Lecture 18
Transfer Learning and Computer Vision I

04 April 2016

Taylor B. Arnold
Yale Statistics
STAT 365/665

Notes:

- Problem set 6 is online and due **this** Friday, April 8th
- Problem set 7 is online and due **next** Friday, April 15th
- No class next week

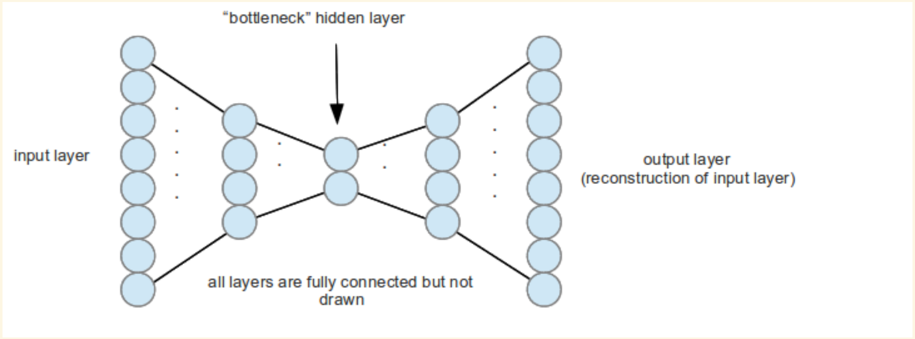
Autoencoders and Transfer Learning

An Autoencoder

An autoencoder is a learning algorithm that attempts to learn a compressed representation of its output. It is conceptually an unsupervised task, but one that neural networks are quite good at.

Neural network Autoencoder

To build a neural network auto-encoder, we simply build a neural network where the input is equal to the output. This is only meaningful when there exists a hidden layer with fewer nodes than the input layer (otherwise we can perfectly reconstruct the image):


"bottleneck" hidden layer
input layer
output layer
(reconstruction of input layer)
all layers are fully connected but not drawn

The botteneck layer

Notice that by storing the values in the hidden layer with the fewest nodes, we get a compression of the input into a lower dimensional space (other hidden layers also do this, but may do a smaller amount of compression).

Dimensionality reduction

We can reason that for many tasks, if this lower dimensional representation does a good job of reconstructing the image, there is enough information contained in that layer to also do learning tasks. If the degree of compression is high, this can aid in learning because the input dimensionality has been reduced.

Dimensionality reduction

We can reason that for many tasks, if this lower dimensional representation does a good job of reconstructing the image, there is enough information contained in that layer to also do learning tasks. If the degree of compression is high, this can aid in learning because the input dimensionality has been reduced.

This is exactly the same as computing principal components and then using the first few components to do regression on.

Dimensionality reduction

We can reason that for many tasks, if this lower dimensional representation does a good job of reconstructing the image, there is enough information contained in that layer to also do learning tasks. If the degree of compression is high, this can aid in learning because the input dimensionality has been reduced.

This is exactly the same as computing principal components and then using the first few components to do regression on.

It turns out, very helpfully, that the hidden layers of non-autoencoding networks perform a supervised dimensionality reduction. We can often think of the inner layers in the same way as the bottleneck layers for an autoencoder.

Autoencoder demo

A live demo of an autoencoder on the MNIST-10 dataset:

> *http://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html*

Transfer learning

Remember when we were predicting crimes on the City of Chicago dataset; many of you were quite concerned about why the following worked so well:

1. Train a model on the output Z, using the covariate matrix X
2. Use the predicted values, along with X, to learn a new output Y

Transfer learning

Remember when we were predicting crimes on the City of Chicago dataset; many of you were quite concerned about why the following worked so well:

1. Train a model on the output Z, using the covariate matrix X
2. Use the predicted values, along with X, to learn a new output Y

We can do this with neural networks in a more natural way by:

1. Train a neural network on an output Z, using the inputs X
2. Remove the learned output layer from this network, and attach another output layer to capture a new output Y
3. Train this new model, either using the weights from the first step as a starting point (pre-training) or freezing them (transfer learning)

Does this process make more intuitive sense now that we can see the first step as supervised dimensionality reduction?

Why is this all important?

In order to get state-of-the-art results using neural networks, we need to train very large and deep models. This requires a lot of data and computing power, both of which most users do not have. To circumvent this, some combination of pre-training or transfer learning is used instead.

This is largely why understanding the recent history of computer vision models is so important; most computer visions tasks will require one to use and modify these models in some form.

Autoencoding, Transfer Learning, and Pre-training

Good references for these topics are given by

Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., and Bergstra, J. (2011). Unsupervised and transfer learning challenge: a deep learning approach.

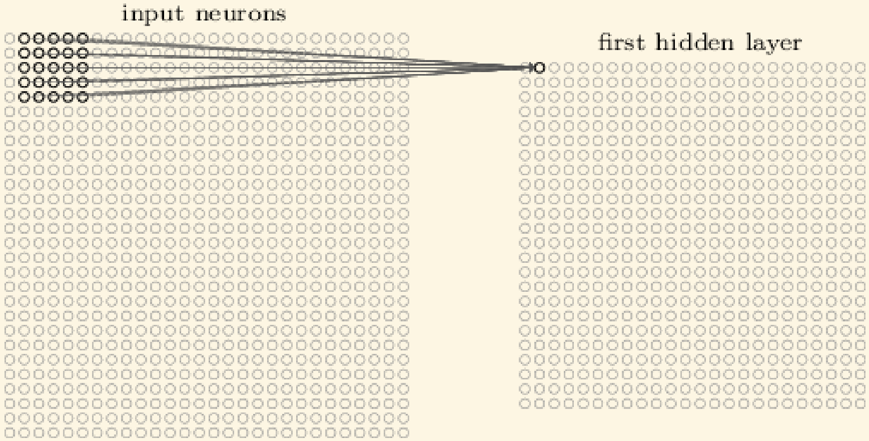
And,

Long, Jonathan L., Ning Zhang, and Trevor Darrell. "Do Convnets Learn Correspondence?." Advances in Neural Information Processing Systems. 2014.

Computer Vision I

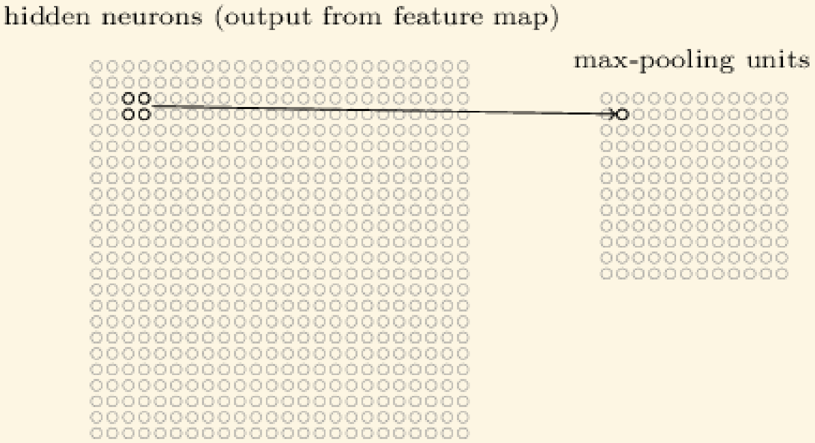
CNN Review

A 5x5 kernel with a stride of 1:



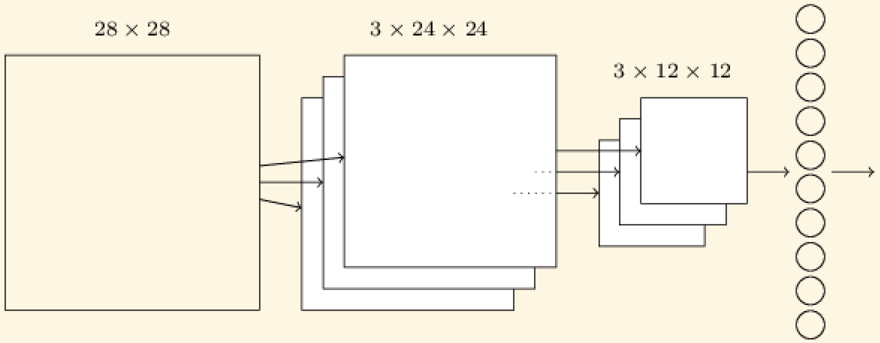
CNN Review

Max pooling with 2x2 kernel and stride of 2:



CNN Review

A complete CNN model:



Convolutional Models in Computer Vision

There is a long history of specific advances and uses of convolutional neural networks. Today, I'll focus on the following set of models:

- LeNet-5 (1998)
- AlexNet (2012)
- OverFeat (2013)
- VGG-16, VGG-19 (2014)
- GoogLeNet (2014)
- PReLUnet (2015)
- ResNet-50, ResNet-101, ResNet-152 (2015)
- SqueezeNet (2016)
- Stochastic Depth (2016)
- ResNet-200, ResNet-1001 (2016)

When you hear about these models people may be referring to: the architecture, the architecture and weights, or just to the general approach.

LeNet-5 (1998)

LeNet was one of first models to really show the power of convolutional neural networks. It was first applied to the MNIST-10 dataset, created by a similar group of individuals:

> *LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.*

PROC. OF THE IEEE, NOVEMBER 1998 7

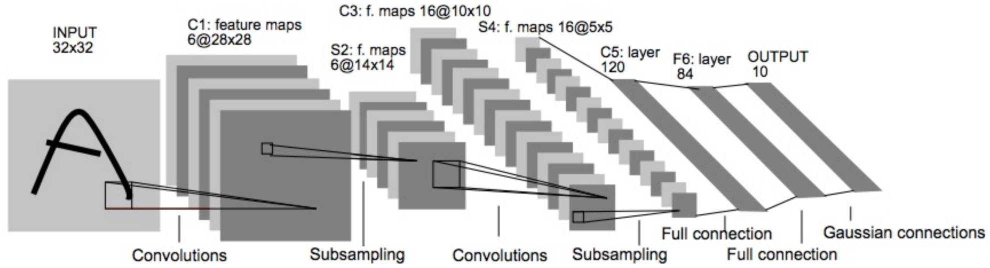
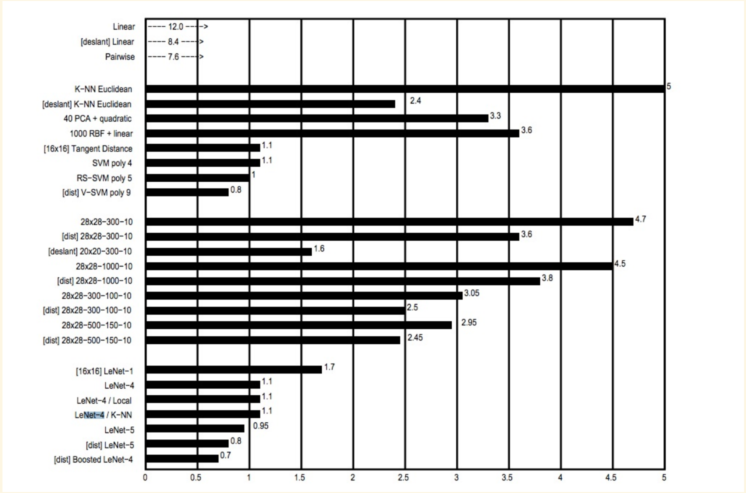


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.


Linear
---- 12.0 ---->
[deslant] Linear
---- 8.4 ---->
Pairwise
---- 7.6 ---->
K−NN Euclidean
5
[deslant] K−NN Euclidean
2.4
40 PCA + quadratic
3.3
1000 RBF + linear
3.6
[16x16] Tangent Distance
1.1
SVM poly 4
1.1
RS−SVM poly 5
1
[dist] V−SVM poly 9
0.8
28x28−300−10
4.7
[dist] 28x28−300−10
3.6
[deslant] 20x20−300−10
1.6
28x28−1000−10
4.5
[dist] 28x28−1000−10
3.8
28x28−300−100−10
3.05
[dist] 28x28−300−100−10
2.5
28x28−500−150−10
2.95
[dist] 28x28−500−150−10
2.45
[16x16] LeNet−1
1.7
LeNet−4
1.1
LeNet−4 / Local
1.1
LeNet−4 / K−NN
1.1
LeNet−5
0.95
[dist] LeNet−5
0.8
[dist] Boosted LeNet−4
0.7
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Python demo I: LeNet-5 for MNIST10

Python demo II: LeNet-5 with “Distortions” (i.e., Data augmentation)

ImageNet Large Scale Visual Recognition Challenges

ImageNet Overview (through 2014)

An excellent summery paper of the ILSVCR challeneg, that describes in more detail than I will here, is given by:

> *Olga Russakovsky, et al. "ImageNet Large Scale Visual Recognition Challenge". arXiv preprint arXiv:1409.0575v3 (2015).*

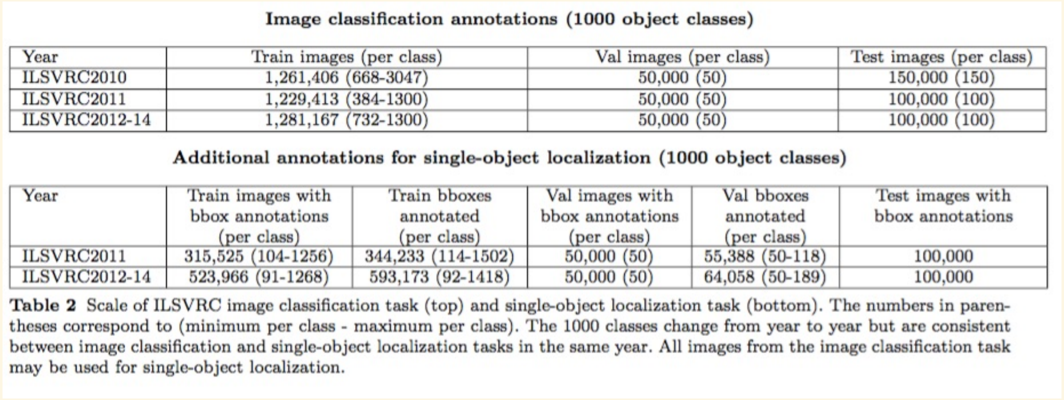


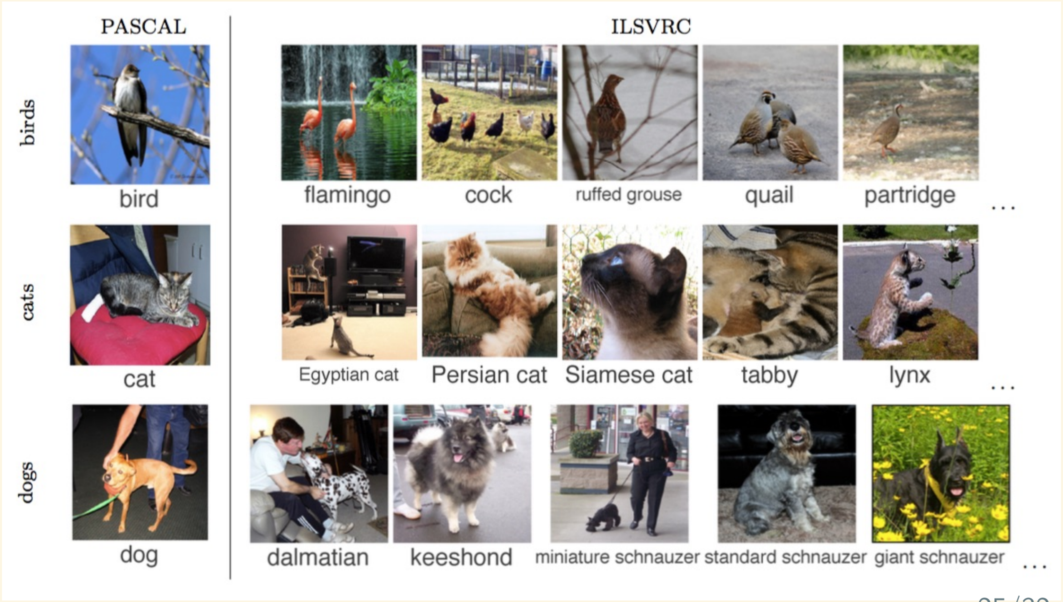
Image classification annotations (1000 object classes)

Year	Train images (per class)	Val images (per class)	Test images (per class)
ILSVRC2010	1,261,406 (668-3047)	50,000 (50)	150,000 (150)
ILSVRC2011	1,229,413 (384-1300)	50,000 (50)	100,000 (100)
ILSVRC2012-14	1,281,167 (732-1300)	50,000 (50)	100,000 (100)

Additional annotations for single-object localization (1000 object classes)

Year	Train images with bbox annotations (per class)	Train bboxes annotated (per class)	Val images with bbox annotations (per class)	Val bboxes annotated (per class)	Test images with bbox annotations
ILSVRC2011	315,525 (104-1256)	344,233 (114-1502)	50,000 (50)	55,388 (50-118)	100,000
ILSVRC2012-14	523,966 (91-1268)	593,173 (92-1418)	50,000 (50)	64,058 (50-189)	100,000

Table 2 Scale of ILSVRC image classification task (top) and single-object localization task (bottom). The numbers in parentheses correspond to (minimum per class - maximum per class). The 1000 classes change from year to year but are consistent between image classification and single-object localization tasks in the same year. All images from the image classification task may be used for single-object localization.


PASCAL
ILSVRC
birds
cats
dogs
bird
flamingo
cock
ruffed grouse
quail
partridge
...
cat
Egyptian cat
Persian cat
Siamese cat
tabby
lynx
...
dog
dalmatian
keeshond
miniature schnauzer
standard schnauzer
giant schnauzer
...

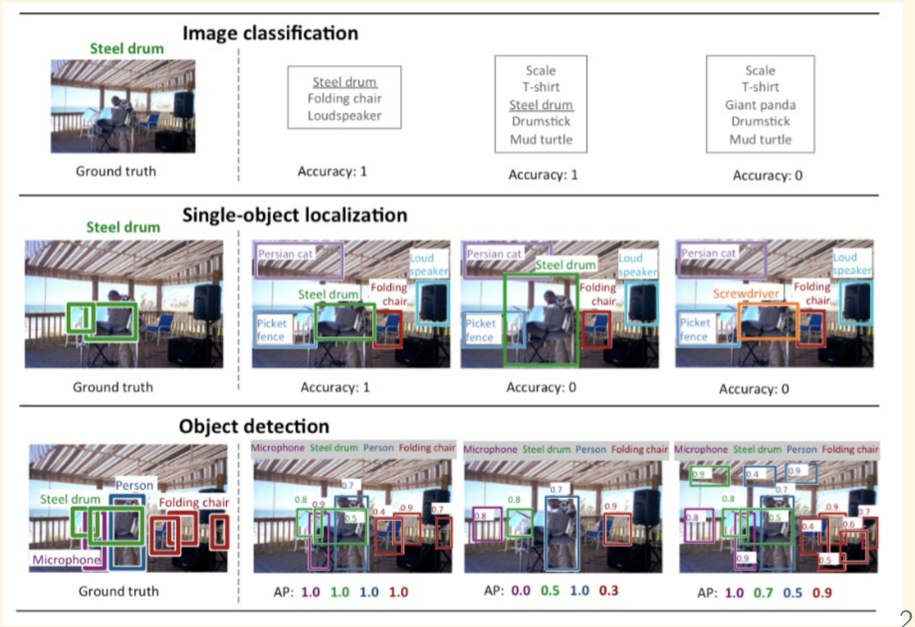


Image classification
Steel drum
Ground truth
Steel drum
Folding chair
Loudspeaker
Accuracy: 1
Scale
T-shirt
Steel drum
Drumstick
Mud turtle
Accuracy: 1
Scale
T-shirt
Giant panda
Drumstick
Mud turtle
Accuracy: 0
Single-object localization
Steel drum
Ground truth
Persian cat
Loud speaker
Steel drum
Folding chair
Picket fence
Accuracy: 1
Persian cat
Steel drum
Loud speaker
Folding chair
Picket fence
Accuracy: 0
Persian cat
Loud speaker
Screwdriver
Folding chair
Picket fence
Accuracy: 0
Object detection
Person
Steel drum
Folding chair
Microphone
Ground truth
Microphone Steel drum Person Folding chair
AP: 1.0 1.0 1.0 1.0
Microphone Steel drum Person Folding chair
AP: 0.0 0.5 1.0 0.3
Microphone Steel drum Person Folding chair
AP: 1.0 0.7 0.5 0.9

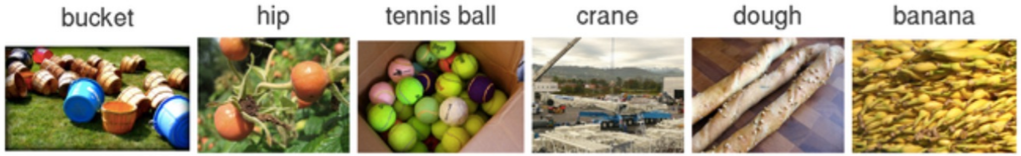


Fig. 8 Images marked as "difficult" in the ILSVRC2012 single-object localization validation set. Please refer to Section 4.2 for details.

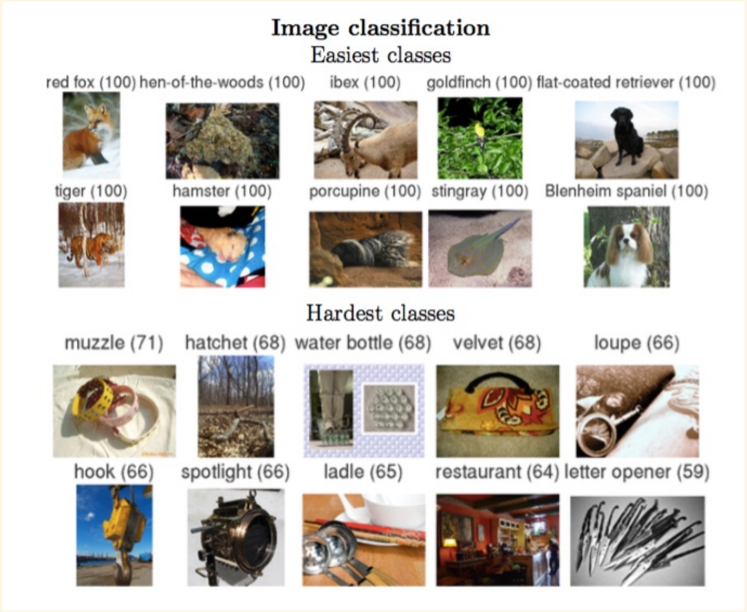
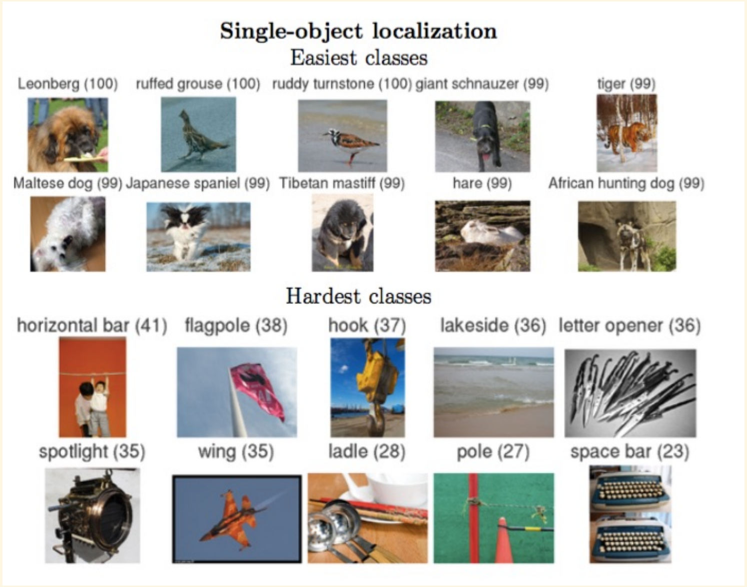
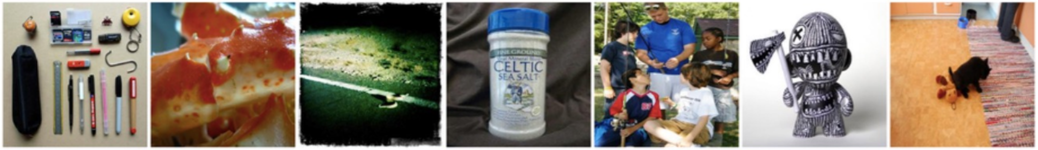


Image classification
Easiest classes
red fox (100)
hen-of-the-woods (100)
ibex (100)
goldfinch (100)
flat-coated retriever (100)
tiger (100)
hamster (100)
porcupine (100)
stingray (100)
Blenheim spaniel (100)
Hardest classes
muzzle (71)
hatchet (68)
water bottle (68)
velvet (68)
loupe (66)
hook (66)
spotlight (66)
ladle (65)
restaurant (64)
letter opener (59)


Single-object localization
Easiest classes
Leonberg (100)
ruffed grouse (100)
ruddy turnstone (100)
giant schnauzer (99)
tiger (99)
Maltese dog (99)
Japanese spaniel (99)
Tibetan mastiff (99)
hare (99)
African hunting dog (99)
Hardest classes
horizontal bar (41)
flagpole (38)
hook (37)
lakeside (36)
letter opener (36)
spotlight (35)
wing (35)
ladle (28)
pole (27)
space bar (23)



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

Fig. 15 Representative validation images that highlight common sources of error. For each image, we display the ground truth in blue, and top 5 predictions from GoogLeNet follow (red = wrong, green = right). GoogLeNet predictions on the validation set images were graciously provided by members of the GoogLeNet team. From left to right: Images that contain multiple objects, images of extreme closeups and uncharacteristic views, images with filters, images that significantly benefit from the ability to read text, images that contain very small and thin objects, images with abstract representations, and example of a fine-grained image that GoogLeNet correctly identifies but a human would have significant difficulty with.

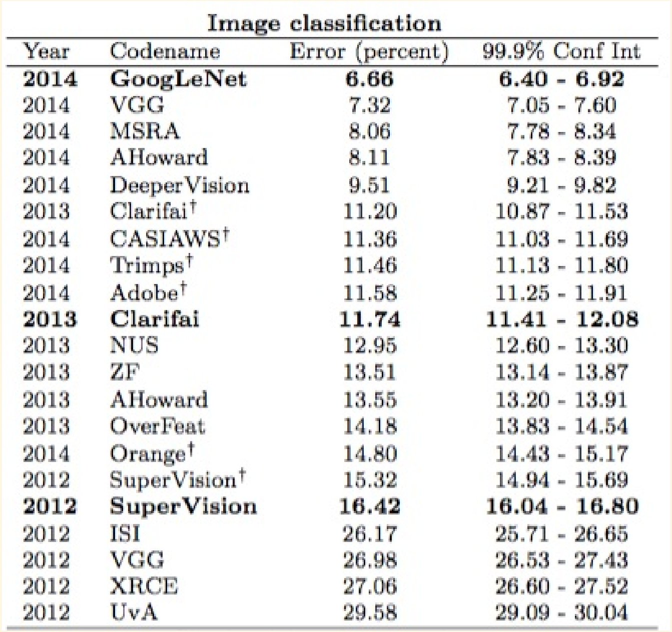
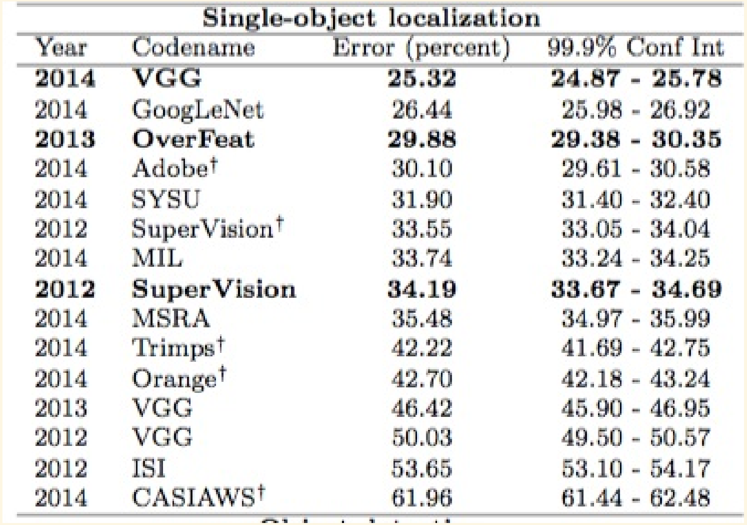


Image classification

Year	Codename	Error (percent)	99.9% Conf Int
2014	**GoogLeNet**	**6.66**	**6.40 - 6.92**
2014	VGG	7.32	7.05 - 7.60
2014	MSRA	8.06	7.78 - 8.34
2014	AHoward	8.11	7.83 - 8.39
2014	DeeperVision	9.51	9.21 - 9.82
2013	Clarifai†	11.20	10.87 - 11.53
2014	CASIAWS†	11.36	11.03 - 11.69
2014	Trimps†	11.46	11.13 - 11.80
2014	Adobe†	11.58	11.25 - 11.91
2013	**Clarifai**	**11.74**	**11.41 - 12.08**
2013	NUS	12.95	12.60 - 13.30
2013	ZF	13.51	13.14 - 13.87
2013	AHoward	13.55	13.20 - 13.91
2013	OverFeat	14.18	13.83 - 14.54
2014	Orange†	14.80	14.43 - 15.17
2012	SuperVision†	15.32	14.94 - 15.69
2012	**SuperVision**	**16.42**	**16.04 - 16.80**
2012	ISI	26.17	25.71 - 26.65
2012	VGG	26.98	26.53 - 27.43
2012	XRCE	27.06	26.60 - 27.52
2012	UvA	29.58	29.09 - 30.04



Single-object localization			
Year	Codename	Error (percent)	99.9% Conf Int
2014	**VGG**	**25.32**	**24.87 - 25.78**
2014	GoogLeNet	26.44	25.98 - 26.92
2013	**OverFeat**	**29.88**	**29.38 - 30.35**
2014	Adobe†	30.10	29.61 - 30.58
2014	SYSU	31.90	31.40 - 32.40
2012	SuperVision†	33.55	33.05 - 34.04
2014	MIL	33.74	33.24 - 34.25
2012	**SuperVision**	**34.19**	**33.67 - 34.69**
2014	MSRA	35.48	34.97 - 35.99
2014	Trimps†	42.22	41.69 - 42.75
2014	Orange†	42.70	42.18 - 43.24
2013	VGG	46.42	45.90 - 46.95
2012	VGG	50.03	49.50 - 50.57
2012	ISI	53.65	53.10 - 54.17
2014	CASIAWS†	61.96	61.44 - 62.48