Lecture 19
Computer Vision II

06 April 2016

Taylor B. Arnold
Yale Statistics
STAT 365/665

Yale

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.

AlexNet (2012)

A model out of the University of Toronto, now known as AlexNet, became the first CNN to produce state-of-the-art classification rates on the ILSVRC-2012 dataset:

> *Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.*

AlexNet contributions

AlexNet was the first to put together several key advances, all of which we have already used or discussed in this class:

1. relu units
2. dropout
3. data augmentation
4. multiple GPUs

While not all invented by the AlexNet group, they were the first to put them all together and figure out how to train a deep neural network.

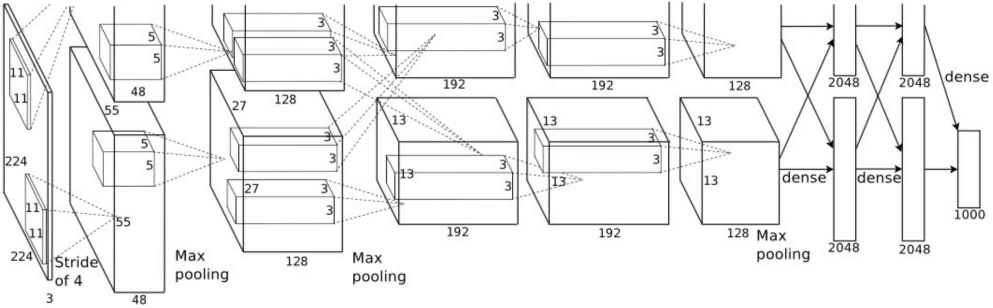


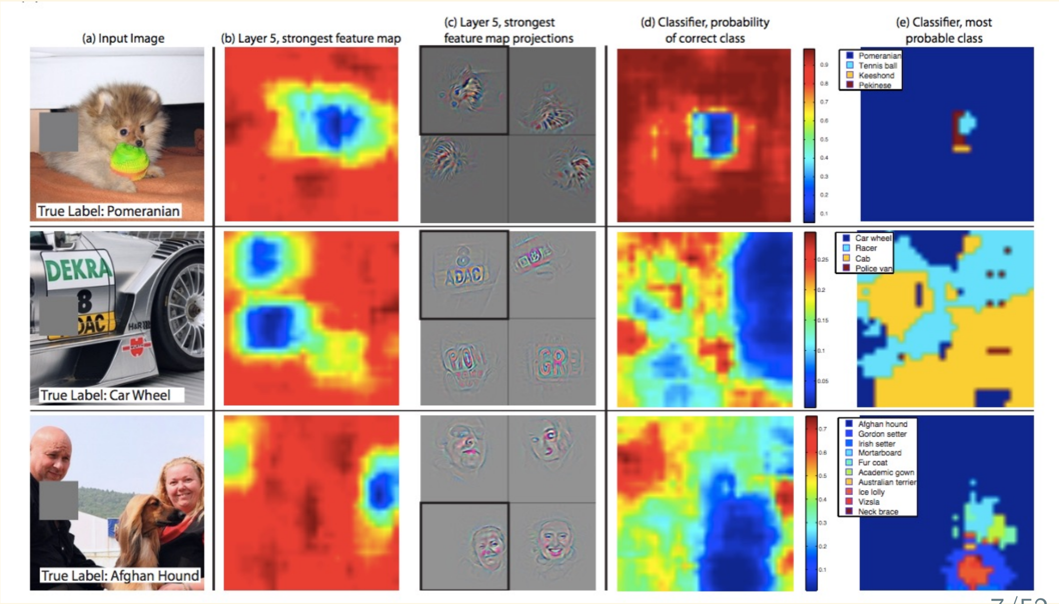
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Visualizing CNNs (2013)

Following the success of AlexNet, the year 2013 saw a much larger number of neural network entrants into the ILSVRC competition. The winning entry came about due to the visualization techniques described in the following paper:

> *Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." Computer vision–ECCV 2014. Springer International Publishing, 2014. 818-833.*

Their incredibly diverse set of techniques allowed the team to tweak the AlexNet architecture to get even better results.


(a) Input Image
(b) Layer 5, strongest feature map
(c) Layer 5, strongest feature map projections
(d) Classifier, probability of correct class
(e) Classifier, most probable class
True Label: Pomeranian
Pomeranian
Tennis ball
Keeshond
Pekinese
True Label: Car Wheel
Car wheel
Racer
Cab
Police van
True Label: Afghan Hound
Afghan hound
Gordon setter
Irish setter
Mortarboard
Fur coat
Academic gown
Australian terrier
Ice lolly
Vizsla
Neck brace

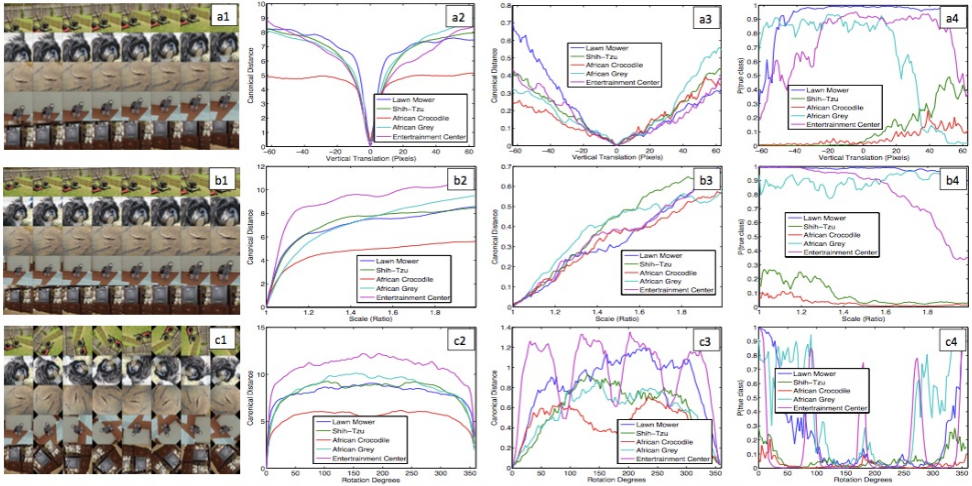


Figure 5. Analysis of vertical translation, scale, and rotation invariance within the model (rows a-c respectively). Col 1: 5 example images undergoing the transformations. Col 2 & 3: Euclidean distance between feature vectors from the original and transformed images in layers 1 and 7 respectively. Col 4: the probability of the true label for each image, as the image is transformed.

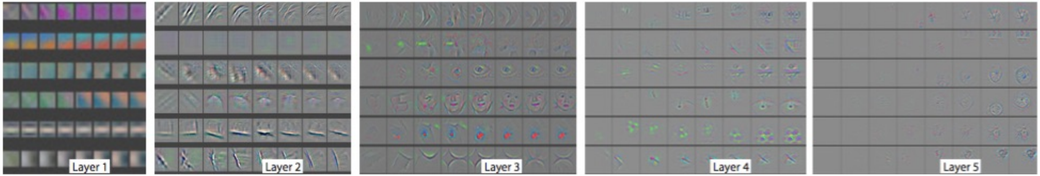


Figure 4. Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, we show a randomly chosen subset of features at epochs [1,2,5,10,20,30,40,64]. The visualization shows the strongest activation (across all training examples) for a given feature map, projected down to pixel space using our deconvnet approach. Color contrast is artificially enhanced and the figure is best viewed in electronic form.

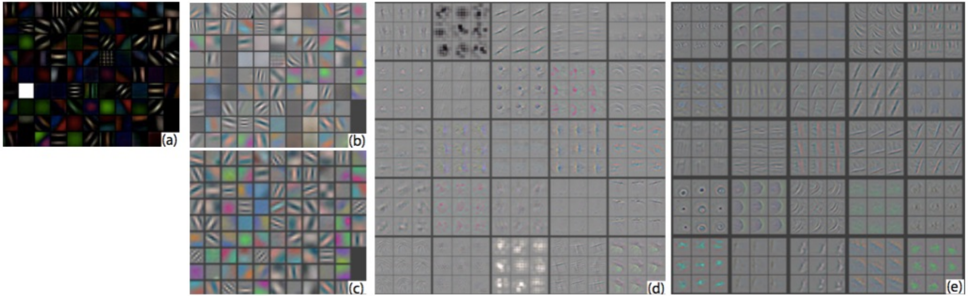


Figure 6. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from (Krizhevsky et al., 2012). (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from (Krizhevsky et al., 2012). (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

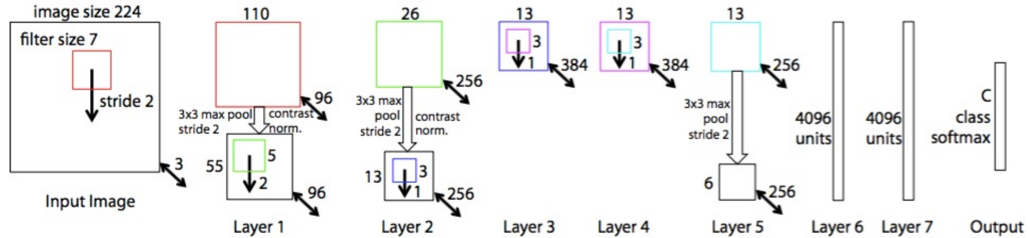


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

A demo of applying these techniques to the MNIST dataset with ConvNetJS:

`http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html`

OverFeat (2013)

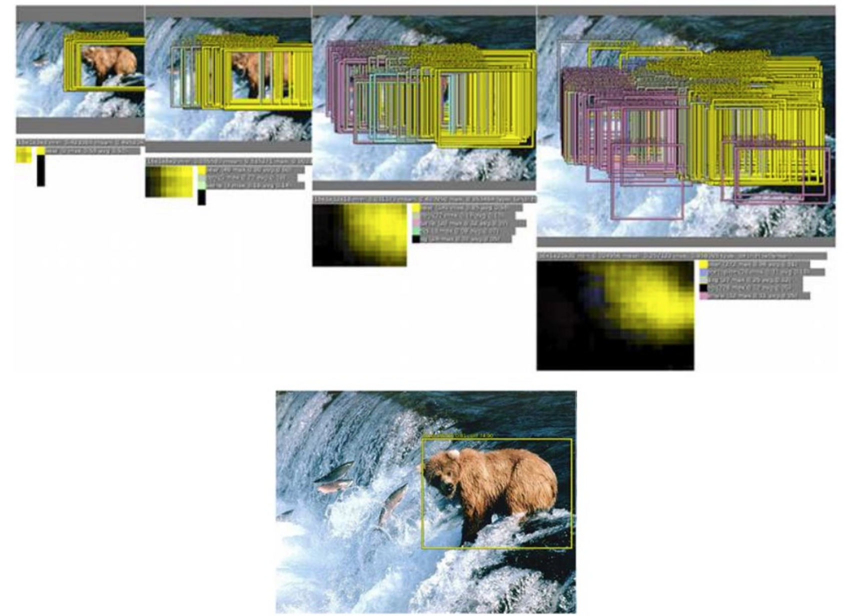
The 2013 competition also brought about the incredibly influential OverFeat model from a team based at NYU:

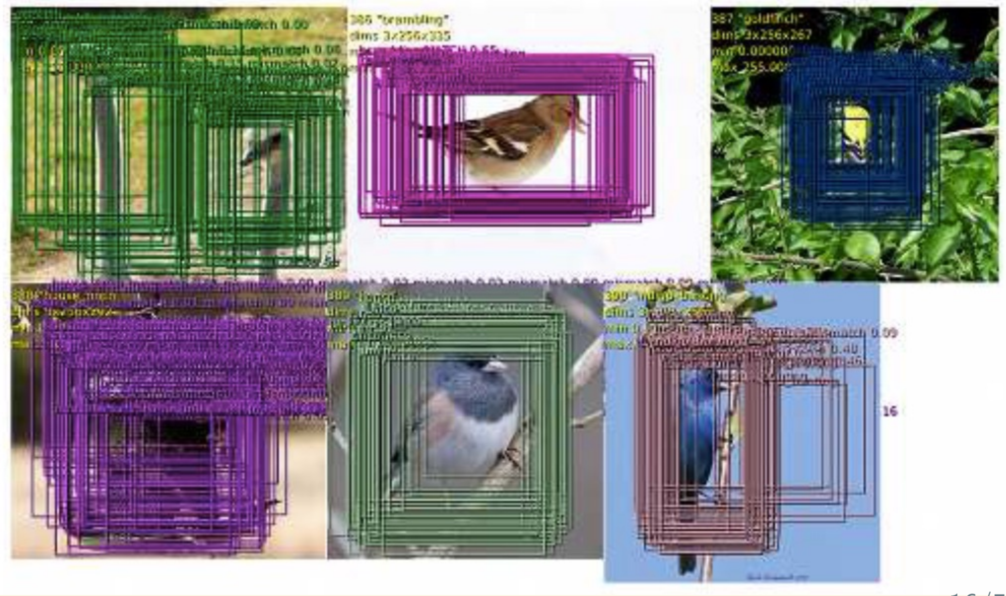
> *Sermanet, Pierre, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." arXiv preprint arXiv:1312.6229 (2013).*

The won the image localization task, by trying to solve localization and identification in a unified process. I'll give a very simplified version of what they did (the paper is a great read, and I suggest working through it if you are interested in computer vision).

Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

Table 1: **Architecture specifics for *fast* model.** The spatial size of the feature maps depends on the input image size, which varies during our inference step (see Table 5 in the Appendix). Here we show training spatial sizes. Layer 5 is the top convolutional layer. Subsequent layers are fully connected, and applied in sliding window fashion at test time. The fully-connected layers can also be seen as 1x1 convolutions in a spatial setting. Similar sizes for *accurate* model can be found in the Appendix.




385 "brambling"
dims 3x256x335
387 "goldfinch"
dims 3x256x267

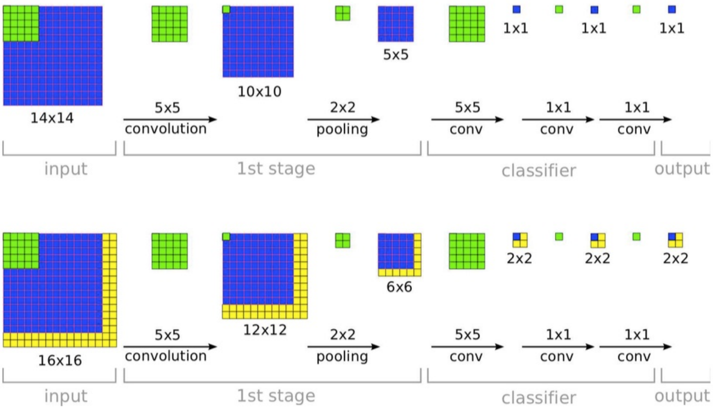


Figure 5: **The efficiency of ConvNets for detection.** During training, a ConvNet produces only a single spatial output (top). But when applied at test time over a larger image, it produces a spatial output map, e.g. 2x2 (bottom). Since all layers are applied convolutionally, the extra computation required for the larger image is limited to the yellow regions. This diagram omits the feature dimension for simplicity.

Python demo I: OverFeat adaptation of AlexNet (2012)

VGG-16, VGG-19 (2014)

One of the top entries from 2014, by an Oxford-based team, took advantage of significantly deeper models.

> *Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).*

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

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 **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					

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

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

Python demo II: Pre-trained VGG-19 Model

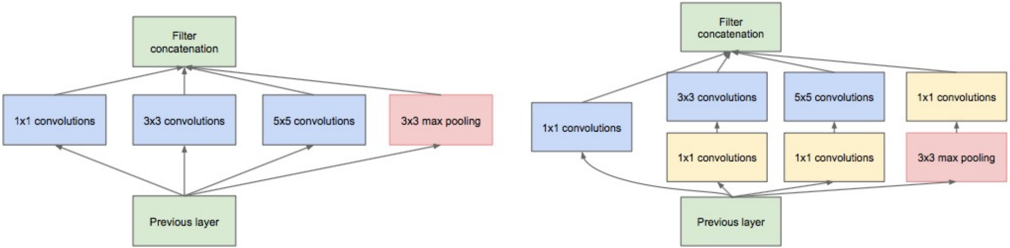
GoogLeNet (2014)

The winning entry from 2014, by Google, also took advantage of much deeper architectures:

> *Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9. 2015.*

They called their model GoogLeNet in honor of the original LeNet architecture.


WE NEED TO GO
DEEPER
memegenerator.net

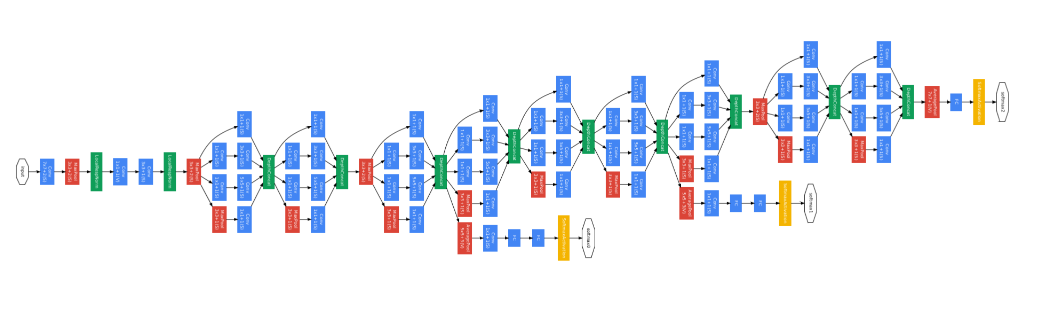


(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture



Python demo III: GoogLeNet - Inception Module

Relative Confusion	**A1**	**A2**
Human succeeds, GoogLeNet succeeds	1352	219
Human succeeds, GoogLeNet fails	72	8
Human fails, GoogLeNet succeeds	46	24
Human fails, GoogLeNet fails	30	7
Total number of images	1500	258
Estimated GoogLeNet classification error	6.8%	5.8%
Estimated human classification error	5.1%	12.0%

Table 9 Human classification results on the ILSVRC2012-2014 classification test set, for two expert annotators A1 and A2. We report top-5 classification error.

Batch Normalization (2015)

Not a model architecture itself, but one very useful new tweak in the past year has been Batch Normalization, first presented in this paper:

> *Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167*

Python demo IV: Batch normalization

PReLUnet (2015)

Microsoft's first contribution in 2015 was the idea of using a modifed ReLU activation function:

> *He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE International Conference on Computer Vision. 2015.*

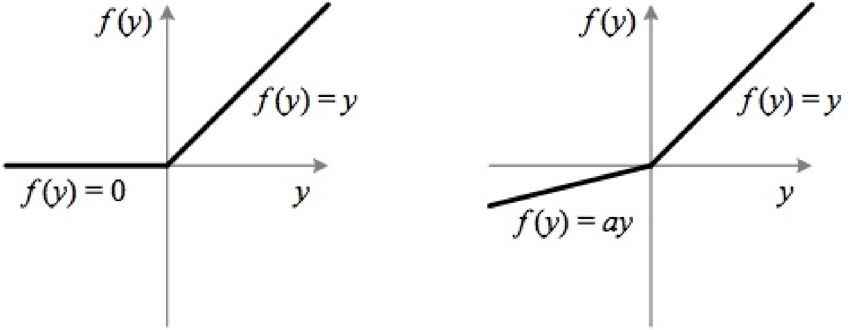


Figure 1. ReLU vs. PReLU. For PReLU, the coefficient of the negative part is not constant and is adaptively learned.

ResNet-50, -101, -152 (2015)

Finally, in the 2015 competition, Microsoft produced an model which is extremely deeper than any previously used. These models are known as ResNet, with their depth given as an suffix.

> *He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." arXiv preprint arXiv:1512.03385 (2015).*

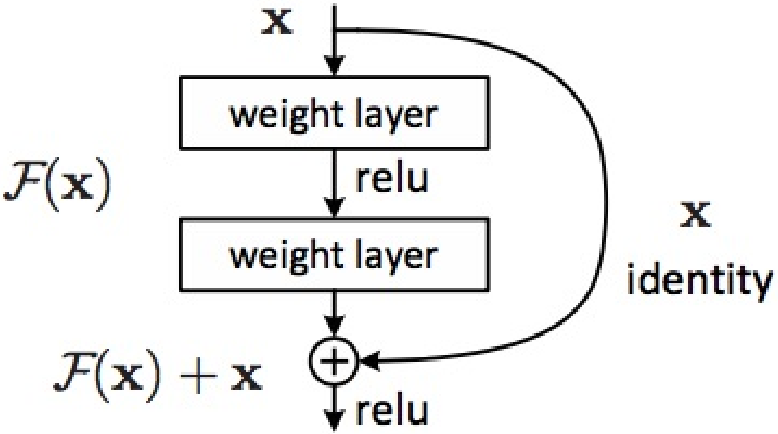


Figure 2. Residual learning: a building block.

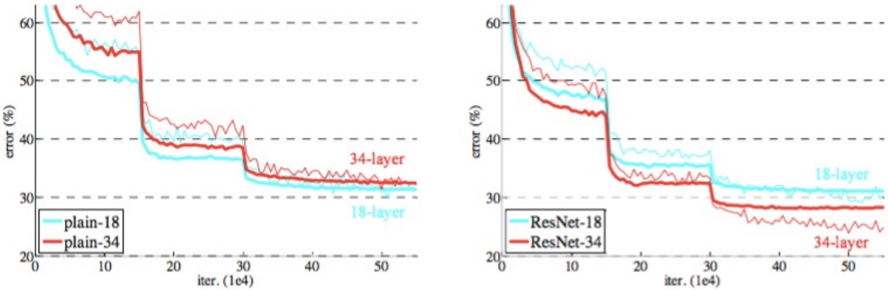


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

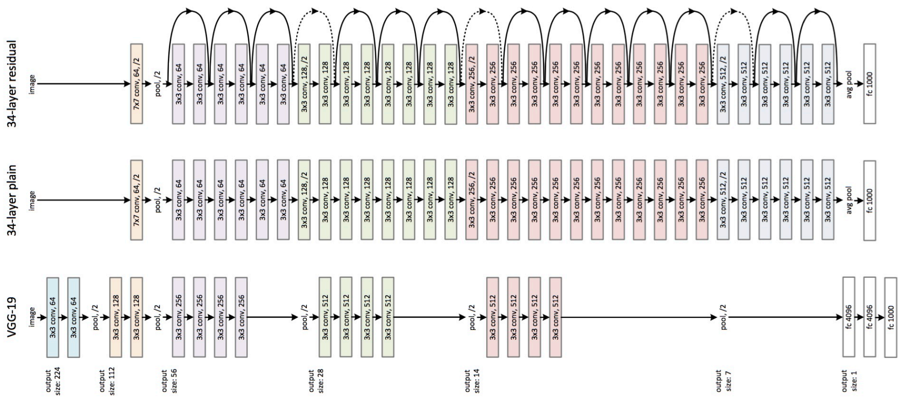


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Python demo V: ResNet Unit (2015)

Stochastic Depth Models (2016)

Another tweak on the ResNet architecture, which subsamples layers in the network:

> *Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, Kilian Weinberger. "Deep Networks with Stochastic Depth", arXiv preprint arXiv:1603.09382 (2016).*

Notice how this seems like an almost obvious thing to try given the ResNet architecture, but less-so in a generic neural network.

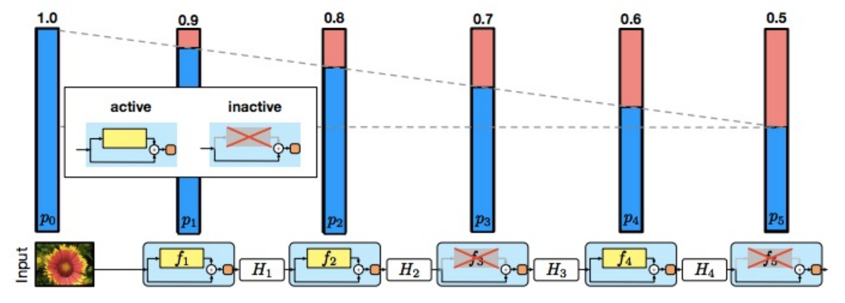


Fig. 2. The linear decay of p_ℓ illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.

ResNet-200, -1001 (2016)

Microsoft's update to last year's model. Posted only two weeks ago!

> *He, Kaiming, et al. "Identity Mappings in Deep Residual Networks." arXiv preprint arXiv:1603.05027 (2016).*

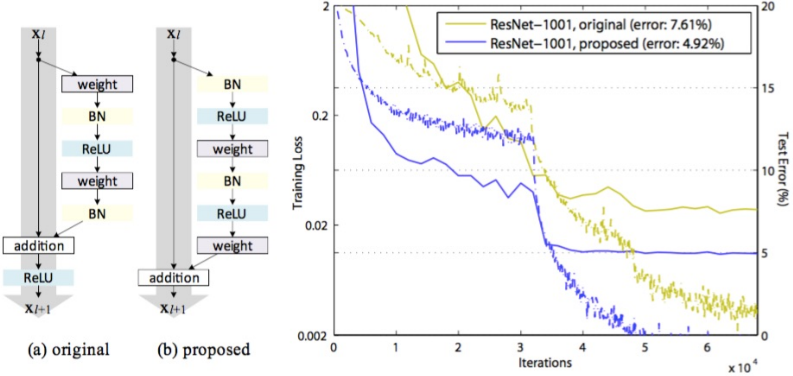


Figure 1. Left: (a) original Residual Unit in [1]; (b) proposed Residual Unit. The grey arrows indicate the easiest paths for the information to propagate, corresponding to the additive term "$\mathbf{x}_l$" in Eqn.(4) (forward propagation) and the additive term "1" in Eqn.(5) (backward propagation). **Right**: training curves on CIFAR-10 of **1001-layer** ResNets. Solid lines denote test error (y-axis on the right), and dashed lines denote training loss (y-axis on the left). The proposed unit makes ResNet-1001 easier to train.

SqueezeNet (2016)

A new line of research involves looking at ways to produce near state-of-the-art results with a minimal model size (or minimal computational cost):

> *Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size." arXiv preprint arXiv:1602.07360 (2016).*

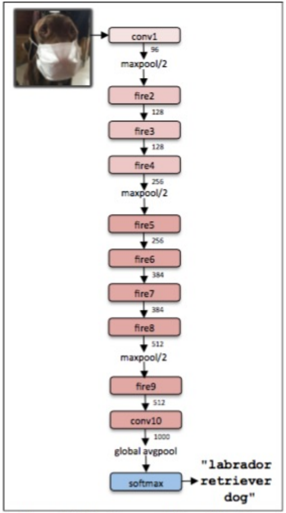


Figure 2. The SqueezeNet architecture

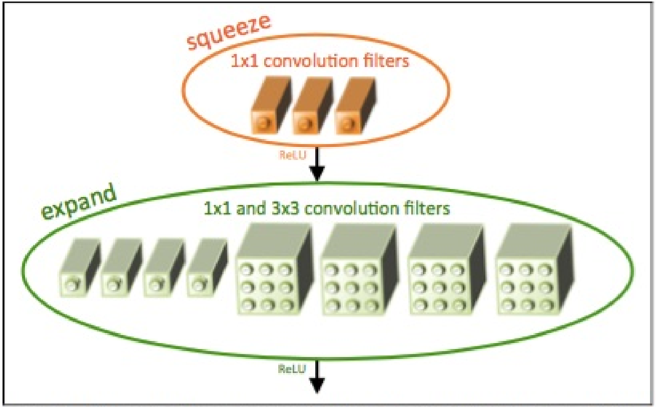
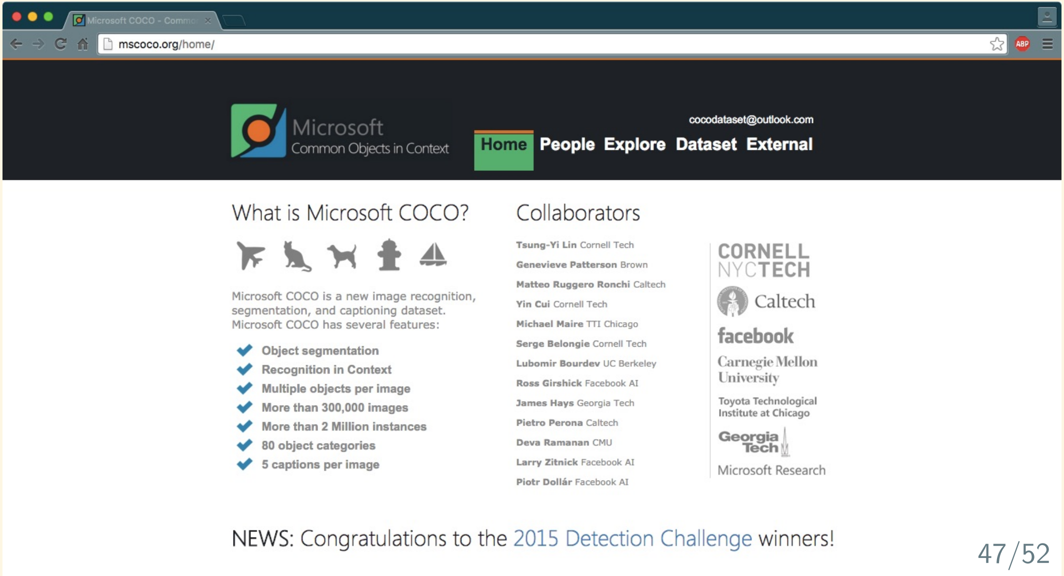


Figure 1. Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

Table 2. Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

DNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD [4]	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning [9]	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression[8]	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.92MB	**258x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.52MB	**461x**	57.5%	80.3%

Microsoft Common Images in Context (MS COCO)



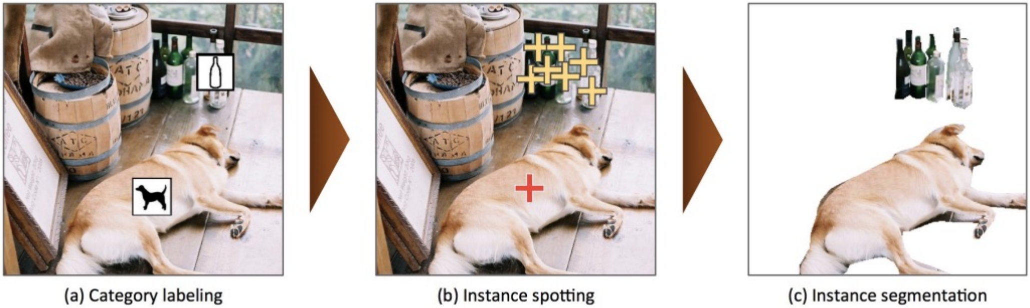
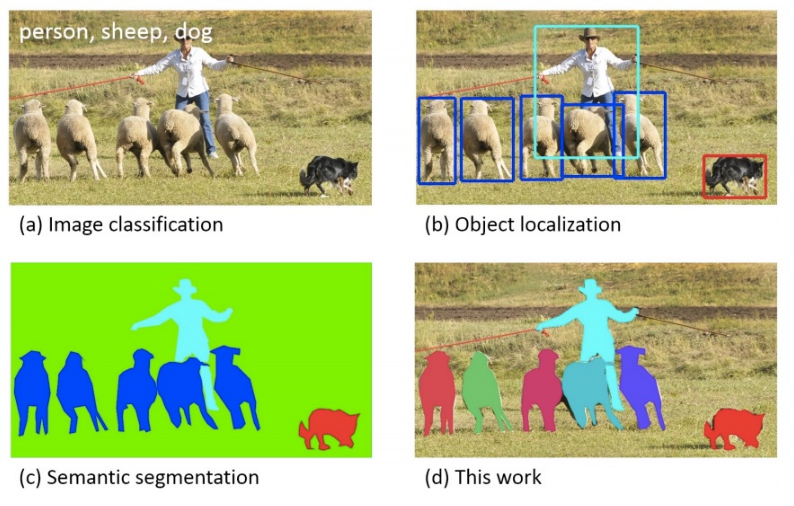


Fig. 3: Our annotation pipeline is split into 3 primary tasks: (a) labeling the categories present in the image (§4.1), (b) locating and marking all instances of the labeled categories (§4.2), and (c) segmenting each object instance (§4.3).


person, sheep, dog
(a) Image classification
(b) Object localization
(c) Semantic segmentation
(d) This work

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

More information is found on their website, `http://mscoco.org/`, and in the paper describing the dataset:

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." Computer Vision–ECCV 2014. Springer International Publishing, 2014. 740-755.

I should mention that Microsoft's own entry (ResNet) essentially swept every winning metric with their technique.

Video Classification

I don't want to make it sound as though the only interesting research in computer vision with neural networks involves one of these large public competitions (though the biggest conceptual advances have come out of these). For example, an influential paper doing video scene classification

> *Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-scale video classification with convolutional neural networks." In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1725-1732. 2014.*

Which yields a fantastic video demonstration of the output:

`https://www.youtube.com/watch?v=qrzQ_AB1DZk`