Administrative

- In A1 we caught a small number of questionable cases of similar code. Keep the Honor Code in mind.

- Moving Project milestone to Monday Feb 16 (was Friday Feb 13 before)

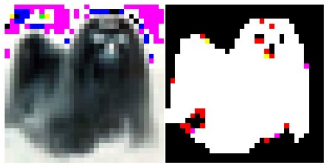
- Terminal lets you use faster than **xlarge**, but don't use them. OpenBLAS is not configured optimally and you aren't seeing any improvements, just burning cash. We are fixing this for A3.

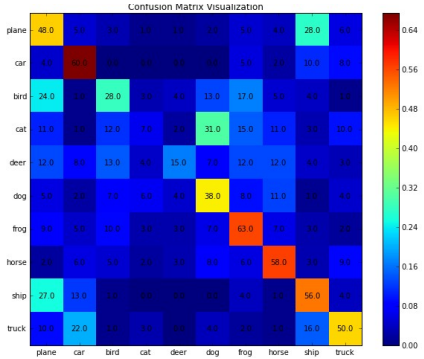
Lecture 9:

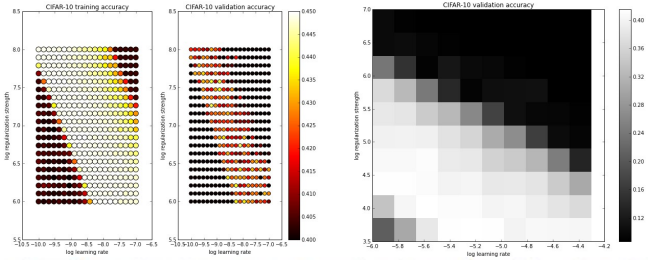
What makes ConvNets tick
&
Transfer Learning

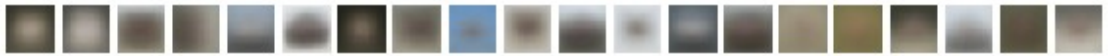
Assignment #1 Extra Credit: Cool submissions

- **Enhao**: HOG+Color Histogram+Spatial Pyramid Pooling+PCA, Hierarchical hyperparameter search
- **Amani**: A full Bag-of-words model, and implementation of all SVMs (One vs. All, One vs. One, Structured SVM)
- **Yilun**: LBP features, got 45%, nice!
- **Charles:** Confusion matrix
- **Rohit**: Prewitt filtering preprocessing (improves 43% -> 53%)
- **Pranav:** Similar idea with sobel filters, 50%
- **Vignesh:** Polynomial features **[x, x**2, x**3]**, boost to 51%.
- **John:** patch k-means features, 51%
- **Chris**: Nice CV plots
- **Albert:** F/B segmentation features, 53%









Chain your gradients…

forward pass:

$$
\begin{aligned}
s_1 &= W_1 x + b_1 \\
a_1 &= \max(0, s_1) \\
s_2 &= W_2 a_1 + b_2 \\
L &= f(s_2)
\end{aligned}
$$

softmax loss

Chain your gradients…

forward pass:

$$s_1 = W_1 x + b_1$$
$$a_1 = \max(0, s_1)$$
$$s_2 = W_2 a_1 + b_2$$
$$L = f(s_2)$$ ← softmax loss

---

backward pass:

$$ds_2 = \ldots$$
$$dW_2 = \ldots$$
$$db_2 = \ldots$$
$$da_1 = \ldots$$

$$ds_1 = \ldots$$
$$dW_1 = \ldots$$
$$db_1 = \ldots$$

short one-liners that implement small piece of chain rule:

local gradient * chained gradient

Chain your gradients…

forward pass:

$$s_1 = W_1 x + b_1$$
$$a_1 = \max(0, s_1)$$
$$s_2 = W_2 a_1 + b_2$$
$$L = f(s_2)$$

softmax loss

---

backward pass:

$$ds_2 = \ldots$$
$$dW_2 = \ldots$$
$$db_2 = \ldots$$
$$da_1 = \ldots$$

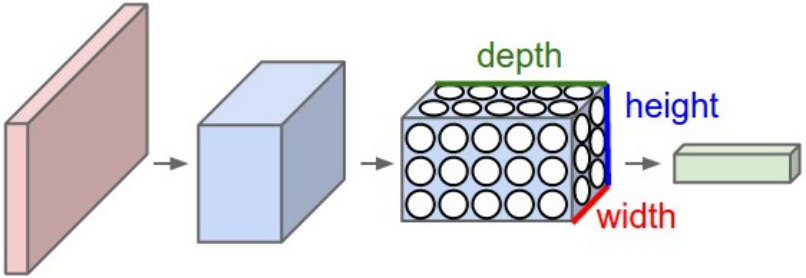
$$ds_1 = \ldots$$
$$dW_1 = \ldots$$
$$db_1 = \ldots$$

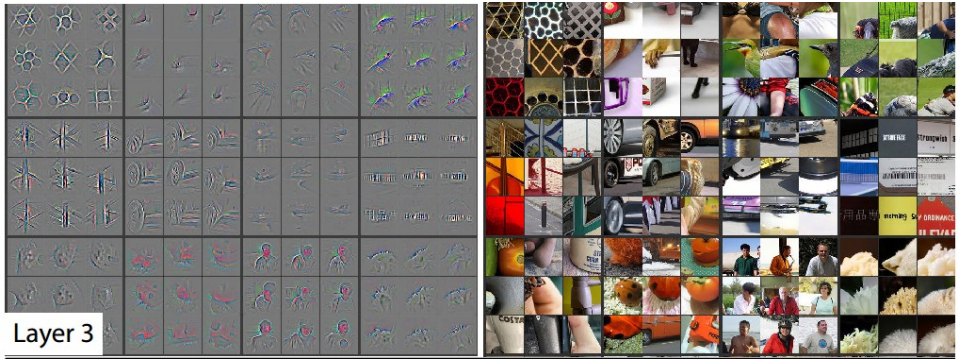
very similar to assignment #1

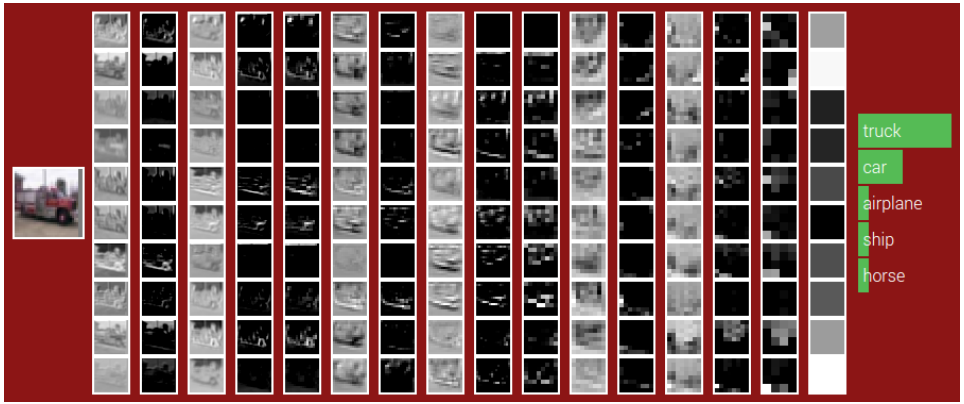
short one-liners that implement small piece of chain rule:

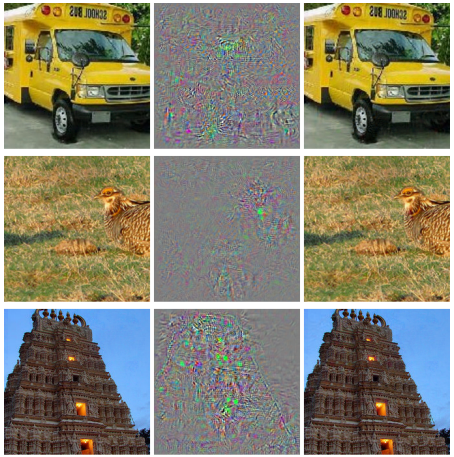
local gradient * chained gradient

Where we are...


depth
height
width


Layer 3


truck
car
airplane
ship
horse



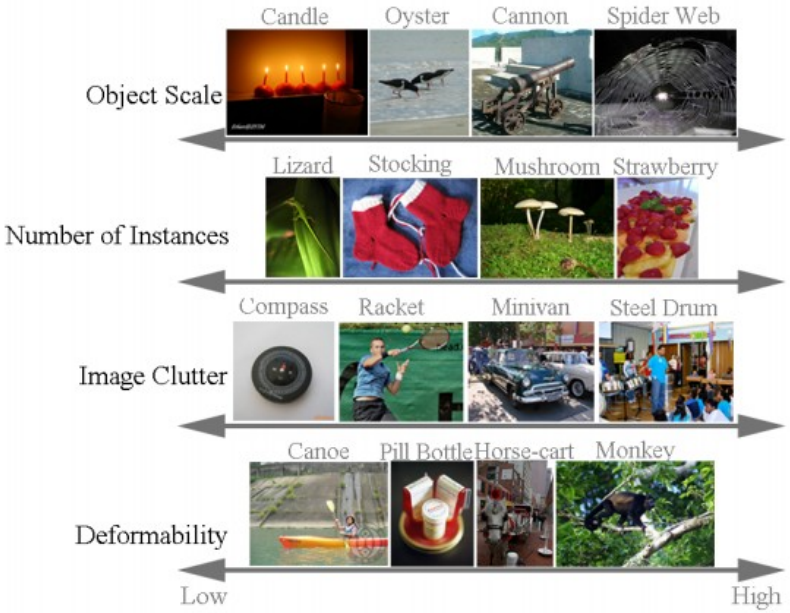
Left over from last class...

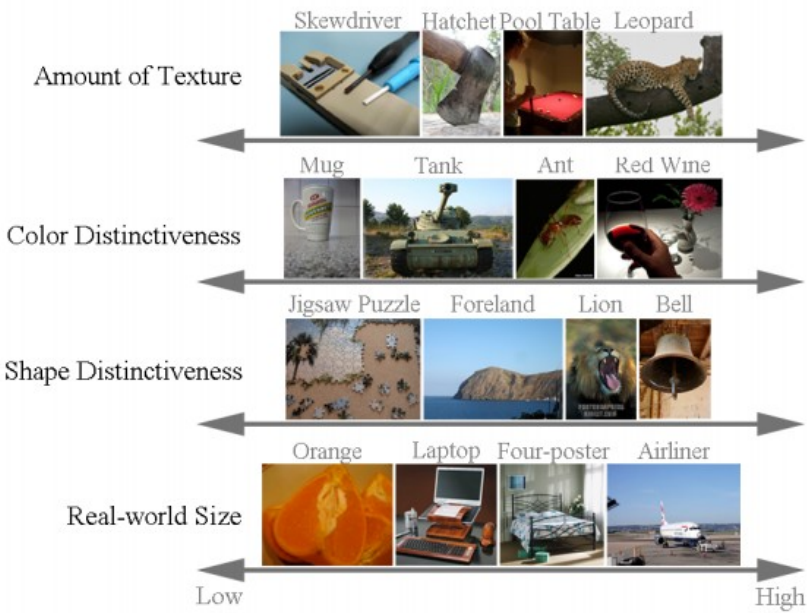
Question: When does CNN work well and when does it not?

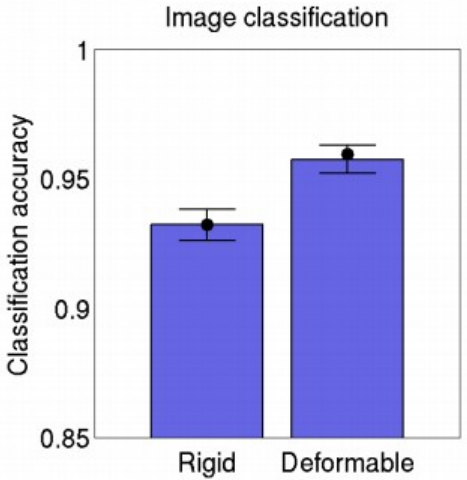
ImageNet (ILSVRC competition) analysis

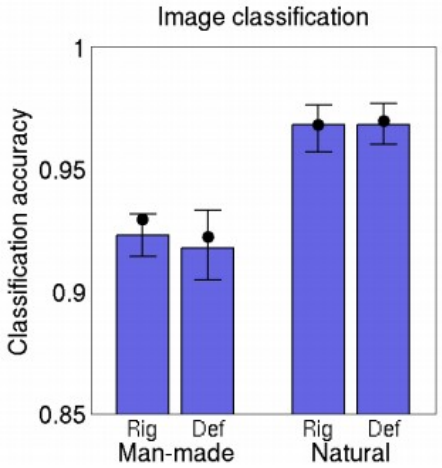
1. Detecting avocados to zucchinis: what have we done, and where are we going?
2. ImageNet Large Scale Visual Recognition Challenge

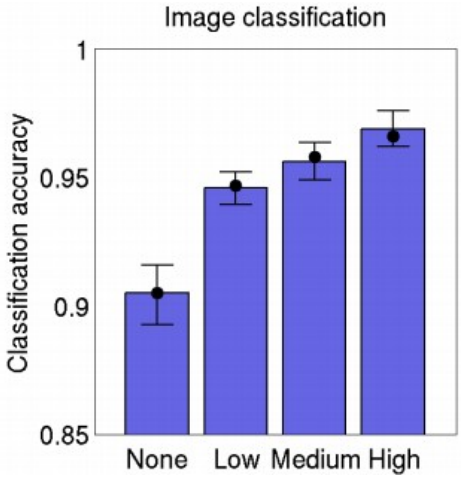
[Olga Russakovsky et al.]











(Amount of texture)

Image classification

Easiest classes

red fox (100) hen-of-the-woods (100) ibex (100) goldfinch (100) flat-coated retriever (100)

tiger (100) hamster (100) porcupine (100) stingray (100) Blenheim spaniel (100)

Hardest classes

muzzle (71) hatchet (68) water bottle (68) velvet (68) loupe (66)

hook (66) spotlight (66) ladle (65) restaurant (64) letter opener (59)

CNN vs. Human

[What I learned from competing against a ConvNet on ImageNet]
Karpathy, 2014: http://bit.ly/humanvsconvnet

Try it out yourself: http://cs.stanford.edu/people/karpathy/ilsvrc/


terrier
terrier
Australian terrier
soft-coated wheaten terrier
Dandie Dinmont, Dandie Dinmont terrier
Airedale, Airedale terrier

:’(

	GoogLeNet correct	GoogLeNet wrong
Human correct	1352/1500 	72/1500 • Objects very small or thin • Abstract representations • Image filters
Human wrong	46/1500 • Fine-grained recognition • Class unawareness • Insufficient training data	30/1500 • Multiple objects • Incorrect annotations

GoogLeNet: 6.7%
Team Human: 5.1% phew...



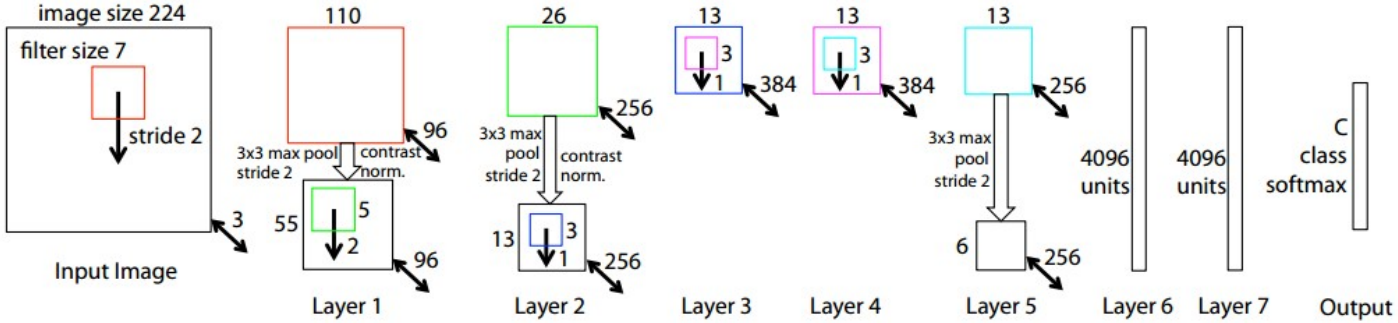



rule, ruler	king crab, Alaska crab	sidewinder	saltshaker, salt shaker	reel	hatchet	schipperke
pencil box, pencil case	pizza, pizza pie	maze, labyrinth	pill bottle	stethoscope	vase	schipperke
rubber eraser, rubber	strawberry	gar, garfish	water bottle	whistle	pitcher, ewer	groenendael
ballpoint, ballpoint pen	orange	valley, vale	lotion	ice lolly, lolly	coffeepot	doormat, welcome mat
pencil sharpener	fig	hammerhead	hair spray	hair spray	mask	teddy, teddy bear
carpenter's kit, tool kit	ice cream, icecream	sea snake	beer bottle	maypole	cup	jigsaw puzzle

What makes a ConvNet tick?

Understanding the source of ConvNet performance

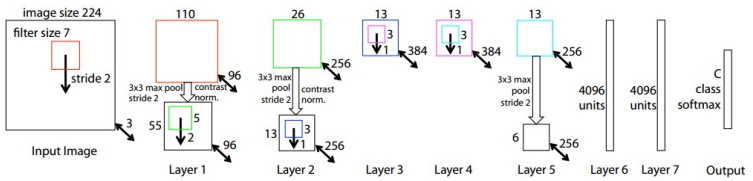
Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]



Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1

Understanding the source of ConvNet performance

Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]



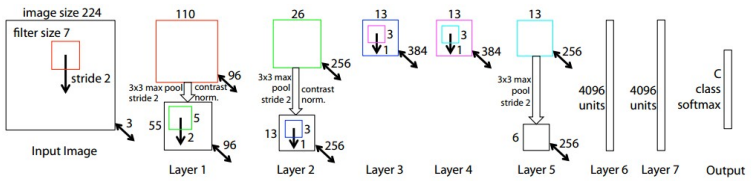
- Remove 2 FC layers (6,7): lose some small performance
- Remove 2 Conv layers (3,4): lose about equal performance
- Remove 2FC 2Conv (3,4,6,7): Very bad (71% error)

=> Depth is important

Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1

Understanding the source of ConvNet performance

Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]



- Remove 2 FC layers (6,7): lose some small performance
- Remove 2 Conv layers (3,4): lose about equal performance
- Remove 2FC 2Conv (3,4,6,7): Very bad (71% error)

=> Depth is important

- Changing size of FC layers: little to no improvement
- Changing size of Conv layers: reasonable improvement!

Error %	Train Top-1	Val Top-1	Val Top-5
Our replication of (Krizhevsky et al., 2012), 1 convnet	35.1	40.5	18.1
Removed layers 3,4	41.8	45.4	22.1
Removed layer 7	27.4	40.0	18.4
Removed layers 6,7	27.4	44.8	22.4
Removed layer 3,4,6,7	71.1	71.3	50.1
Adjust layers 6,7: 2048 units	40.3	41.7	18.8
Adjust layers 6,7: 8192 units	26.8	40.0	18.1
Our Model (as per Fig. 3)	33.1	38.4	16.5
Adjust layers 6,7: 2048 units	38.2	40.2	17.6
Adjust layers 6,7: 8192 units	22.0	38.8	17.0
Adjust layers 3,4,5: 512,1024,512 maps	18.8	**37.5**	**16.0**
Adjust layers 6,7: 8192 units and Layers 3,4,5: 512,1024,512 maps	**10.0**	38.3	16.9

Return of the Devil in the Details: Delving Deep into Convolutional Nets
[Chatfield et al. 2014]

(Finetuning on PASCAL VOC 2007)

						mAP (high = good)
(p) CNN M	–	(C)	f	s	4K	**79.89**
(x) CNN M 2048	–	(C)	f	s	2K	**80.10**
(y) CNN M 1024	–	(C)	f	s	1K	**79.91**
(z) CNN M 128	–	(C)	f	s	128	**78.60**

surprisingly small drop for 128 dimensions

number of neurons in last FC layer

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

ConvNet Configuration					
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**
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**
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Normalization doesn't do anything

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

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

ConvNet Configuration					
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**
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**
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**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

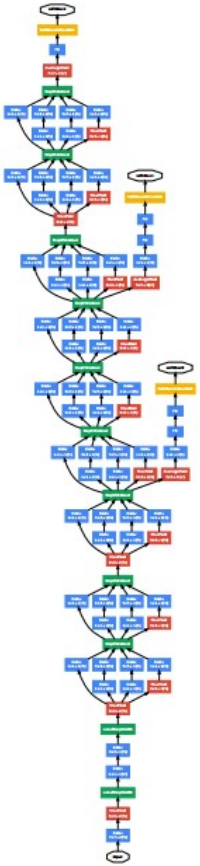
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

More depth = better performance

well almost, looks like D with 16 weight layers is a sweet spot

[Going deeper with convolutions, Szegedy et al., 2014]



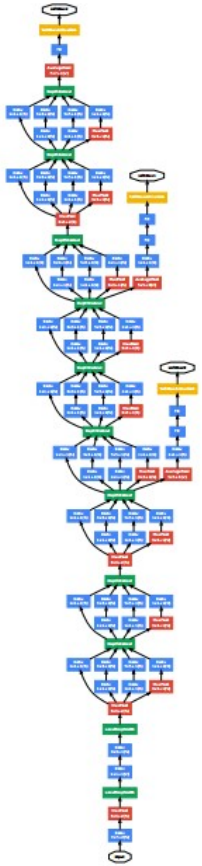
GoogLeNet

12x less params than Krizhevsky et al.

=> ~5M params

Q: How to reduce the number of parameters?

[Going deeper with convolutions, Szegedy et al., 2014]



GoogLeNet

12x less params than Krizhevsky et al.

=> ~5M params

Q: How to reduce the number of parameters?

A: Throw away the FC layers (only part of their answer (Inception modules))

After last pooling layer, volume is of size **[7x7x1024]**

Normally you would place the first 4096-D FC layer here (Many M params)

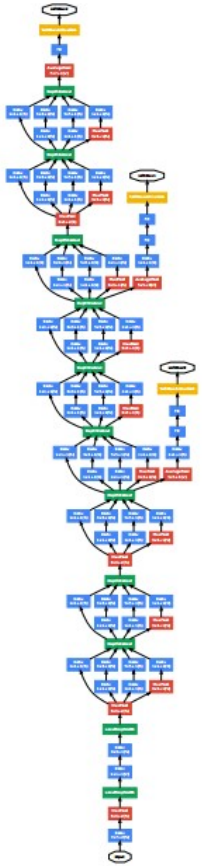
Instead: use Average pooling in each depth slice:

=> **[1x1x1024]**

performance actually improves 0.6% (less overfitting?)

[Going deeper with convolutions, Szegedy et al., 2014]

GoogLeNet



Fun feature: multiple Softmaxes along the way
(Very experimental!)

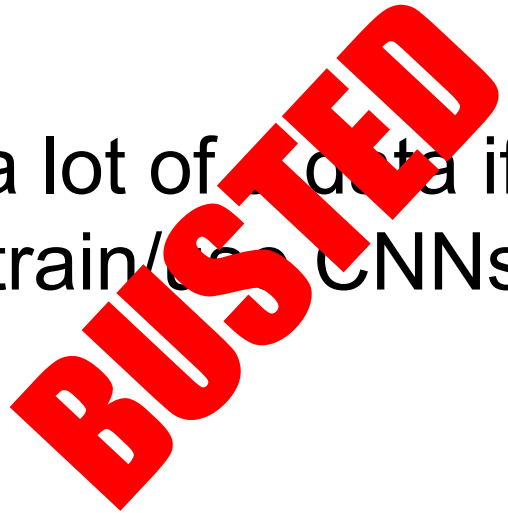
Summary: What makes ConvNets tick?

- depth
- small filter sizes
- Conv layers > FC layers

“You need a lot of a data if you want to train/use CNNs”

Transfer Learning

"You need a lot of data if you want to train/use CNNs"

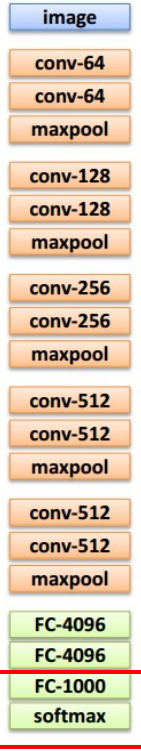


Transfer Learning with CNNs

1. Train on Imagenet

Transfer Learning with CNNs

1. Train on Imagenet



2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

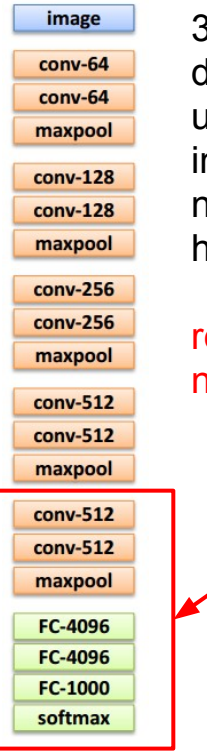
i.e. swap the Softmax layer at the end

Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end



3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

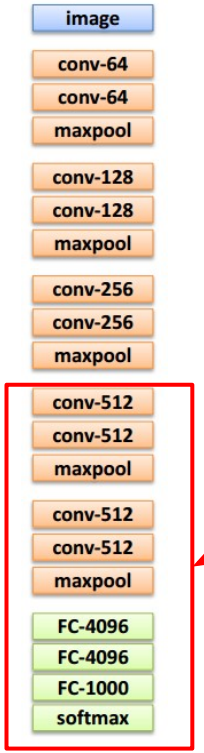
retrain bigger portion of the network, or even all of it.

Transfer Learning with CNNs

1. Train on Imagenet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

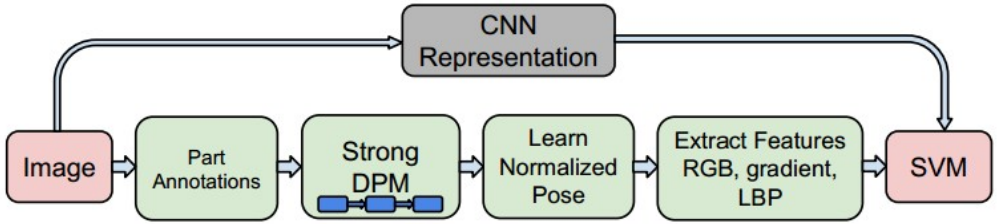


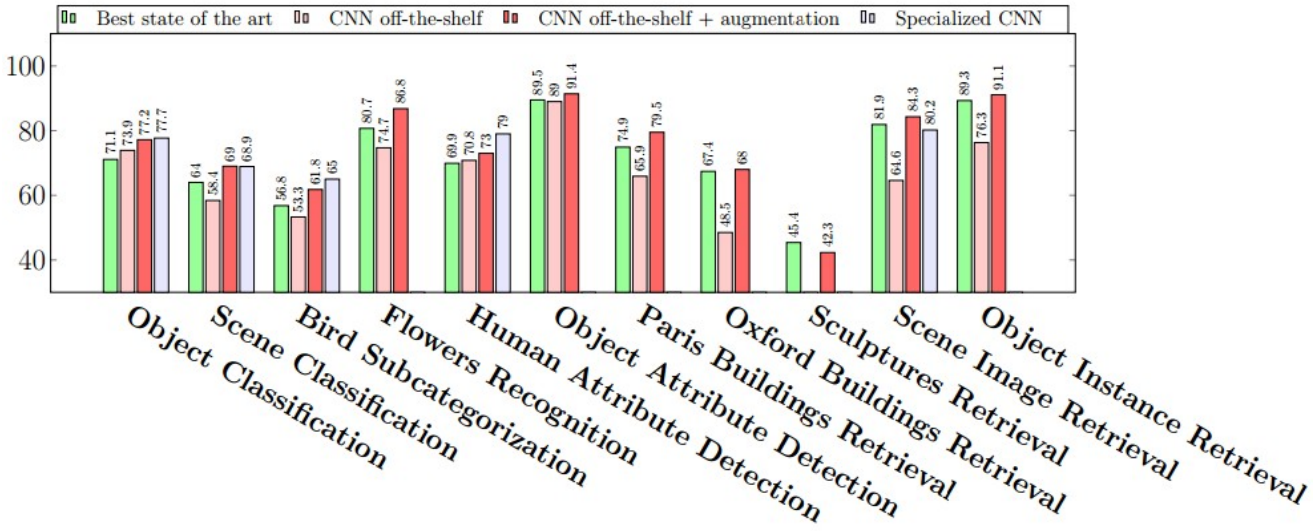
3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

retrain bigger portion of the network, or even all of it.

tip: use only ~1/10th of the original learning rate in finetuning to player, and ~1/100th on intermediate layers

CNN Features off-the-shelf: an Astounding Baseline for Recognition
[Razavian et al, 2014]

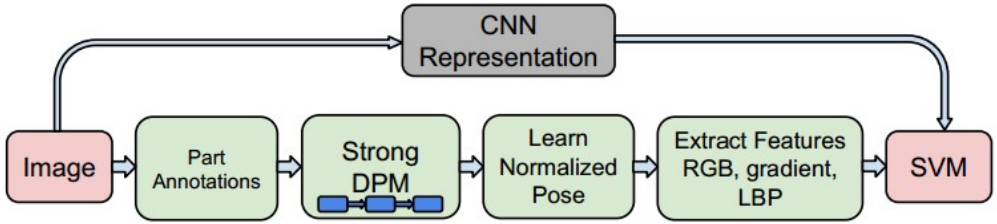


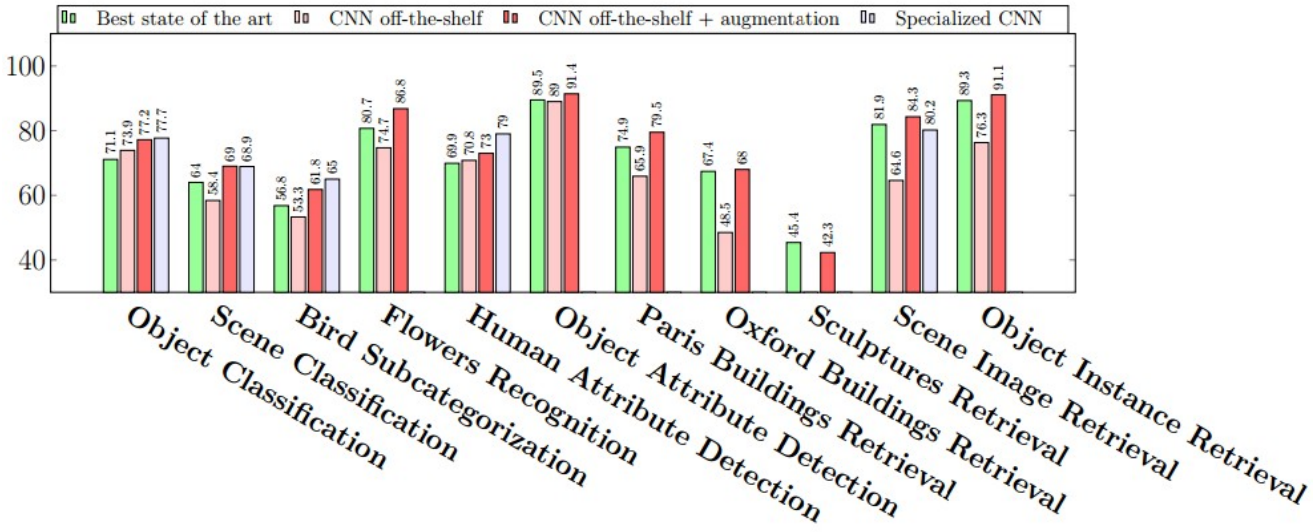


CNN Features off-the-shelf: an Astounding Baseline for Recognition [Razavian et al, 2014]

DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition [Donahue, Jia*, et al., 2013]*

	$DeCAF_6$	$DeCAF_7$
LogReg	**40.94 ± 0.3**	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao et al. (2010)	38.0	



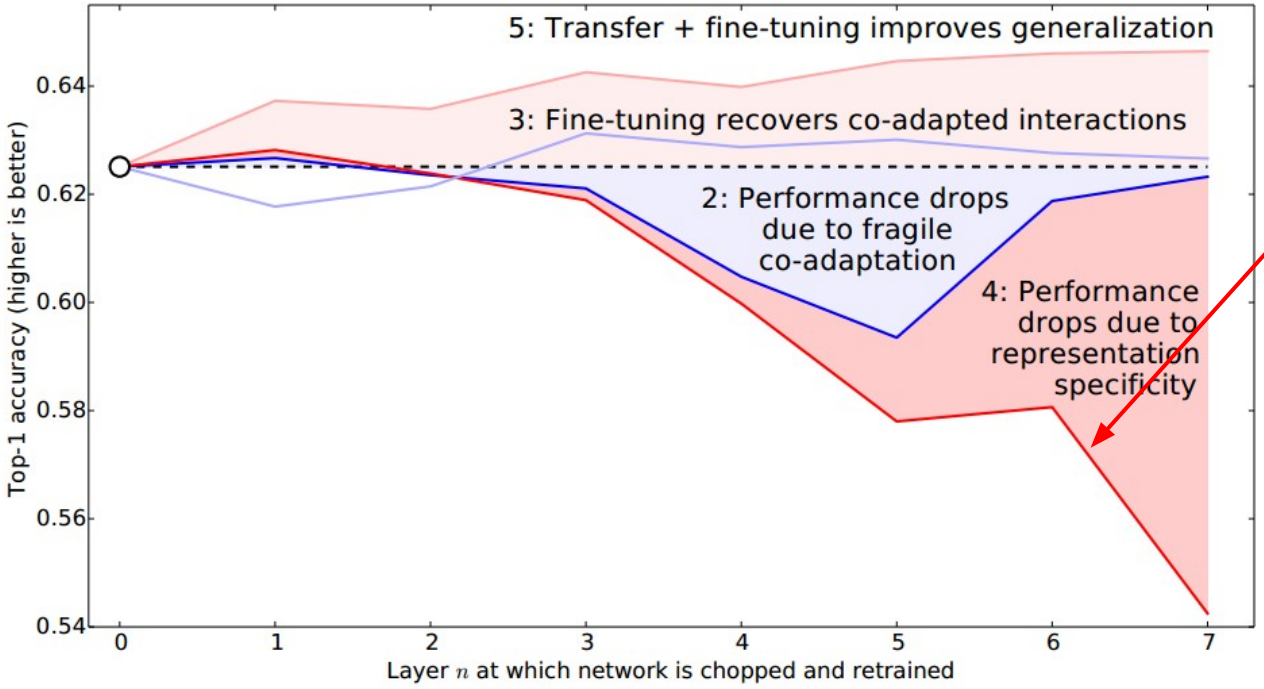


Visualizing and Understanding Convolutional Networks
[Zeiler and Fergus, 2013]

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

Q: If we were to use only one layer, which layer should we transfer from?

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

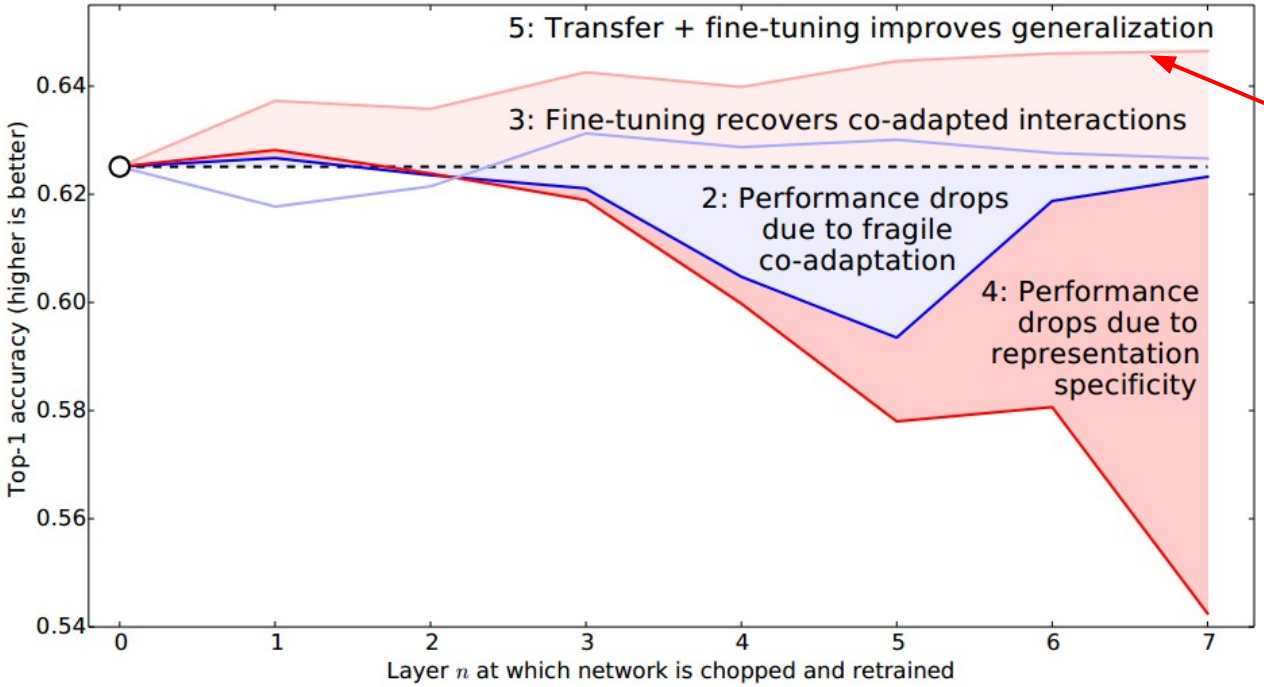


Split ImageNet classes in half to two sets: A/B.

Train on A, fix the first n layers, reinit layers n+, train on B, test on B val.

=> performance degrades because representation higher up is too A-specific

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

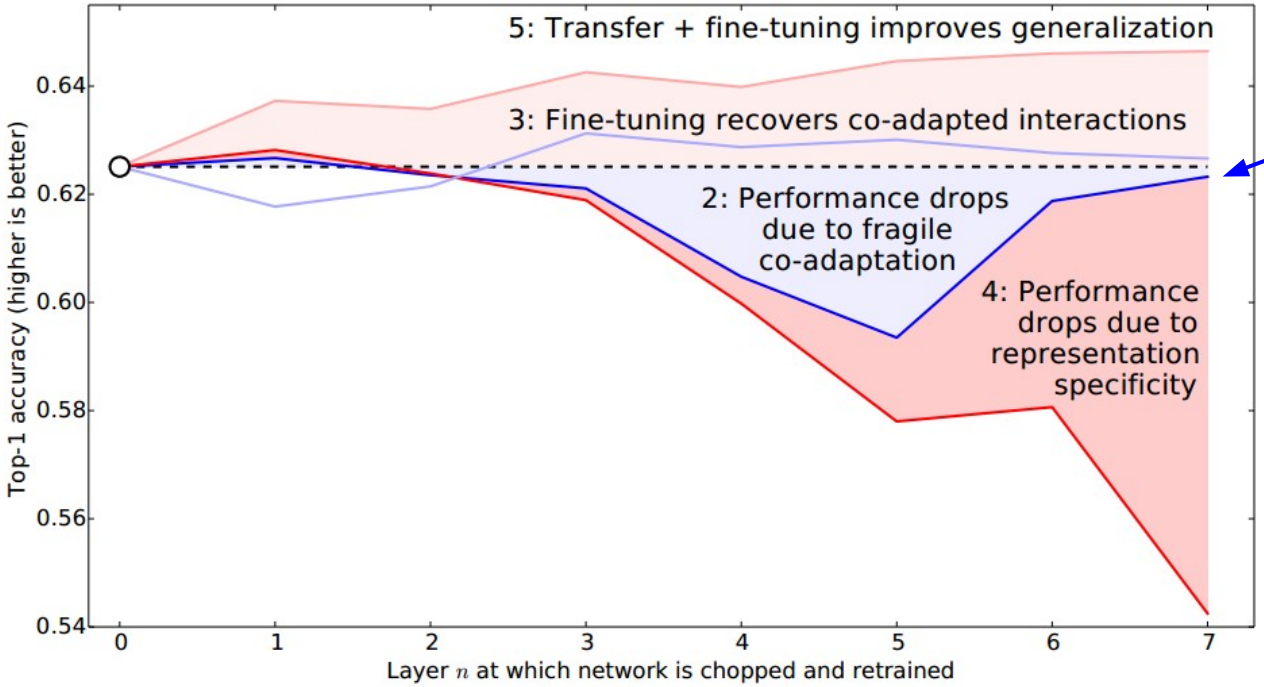


Split ImageNet classes in half to two sets: A/B.

Train on A, reinit layers n+, train on B, test on B val.

=> the information from once seeing data from A seems to linger, gives better generalization

How transferable are features in deep neural networks?
[Yosinski et al., 2014]

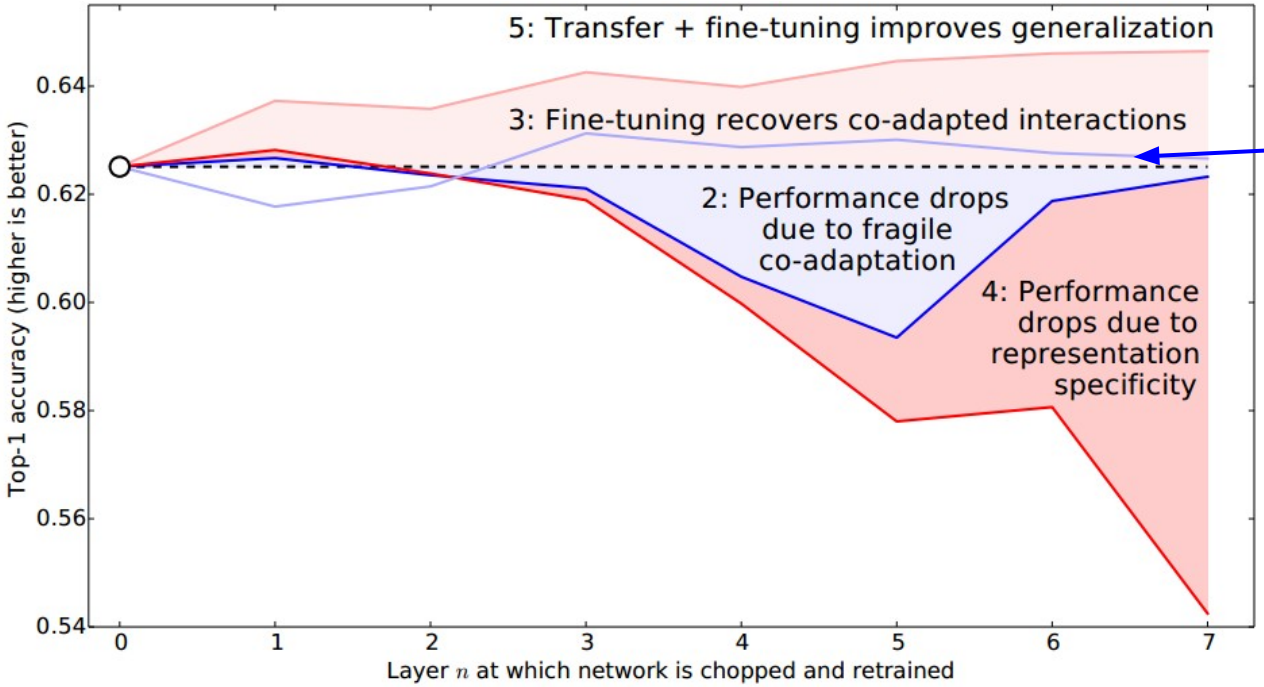


Split ImageNet classes in half to two sets: A/B.

Train on B, fix the first n layers, re-initialize layers n+, train on B again and test on B val

=> performance degrades when n = 4. wat.

How transferable are features in deep neural networks?
[Yosinski et al., 2014]



Split ImageNet classes in half to two sets: A/B.

Train on B, reinitialize layers n+, train on B again and test on B val.

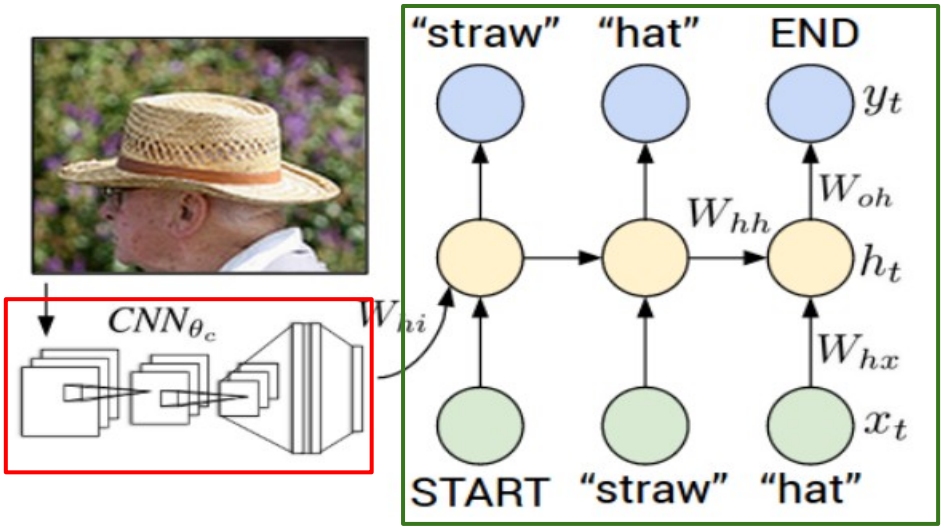
=> performance doesn't degrade anymore

	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?

	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?

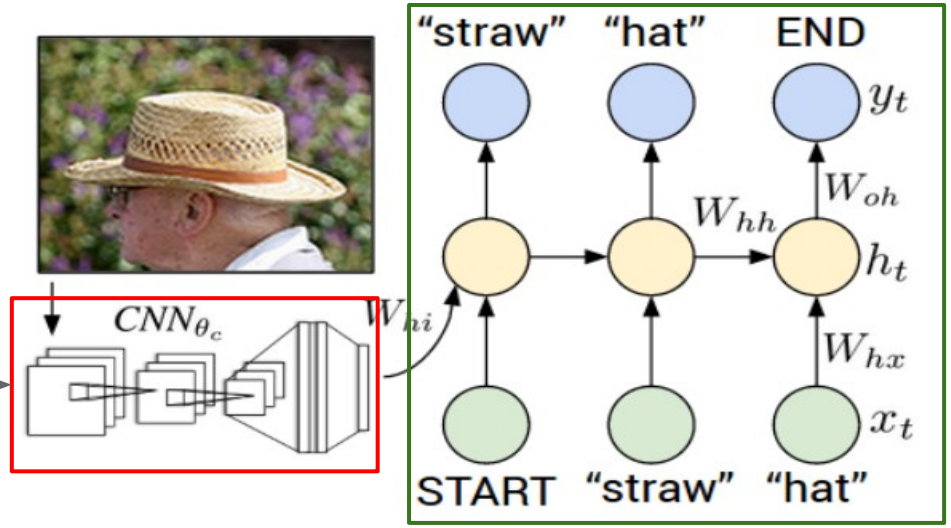
	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

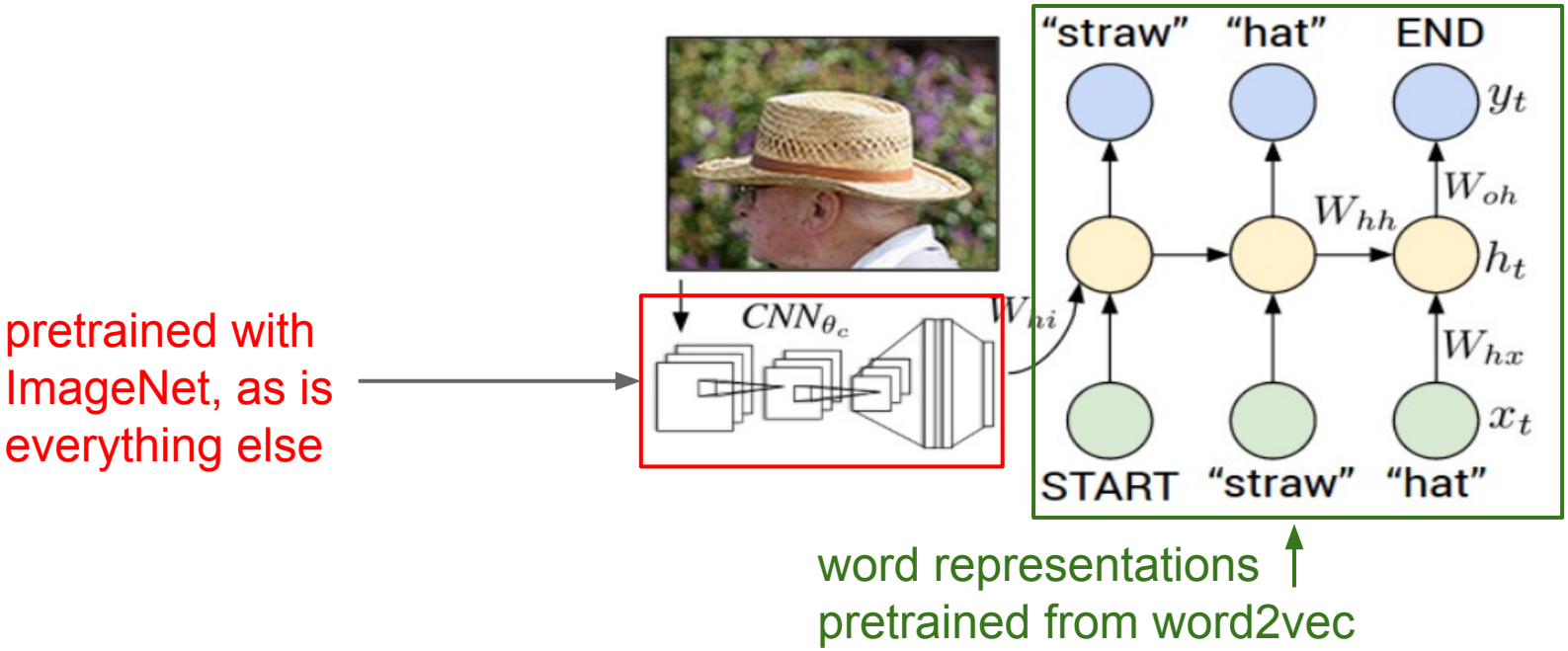


Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

pretrained with ImageNet, as is everything else



Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)



Takeaway for your projects/beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there.
2. Transfer learn to your dataset

Caffe ConvNet library has a **“Model Zoo”** of pretrained models:
https://github.com/BVLC/caffe/wiki/Model-Zoo

Summary:

- A lot of CNN's power comes from middle Conv layers, and depth seems to be important.
- Transfer Learning can be very helpful with small-medium data

Next Lecture:

CNN Tips/Tricks:
squeezing out the last few percent