Chapter 9: Convolutional Networks

Laurence & Archy
4/8/16

Back propagation arrives in 1986

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

Rumelhart et al (1986)

Back propagation arrives in 1986

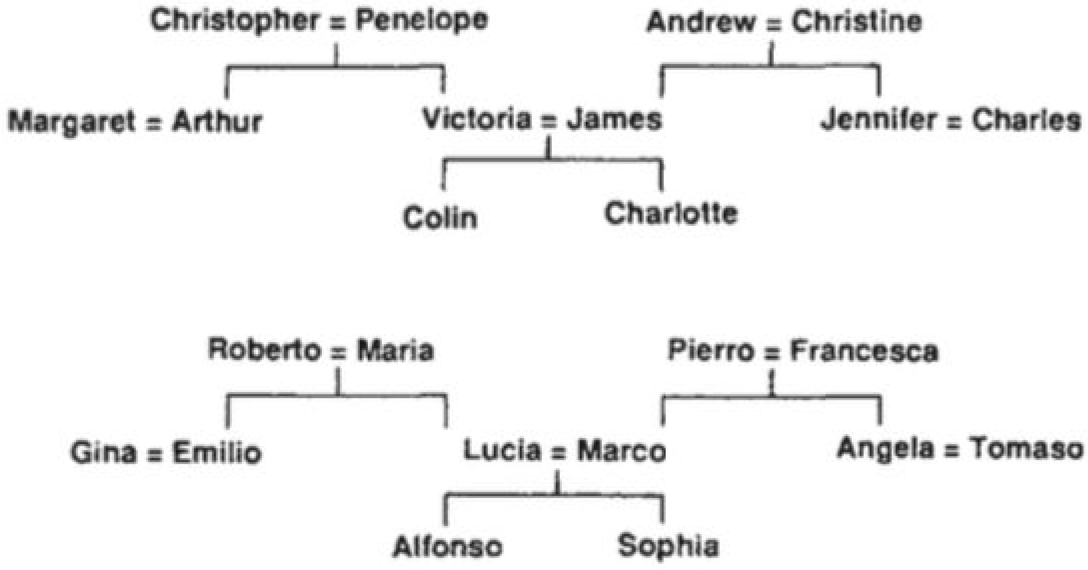


Fig. 2 Two isomorphic family trees. The information can be

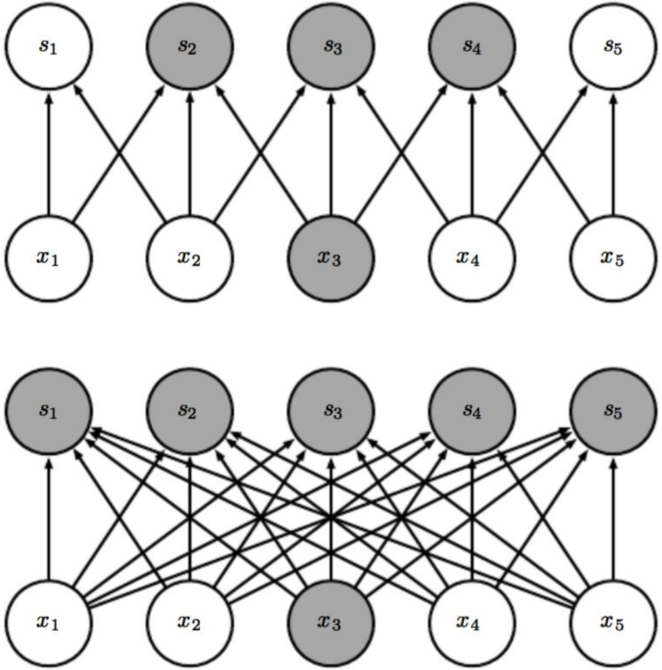
Rumelhart et al (1986)

Back propagation arrives in 1986

‘The most obvious drawback… is that the error-surface may contain local minima so that the gradient descent is not guaranteed to find a global minimum. However… adding a few connections creates extra dimensions in weight-space and these dimensions provide paths around the barriers that create poor local minima in lower dimensional subspaces’

Rumelhart et al (1986)

From feedforward nets to ConvNets



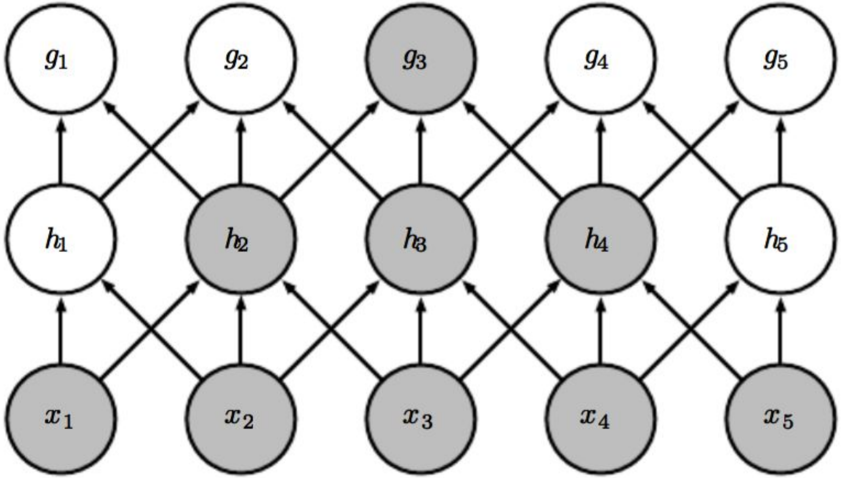
Convolutional feedforward net

Dense feedforward net

Sparse interactions ...make it fast - less matrix multiplication, and fewer parameters to learn whilst...

Figure 9.2

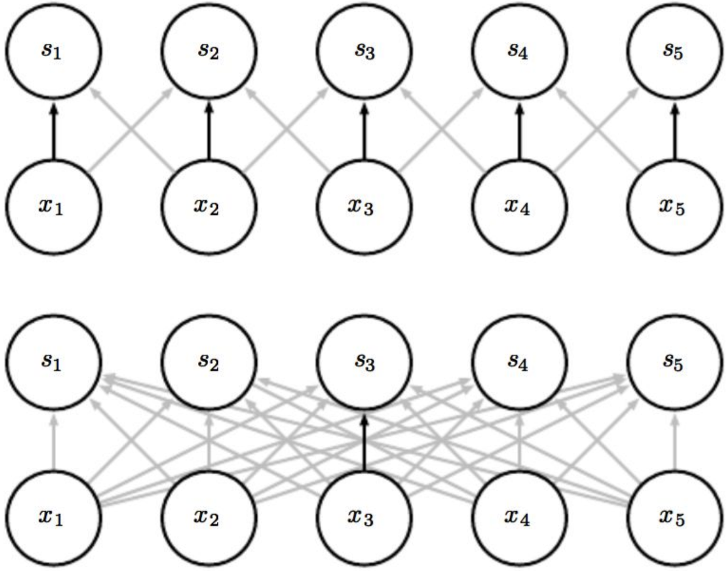
From feedforward nets to ConvNets



… operations requiring larger bits of the image can be found at deeper layers

Figure 9.4

From feedforward nets to ConvNets



Parameter sharing ('tied weights')
...reduces storage requirements
... produces **equivariance**
(consider **images** and **timeseries)**

- Doesn't handle rotations, rescalings

Figure 9.5

Computational efficiency really matters

Convolution: 2 x (319 x 280) computations; (and 2 parameters)
Matrix Multiplication: (320 x 280) x (319 x 280) computations

Figure 9.6

Computational efficiency still a challenge

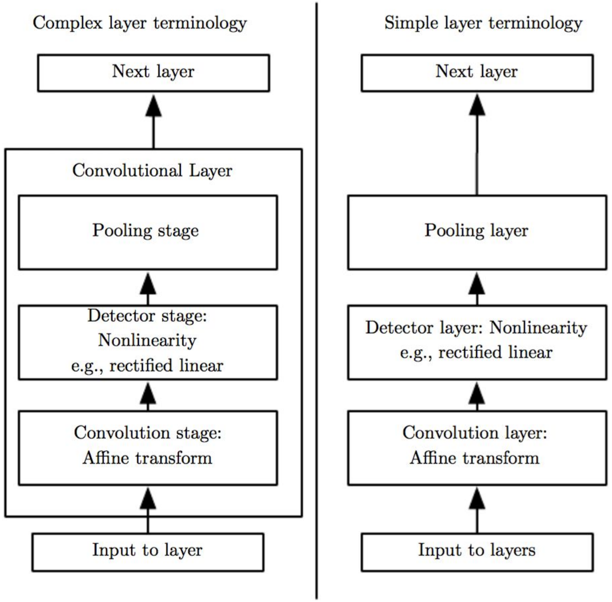
A ConvNet will still typically contain millions of units - needs parallelisation

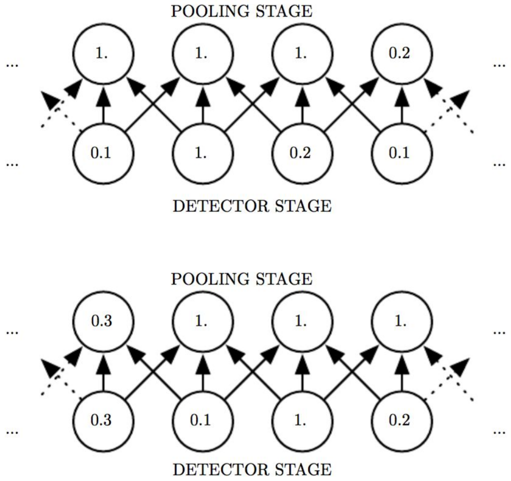
But tricks (section 9.8):

- Fourier domain convolution: convert both kernel and input to frequency domain, point-wise multiple, inverse transform
- Separate multi-dimensional kernels into multiple 1-D convolutions

ConvNets: pool to be cool

Pooling stage - looks for maximum in neighbourhood
Produces further invariance to position

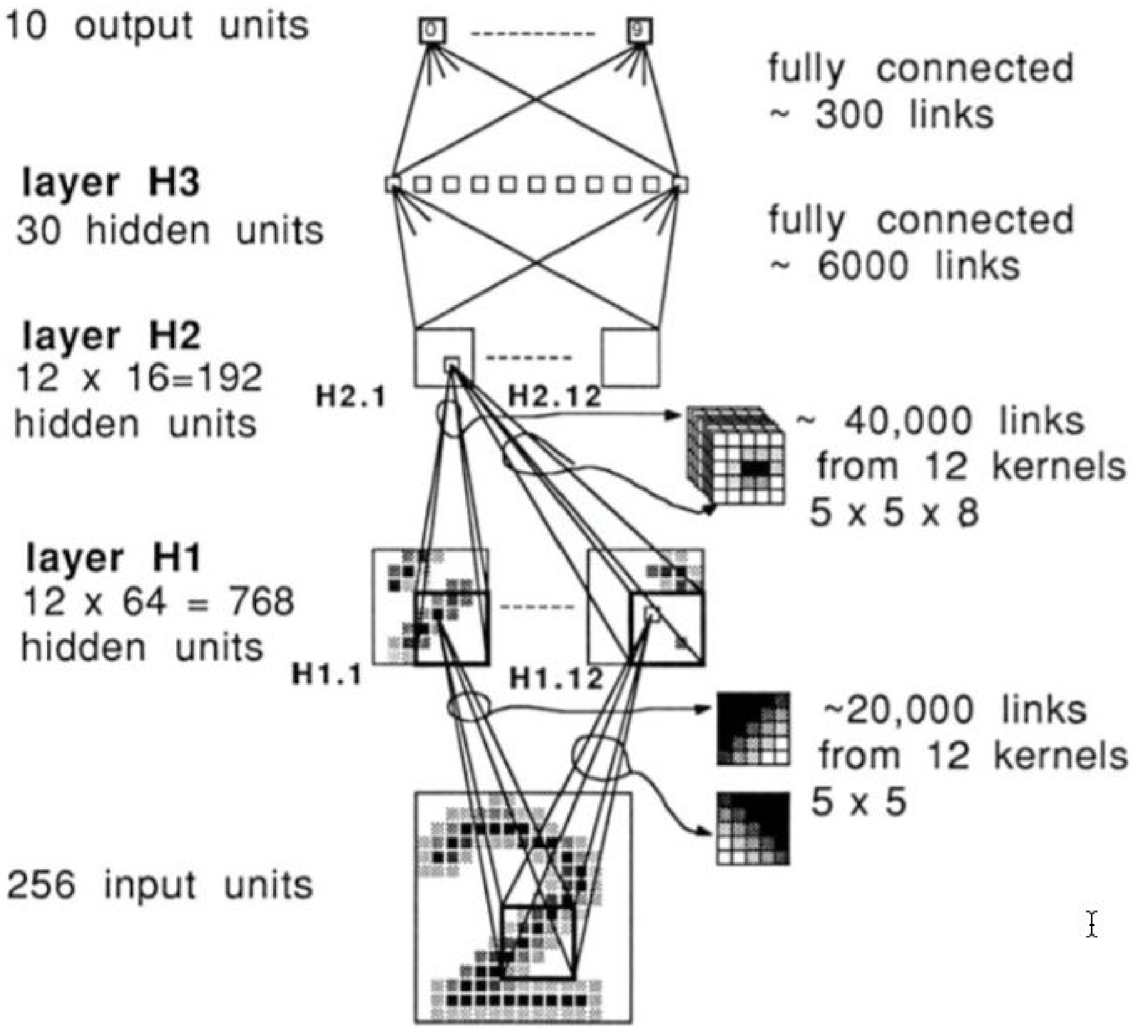




Can think of convolution/pooling as an ‘infinitely strong prior’ on a fully connected net

Figure 9.7/9.8

LeCun applies it to handwriting



LeCun (1989;1990)

LeCun applies it to handwriting

Clever observation 1: it's really fast to train

'Because of the redundant nature of the data and because of the constraints imposed upon the network, the learning time was relatively short considering the size of the training set'

LeCun (1989;1990)

LeCun applies it to handwriting

Clever observation 2: Position invariance

‘One reason for this is that the salient features of a distorted character might be displaced slightly from their position in a typical character’

LeCun (1989;1990)

LeCun applies it to handwriting

Clever observation 3: we can still get higher feature coding in space

‘Since the *precise* location of a feature is not relevant to the classification, we can afford *to* lose some position information in the process. Nevertheless, *approximate* position information must be preserved, to allow the next levels to detect higher order, more complex features (Fukushima 1980; Mozer 1987). ‘

LeCun (1989;1990)

LeCun 1990 demo

Some practicalities

Zero-padding (cf. ‘valid’ vs. ‘same’ vs. ‘full’ in MATLAB)

Each output normally has multiple convolutions at each position (to give multiple filters etc.)

There are other ways of reducing computational cost, e.g. ‘striding’

There are also other ways of joining layers locally: ‘locally connected’ vs. convolutional vs. ‘tiled convolution’

Table 9.1 provides nice examples of how to use on different data types

How do we do backprop on a kernel?

Remember that convolution can always be written as a (very large, sparse) matrix

Writing the transpose of this matrix helps us in computing the derivatives (but it seems like this will be massive?)

(section 9.5… got a bit lost).

They work well on GPUs

They're fast

'Our implementation allows for training large CNNs within days instead of months….

One epoch takes 35 GPU minutes but more than 35 CPU hours'

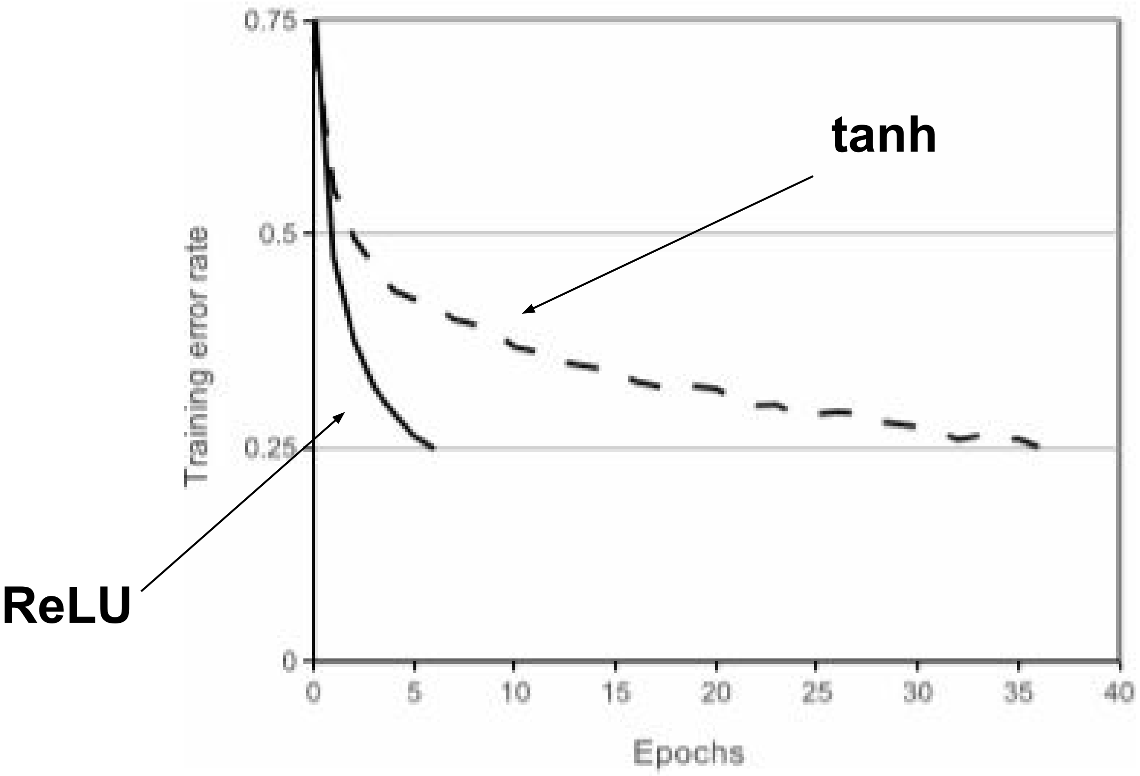
And accurate

On MNIST the best network achieved a recognition test error rate of 0.35%, on NORB 2.53% and on CIFAR10 19.51%. Our results are raising the bars for all three benchmarks

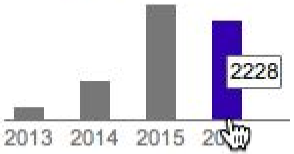
Ciresan (2011)

All hail Hinton

Ready, set, ReLu



Cited by 6040



Krizhevsky, Sutskever, Hinton (2012)

All hail Hinton

Dang I love dropout

‘This technique reduces complex co-adaptations of neurons…

...it is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons...

… without dropout, our network exhibited considerable overffitting. ‘

Krizhevsky, Sutskever, Hinton (2012)

All hail Hinton

Add some data augmentation

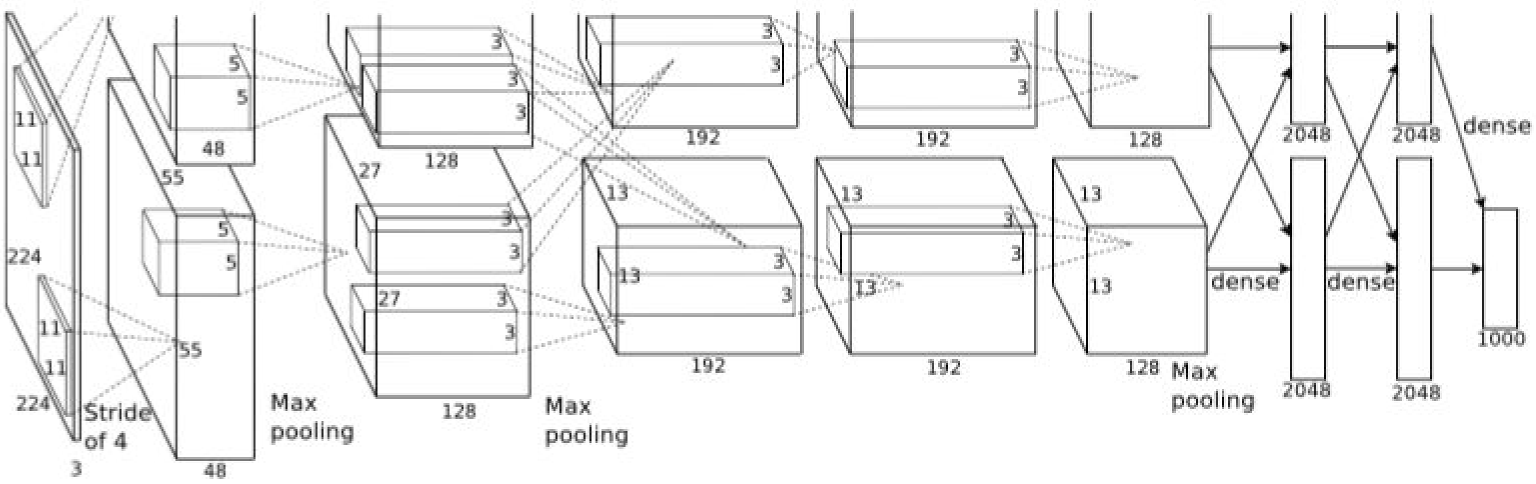
The first form of data augmentation consists of **generating image translations and horizontal reflections**. We do this by extracting random 224 × 224 patches (and their horizontal reflections) from the 256×256 images and training our network on these extracted patches....

The second form of data augmentation consists of **altering the intensities of the RGB channels** in training images.

Krizhevsky, Sutskever, Hinton (2012)

All hail Hinton

Network depth through the roof



Krizhevsky, Sutskever, Hinton
(2012)

All hail Hinton

It works rather well

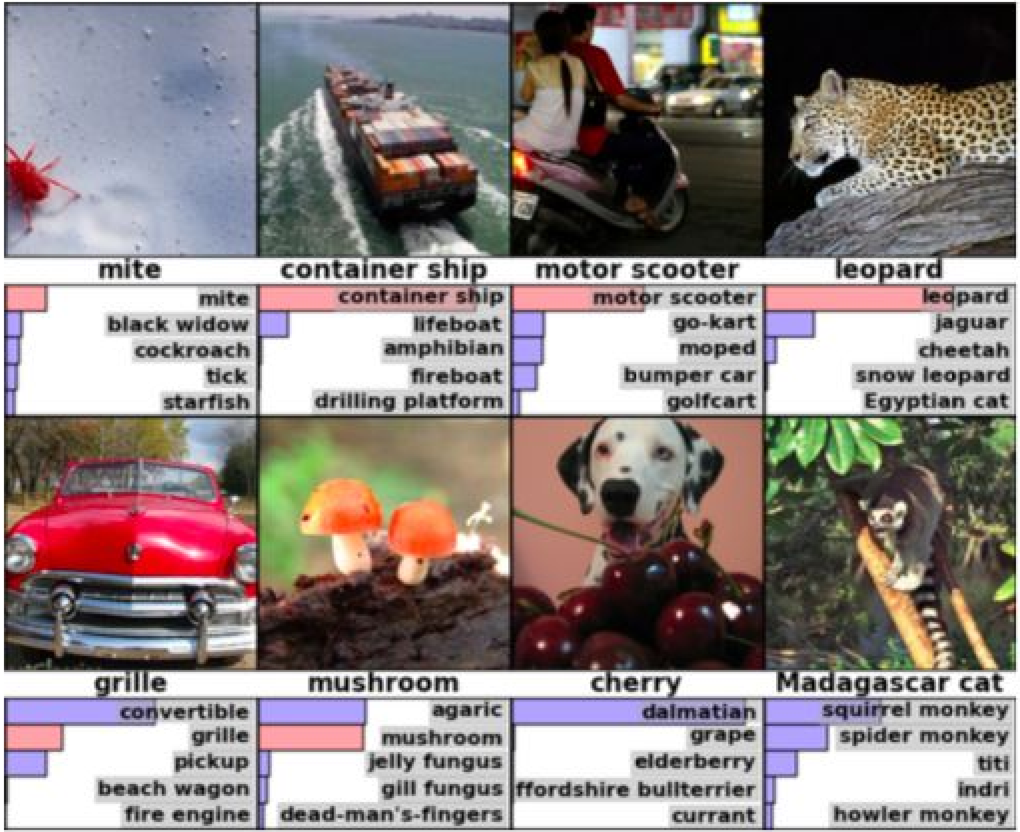
Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Krizhevsky, Sutskever, Hinton (2012)

All hail Hinton

Example labellings



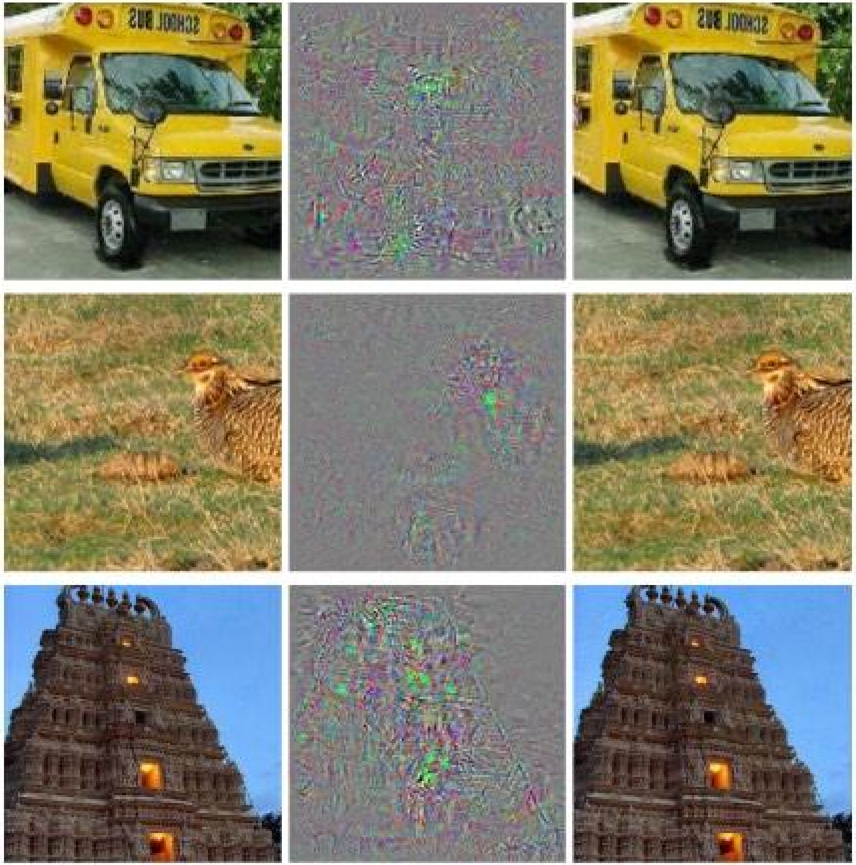
Krizhevsky, Sutskever, Hinton (2012)

All hail Hinton

Final layer similarities

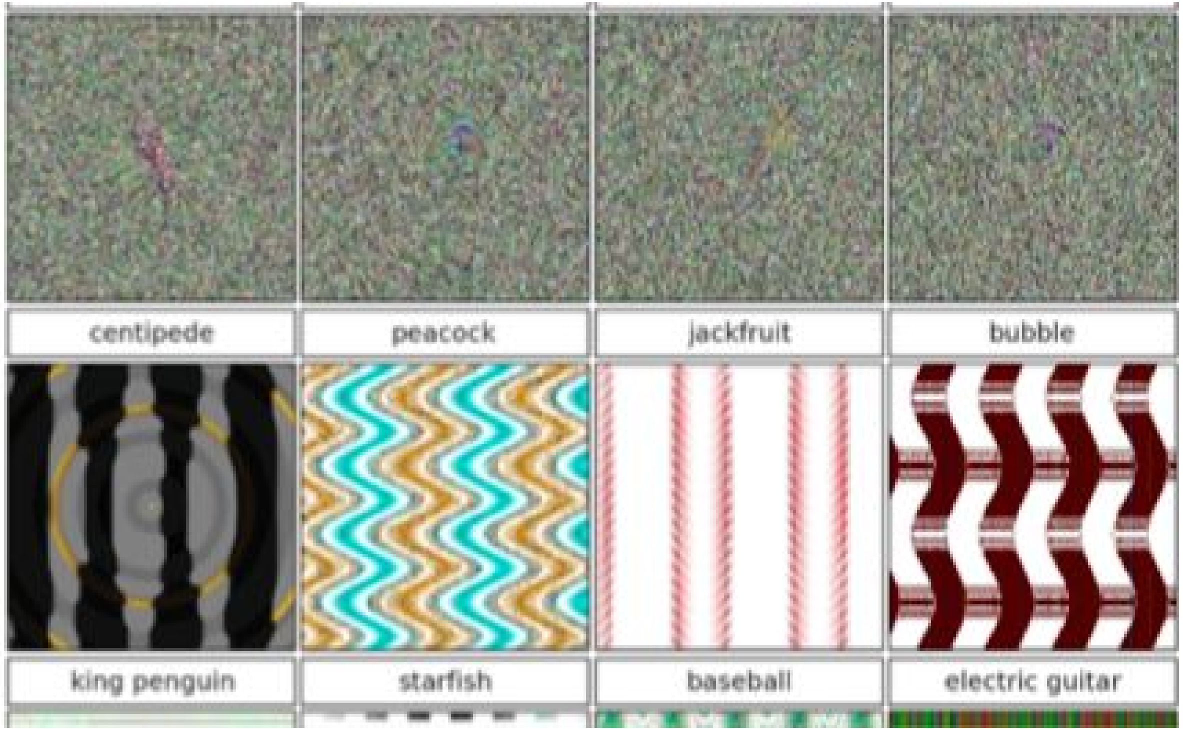
Krizhevsky, Sutskever, Hinton (2012)

ConvNets can be fragile



Szegedy et al (2013)

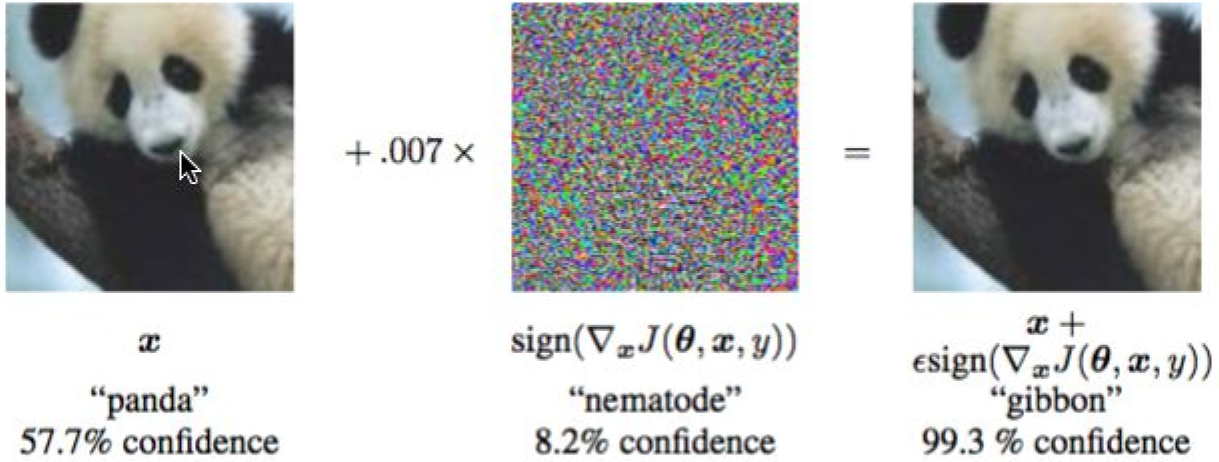
ConvNets can be fragile



Nguyen et al (2014)

ConvNets can be fragile

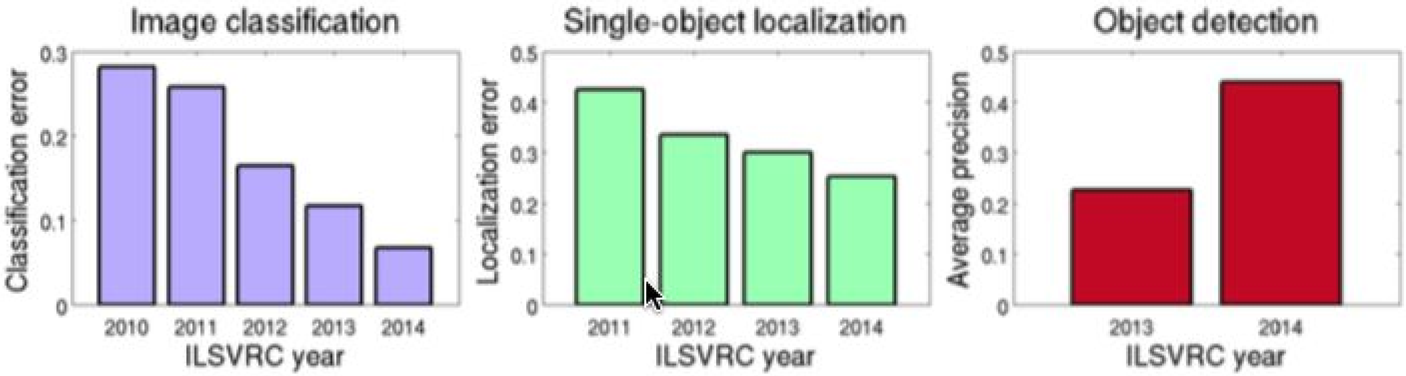
‘Instead, these algorithms have **built a Potemkin village** that works well on naturally occurring data, but is exposed as a fake when one visits points in space that do not have high probability in the data distribution’



Goodfellow et al (2015)

Bye bye humans

Progress on ImageNet keeps improving



Russakovsky et al (2014)

Bye bye humans

Google LeNet

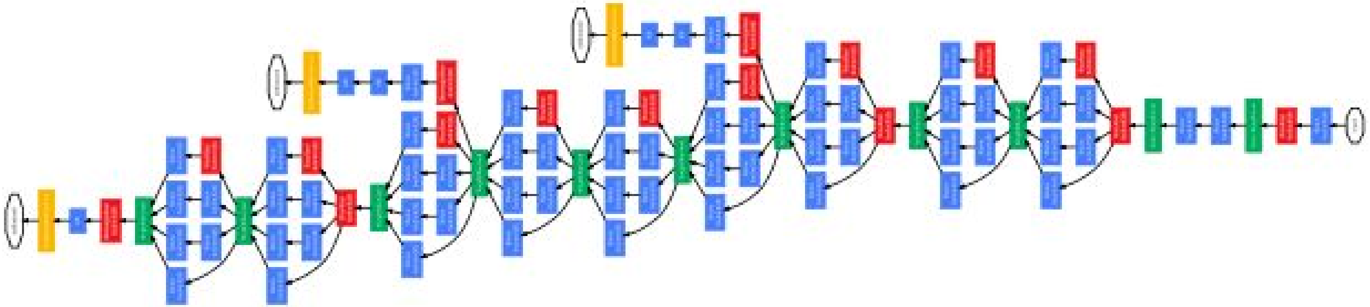


Figure 3: GoogLeNet network with all the bells and whistles

Szegedy et al (2014)

Bye bye humans

Google LeNet is probably better than a human

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.

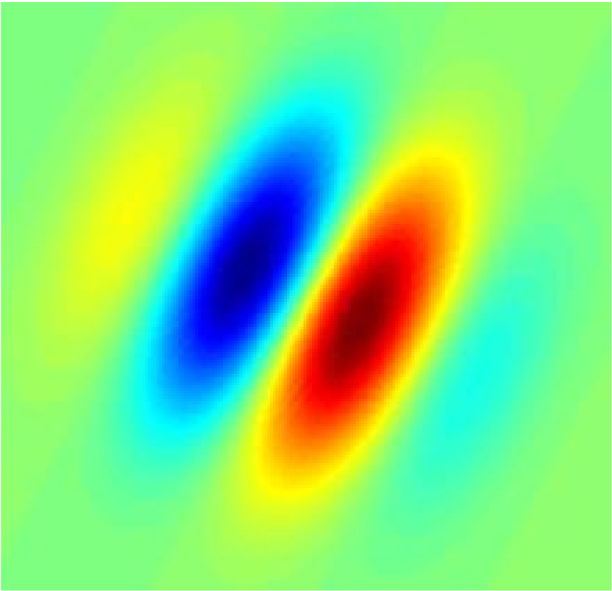
Russakovsky et al (2014)

(Aside: interesting blog)

Andrej Karpathy on labelling images

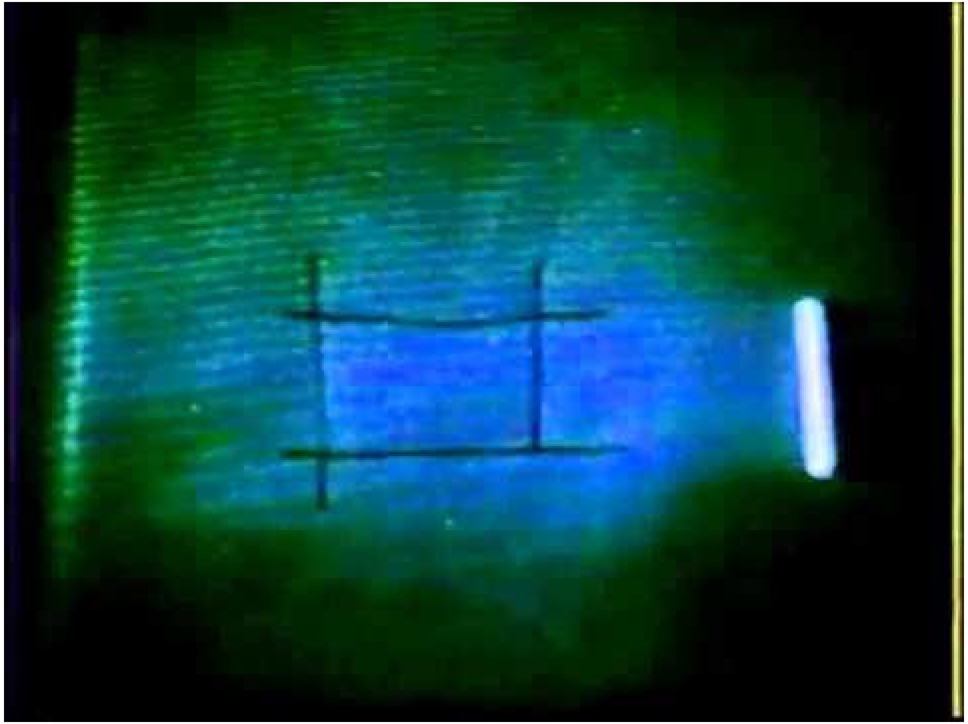
http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

Links to neuroscience



Simple cell: convolution?
(n.b. Reverse correlation)

Links to neuroscience



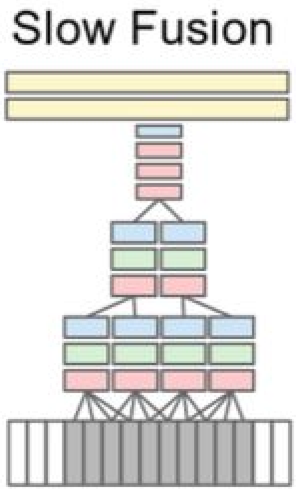
Complex cell: max pooling?

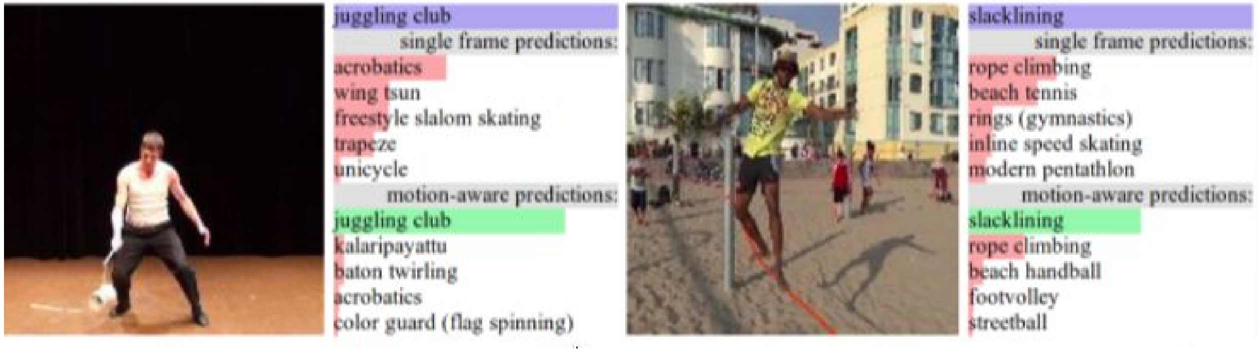
Va-va-video

Karpathy et al (2014)

Va-va-video

Convolving over time and space



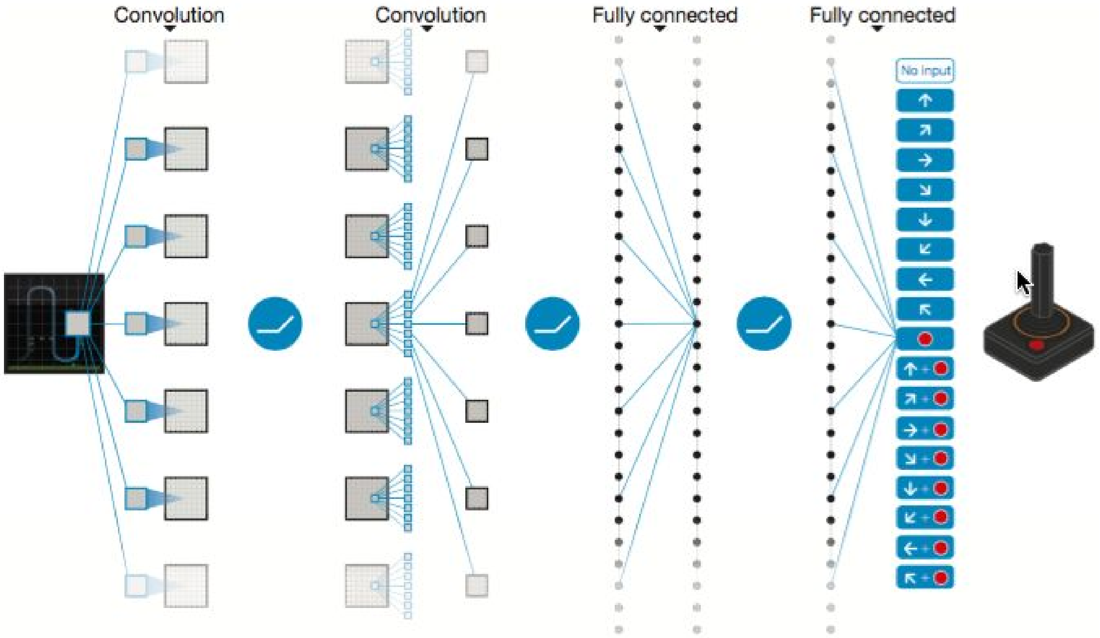


Karpathy et al (2014)

Convolution in DQN

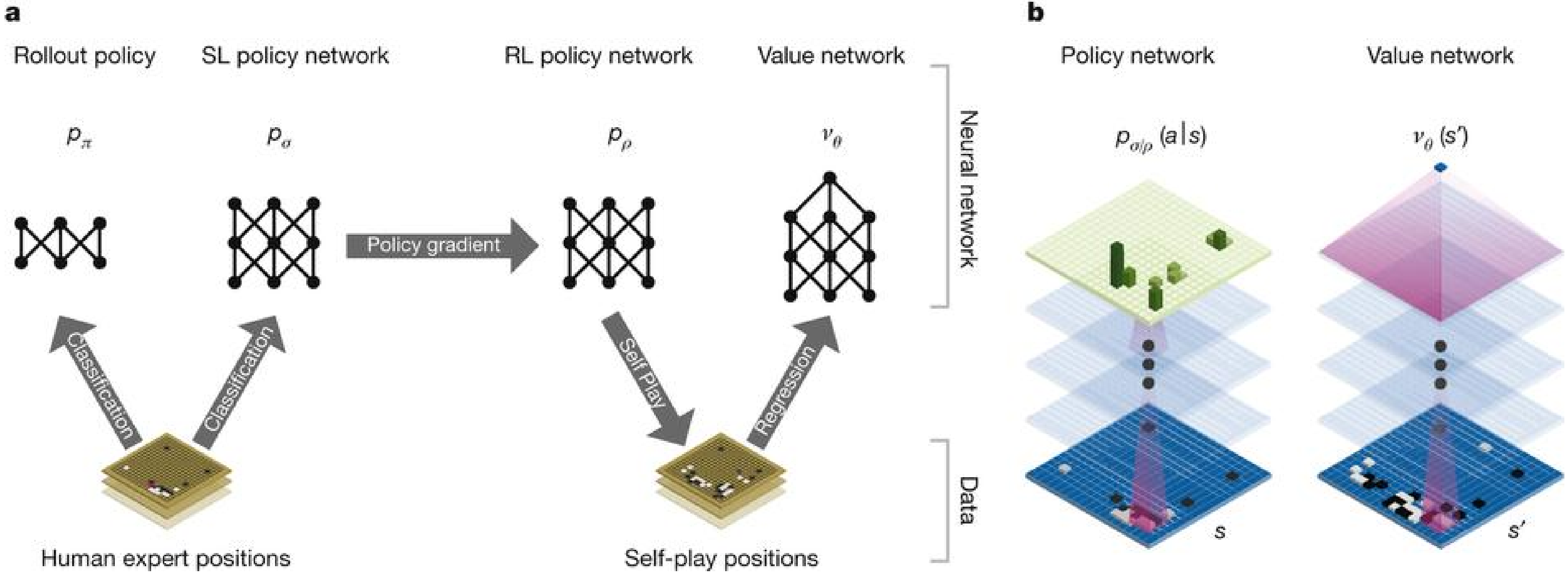
From pictures to playing

'The first hidden layer convolves 32 filters of 8 3 8 with stride 4 with the input image and applies a rectifier nonlinearity[31,32]. The second hidden layer con- volves 64 filters of 4 3 4 with stride 2, again followed by a rectifier nonlinearity. This is followed by a third convolutional layer that convolves 64 filters of 3 3 3 with stride 1 followed by a rectifier.'



Mnih et al (2015)

Convolution in DQN



Silver et al (2016)

Convolution in DQN

Before you Go Go

The input to the policy network is a 19 × 19 × 48 image stack consisting of **48 feature planes**. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. **Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1**, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used ***k* = 192 filters**

Silver et al (2016)

Further reading

Michael Nielsen’s excellent ebook
http://neuralnetworksanddeeplearning.com/chap6.html#recent_progress_in_image_recognition

Andrej Karpathy’s very convivial blog
http://karpathy.github.io/neuralnets/