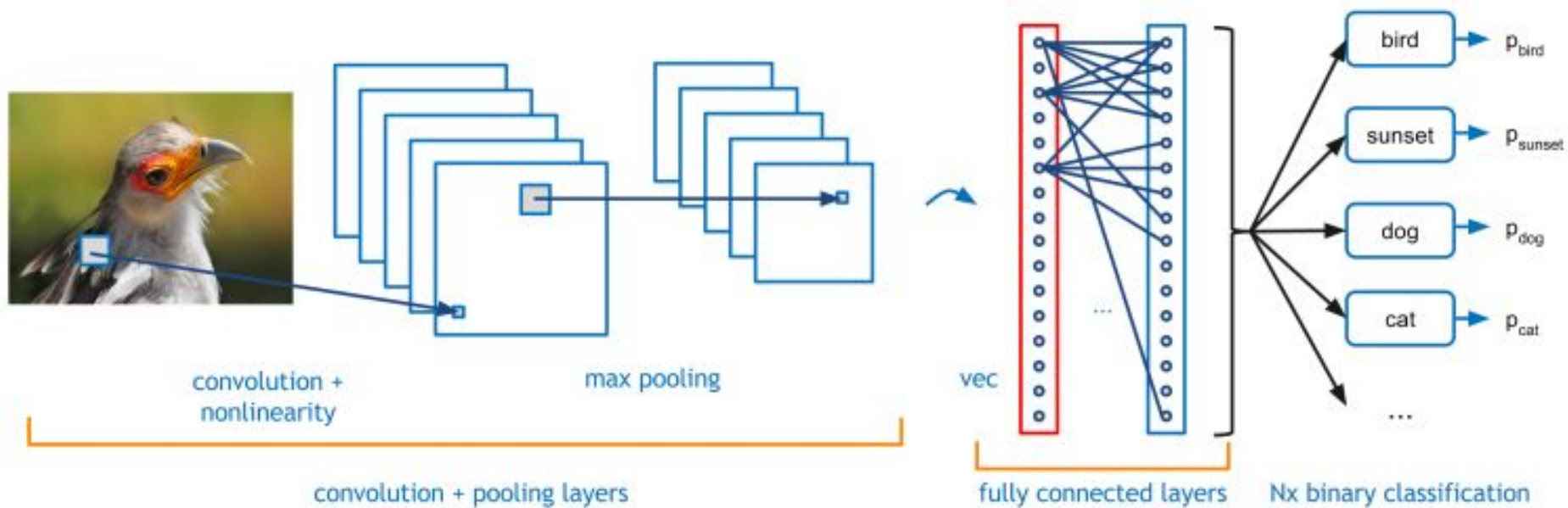
Ah yes, [Everglades](#) is truly beautiful.

BIRD?

**YES**

Dude, that is such a bird.





All in a day's work with Caffe



# What is Caffe?

**Open framework, models, and worked examples**  
for deep learning

- 4.5 years old
- 7,000+ citations, 250+ contributors, 24,000+ stars
- 15,000+ forks, >1 pull request / day average at peak
- focus has been vision, but also handles 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

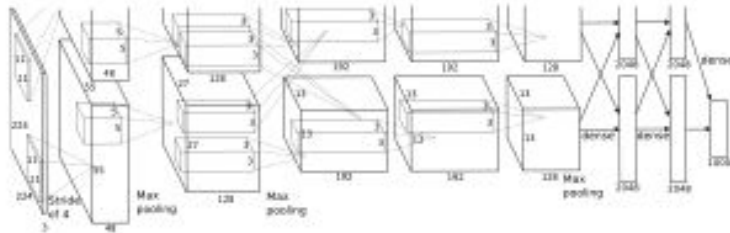
New Issues

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**.

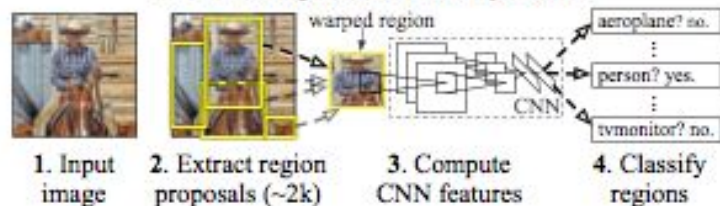


# Reference Models

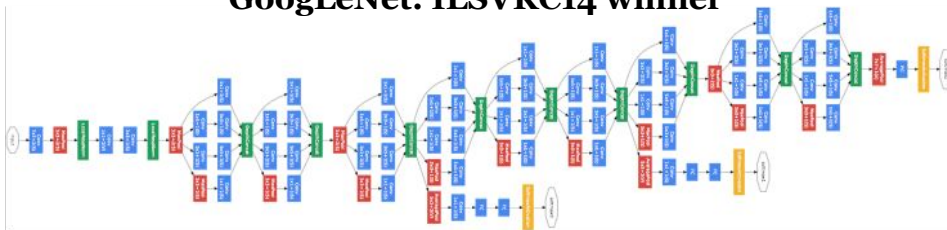
**AlexNet: ImageNet Classification**



**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)

CAFFE

EXAMPLES + APPLICATIONS



# Share a Sip of Brewed Models

[demo.caffe.berkeleyvision.org](http://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/>



## 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

*B. Zhou et al. NIPS 14*

# 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).

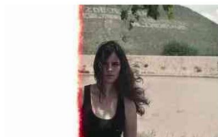
Ethereal



HDR



Melancholy



Minimal



Other Styles:

[Vintage](#)

[Long Exposure](#)

[Noir](#)

[Pastel](#)

[Macro](#)

... and so on.

# Object Detection

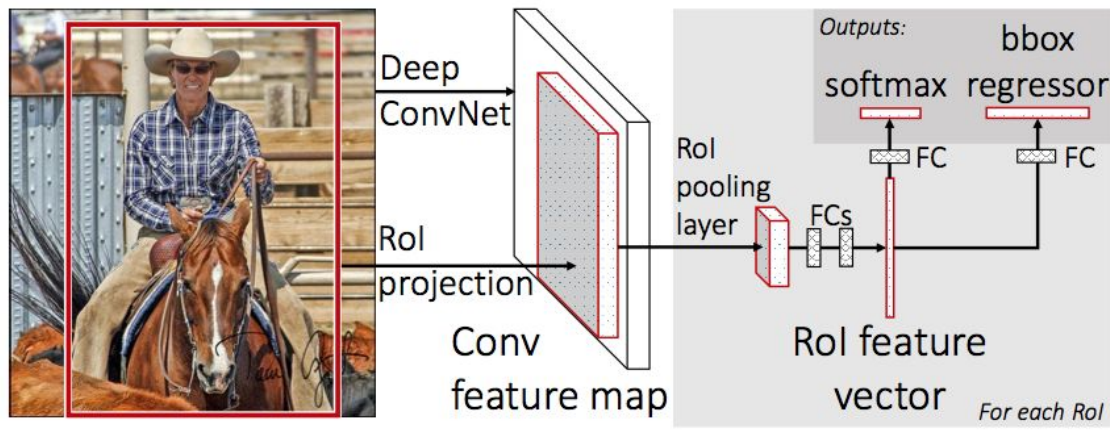
## R-CNNs: Region-based Convolutional Networks

### 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



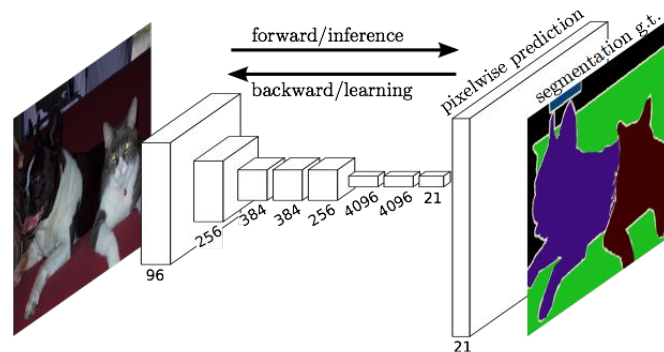
papers + code online

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

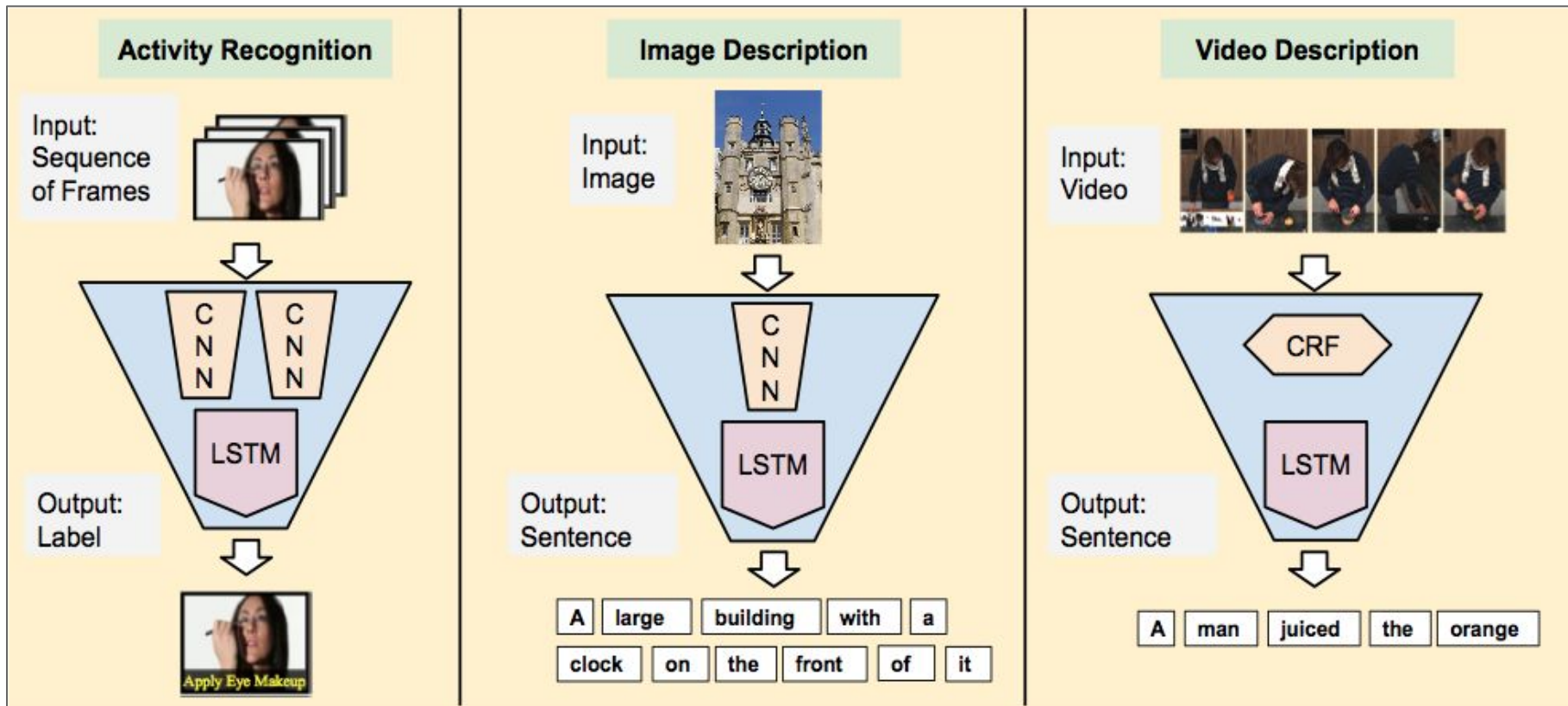


## Applications

- semantic segmentation
- denoising
- depth estimation
- optical flow



# Visual Sequence Tasks





# Recurrent Networks for Sequences

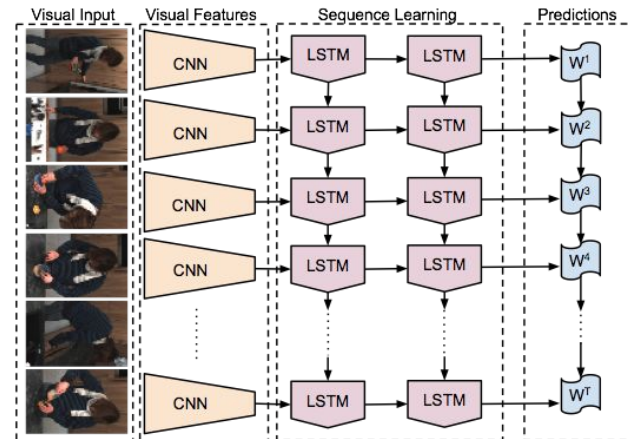
Recurrent Nets and Long Short Term Memories (LSTM) are sequential models

- 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)

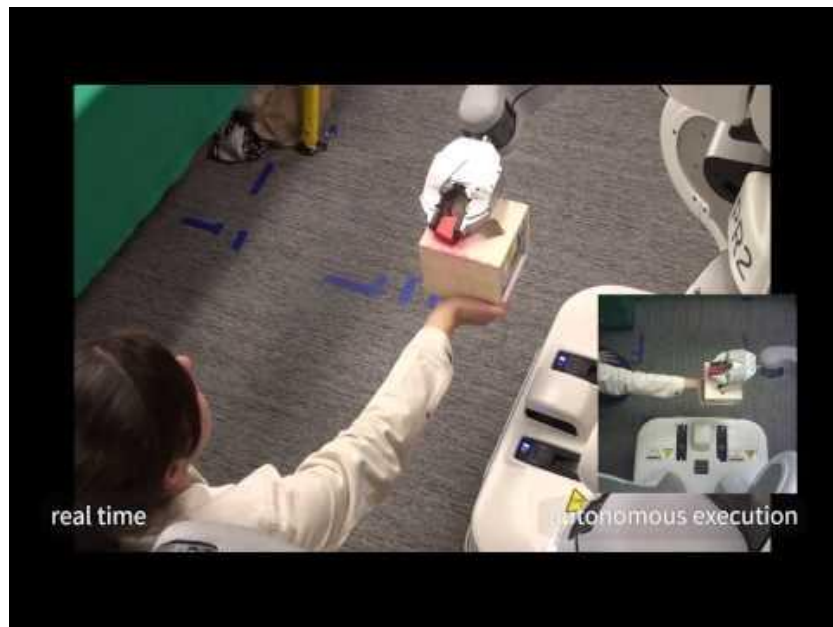


**LRCN:**

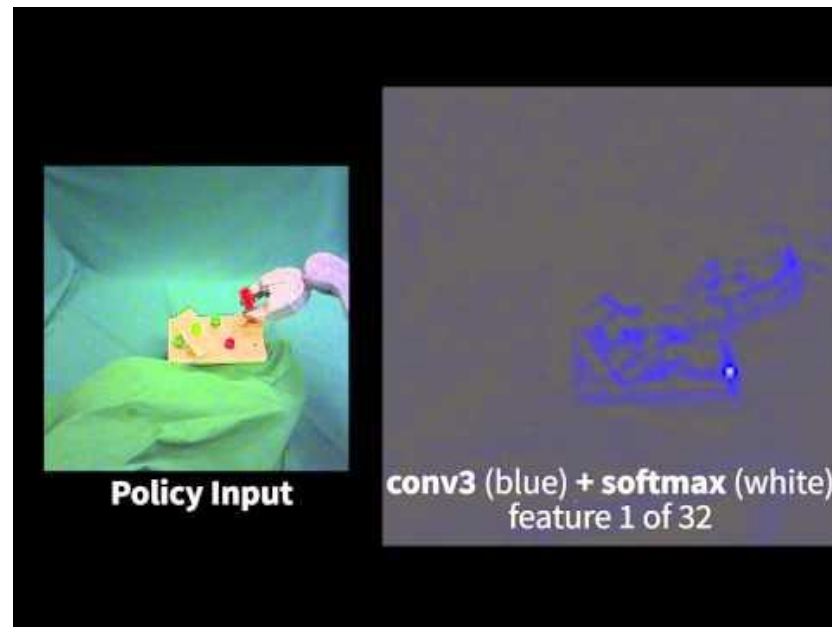
recurrent + convolutional  
for visual sequences



# Deep Visuomotor Control

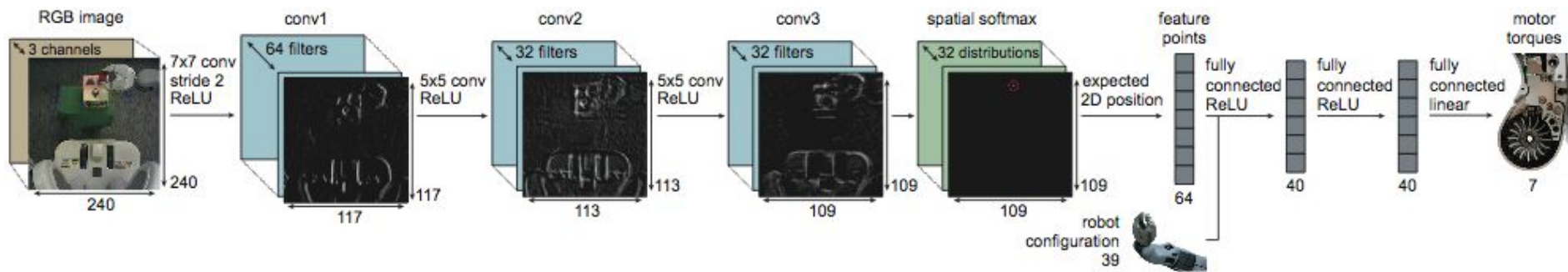


example experiments



feature visualization

# 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](#)

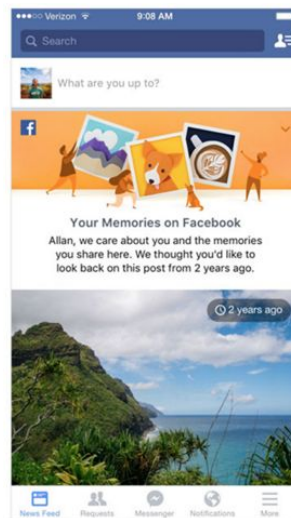
# Caffeinated Companies



... 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  
include [fb-caffe-exts](#) thanks Andrew!



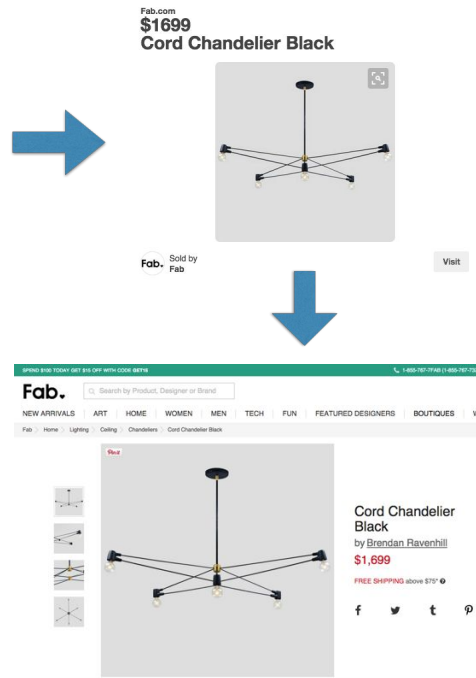
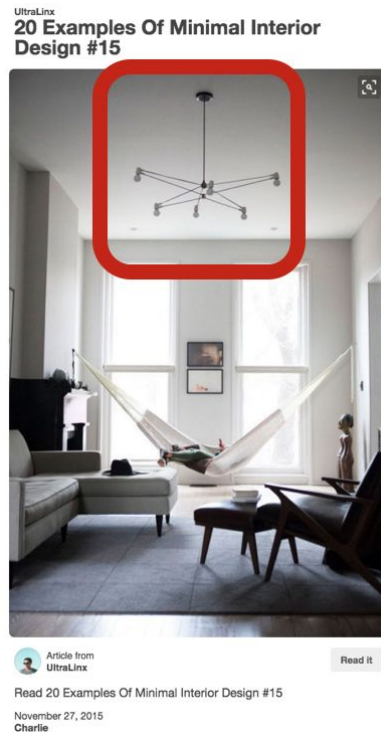
**On This Day**  
highlight content



**Automatic Alt Text**  
recognize photo content  
for accessibility

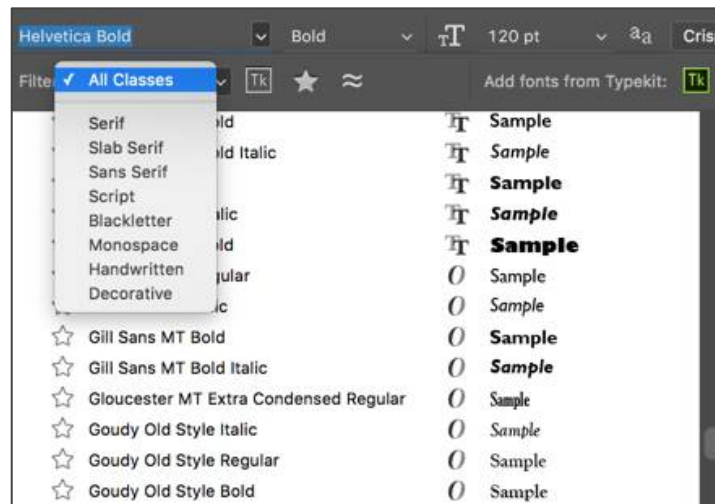
# Caffe at Pinterest

- 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, ...



# 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

- curate news and restaurant photos for recommendation
- arrange user photo albums



**News Image Recommendation**  
select and crop images for news

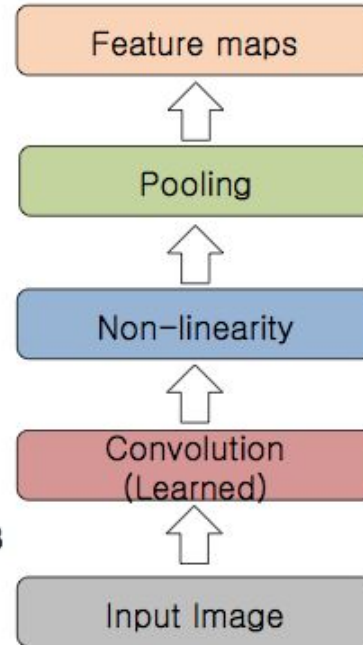
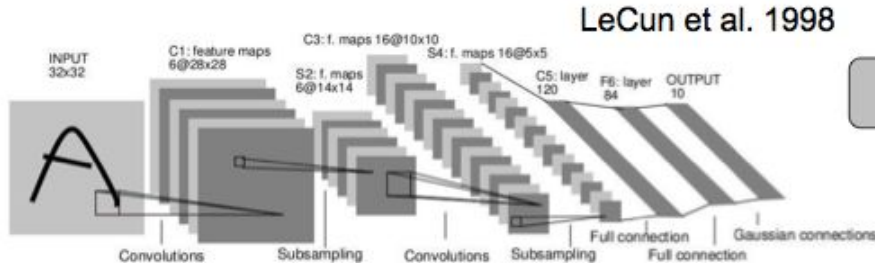
# Classification

instant recognition the Caffe way

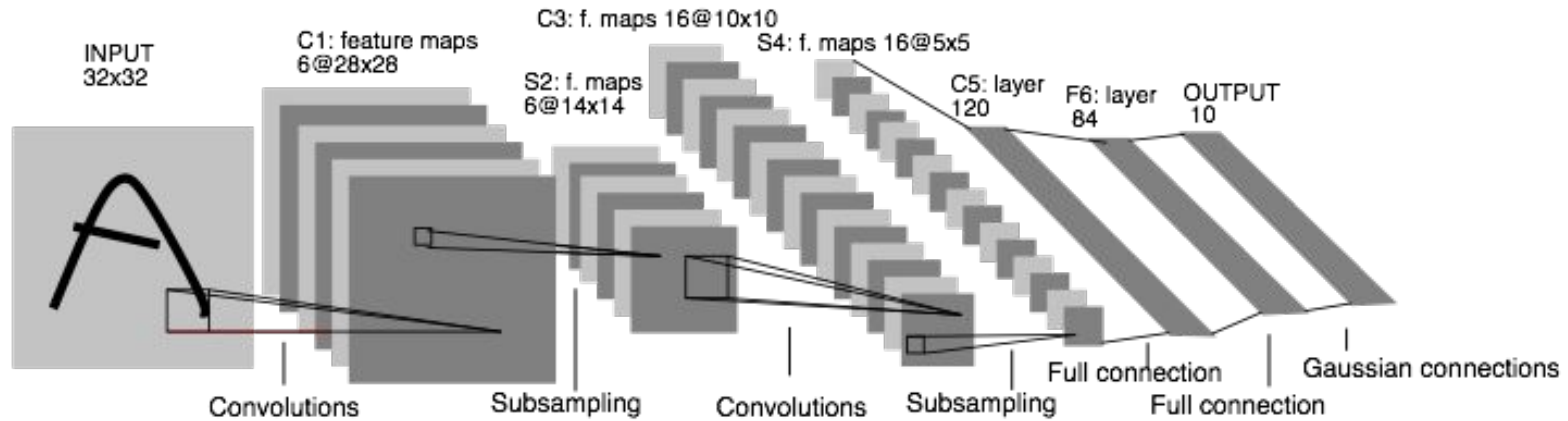
[see notebook](#)

# Convolutional Network

- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Supervised
- Train convolutional filters by back-propagating classification error

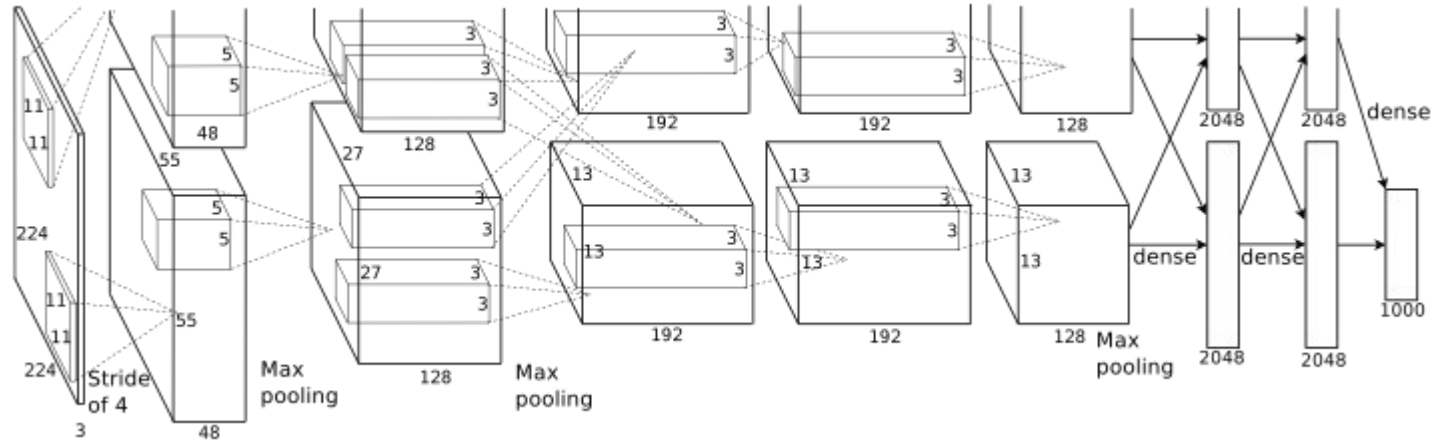


# Convolutional Networks: 1989



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [ LeNet ]

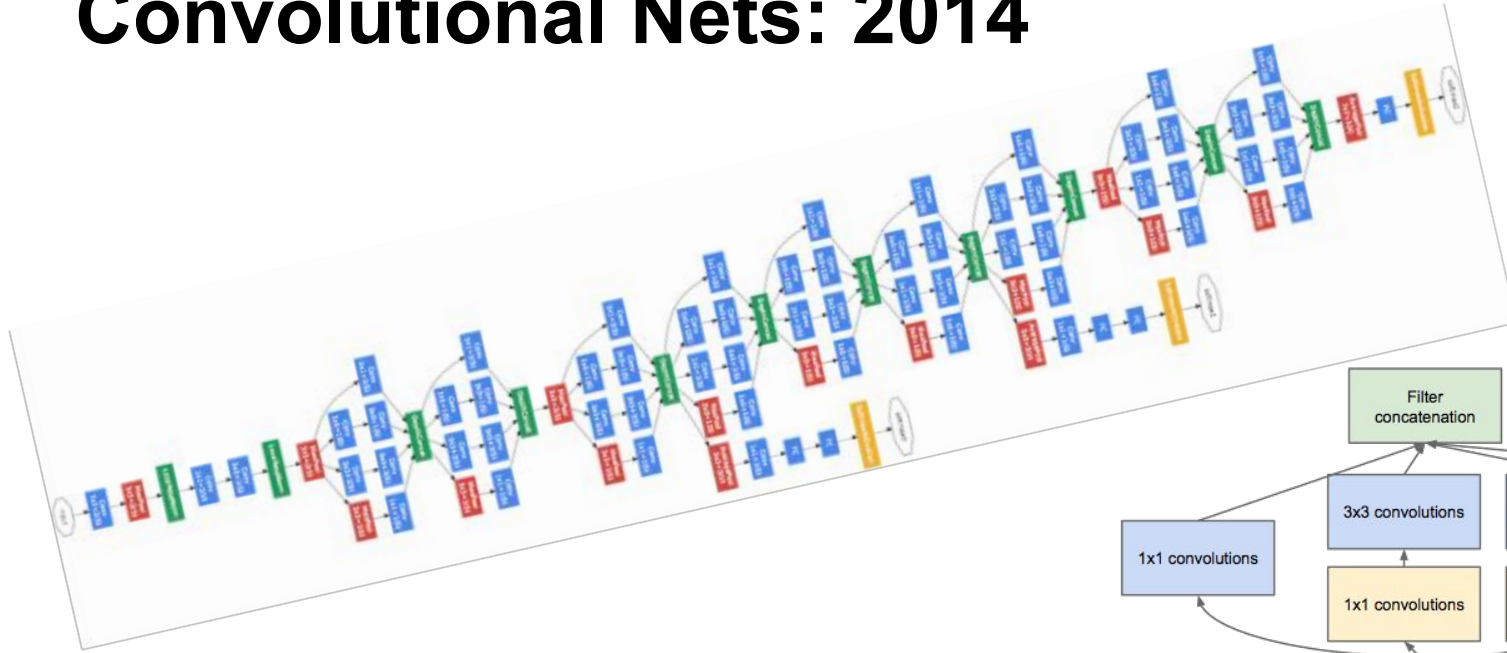
# Convolutional Nets: 2012



AlexNet: a layered model composed of convolution, subsampling, and further operations followed by a holistic representation and all-in-all a landmark classifier on ILSVRC12. [ AlexNet ]

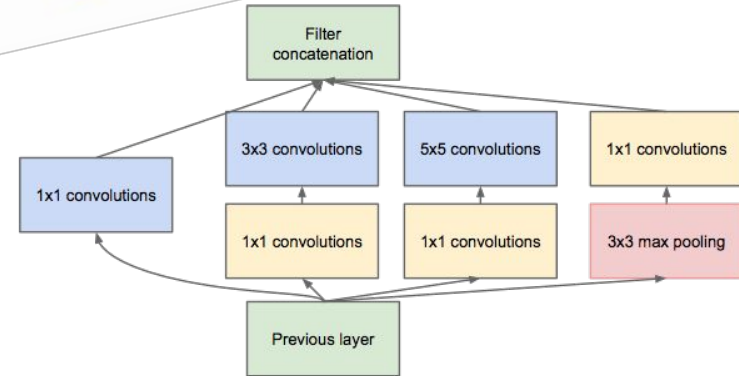
- + data
- + gpu
- + non-saturating nonlinearity
- + regularization

# Convolutional Nets: 2014



ILSVRC14 Winners: **~6.6% Top-5 error**

- GoogLeNet: composition of multi-scale dimension-reduced modules (pictured)
- VGG: 16 layers of 3x3 convolution interleaved with max pooling + 3 fully-connected layers



- + depth
- + data
- + dimensionality reduction

# Learning LeNet

back to the future of visual recognition

[see notebook](#)

# Deep Learning, as it is executed...

What should a 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" ...}

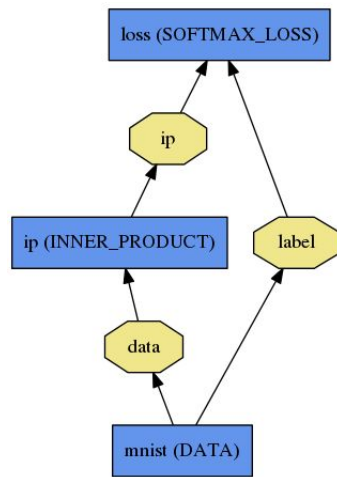
**layer** { name: "conv" ...}

**layer** { name: "pool" ...}

... more layers ...

**layer** { name: "loss" ...}

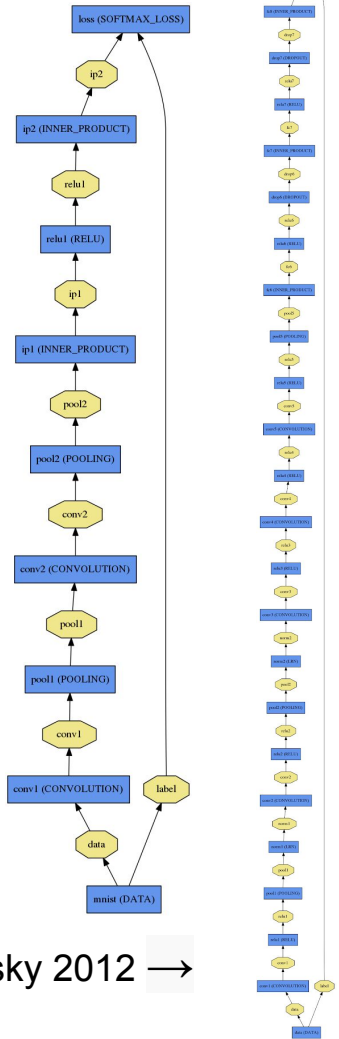
- Caffe creates and checks the net from the definition.
- Data and derivatives flow through the net as *blobs* – an array interface



LogReg ↑

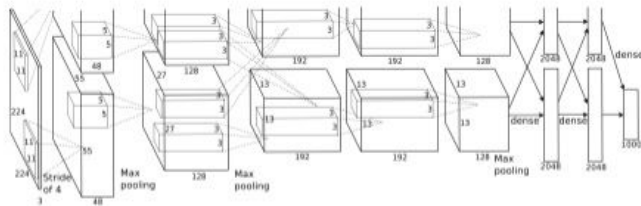
LeNet →

ImageNet, Krizhevsky 2012 →



# Forward / Backward the essential Net computations

Forward:  
inference  $f_W(x)$



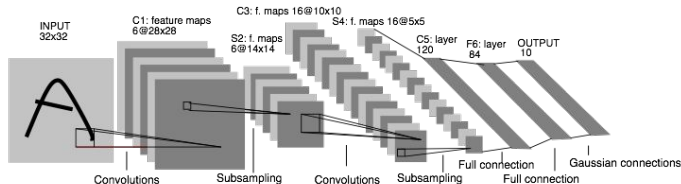
“espresso”  
+ loss

$\nabla f_W(x)$  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.

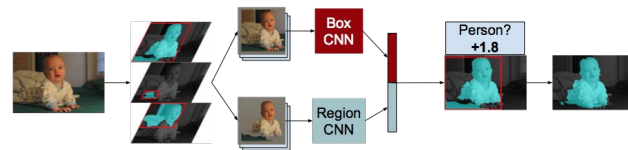
# DAG

Many current deep models have linear structure

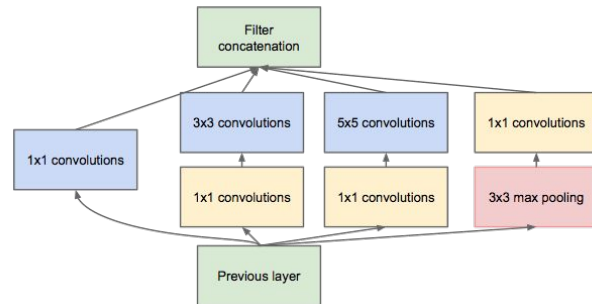


but Caffe nets can have any directed acyclic graph (DAG) structure.

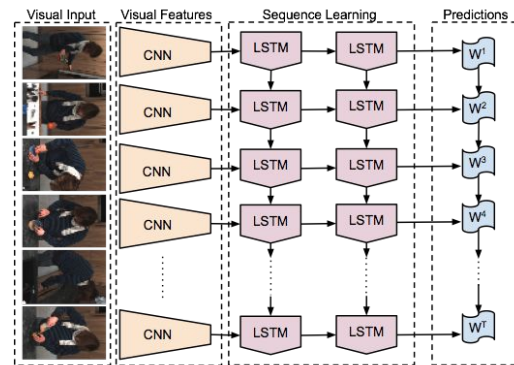
Define bottoms and tops  
and Caffe will connect the net.



SDS two-stream net



GoogLeNet Inception Module



LRCN joint vision-sequence model

# Layer Protocol

**Setup:** run once for initialization.

**Forward:** make output given input.

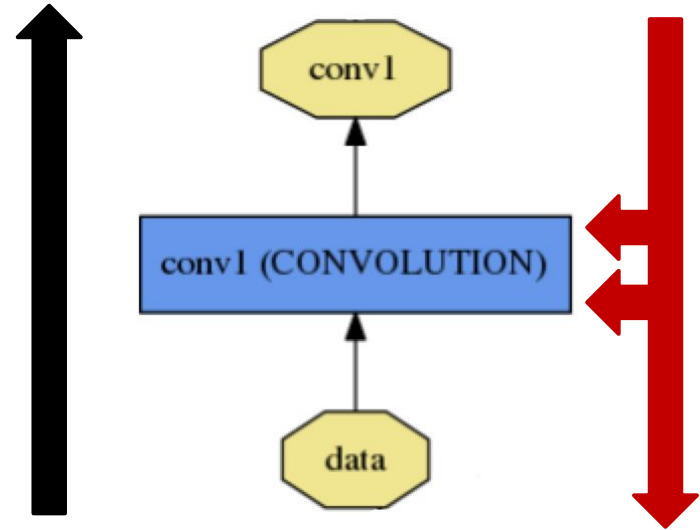
**Backward:** make gradient of output

- w.r.t. bottom
- w.r.t. parameters (if needed)

**Reshape:** set dimensions.

## *Compositional Modeling*

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



[Layer Development Checklist](#)

```

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.

```

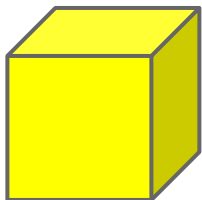
layer {
  type: "Python"
  python_param {
    module: "layers"
    layer: "EuclideanLoss"
  }
}

```

# Blob

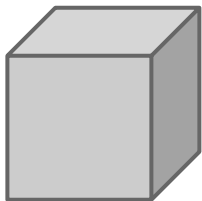
Blobs are N-D arrays for storing and communicating information.

- hold data, derivatives, and parameters
- lazily allocate memory
- shuttle between CPU and GPU



## Data

Number x  $K$  Channel x Height x Width  
256 x 3 x 227 x 227 for ImageNet train input



## Parameter: Convolution Weight

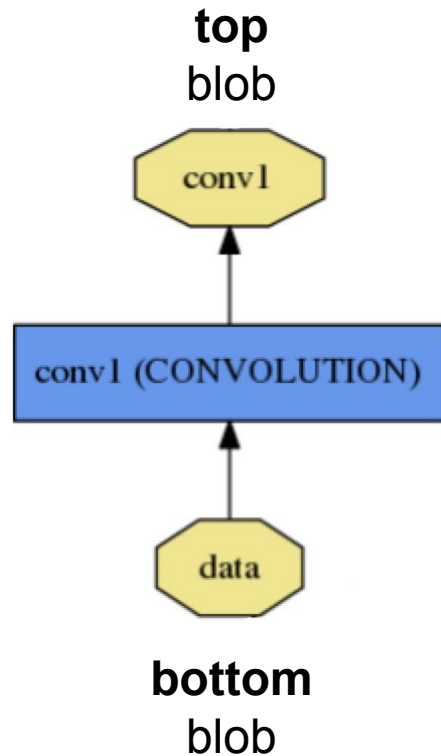
$N$  Output x  $K$  Input x Height x Width  
96 x 3 x 11 x 11 for CaffeNet conv1



## Parameter: Convolution Bias

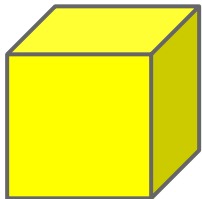
96 x 1 x 1 x 1 for CaffeNet conv1

```
name: "conv1"  
type: CONVOLUTION  
bottom: "data"  
top: "conv1"  
... definition ...
```



# Blob

Blobs provide a unified memory interface.



**Reshape(num, channel, height, width)**

- declare dimensions
- make *SyncedMem* -- but only lazily allocate

**cpu\_data(), mutable\_cpu\_data()**

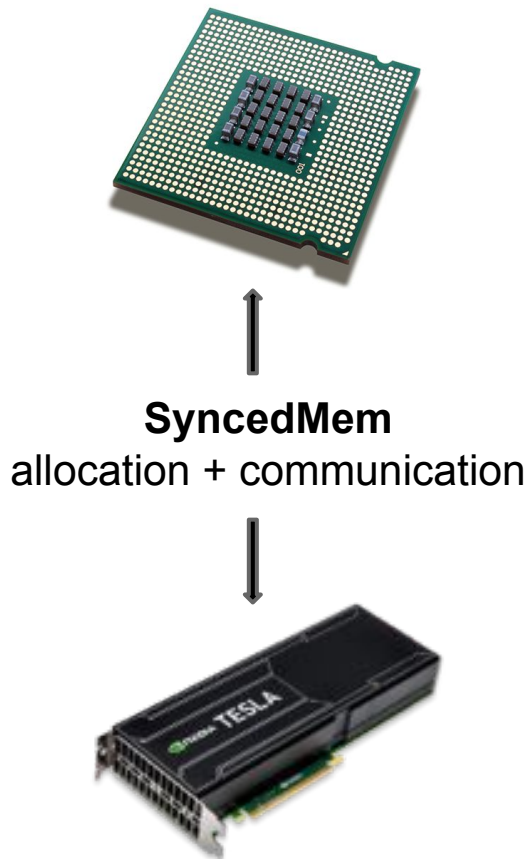
- host memory for CPU mode

**gpu\_data(), mutable\_gpu\_data()**

- device memory for GPU mode

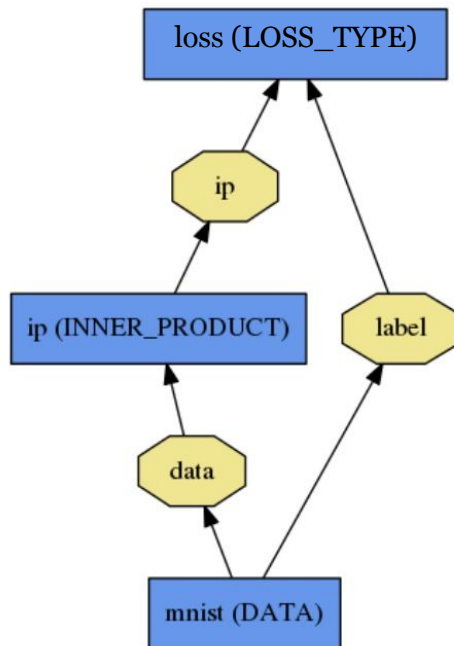
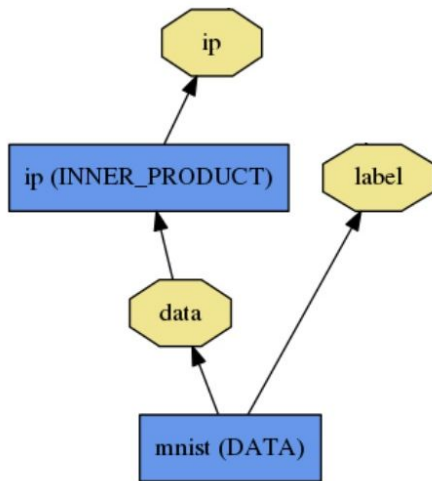
**{cpu,gpu}\_diff(), mutable\_{cpu,gpu}\_diff()**

- derivative counterparts to data methods
- easy access to data + diff in forward / backward



# Loss

What kind of model is this?



## Classification

SoftmaxWithLoss

HingeLoss

## Linear Regression

EuclideanLoss

## Attributes / Multiclassification

SigmoidCrossEntropyLoss

Others...

New Task

NewLoss

Define the task by the **loss**.



# Protobuf Model Format

- Strongly typed format
- 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];  
}
```

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

# 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"`

`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`

Stochastic Gradient Descent (SGD) + momentum ·

Adaptive Gradient (ADAGRAD) · Nesterov's Accelerated Gradient (NAG)

# Solver Showdown: MNIST Autoencoder

## AdaGrad

```
I0901 13:36:30.007884 24952 solver.cpp:232] Iteration 65000, loss = 64.1627
I0901 13:36:30.007922 24952 solver.cpp:251] Iteration 65000, Testing net (#0) # train set
I0901 13:36:33.019305 24952 solver.cpp:289] Test loss: 63.217
I0901 13:36:33.019356 24952 solver.cpp:302]      Test net output #0: cross_entropy_loss = 63.217 (* 1 = 63.217 loss)
I0901 13:36:33.019773 24952 solver.cpp:302]      Test net output #1: l2_error = 2.40951
```

## SGD

```
I0901 13:35:20.426187 20072 solver.cpp:232] Iteration 65000, loss = 61.5498
I0901 13:35:20.426218 20072 solver.cpp:251] Iteration 65000, Testing net (#0) # train set
I0901 13:35:22.780092 20072 solver.cpp:289] Test loss: 60.8301
I0901 13:35:22.780138 20072 solver.cpp:302]      Test net output #0: cross_entropy_loss = 60.8301 (* 1 = 60.8301 loss)
I0901 13:35:22.780146 20072 solver.cpp:302]      Test net output #1: l2_error = 2.02321
```

## Nesterov

```
I0901 13:36:52.466069 22488 solver.cpp:232] Iteration 65000, loss = 59.9389
I0901 13:36:52.466099 22488 solver.cpp:251] Iteration 65000, Testing net (#0) # train set
I0901 13:36:55.068370 22488 solver.cpp:289] Test loss: 59.3663
I0901 13:36:55.068410 22488 solver.cpp:302]      Test net output #0: cross_entropy_loss = 59.3663 (* 1 = 59.3663 loss)
I0901 13:36:55.068418 22488 solver.cpp:302]      Test net output #1: l2_error = 1.79998
```

# Weight Sharing

- Just give the parameter blobs explicit names using the `param` field
- Layers specifying the same `param` name will share that parameter, accumulating gradients accordingly

```
layer: {  
  name: 'innerproduct1'  
  type: INNER_PRODUCT  
  inner_product_param {  
    num_output: 10  
    bias_term: false  
    weight_filler {  
      type: 'gaussian'  
      std: 10  
    }  
  }  
  param: 'sharedweights'  
  bottom: 'data'  
  top: 'innerproduct1'  
}  
layer: {  
  name: 'innerproduct2'  
  type: INNER_PRODUCT  
  inner_product_param {  
    num_output: 10  
    bias_term: false  
  }  
  param: 'sharedweights'  
  bottom: 'data'  
  top: 'innerproduct2'  
}
```

# Recipe for Brewing

- Convert the data to Caffe-format
  - lmdb, leveldb, hdf5 / .mat, list of images, etc.
- Define the Net
- Configure the Solver
- `caffe train -solver solver.prototxt -gpu 0`
- Examples are your friends
  - `caffe/examples/mnist, cifar10, imagenet`
  - `caffe/examples/*.ipynb`
  - `caffe/models/*`

# Brewing Models

from logistic regression to non-linearity

[see notebook](#)

# Take a pre-trained model and fine-tune to new tasks

[DeCAF] [Zeiler-Fergus] [OverFeat]

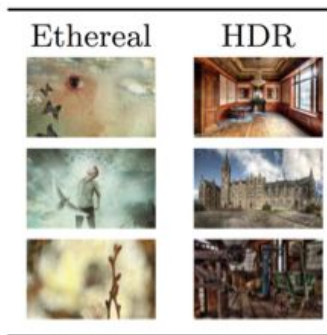
Lots of Data



ImageNet

image by Andrej Karpathy

Your Task



Style  
Recognition



Dogs vs.  
Cats  
top 10 in  
10 minutes

© kaggle.com



# From ImageNet to Style

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
    ...
  }
}
```

Input:  
A different source

new name =  
new params

Last Layer:  
A different classifier

# From ImageNet to Style

```
> 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()
```

Vintage HDR Melancholy Minimal



# Fine-tuning

transferring features to style recognition

[see notebook](#)

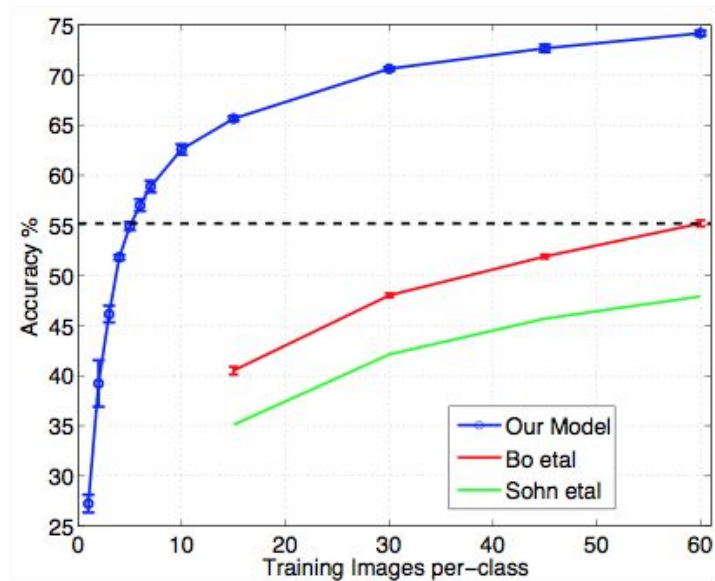
# When to Fine-tune?

A good first step

- More robust optimization – good initialization helps
- Needs less data
- Faster learning

State-of-the-art results in

- recognition
- detection
- segmentation



[Zeiler-Fergus]

# Fine-tuning Tricks

## Learn the last layer first

- Caffe layers have local learning rates: `param { lr_mult: 1 }`
- Freeze all but the last layer for fast optimization and avoiding early divergence by setting `lr_mult: 0` to fix a parameter.
- Stop if good enough, or keep fine-tuning

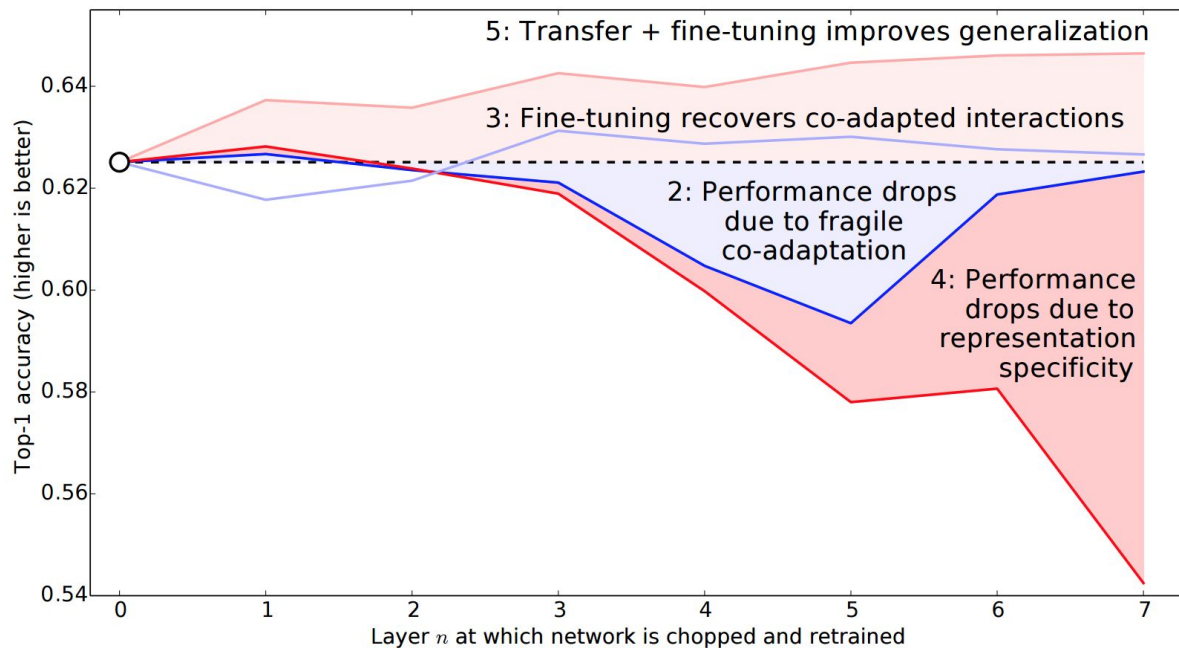
## Reduce the learning rate

- Drop the solver learning rate by 10x, 100x
- Preserve the initialization from pre-training and avoid divergence

Do net surgery see notebook on [editing model parameters](#)

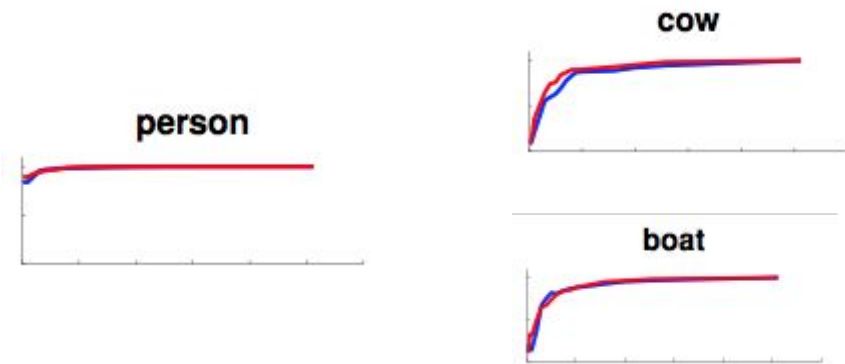
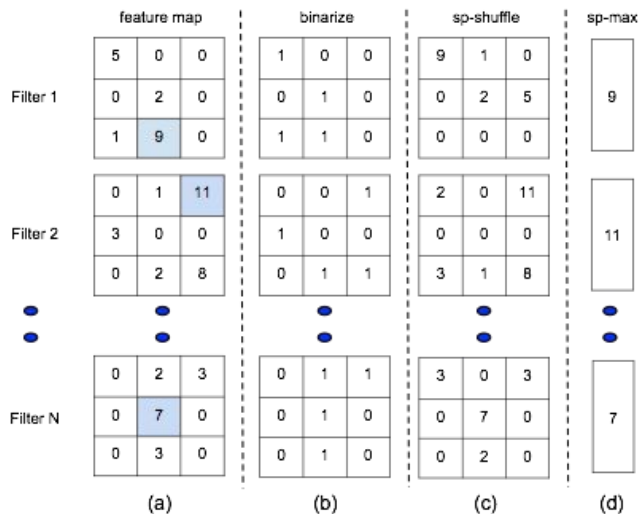
# Transferability of Features

Yosinski et al. NIPS 2014



# After fine-tuning

- Supervised pre-training does not overfit
- Representation is (mostly) distributed
- Sparsity comes “for free” in deep representation



# Editing model parameters

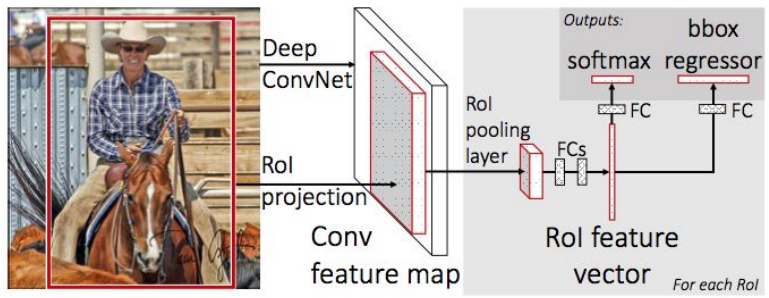
how to do net surgery to set custom weights

[see notebook](#)

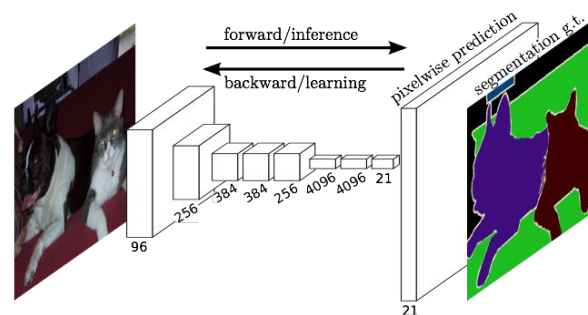


# Up Next The Latest Roast

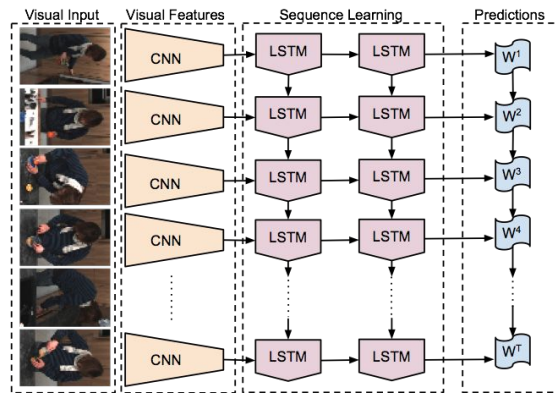
## Detection



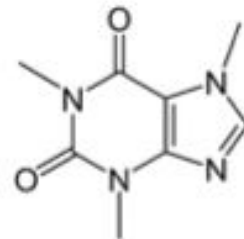
## Pixelwise Prediction



## Sequences



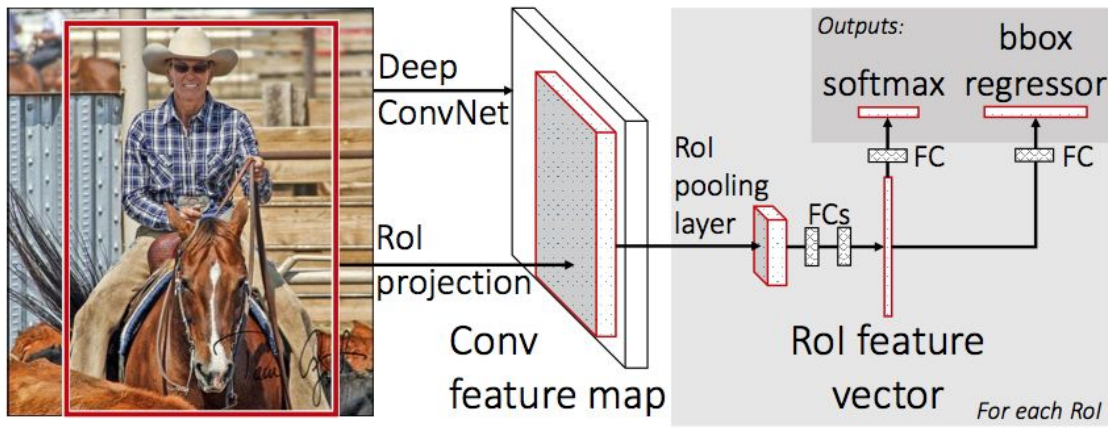
## Framework Future



# Detection

## Fast R-CNN

- convolve once
- project + detect



## Faster R-CNN

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

[arXiv](#) and [code](#) for Fast R-CNN

Ross Girshick, Shaoqing Ren,  
Kaiming He, Jian Sun

# Pixelwise Prediction

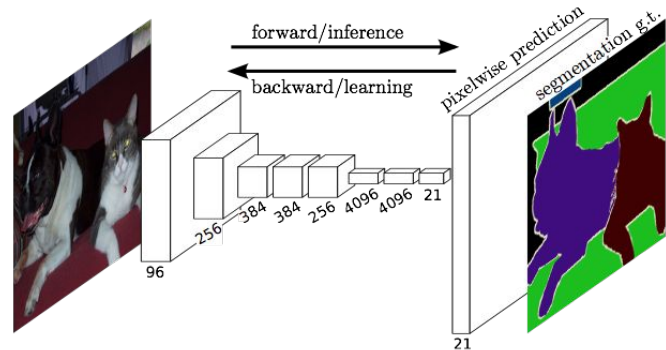
Fully convolutional networks for pixel prediction applied to semantic segmentation

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

Further applications

- depth
- boundaries
- flow + more

CVPR15 [arXiv](#) and [reference models + code](#)



Jon Long\* & Evan Shelhamer\*,  
Trevor Darrell

# Sequences

Recurrent Net and Long Short Term Memory LSTM are sequential models

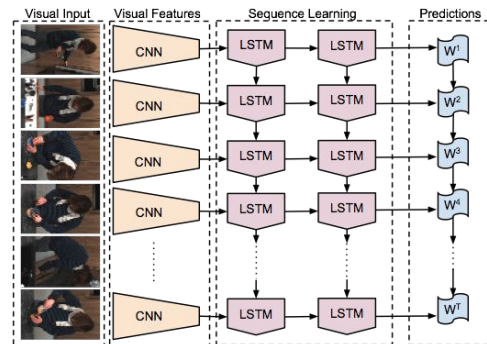
- video
- language
- dynamics

learned by backpropagation through time.

LRCN: Long-term Recurrent Convolutional Network

- activity recognition
- image captioning
- video captioning

CVPR15 [arXiv](#) and [project site](#)



A group of young men playing a game of soccer.

**Jeff Donahue et al.**

# 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

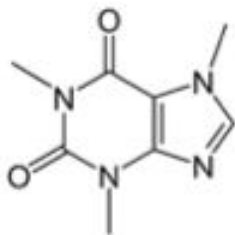
Halide interface for prototyping and experimenting

Widening the Circle continued and closer collaborative development

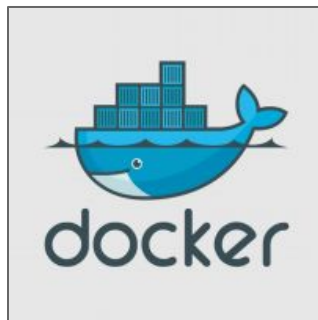
# Next Steps

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

Next Up:



[caffe.berkeleyvision.org](http://caffe.berkeleyvision.org)



[github.com/BVLC/caffe](https://github.com/BVLC/caffe)

Check out Caffe on github

[Run Caffe through Docker](#)  
and NVIDIA Docker for GPU

Join the [caffe-users](#) mailing list

# Help Brewing

## Documentation

- [tutorial documentation](#)
- [hands-on examples](#)

## Modeling, Usage, and Installation

- [caffe-users group](#)
- [gitter.im chat](#)

## Convolutional Nets

- [CS231n online convnet class](#)  
by Andrej Karpathy and Fei-Fei Li
- [Deep Learning Online](#)  
by Michael Nielsen
- [Deep Learning Book](#)  
by Goodfellow, Bengio, Courville



# Thanks to the Caffe Crew



...plus the cold-brew

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 our 150+  
open source contributors  
and vibrant community!



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

# 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.

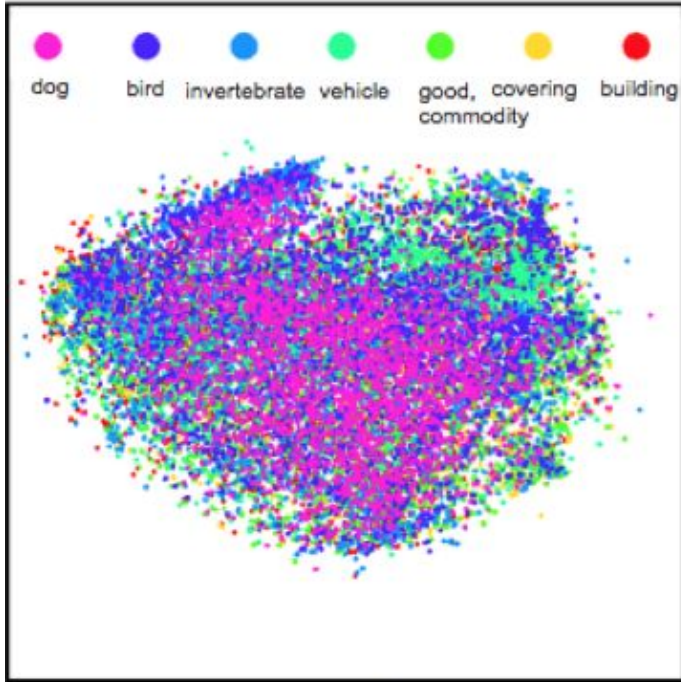
[ Chopra05 ] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. CVPR, 2005.

**END**

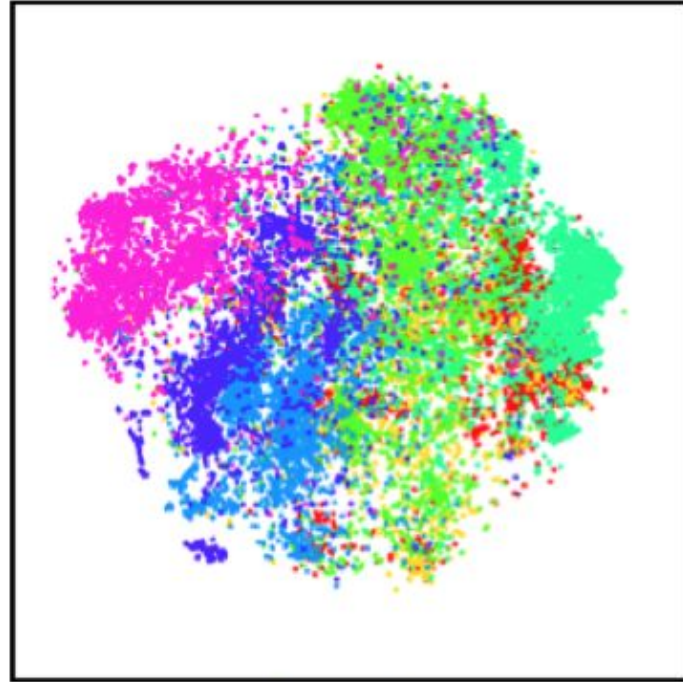
# MORE DETAILS

# Why Deep Learning?

## The Unreasonable Effectiveness of Deep Features



Low-level: Pool<sub>1</sub>



High-level: FC<sub>6</sub>

Classes separate in the deep representations and transfer to many tasks.

[DeCAF] [Zeiler-Fergus]

# Why Deep Learning?

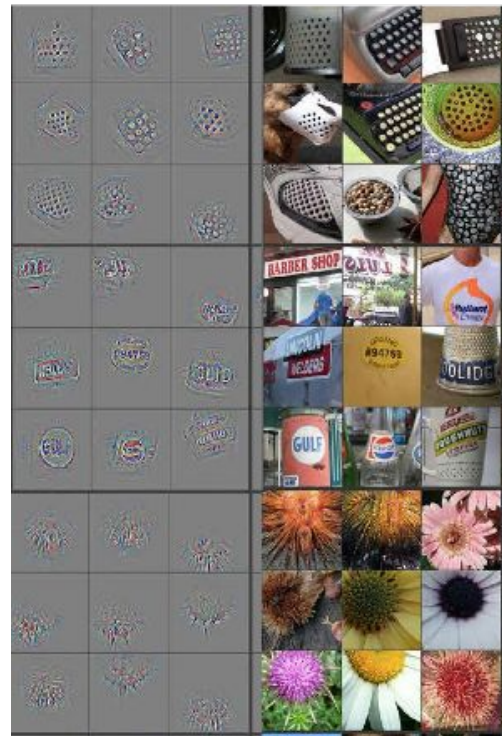
## The Unreasonable Effectiveness of Deep Features



Maximal activations of pool<sub>5</sub> units

[R-CNN]

Rich visual structure of features deep in hierarchy.



conv<sub>5</sub> DeConv visualization

[Zeiler-Fergus]



# Why Deep Learning?

## The Unreasonable Effectiveness of Deep Features

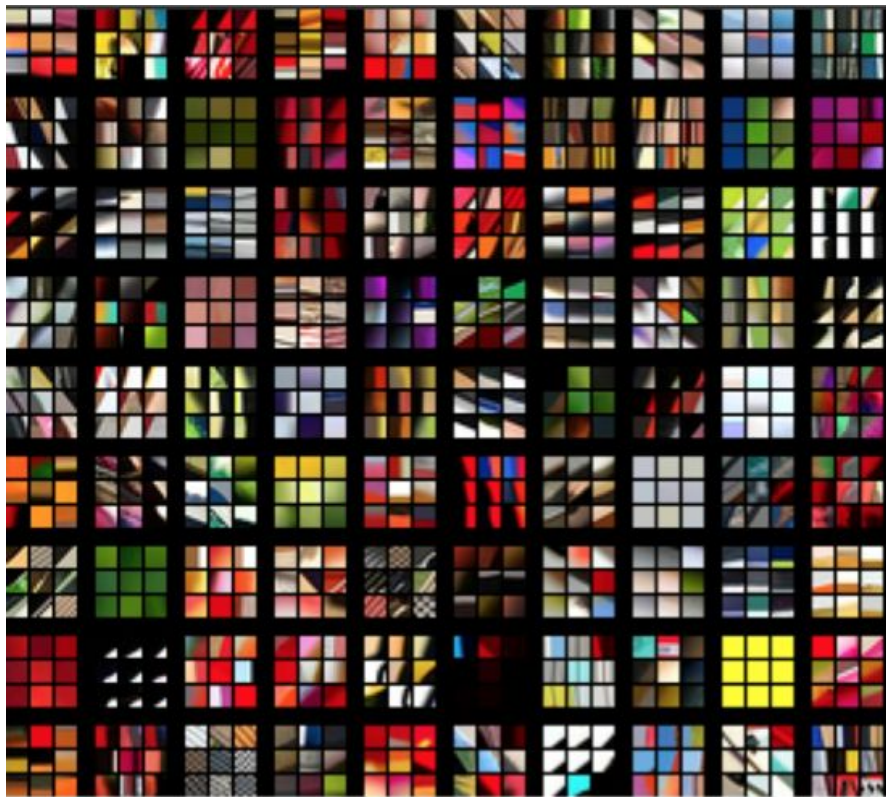
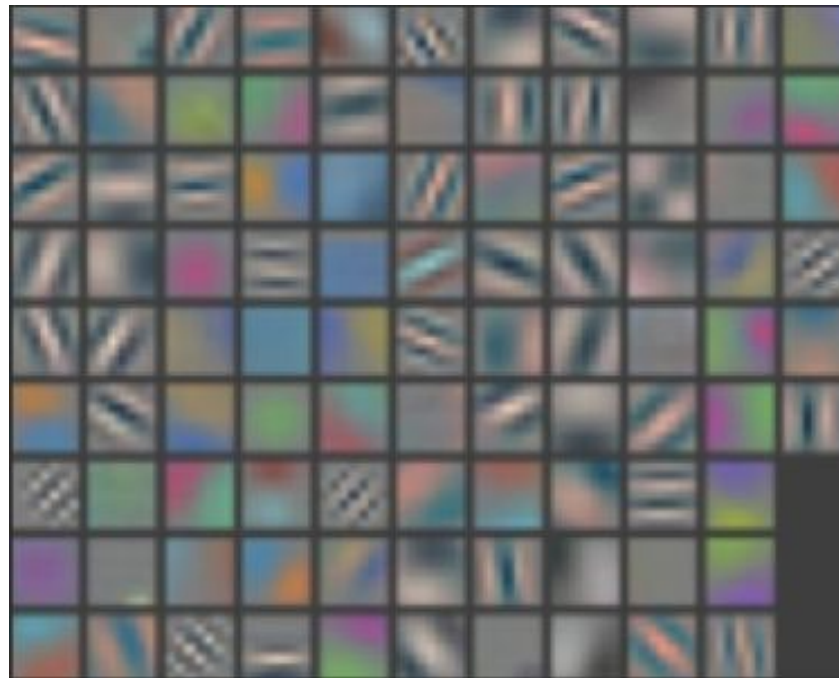


image patches that strongly activate 1st layer filters



1st layer filters

[Zeiler-Fergus]