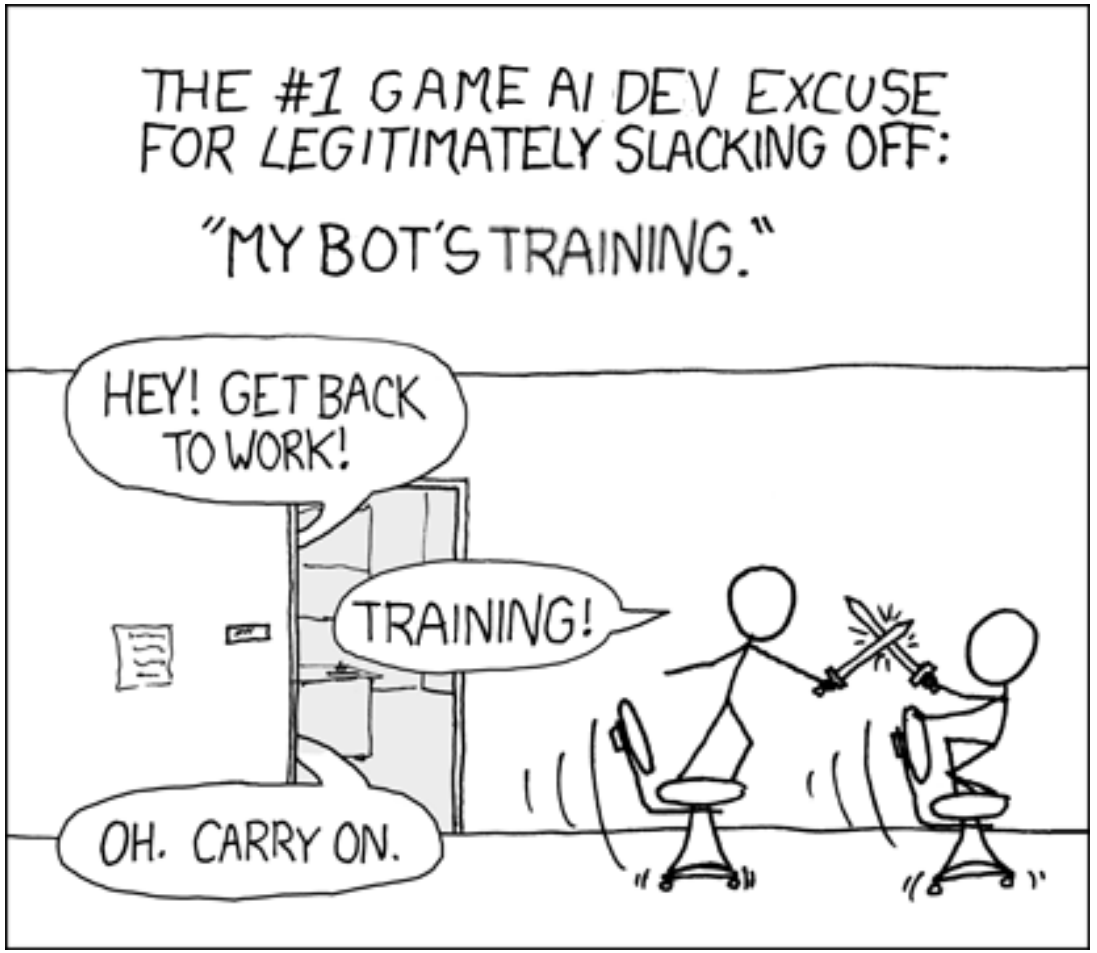
UNIVERSITY of WASHINGTON

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.

Review

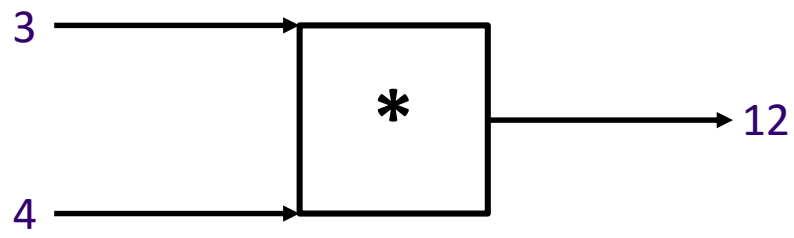
- Bayesian and Computational Statistics
  - Bayesian inference on a coin flip
  - Derive a Posterior from a Prior and Likelihood
  - MCMC for a coin flip
  - MCMC for a bivariate normal
  - Bayesian Linear Regression
  - Bootstrapping for small data sets
  - Bootstrapping for linear regression
  - Computational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

Topics

- Review
- Circuits
- Neural Networks
- Further applications of Neural Networks

Algebraic Operations as Circuit Diagrams

> F(x,y) = x * y

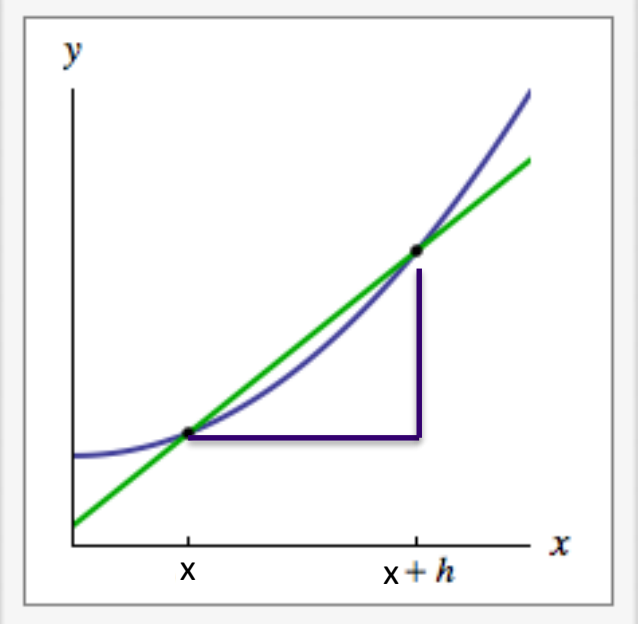


> How can we change the inputs to decrease the output?
> The intuitive answer is to decrease 3 and/or decrease 4.
> Mathematically, the answer is to use the derivative…

The Derivative

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

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



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

The Derivative

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

- This equation works out really nicely for multiplication:

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

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

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

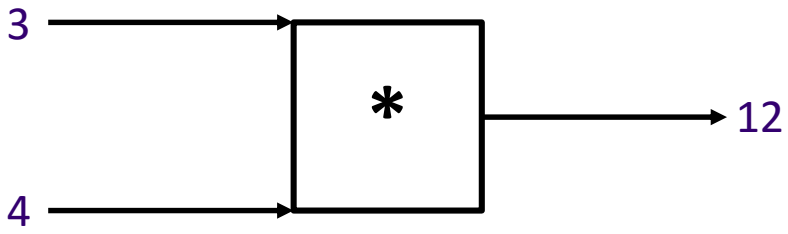
$$\frac{dF(x,y)}{dx} = \lim_{h\to 0} y = y$$

- So…

$$\frac{dF}{dx} = y \qquad \frac{dF}{dy} = x$$

Using the Derivative to change inputs

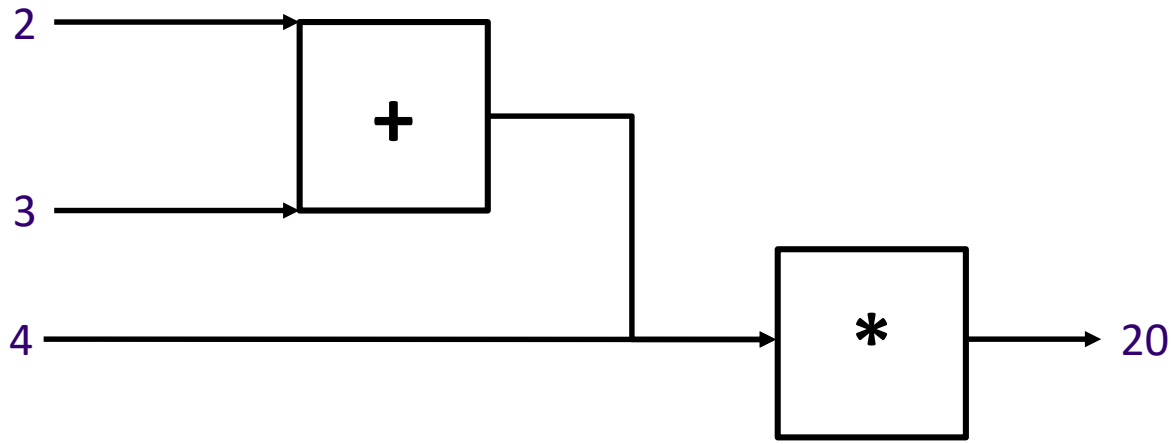
> F(x,y) = x * y



> 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)

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

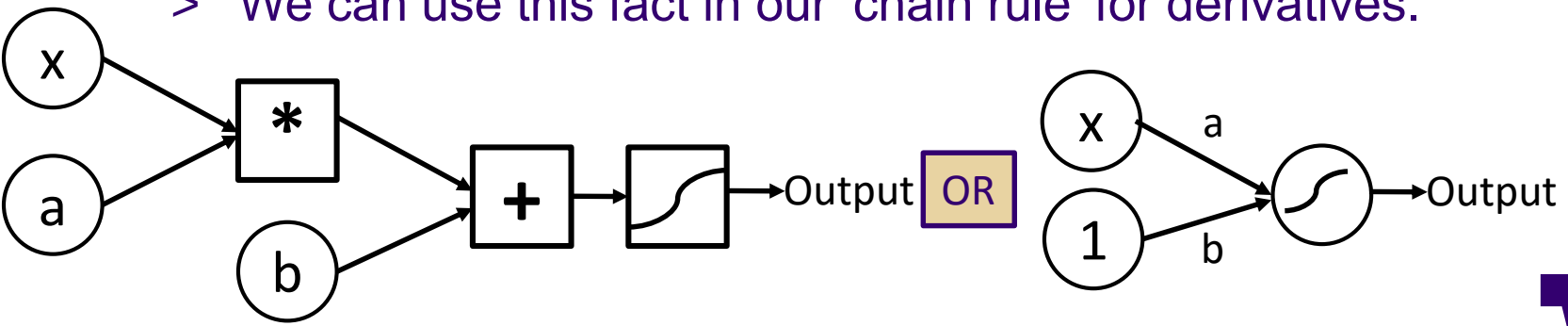
- A neural network is very similar to what we have been doing, except that we bound the outputs between 0 and 1… with a sigmoid function (we've used this before in logistic regression).

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

- 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

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

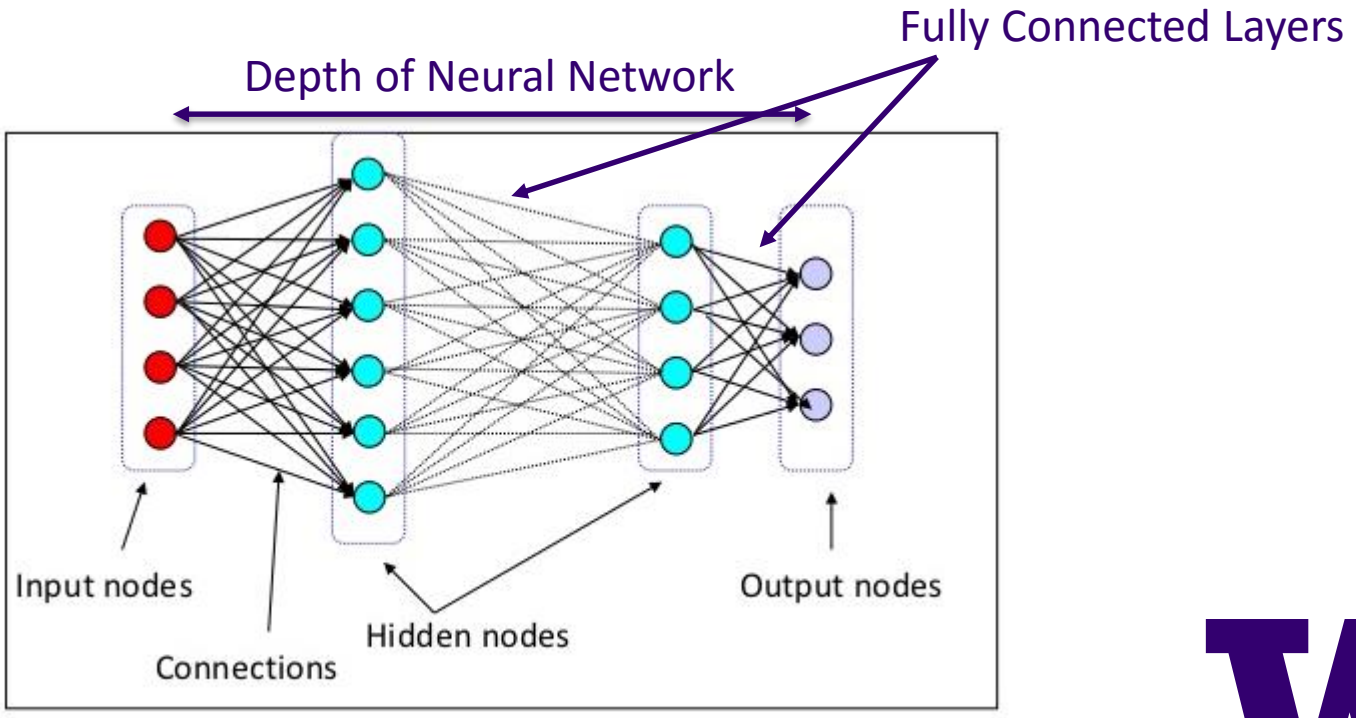
Training a Neural Network

- 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

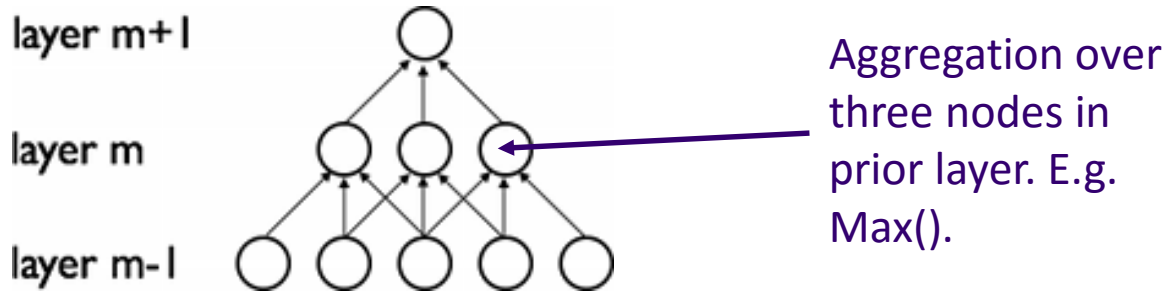
- Neural networks can have many ‘hidden layers’.
- We can make them as deep as we want.
- Neural Networks can also have as many inputs or outputs as we would like as well.



- R-demo

Additional Issues and Using Pooling

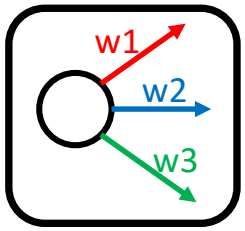
- Larger neural networks can have MANY parameters to fit.
- Solution 1: Provide more training data.
- Solution 2: Reduce parameters via aggregation steps in the model, this is called “pooling”.
- Types of pooling:
  - Maximum: Take the maximum of a group of nodes in the prior layer.
  - Minimum
  - Mean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

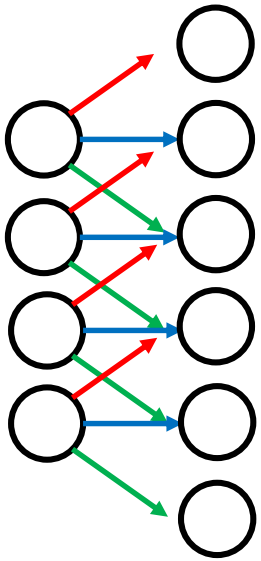


Aggregation over three nodes in prior layer. E.g. Max().

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:



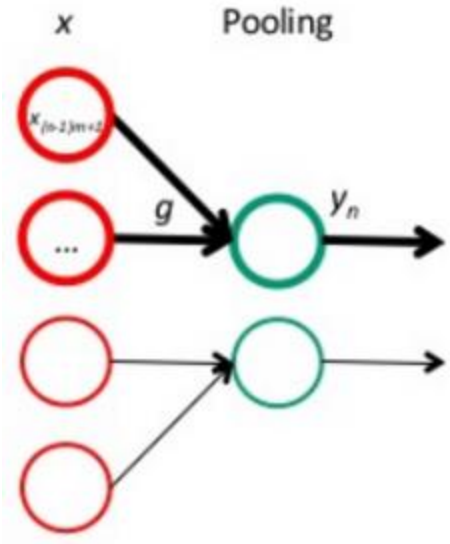


There are only three weights in this layer, w1, w2, and w3

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

Subsampling Layers

- Yet another way to limit the amount of parameters is to group inputs together.
- This is different than pooling because we are not allowing overlap of nearby inputs.
- Types: Mean sampling, Max sampling, Min sampling…



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 → $(-0.2, 0.4, 1.1)$

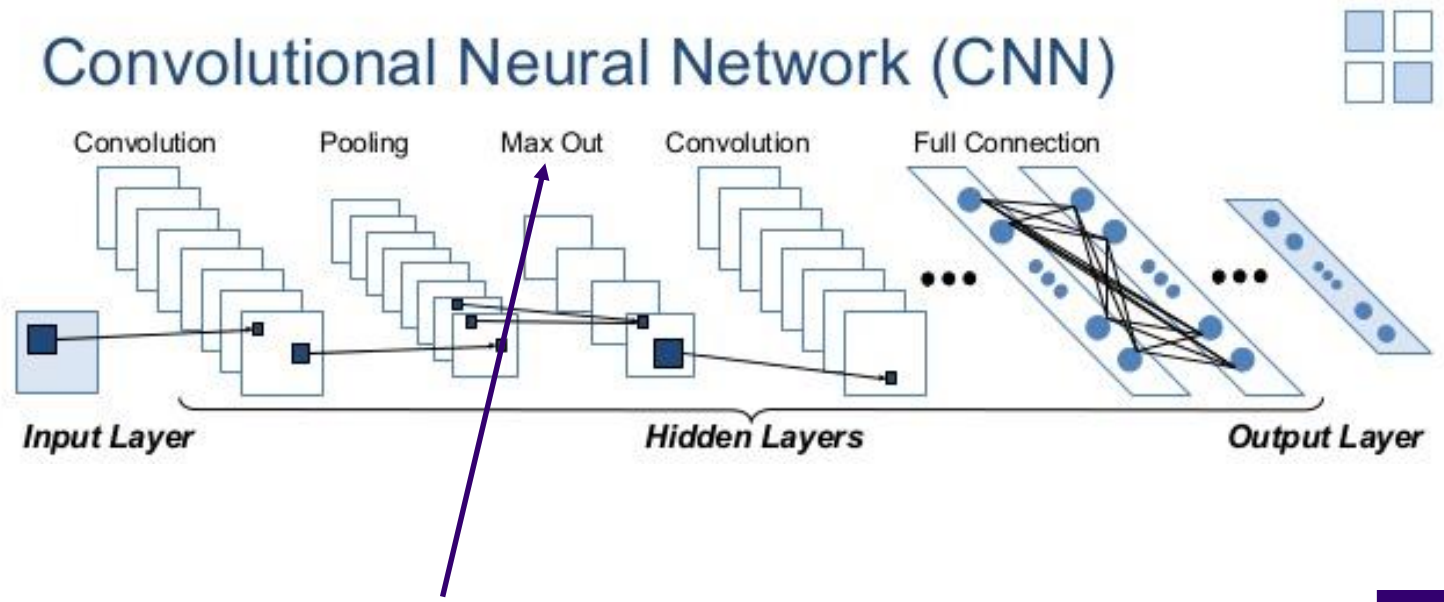
- (Remember, we are using back propagation to pull the right class output up and the wrong classes down for each example.)
- We usually convert these to probabilities via a 'Softmax' function.

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

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

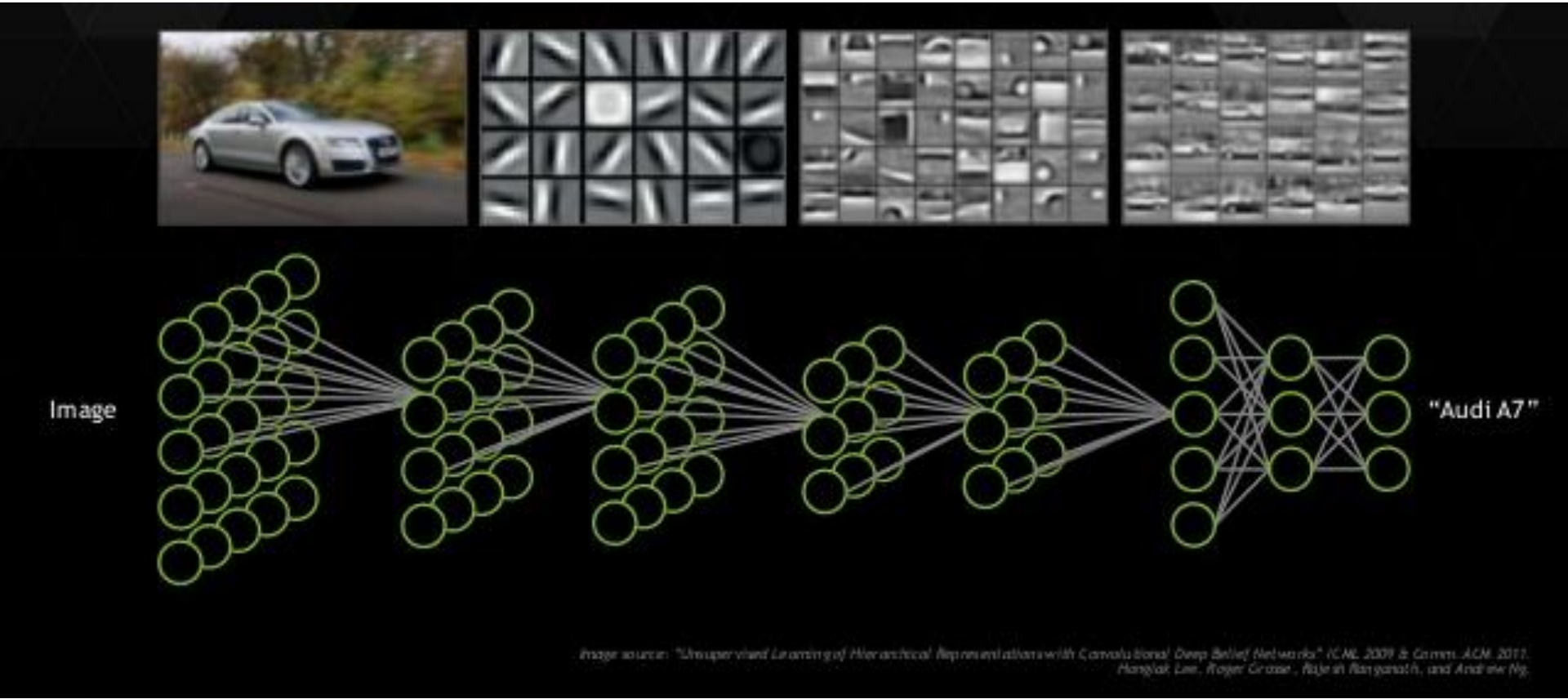
Neural Network Layer Shorthand

- 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:



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)

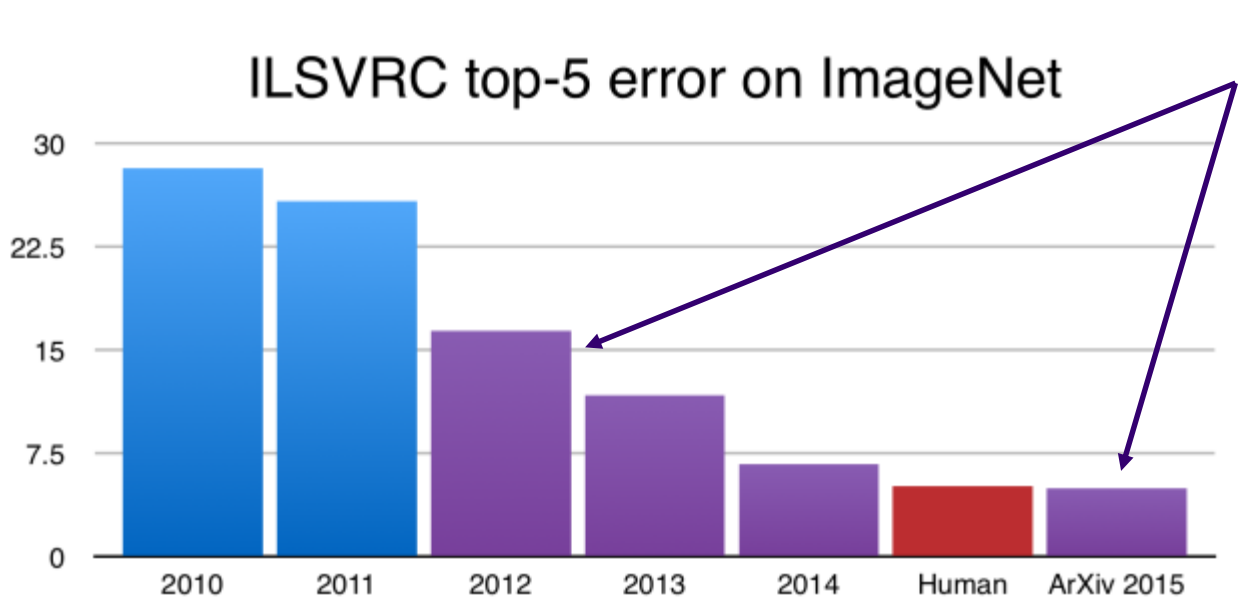
- Another possibility is to train part of the network that learns basic shapes and edges on a “style picture” and train the rest on a “source picture”. The loss function here is how close small sections of the source resemble the style.

Recoloring of black and white photos

https://vimeo.com/139123754

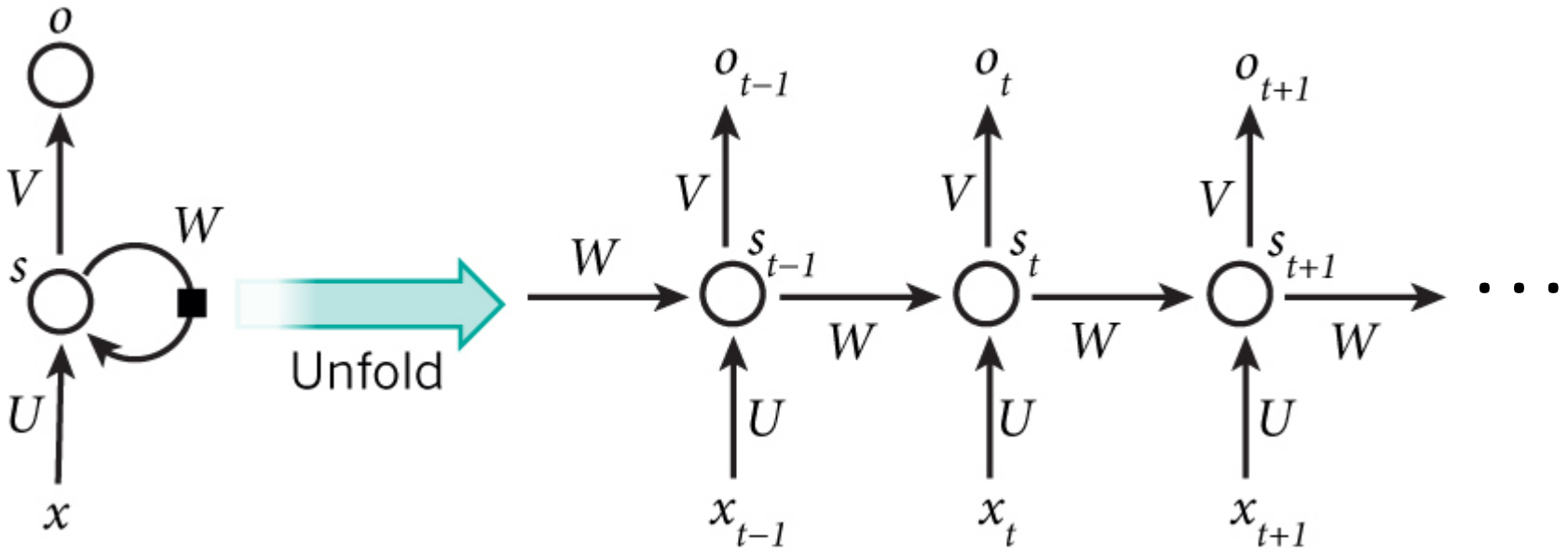
CNN Performance Benchmarks

- CNN models were benchmarked on the MNIST handwritten digit recognition set, but there are only 70K examples. More needed now.
- Now, models are benchmarked on ImageNet ILSVRC set.
  - ~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'.



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

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/

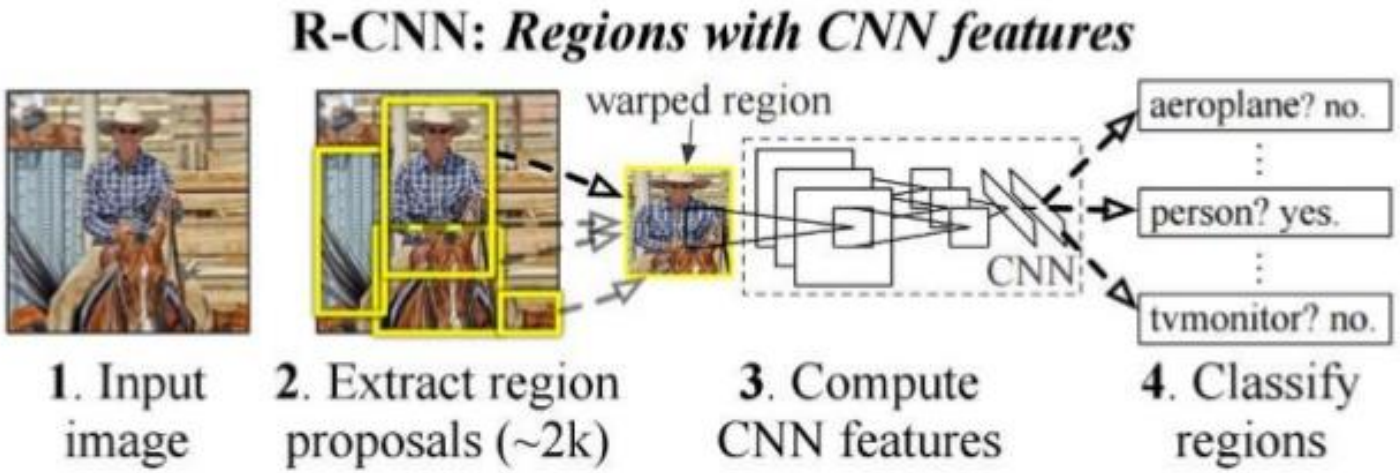
https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3#.jstk7fk1b

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

https://twitter.com/DeepDrumpf

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]

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.

* Microsoft Research 2015

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: (Dec 15/Jan 16)
- https://www.youtube.com/watch?v=1PGLj-uKT1w
- https://www.technologyreview.com/s/600889/google-unveils-neural-network-with-superhuman-ability-to-determine-the-location-of-almost/
- https://scholar.google.com/scholar?as_vis=1&q=%22neural+network%22&hl=en&scisbd=1&as_sdt=1,48

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