

UNIVERSITY *of* WASHINGTON

# Data Science UW

# Methods for Data

# Analysis

---

Introduction to Neural Networks

Extra Topics

Nick McClure



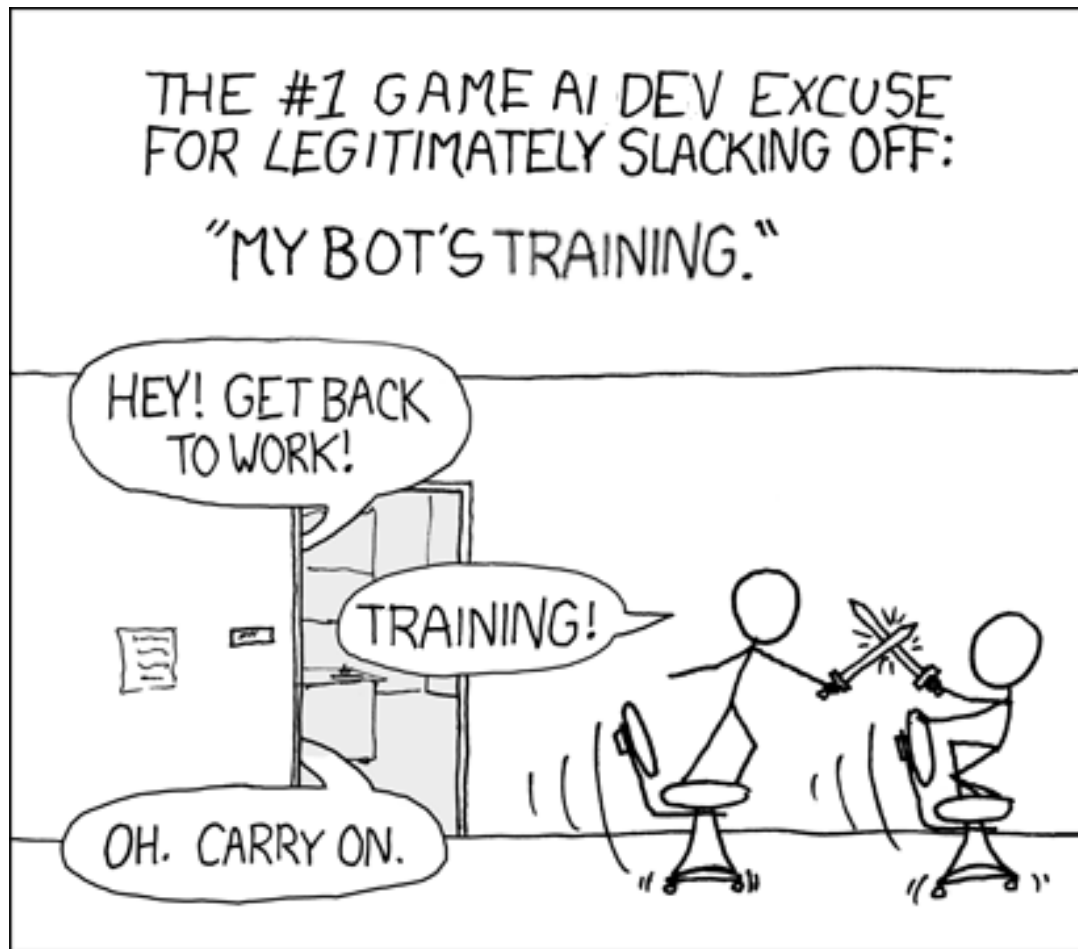
THE #1 GAME AI DEV EXCUSE  
FOR LEGITIMATELY SLACKING OFF:

"MY BOT'S TRAINING."

HEY! GET BACK  
TO WORK!

TRAINING!

OH. CARRY ON.



W

# Review

## > Bayesian and Computational Statistics

- Bayesian inference on a coin flip
- Derive a Posterior from a Prior and Likelihood
- MCMC for a coin flip
- MCMC for a bivariate normal
- Bayesian Linear Regression
- Bootstrapping for small data sets
- Bootstrapping for linear regression
- Computational P values (simulating the Null)
- Confusion Matrix and Cross Validation



# Topics

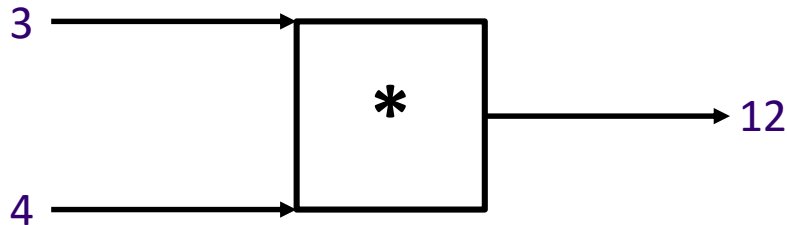
---

- > Circuits
- > Neural Networks
- > Further applications of Neural Networks



# Algebraic Operations as Circuit Diagrams

>  $F(x,y) = x * y$



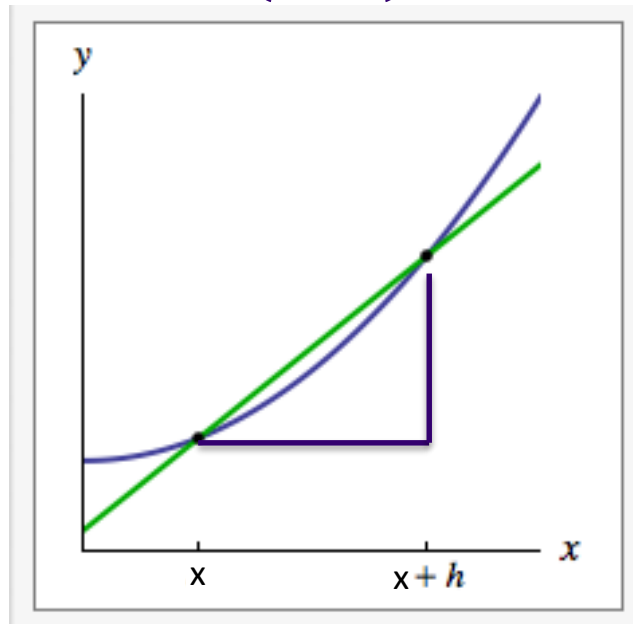
- > How can we change the inputs to decrease the output?
- > The intuitive answer is to decrease 3 and/or decrease 4.
- > Mathematically, the answer is to use the derivative...

**W**

# The Derivative

- > The derivative of  $F$  with respect to  $x$ , tells us how  $F$  changes when we change  $x$  by a slight amount,  $h$ :

$$\frac{dF(x, y)}{dx} = \frac{F(x + h, y) - F(x, y)}{(x + h) - x}$$



W

- > We want to set  $h$  to 0, but we can't so we just take the limit.

# The Derivative

$$\frac{dF(x, y)}{dx} = \lim_{h \rightarrow 0} \frac{F(x + h, y) - F(x, y)}{h}$$

> This equation works out really nicely for multiplication:

$$\frac{dF(x, y)}{dx} = \lim_{h \rightarrow 0} \frac{(x + h)y - xy}{h}$$

$$\frac{dF(x, y)}{dx} = \lim_{h \rightarrow 0} \frac{xy + hy - xy}{h}$$

$$\frac{dF(x, y)}{dx} = \lim_{h \rightarrow 0} \frac{hy}{h}$$

$$\frac{dF(x, y)}{dx} = \lim_{h \rightarrow 0} y = y$$

> So...

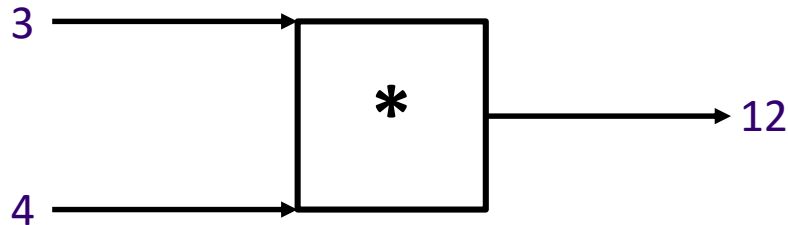
$$\frac{dF}{dx} = y$$

$$\frac{dF}{dy} = x$$

# W

# Using the Derivative to change inputs

>  $F(x,y) = x * y$



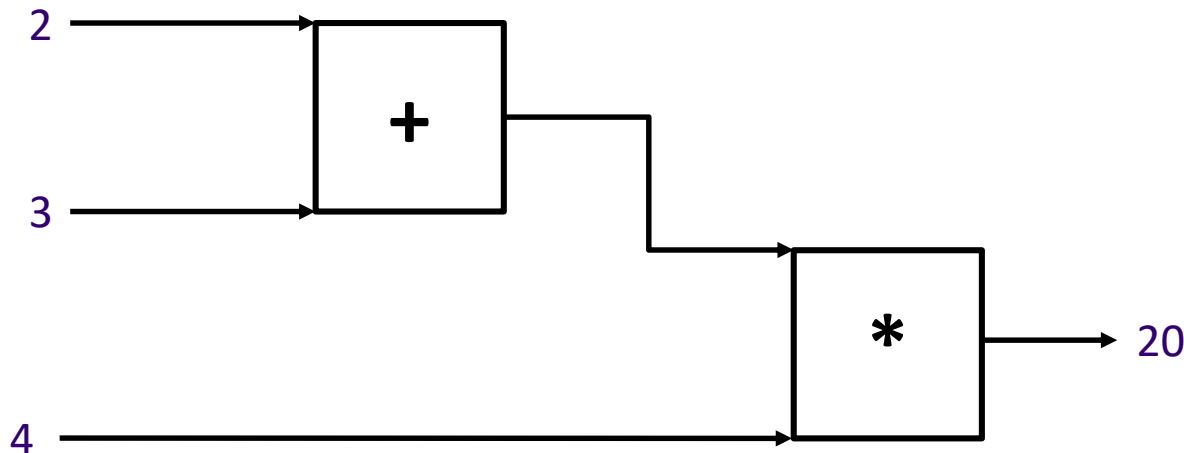
- > Mathematically, the answer is to use the derivative...
  - We know if we increase the (3) input by 1, the output goes up by 4. ( $4*4 = 16$ )

**W**



# Multiple Gates Chained Together

- >  $F(x, y, z) = (x + y) * z$
- > If  $G(a, b) = a + b$ , then  $F(x, y, z) = G(x, y) * z$
- > Or (change notation)  $F = G * z$  (we have solved this problem before)



- > Mathematically, we will use the “chain rule” to get the derivatives:

$$\frac{dF}{dx} = \frac{dF}{dG} \cdot \frac{dG}{dx}$$

$$\frac{dF}{dx} = z \cdot 1$$

- > R Demo



# A Simple Neural Network

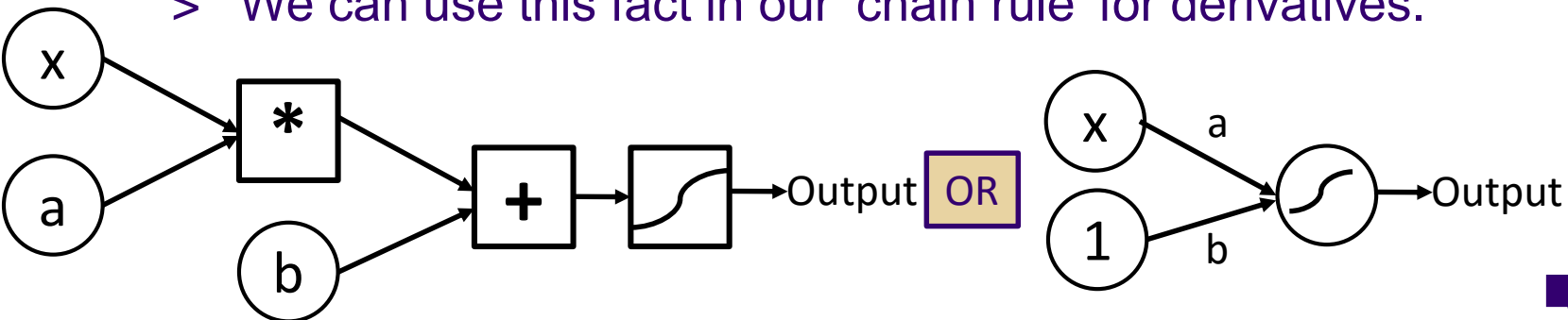
- > A neural network is very similar to what we have been doing, except that we bound the outputs between 0 and 1... with a sigmoid function (we've used this before in logistic regression).

$$F(x, a, b) = \sigma(ax + b) \quad \sigma(x) = \frac{1}{1 + e^{-x}}$$

- > The sigmoid function is nice, because it turns out that the derivative is related to itself:

$$\frac{d\sigma}{dx} = \sigma(x) \cdot (1 - \sigma(x))$$

- > We can use this fact in our 'chain rule' for derivatives.



- > R-demo

W

# Different Gate Functions

- > The sigmoid function is just one example of a gate function.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- > There is also the Rectified Linear Unit (ReLU):

$$\sigma(x) = \max(x, 0)$$

- > Softplus:

$$\sigma(x) = \ln(1 + e^x)$$

- > Leaky ReLU:

$$\sigma(x) = \max(x, ax)$$

– Where  $0 < a < 1$

- > R-demo



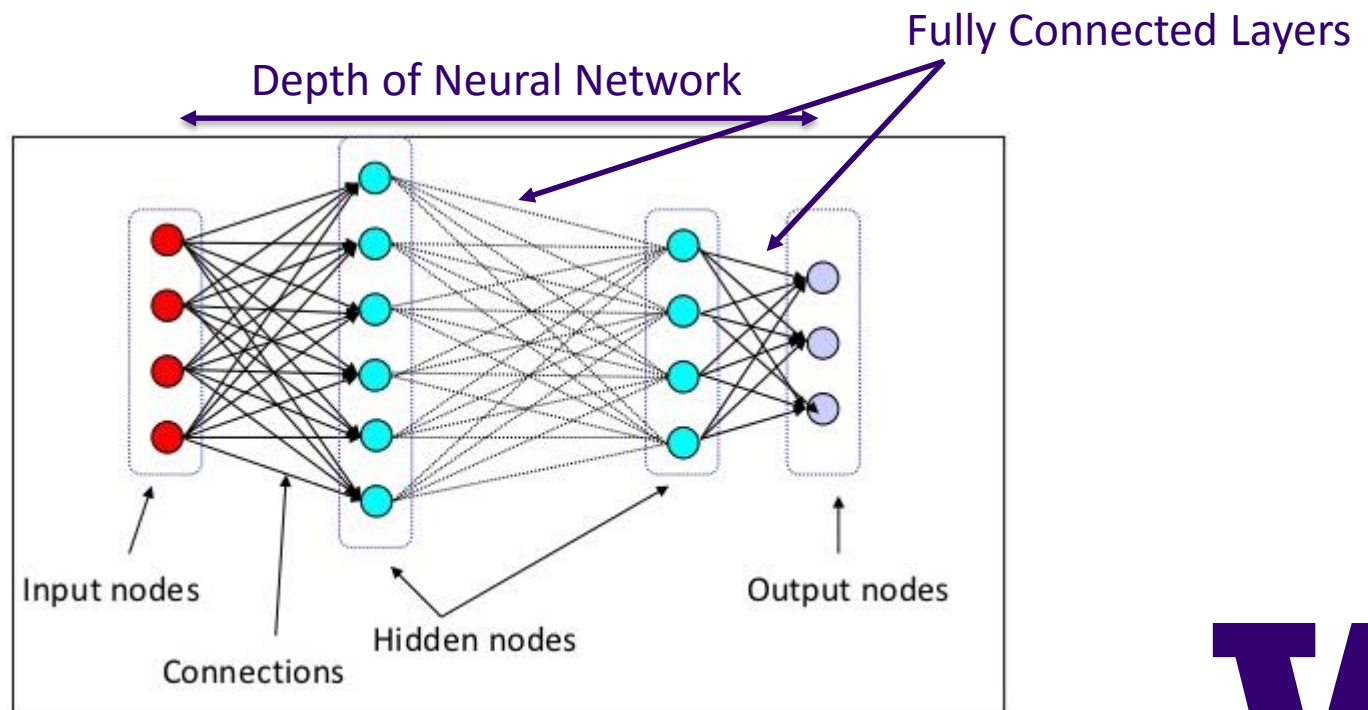
# Training a Neural Network

- > If we want a neural network to learn, we have to tell it a 'loss function'.
- > The idea is we have a labeled data set and we look at each data point. We run the data point forward through the network and then see if we need to increase or decrease the output.
- > We then change the parameters according to the derivatives in the network.
- > We loop through this many times until we reach the lowest error in our training.
- > R-demo



# Larger Neural Networks

- > Neural networks can have many 'hidden layers'.
- > We can make them as deep as we want.
- > Neural Networks can also have as many inputs or outputs as we would like as well.

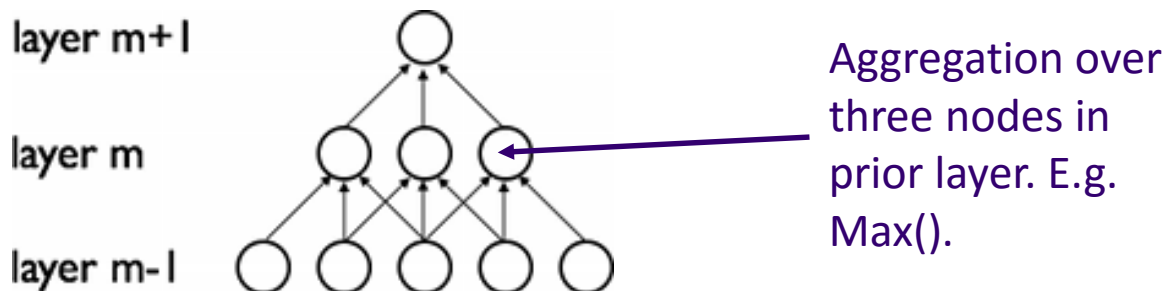


W

- > R-demo

# Additional Issues and Using Pooling/subsampling

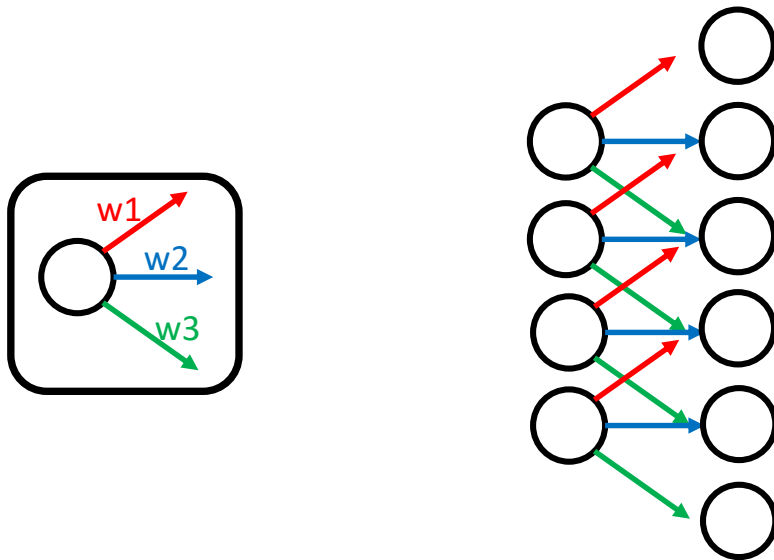
- > Larger neural networks can have MANY parameters to fit.
- > Solution 1: Provide more training data.
- > Solution 2: Reduce parameters via aggregation steps in the model, this is called “pooling”.
- > Types of pooling:
  - Maximum: Take the maximum of a group of nodes in the prior layer.
  - Minimum
  - Mean
  - Median
  - ...



**W**

# Convolution Layers

- > Another solution to reduce the parameters in the models is to use what is called convolution layers.
- > Here we assume that weights are the same on arrows in the same direction, regardless of which data node we start at:



There are only three weights in this layer,  $w_1$ ,  $w_2$ , and  $w_3$

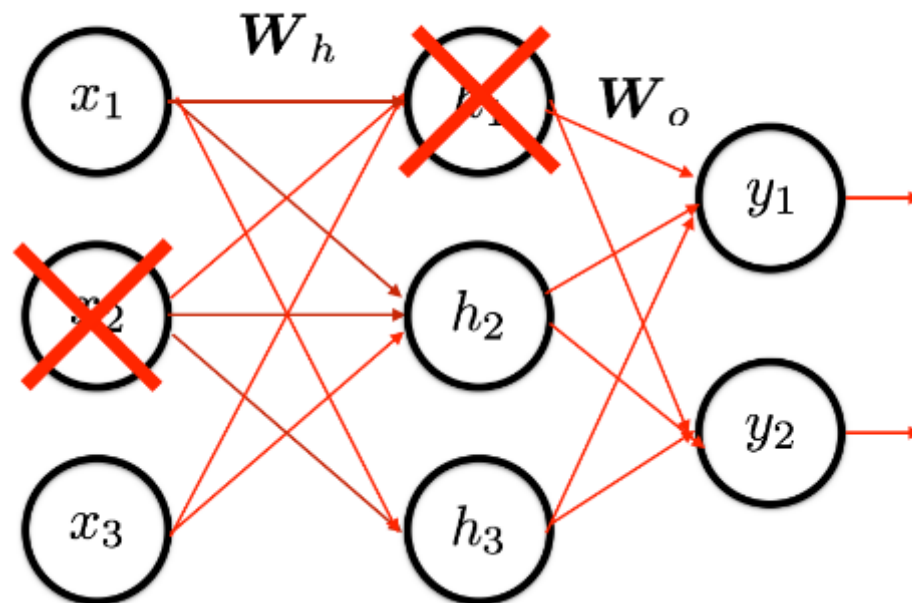
Note: Usually multiple convolutions are performed, creating 'Layers'.

# W

# Dropout

- > Dropout is a way to increase the training strength and regularize the parameters.
- > During training, we randomly select connections (or sets of edges) and set their value to zero.
- > This helps the network find multiple pathways for prediction. Because of this, the network will not rely on just one connection or edge for prediction.
- > This also helps the training of other parameters in the network.

At training time



W



# Convolutional Neural Networks

- > We can use combinations of pooling/subsampling, and convolutional layers to apply this data transformation to images.
- > Images are essentially matrices of input and we want to classify them.
- > The output would be an array of (# of classes) of real numbers.

Example of predicted output for three classes  $\rightarrow (-0.2, 0.4, 1.1)$

- > (Remember, we are using back propagation to pull the right class output up and the wrong classes down for each example.)
- > We usually convert these to probabilities via a 'Softmax' function.

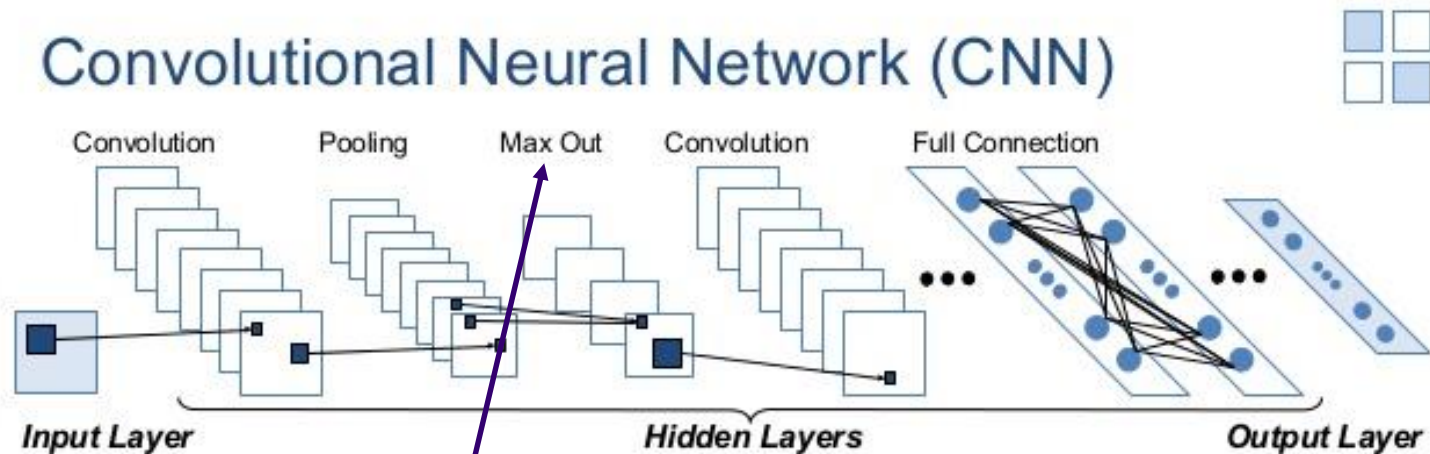
$$P(\text{output} = i) = \frac{e^{x_i}}{\sum e^{x_i}}$$

$$(-0.2, 0.4, 1.1) \rightarrow (0.154, 0.281, 0.565)$$



# Neural Network Layer Shorthand

- > Since neural networks can get quite large, we do not want to write out all the connections nor nodes.
- > Here's how to convey the meaning in a shorter, less complicated way:



“Max of a set of features”  
E.g., Here, it appears to be  
a max of 2 features.

W

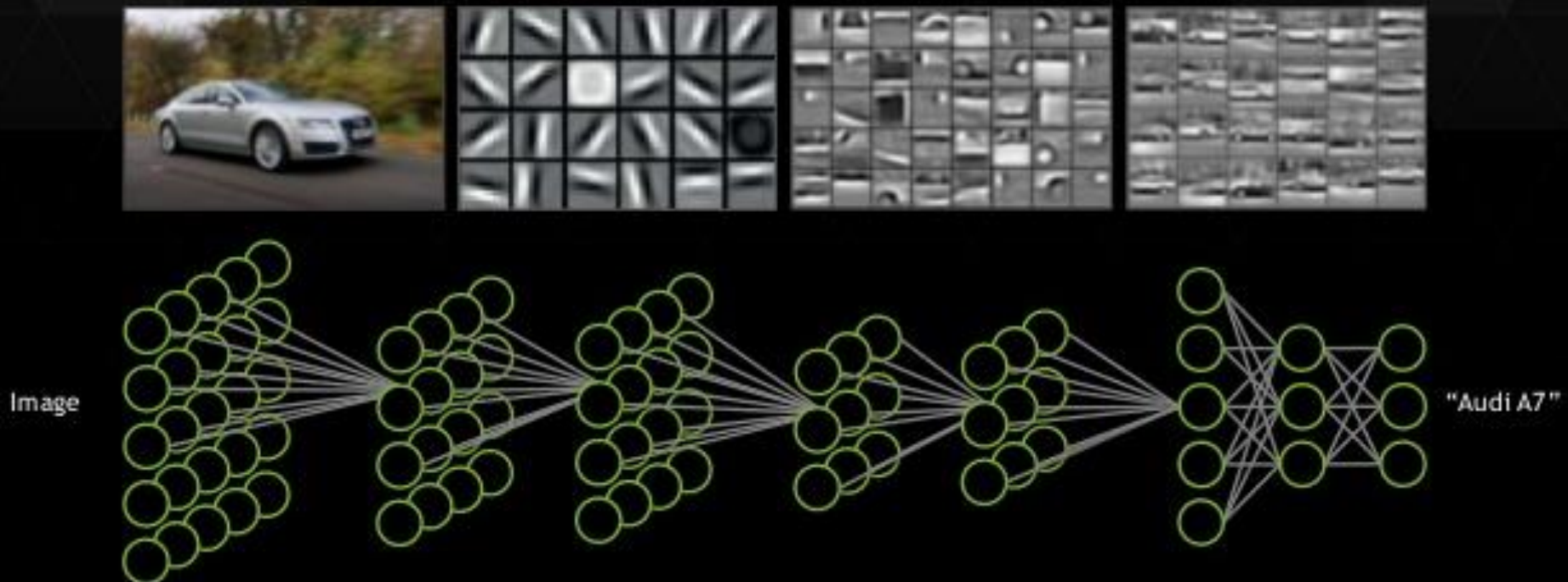
# Neural Network Simple Convolution Animation

- > Nice animation of simple convolutions and pooling:
- > <http://cs231n.github.io/convolutional-networks/>



# More on Convolutional Networks

- > We can intercept the pixels mid way and see what neurons are activated in each image and see what the neurons are learning:





# Deep Dream (2015)

- > We can also back-propagate labels into images and see what the images would then look like.



<https://www.youtube.com/watch?v=DgPaCWJL7XI>

<http://www.fastcodesign.com/3057368/inside-googles-first-deepdream-art-show>



# StyleNet (2015)

- > Another possibility is to train part of the network that learns basic shapes and edges on a “style picture” and train the rest on a “source picture”. The loss function here is how close small sections of the source resemble the style.



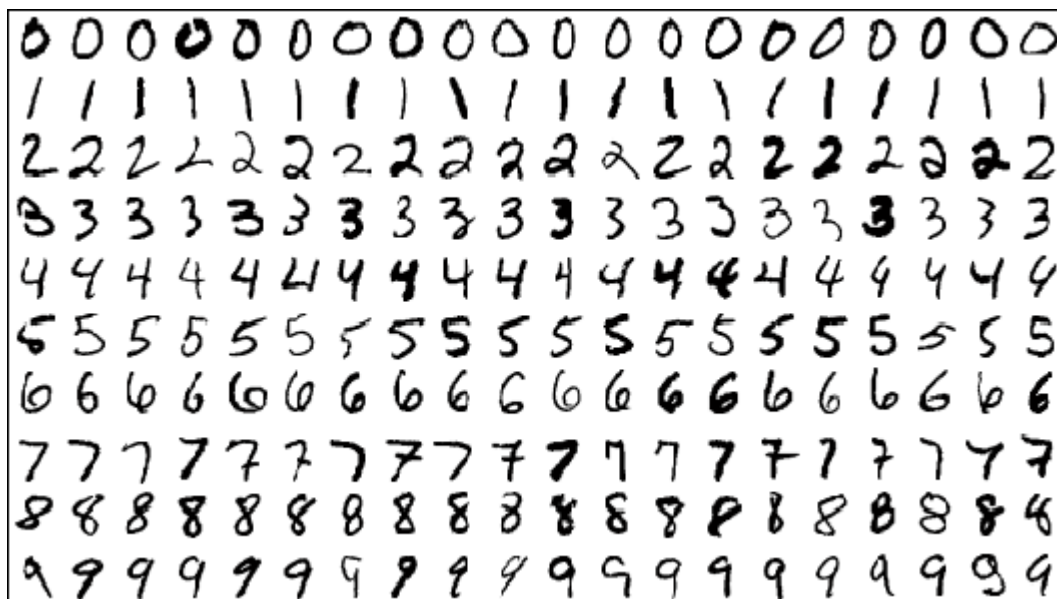
Recoloring of black and white photos

<https://vimeo.com/139123754>



# CNN Performance Benchmarks

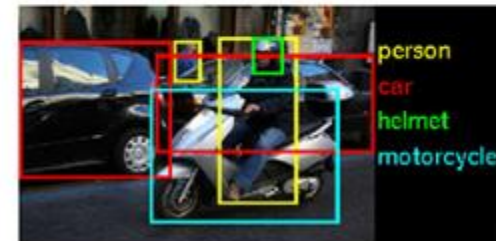
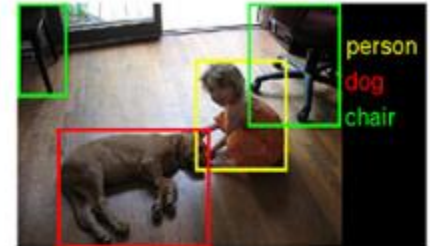
- > CNN models were benchmarked on the MNIST handwritten digit recognition set, but there are only 70K examples.
- > Each image is a matrix of 28x28 pixels.
- > This dataset was created by Yann Lecun, the creator of Conv NNs.
- > He keeps it here:
- > <http://yann.lecun.com/exdb/mnist/>





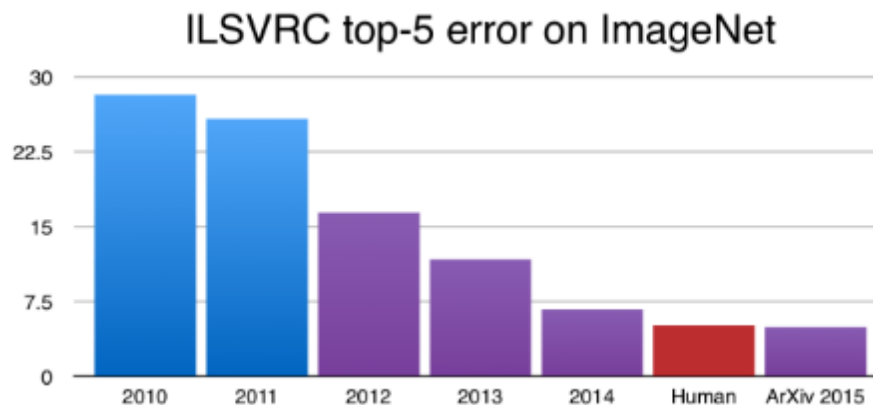
# CNN Performance Benchmarks

- > Now, models are benchmarked on ImageNet ILSVRC set.
- > There is a yearly competition for the highest classification accuracy.
  - ~15 Million photos. Each photo has multiple labels. There are ~80K different labels overall. Labels fall under hierarchies as well.
  - E.g. there are 3,822 different labels under the category 'animal'.



Convolutional  
Neural Networks  
(in purple)

Andrej Karpathy  
in red.



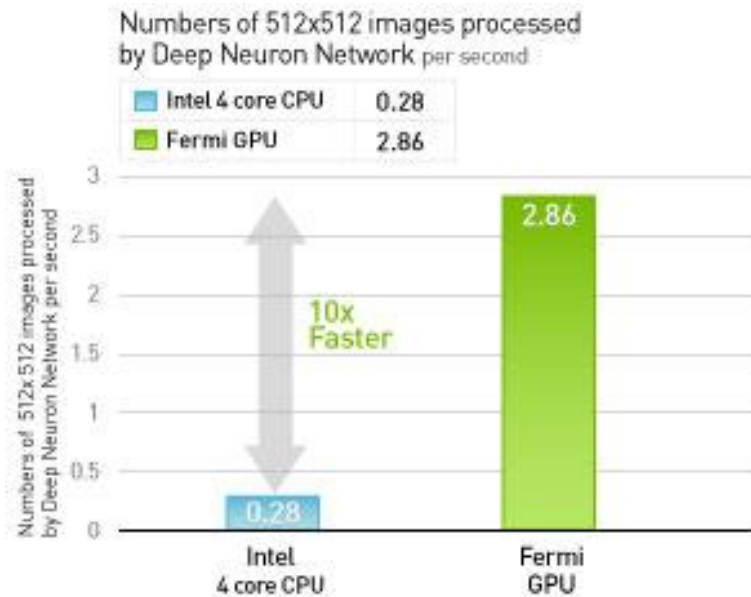
W



# CNN Training Issues

- > These neural networks can have thousands to millions of parameters. How do we deal with training this many?
  - Pooling, convolutions, and dropout help.
  - LOTS of data.
  - Besides this, we need hardware that can calculate an immense number of matrix calculations.
- > Graphic cards (GPUs) are built to handle matrix calculations fast and efficiently.

It's quite common to get 10x to 50x speedup when switching the same code to a GPU.



W

# GPU Technology

- > We need software that can easily transport calculations and data between the CPU and GPU.
  - Torch (Lua), Caffe (C++ w/Python bindings), Theano (Python), and more recently Tensorflow (Python).
- > Recent hardware by NVIDIA is quite spectacular.
  - NVIDIA GPU server cluster: 20 Teraflops!!!!
  - NVIDIA Tesla Series: \$3k-\$10k
  - NVIDIA Compact Deep Learning Super Computer (DGX-1)



The DGX-1 was announced in May 2016 and has a rating of 200 TF. The price has been speculated to be around \$100k. Price will probably drop if NVIDIA mass produces these.



Also note that AWS has GPU instances available.

# CNN Uses

## > Pictures:

- <https://how-old.net/>
- <https://www.what-dog.net/>
- Google's AI research lab, Deepmind, just beat world champion in GO:
  - > <https://deepmind.com/alpha-go>
  - > Five matches, alpha-go won 4-1.
  - > <https://www.tastehit.com/blog/google-deepmind-alphago-how-it-works/>

## > Video:

- Self driving cars: <https://www.youtube.com/watch?v=ZJMtDRbqH40>

## > Speech/Language:

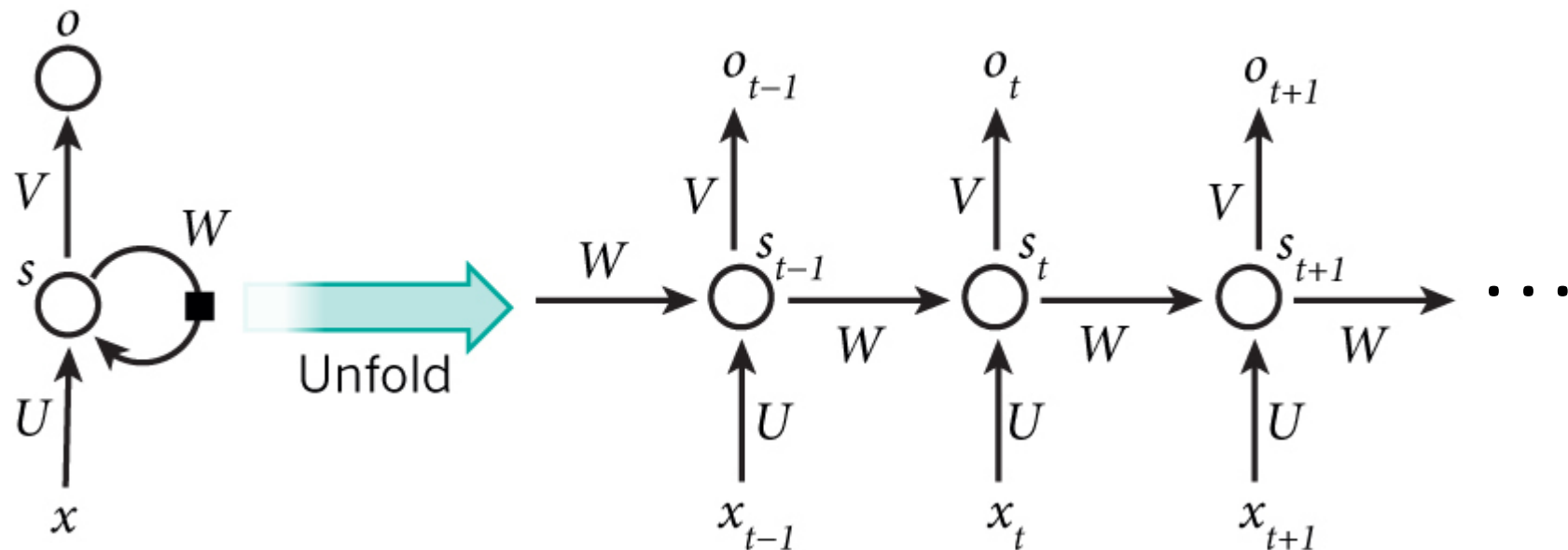
- TayTweets:
  - > <https://twitter.com/tayandyou>
  - > <http://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist> (some parts not entirely work safe)
- Speech Recognition
- Sentiment Analysis

But none of these can deal with sequences... i.e. none of these models can predict the next value by taking into account prior values...

# W

# Recurrent Neural Networks

- > Recurrent Neural Networks (RNN) are networks that can 'see' outputs from prior layers.
- > The most common usage is to predict the next letter from the prior letter. We train this network on large text samples.



$U, V, W$  = Weight Parameters

$S$  = layer

$X$  = input

$O$  = output

**W**

# Recurrent Neural Networks

- > Know that training regular RNNs usually result in diminishing or exploding gradients over time. There are solutions to solve this, and the most common is called LSTM (Long-Short term Memory).
- > LSTM essentially introduces a 'gate', where we have two parameters, a weight of how much to pass on to the next layer and a binary indicator if we forget the prior layer.
- > What are some examples?

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

<https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3#.jstk7fk1b>

<http://blog.otoro.net/2015/12/28/recurrent-net-dreams-up-fake-chinese-characters-in-vector-format-with-tensorflow/>

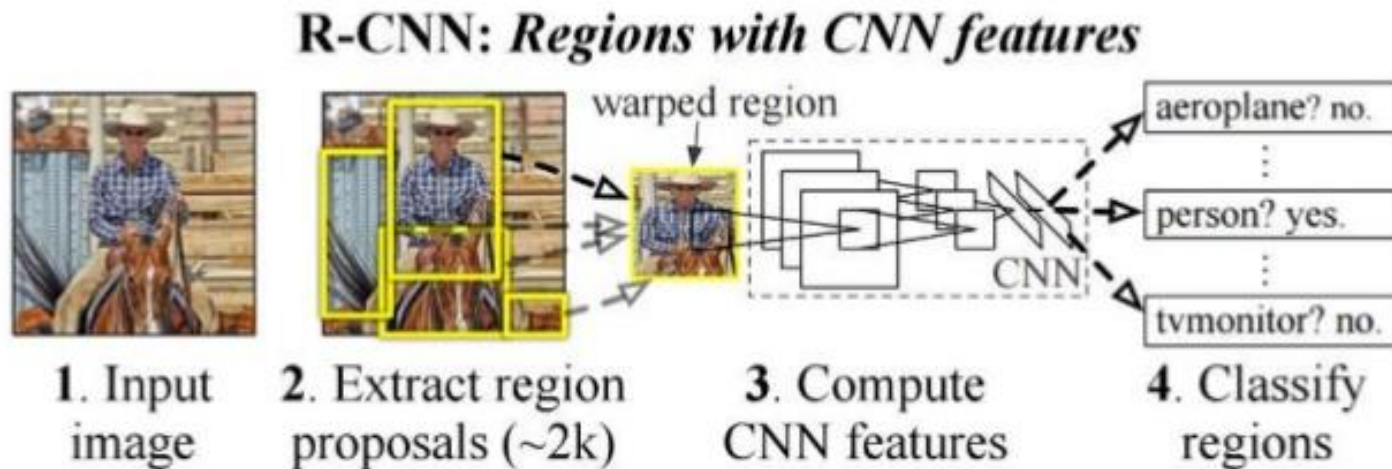
<https://twitter.com/DeepDrumpf>   <https://twitter.com/DeepLearnBern>

<http://arxiv.org/abs/1410.3916v2> (Answering Questions!)



# Regional Convolutional Networks (Regional CNN)

- > We can convolve a whole network over patches of the image and see what regions score highly for a category. (Object Detection)



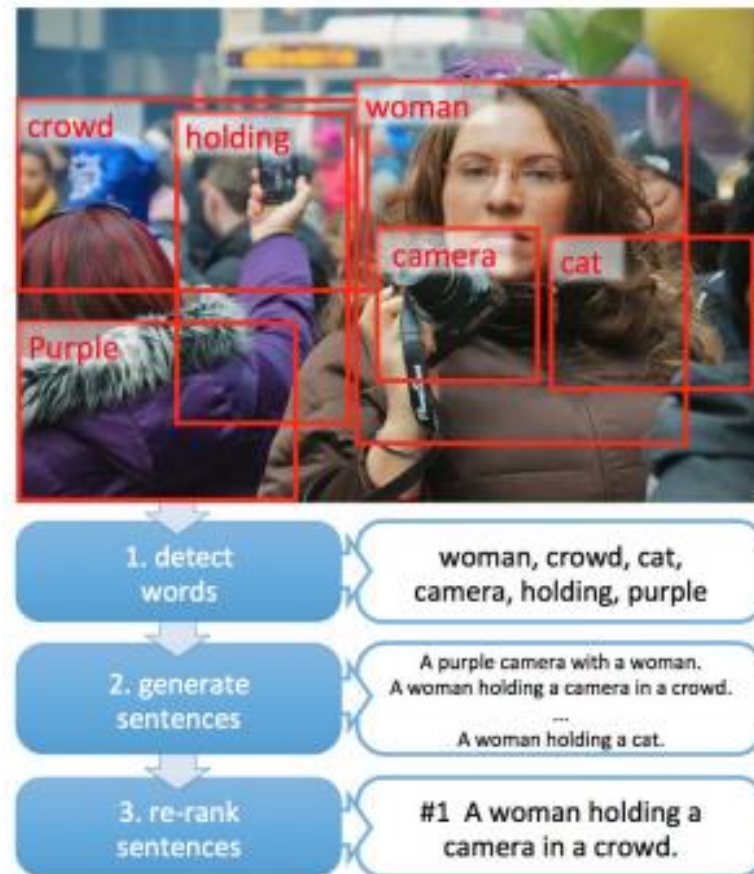
RCNN [Girshick et al. CVPR 2014]

<https://www.youtube.com/watch?v=lr1qVxhGUDw>

W

# Regional CNN + Recurrent NN = Image Captioning

- > We take the object detection output and feed it through a recurrent neural network to generate text describing the image.



W



# More Resources

- > More on CNNs:
- > <https://www.reddit.com/r/MachineLearning/>
- > <https://www.reddit.com/r/deepdream/>
- > <http://karpathy.github.io/>
- > LSTM:
- > <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- > Recent Research: (Jan 16-Mar 16)
- > Residual Conv. Nets: <https://www.youtube.com/watch?v=1PGLj-uKT1w>
- > Determine location of a random image:  
<https://www.technologyreview.com/s/600889/google-unveils-neural-network-with-superhuman-ability-to-determine-the-location-of-almost/>
- > Stochastic Depth Networks: <http://arxiv.org/abs/1603.09382>

Pre-trained models: <http://www.vlfeat.org/matconvnet/pretrained/>

More paper sources: <https://arxiv.org/>

