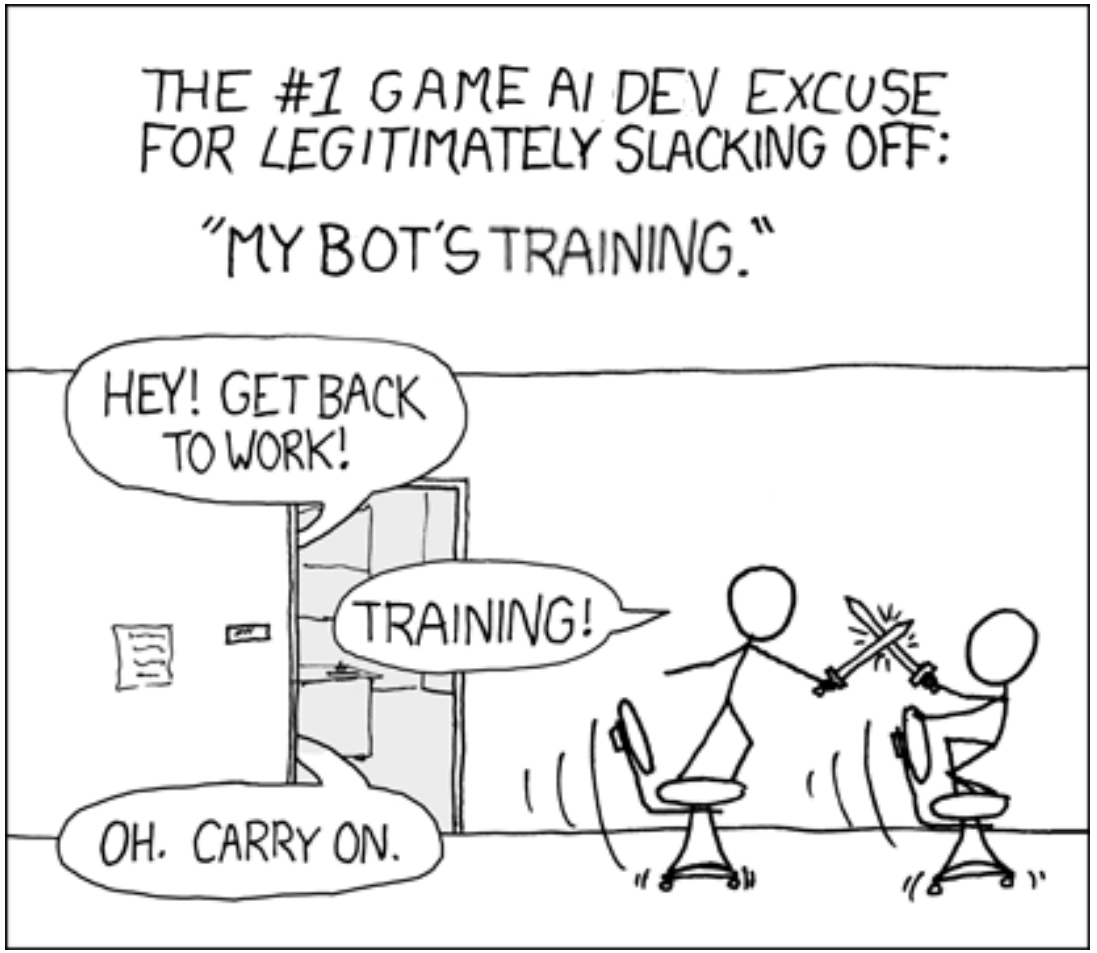
UNIVERSITY of WASHINGTON

Data Science UW Methods for Data Analysis

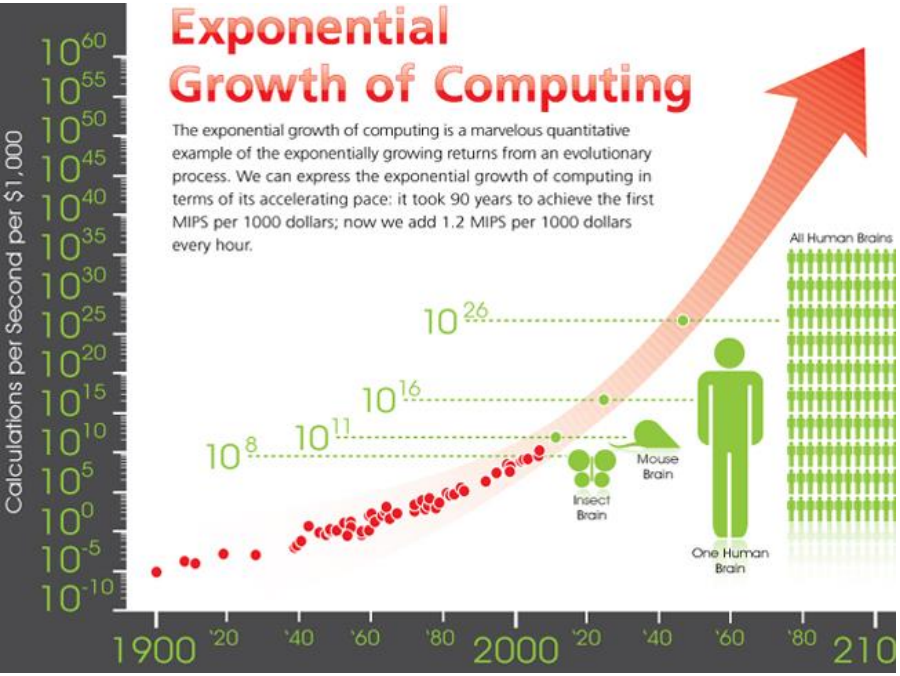
Introduction to Neural Networks
Extra Topics
Nick McClure

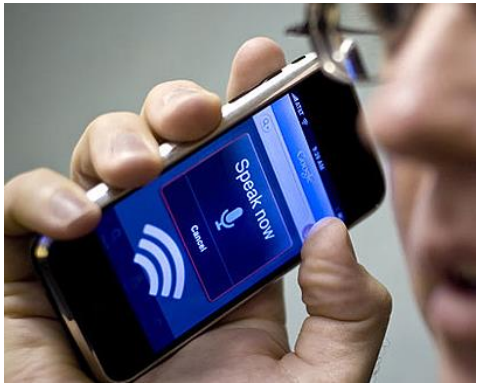

THE #1 GAME AI DEV EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY BOT'S TRAINING."
HEY! GET BACK TO WORK!
TRAINING!
OH. CARRY ON.

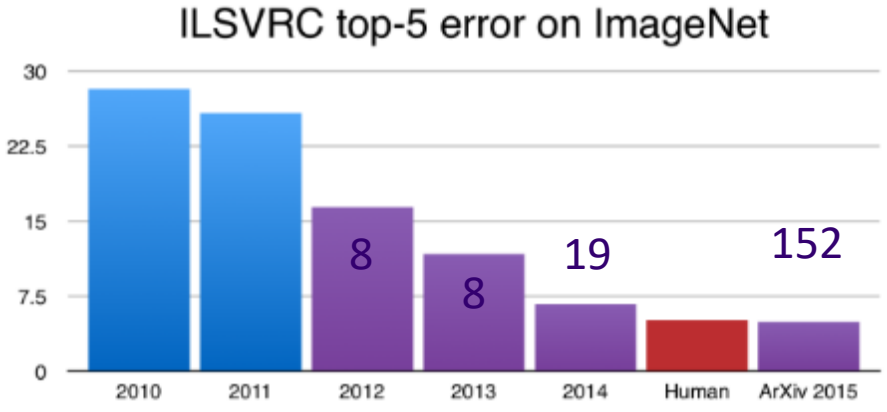
Topics

- Circuits
- Neural Networks
- Further applications of Neural Networks

Why Does Any of This Matter?

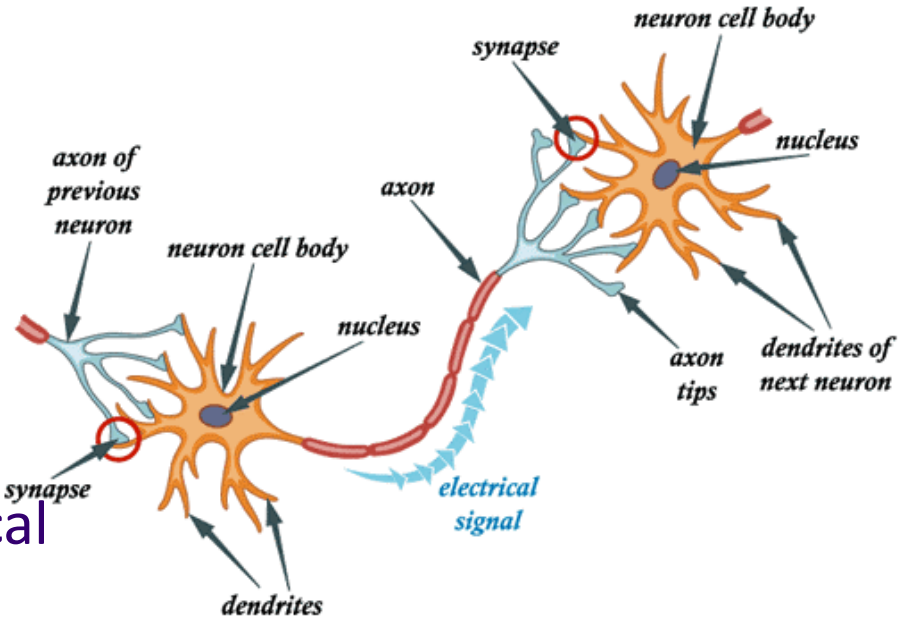






Why Neural Networks?

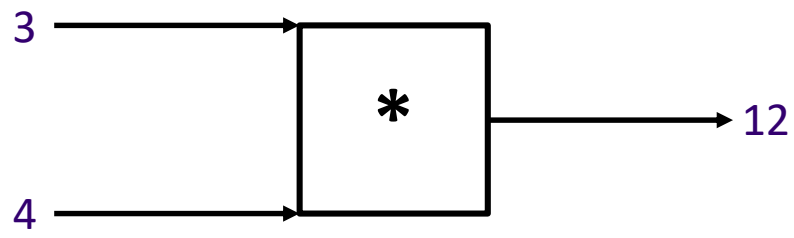
- The human brain can be considered to be one of the best processors. (Estimated to contain ~10^{11} neurons.)
- Studies show that our brain can process the equivalent of ~20Mb/sec just through the optical nerves.
- If we can copy this design, maybe we can solve the "hard for a computer – easy for humans" problems.



- Speech recognition
- Facial identification
- Reading emotions
- Recognizing images
- Sentiment analysis
- Driving a vehicle
- Disease diagnosis

Algebraic Operations as Circuit Diagrams

- F(x,y) = x * y

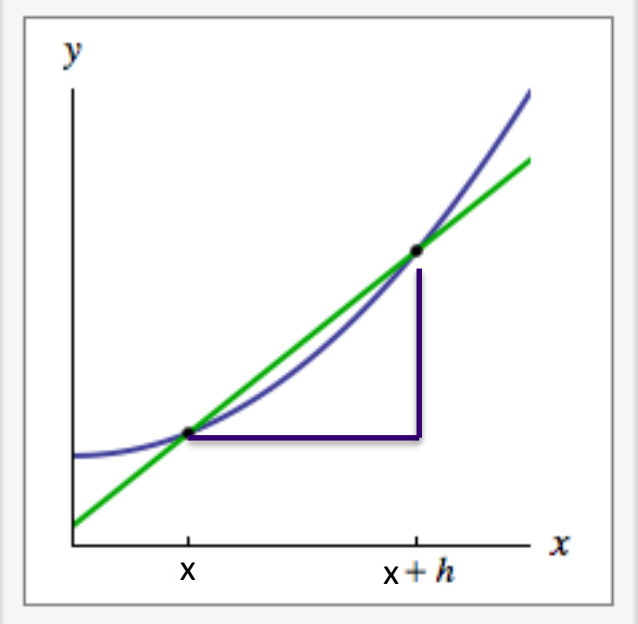


- How can we change the inputs to decrease the output?
- The intuitive answer is to decrease 3 and/or decrease 4.
- Mathematically, the answer is to use the derivative…

The Derivative (i.e. the Gradient)

- 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…

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

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 **addition** too:

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

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

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

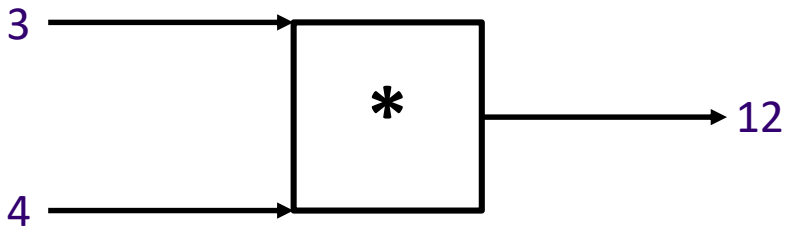
$$\frac{dF(x,y)}{dx} = \lim_{h \to 0} 1 = 1$$

- So…

$$\frac{dF}{dx} = 1 \qquad \frac{dF}{dy} = 1$$

Using the Derivative to change inputs

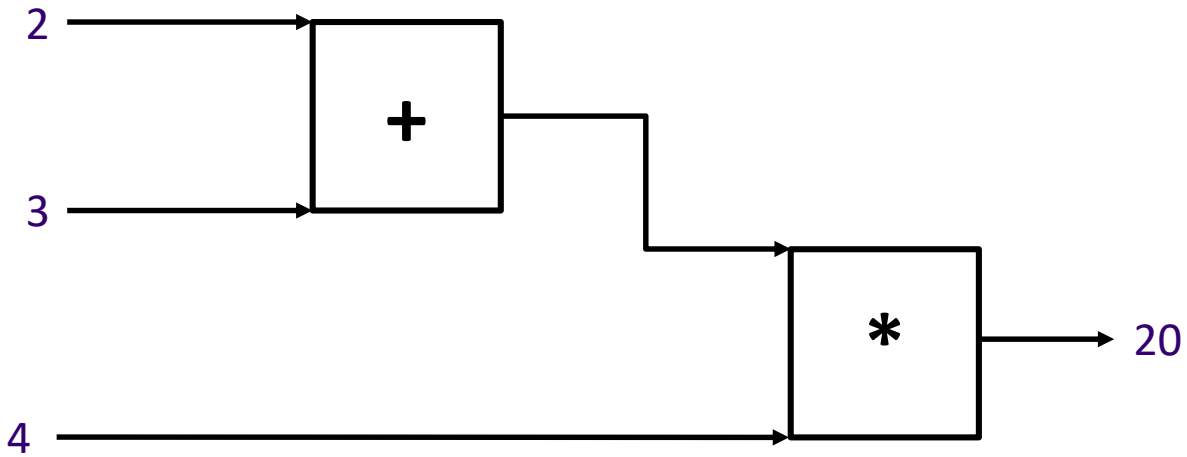
> F(x,y) = x * y



> 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:

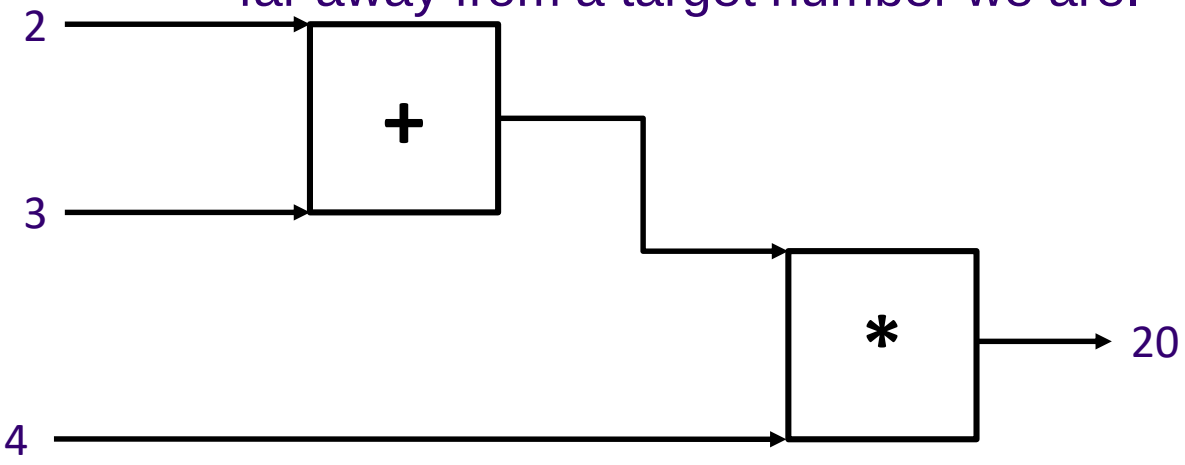
$$\frac{dF}{dx} = \frac{dF}{dG} \cdot \frac{dG}{dx}$$

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

- R Demo

Pursuing a Goal

> If we know how we should change the parameters to decrease or increase the output, we can change the step size according to how far away from a target number we are.



> If we are above a goal of ‘17’, decrease the output, if we are below, increase the output.

> R Demo

A Simple Neural Network

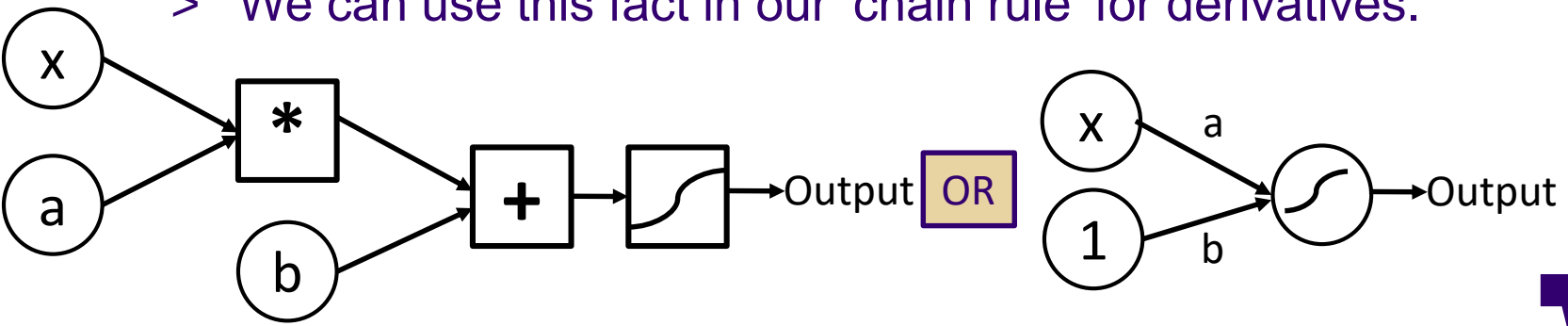
- A neural network is very similar to what we have been doing. So far, we cannot fit to non-linearities. We introduce a non-linear function at the end to address this. For starters, consider the sigmoid function:

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

- 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

Why Activation Functions?

- Turns out, no matter how many additions and multiplications we pile on, we will never be able to model non-linear outputs without having non-linear transformations.
- 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

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

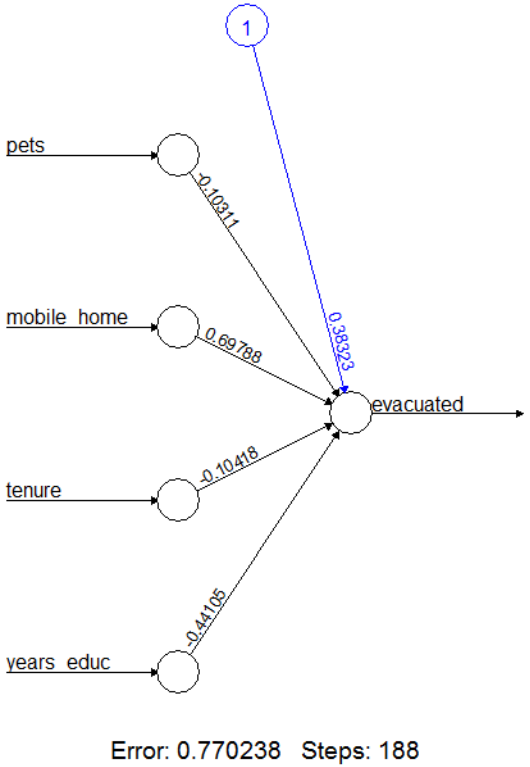
Training a Neural Network

- 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.

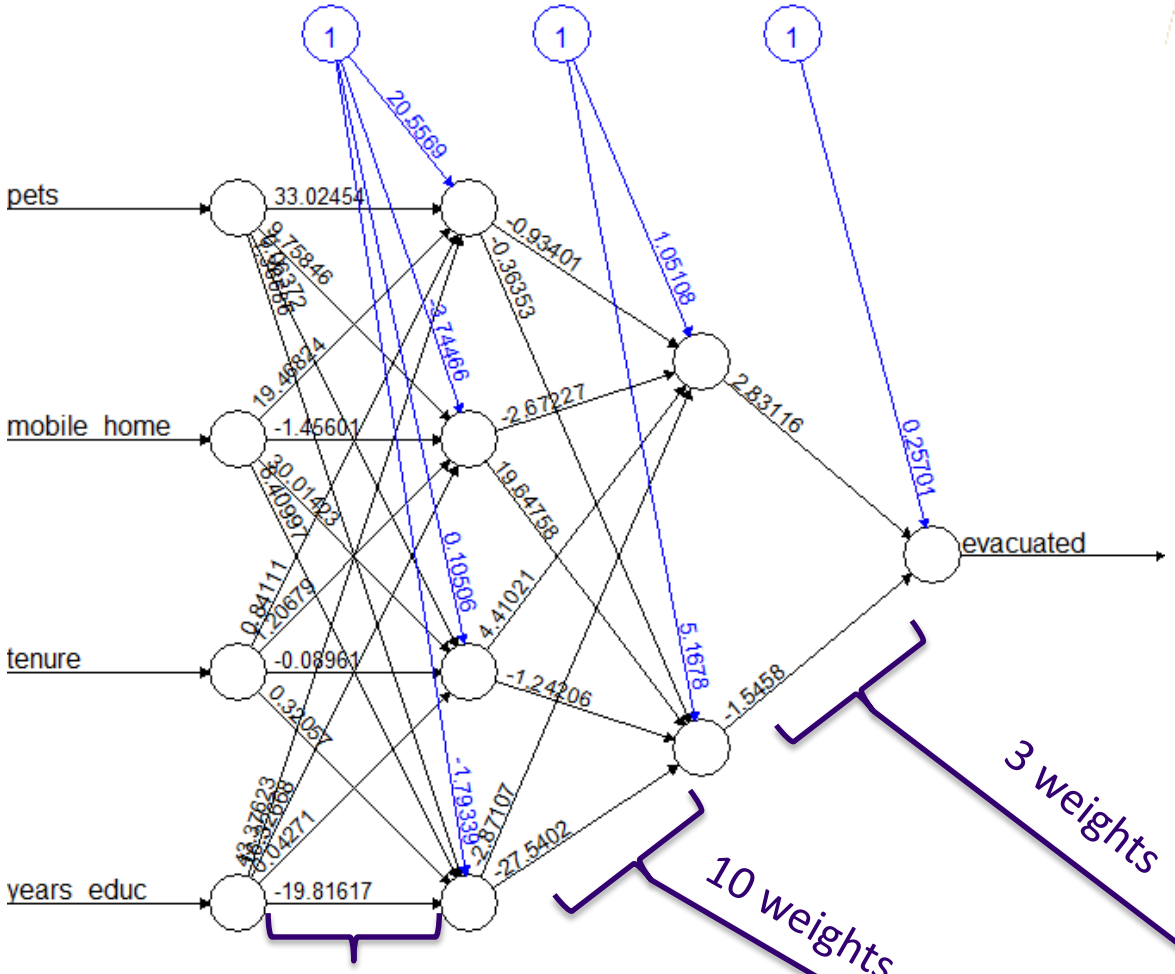
- R-demo

Neural Network = Matrix Multiplication

> A neural network is *mostly* just matrix multiplication…



$$[p \quad m \quad t \quad y \quad 1] \cdot \begin{bmatrix} -0.10311 \\ 0.69788 \\ -0.10418 \\ -0.44105 \\ 0.38323 \end{bmatrix} = output$$



$$((([p \quad m \quad t \quad y]\begin{bmatrix} 33.02 & 9.76 & 0.06 & 1.37 \\ 19.47 & -1.46 & 30.01 & 8.41 \\ 0.84 & 1.21 & -0.09 & 0.32 \\ 43.38 & 16.33 & 0.04 & -19.82 \end{bmatrix} + \begin{bmatrix} 20.56 \\ -3.74 \\ 0.11 \\ -1.79 \end{bmatrix})\begin{bmatrix} -0.93 & -0.36 \\ -2.67 & 19.65 \\ 4.41 & -1.24 \\ -2.87 & -27.54 \end{bmatrix}) + \begin{bmatrix} 1.05 \\ 5.17 \end{bmatrix})\begin{bmatrix} 2.83 \\ -1.55 \end{bmatrix} + 0.26$$

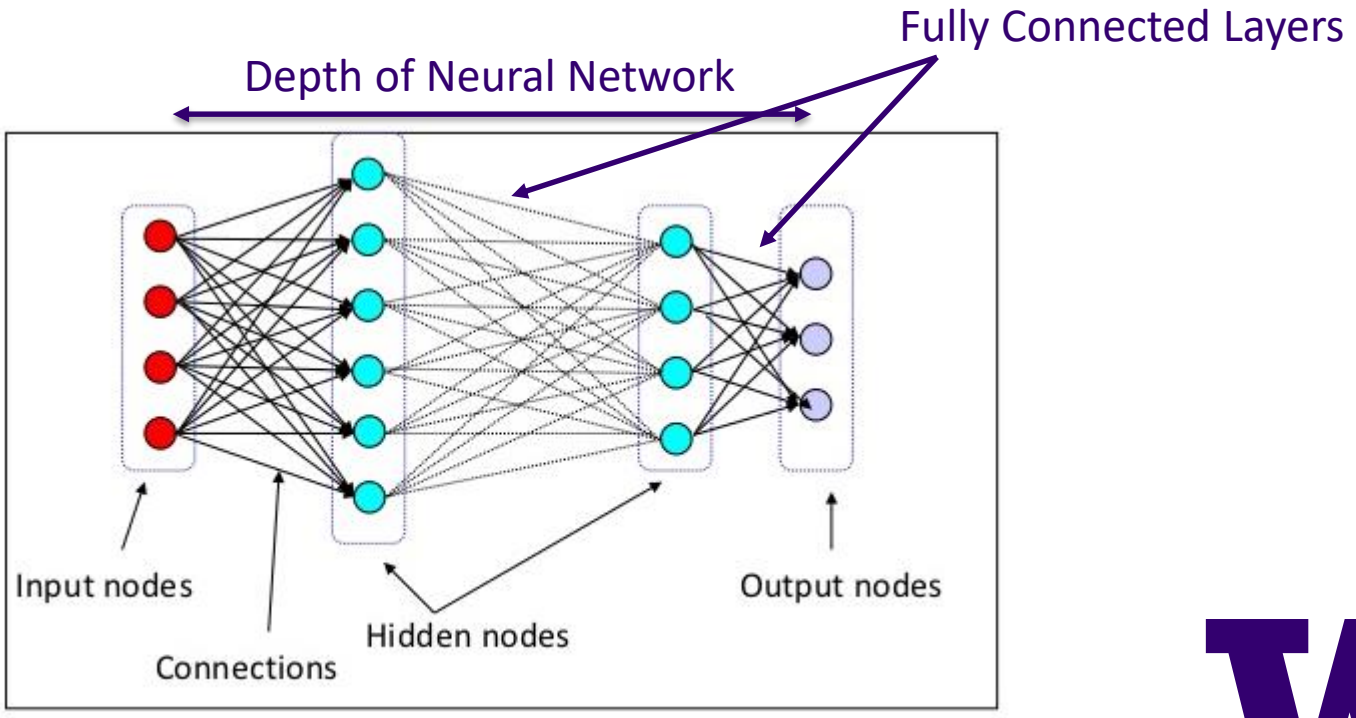
First Step

Second Step

Last Step

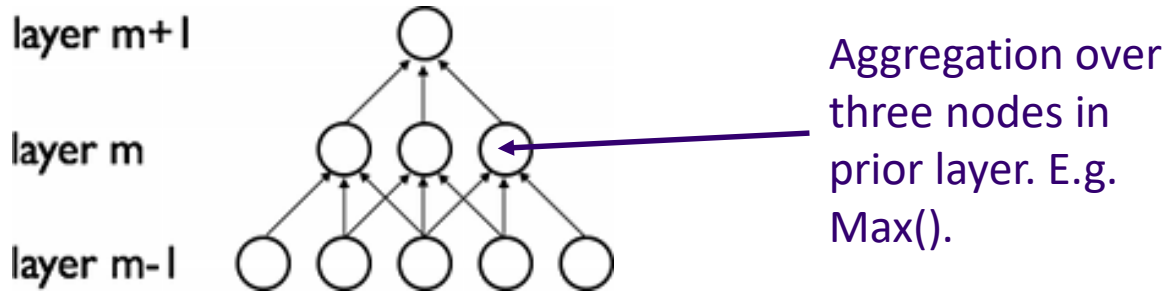
Larger Neural Networks

- 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.



Additional Issues and Using Pooling/subsampling

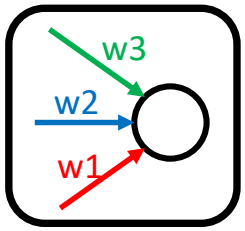
- Larger neural networks can have MANY parameters to fit.
- Solution 1: Provide more training data.
- Solution 2: Reduce parameters via aggregation steps in the model, this is called “pooling”.
- Types of pooling:
  - Maximum: Take the maximum of a group of nodes in the prior layer.
  - Minimum
  - Mean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

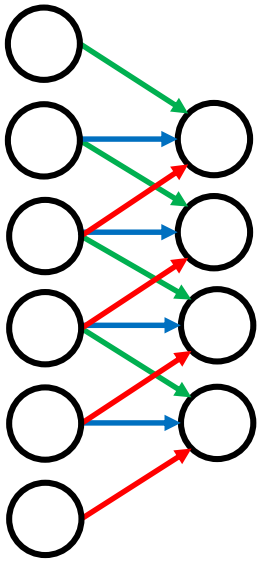


Aggregation over three nodes in prior layer. E.g. Max().

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:



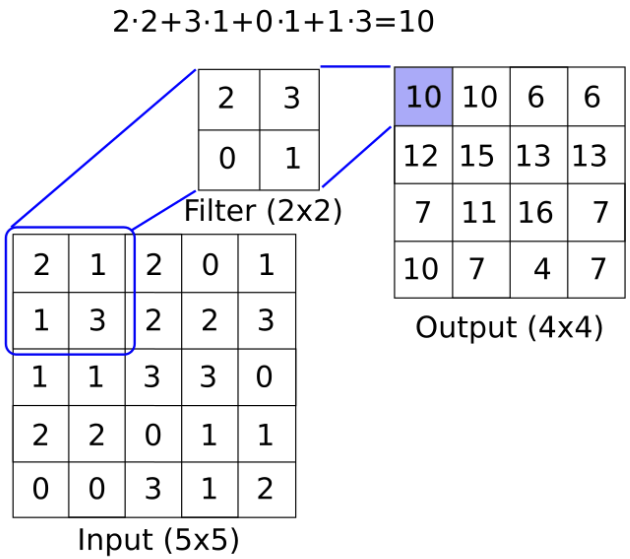


There are only three weights in this layer, w1, w2, and w3

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

Convolution Layers

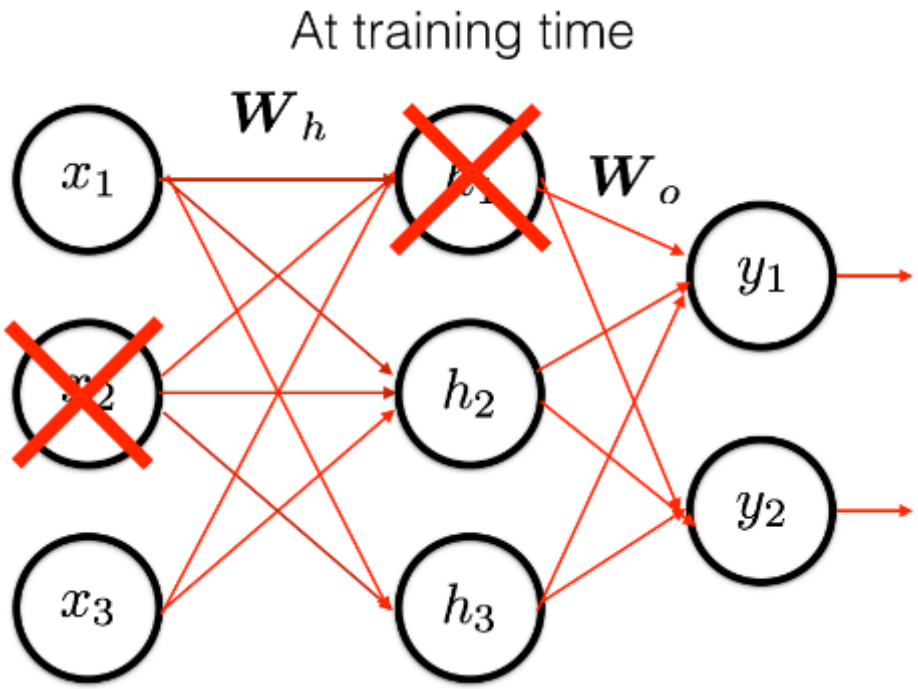
> Convolution can also act on 2 dimensional input:



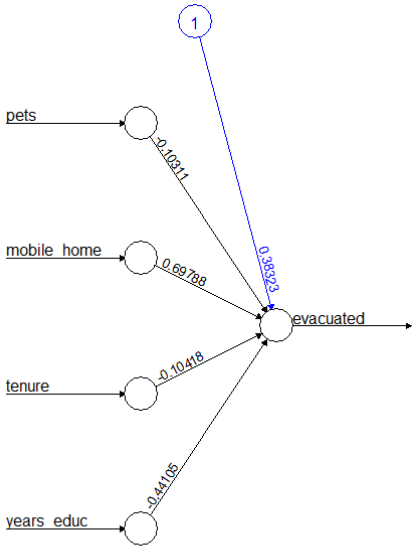
> Great animation of convolution acting on two dimensions:
  - http://cs231n.github.io/convolutional-networks/

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.



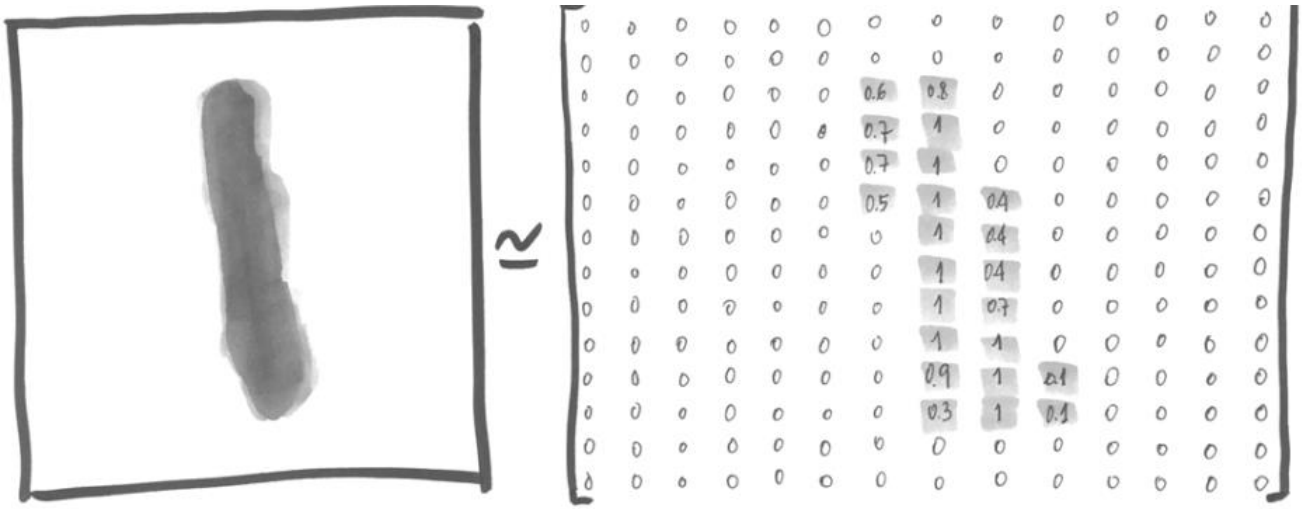
Dealing with 2-Dimensional Input



Error: 0.770238 Steps: 188

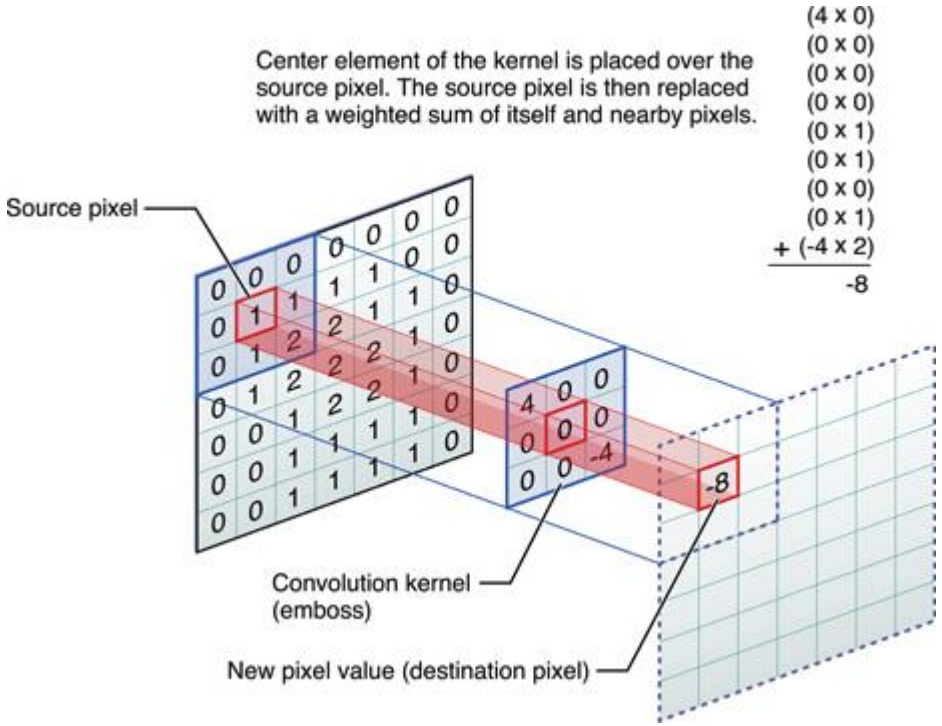
$$\begin{bmatrix} p & m & t & y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p & m & t & y & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.10311 \\ 0.69788 \\ -0.10418 \\ -0.44105 \\ 0.38323 \end{bmatrix} = output$$

> Instead of a 1-dimensional input, we can have 2-dimensions as an input:



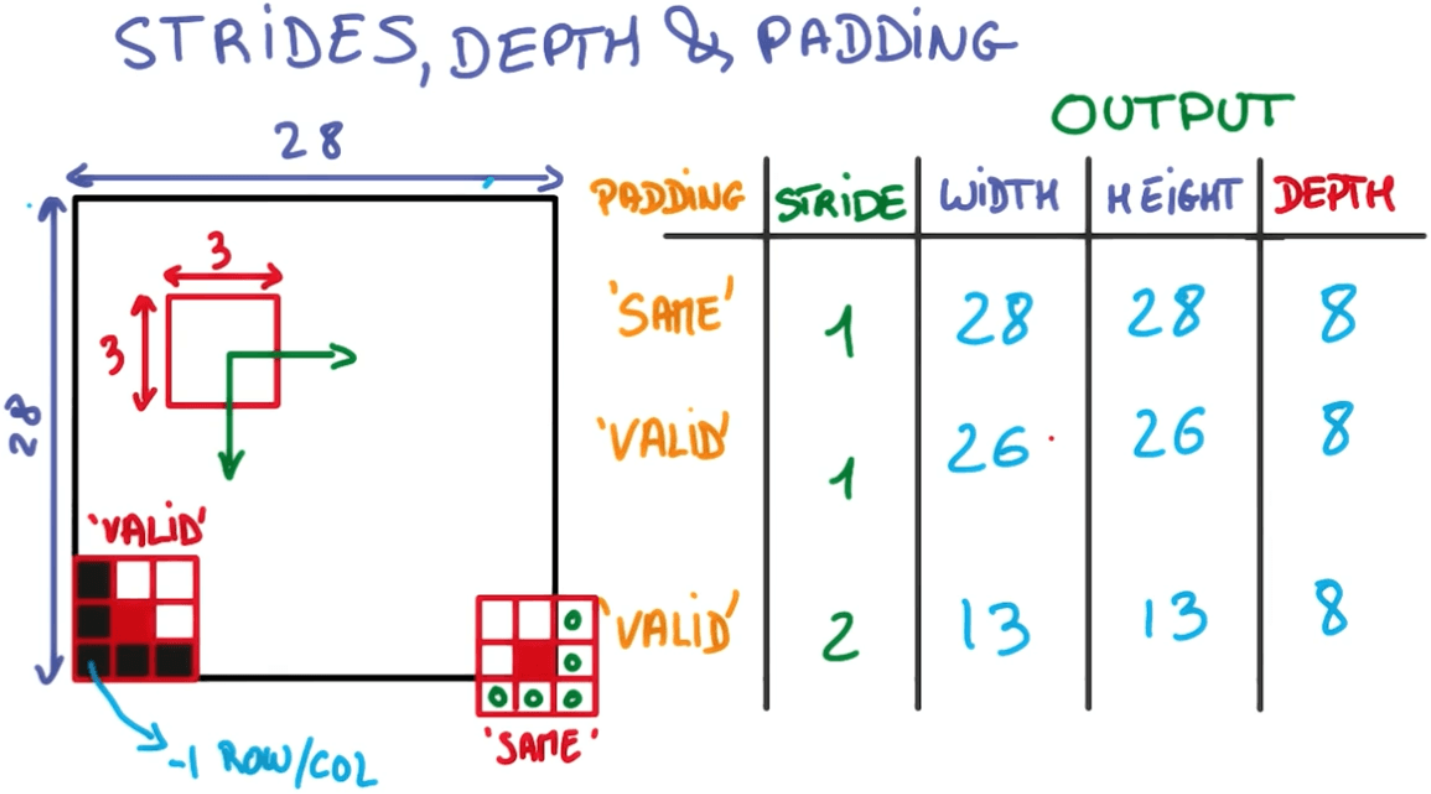
Convolutional Filters on Images

- We can use combinations of pooling/subsampling, and convolutional layers to apply this data transformation to images.
- How does the Convolutional Filter work on 2 dimensions?



Convolutional Filters on Images

> With Convolutional filters, we need to be aware of how we handle the size, strides, and padding.



> In any dimension: if I is the input length, F is the filter size, P is the padding amount, and O is the output size, then

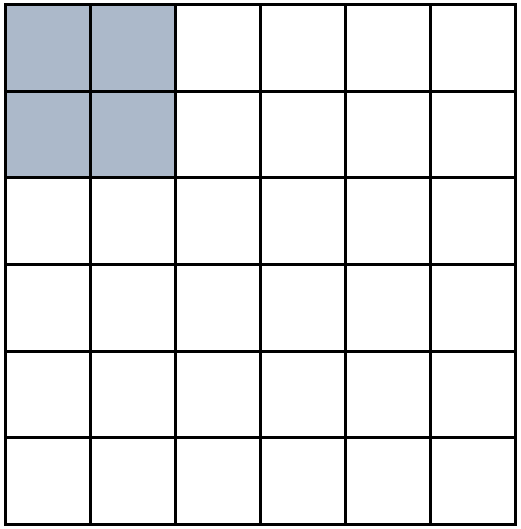
$$O = \frac{I - F + 2P}{S} + 1$$

Convolutional Filters on Images

- Try a simple example:

$$O = \frac{I - F + 2P}{S} + 1$$

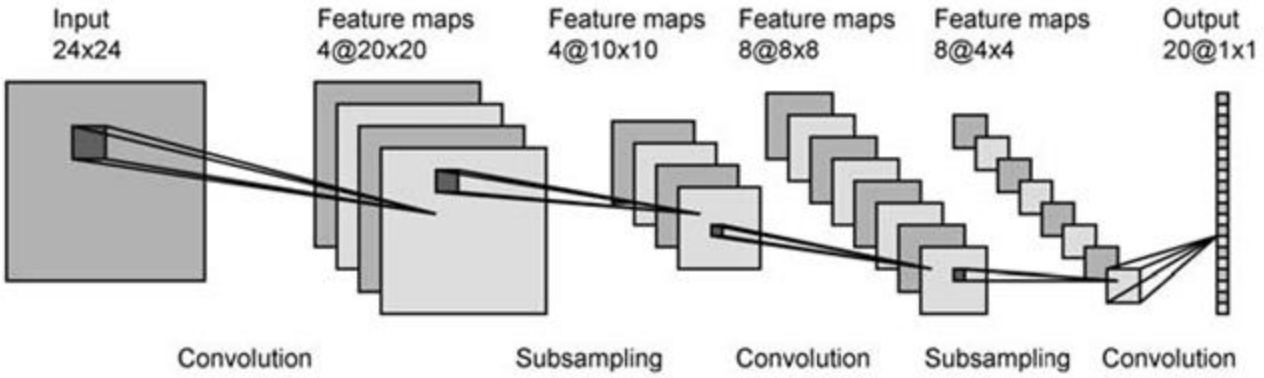
- 6x6 image, 2x2 filter
  - 0 padding, stride=1



  - 0 padding, stride = 2
  - 1 padding, stride = 1

Convolutional Neural Networks

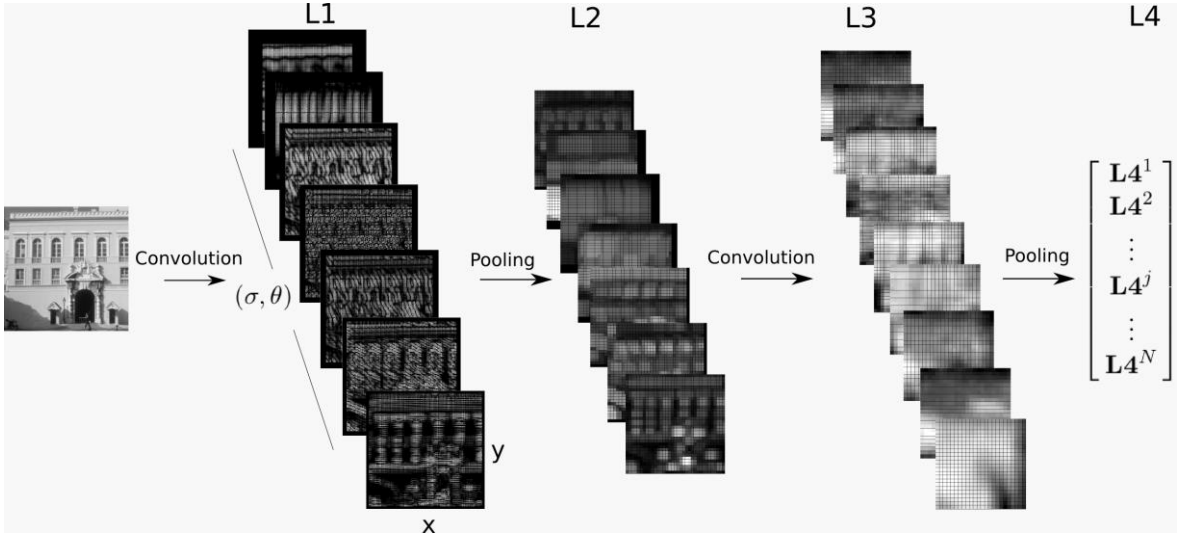
- Note that we are not limited to a filter of 2 dimensions, we can have a 3 dimensional filter that will create multiple feature spaces:



- 24x24 image + 5x5 filter, stride = 1, valid padding = 20x20 output
- Then subsampling a 2x2 window with stride 2 results in a 10x10 output
- 10x10 'image' + 3x3 filter + stride = 1, valid padding = 8x8
- Then subsampling a 2x2 window with stride 2 results in a 4x4 output
- 4x4 'image' + 4x4 filter + stride=0, valid padding = 1x1 output

Convolutional Neural Networks

> Note that we are not limited to a filter of 2 dimensions, we can have a 3 dimensional filter that will create multiple feature spaces:



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 → $(-0.2, 0.4, 1.1)$

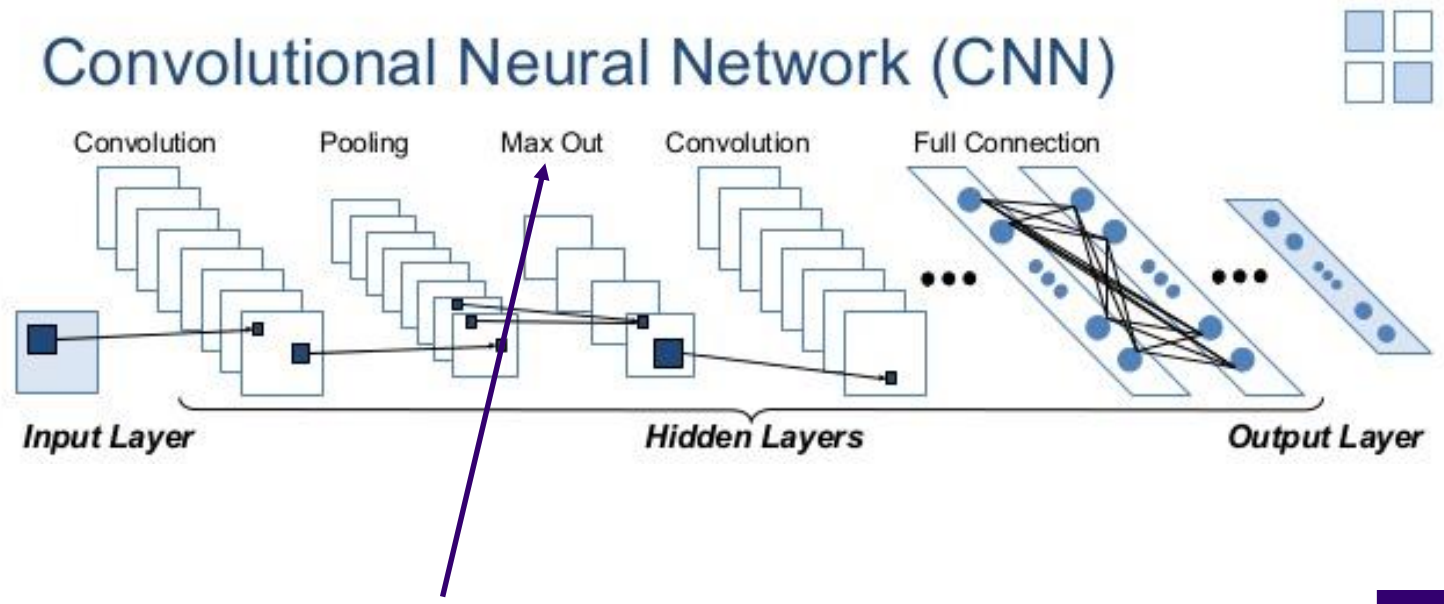
- (Remember, we are using back propagation to pull the right class output up and the wrong classes down for each example.)
- We usually convert these to probabilities via a 'Softmax' function.

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

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

Neural Network Layer Shorthand

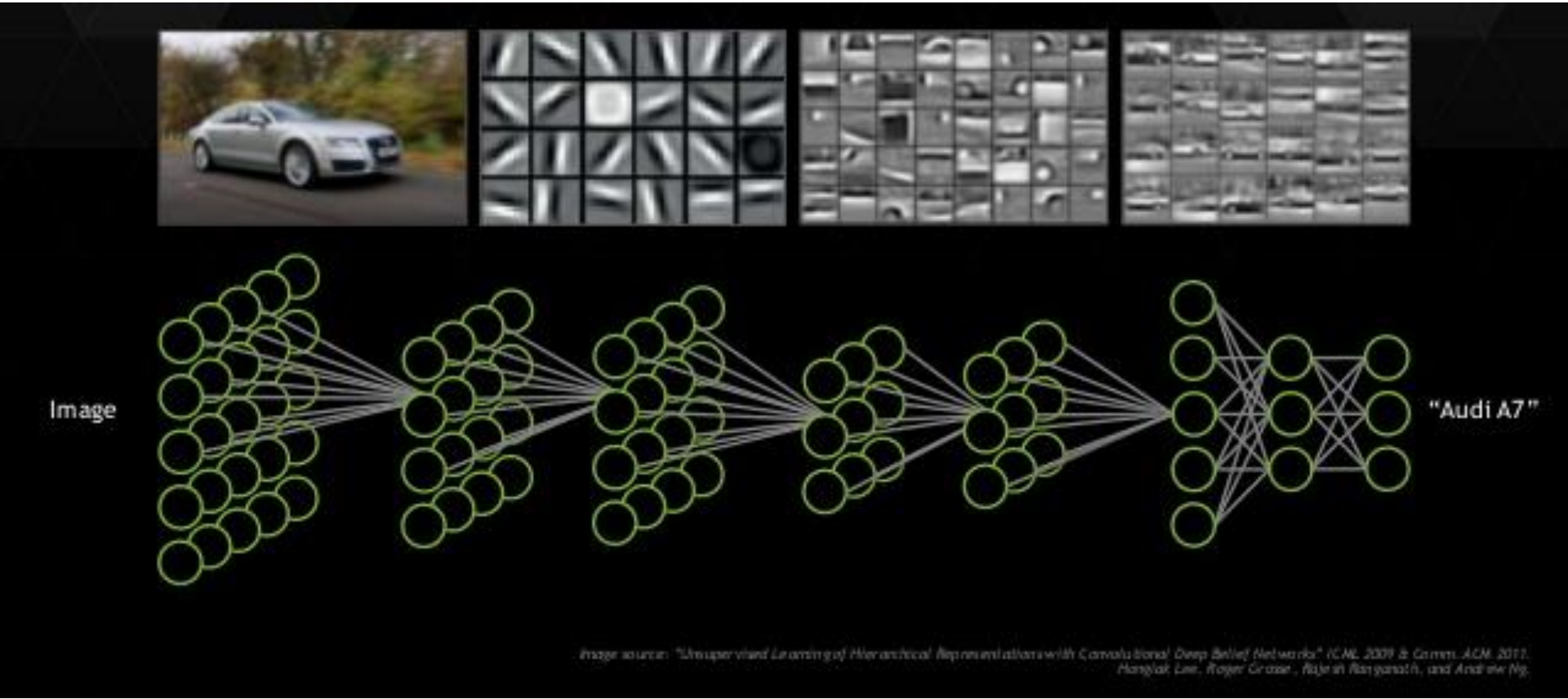
- Since neural networks can get quite large, we do not want to write out all the connections nor nodes.
- Here's how to convey the meaning in a shorter, less complicated way:



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

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)

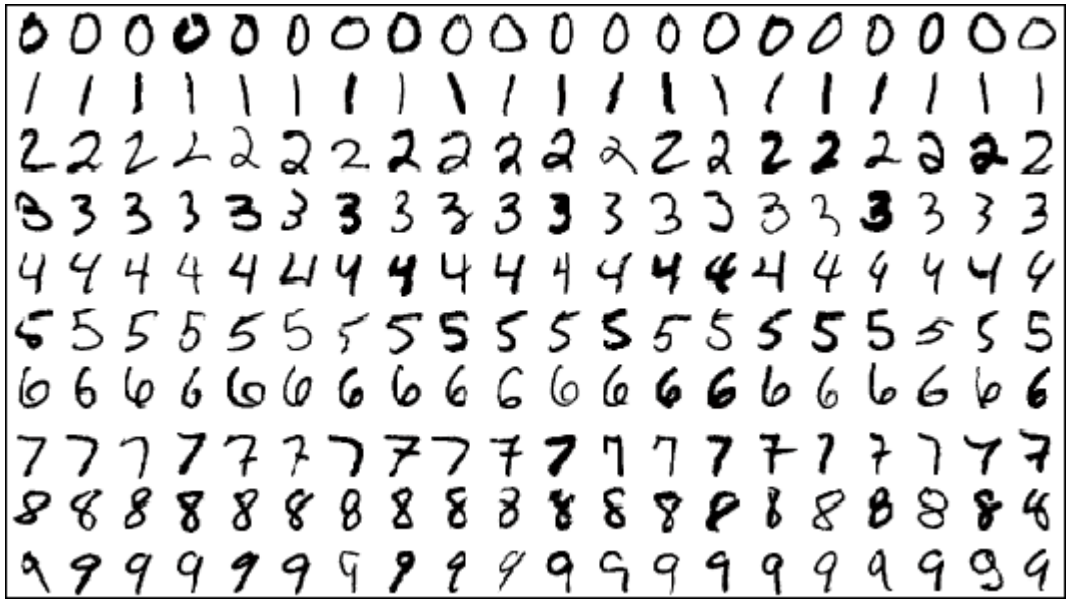
- Another possibility is to train part of the network that learns basic shapes and edges on a “style picture” and train the rest on a “source picture”. The loss function here is how close small sections of the source resemble the style.

Recoloring of black and white photos

https://vimeo.com/139123754

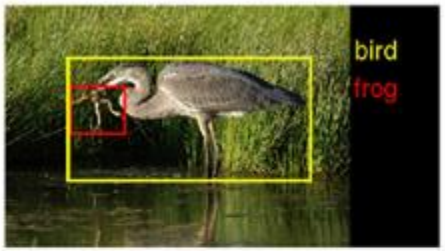
CNN Performance Benchmarks

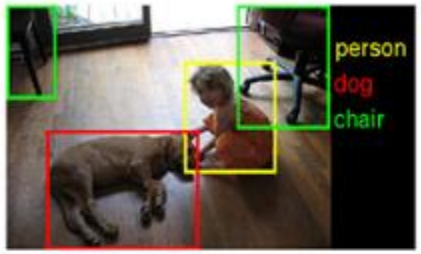
- CNN models were benchmarked on the MNIST handwritten digit recognition set, but there are only 70K examples.
- Each image is a matrix of 28x28 pixels.
- This dataset was created by Yann Lecun, the creator of Conv NNs.
- Data resides here:
- http://yann.lecun.com/exdb/mnist/

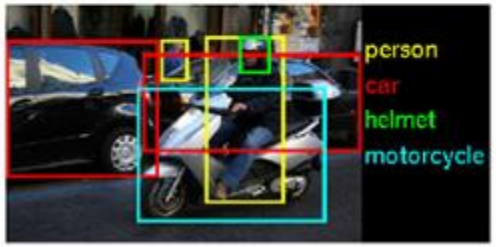


CNN Performance Benchmarks

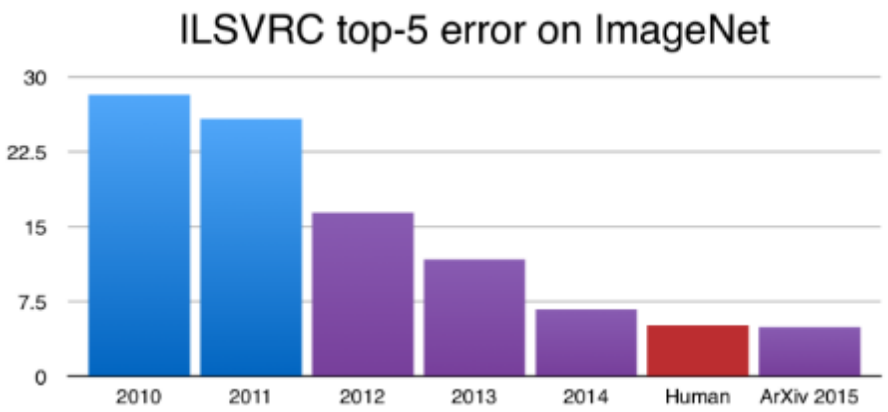
- Now, models are benchmarked on ImageNet ILSVRC set.
- There is a yearly competition for the highest classification accuracy.
  - ~15 Million photos. Each photo has multiple labels. There are ~80K different labels overall. Labels fall under hierarchies as well.
  - E.g. there are 3,822 different labels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







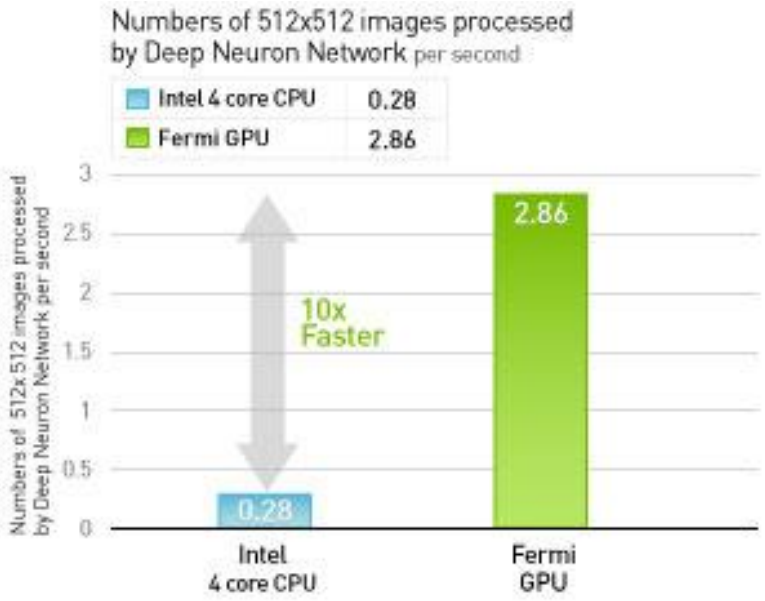
Convolutional Neural Networks (in purple) Andrej Karpathy in red.



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.

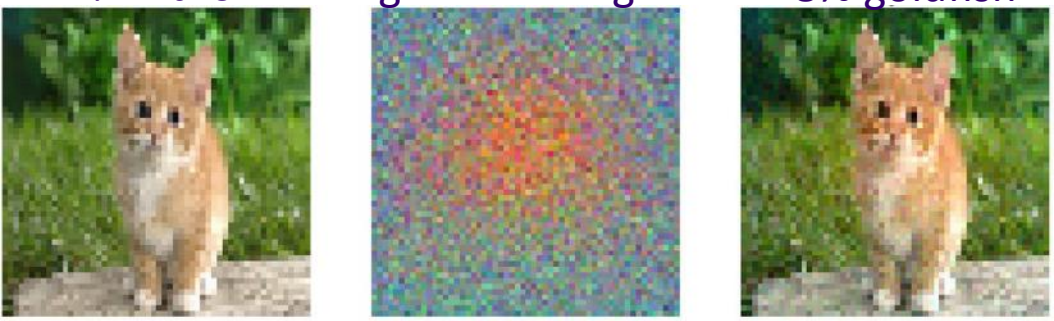
It's quite common to get 10x to 50x speedup when switching the same code to a GPU.



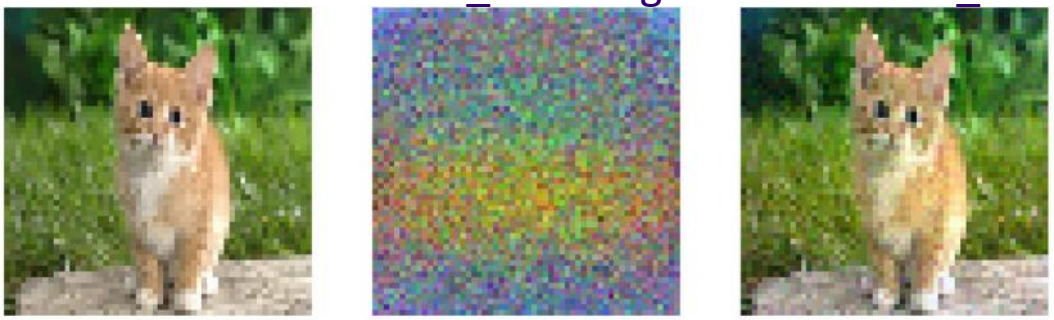
Fooling CNNs

> If the network is shallow with less (or no) regularization, we can fool the CNN by adding in weights that represent other classes.

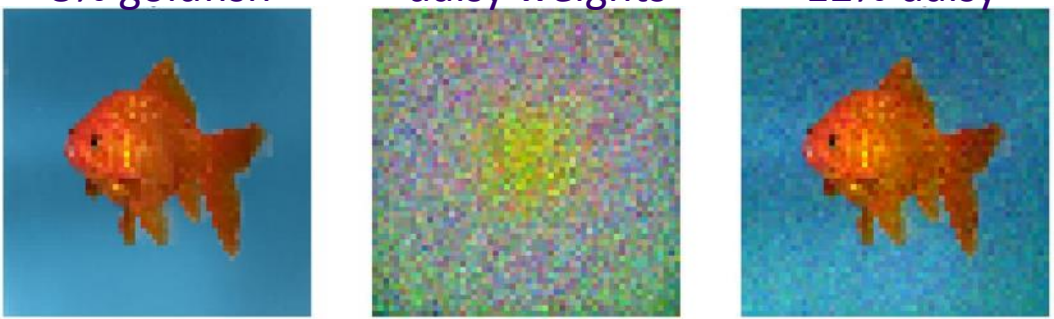
1% kit fox +goldfish weights =8% goldfish



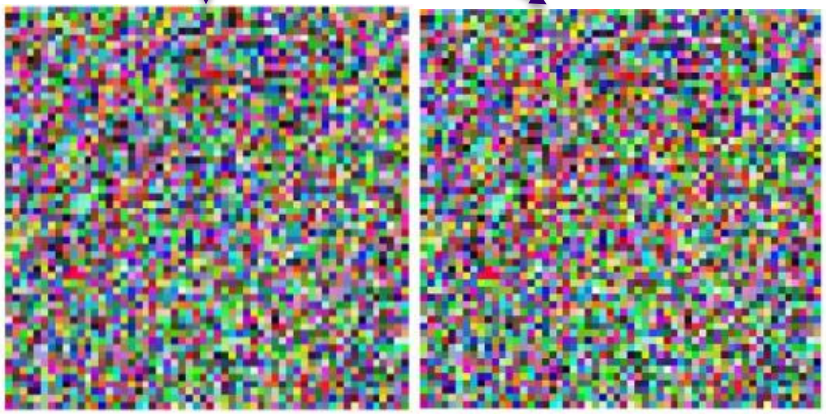
1% kit fox +school_bus weights =4% school_bus



8% goldfish +daisy weights =12% daisy



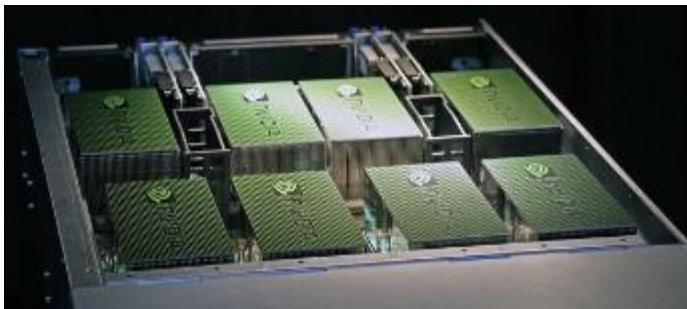
random noise + goldfish weights = 100% goldfish



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.

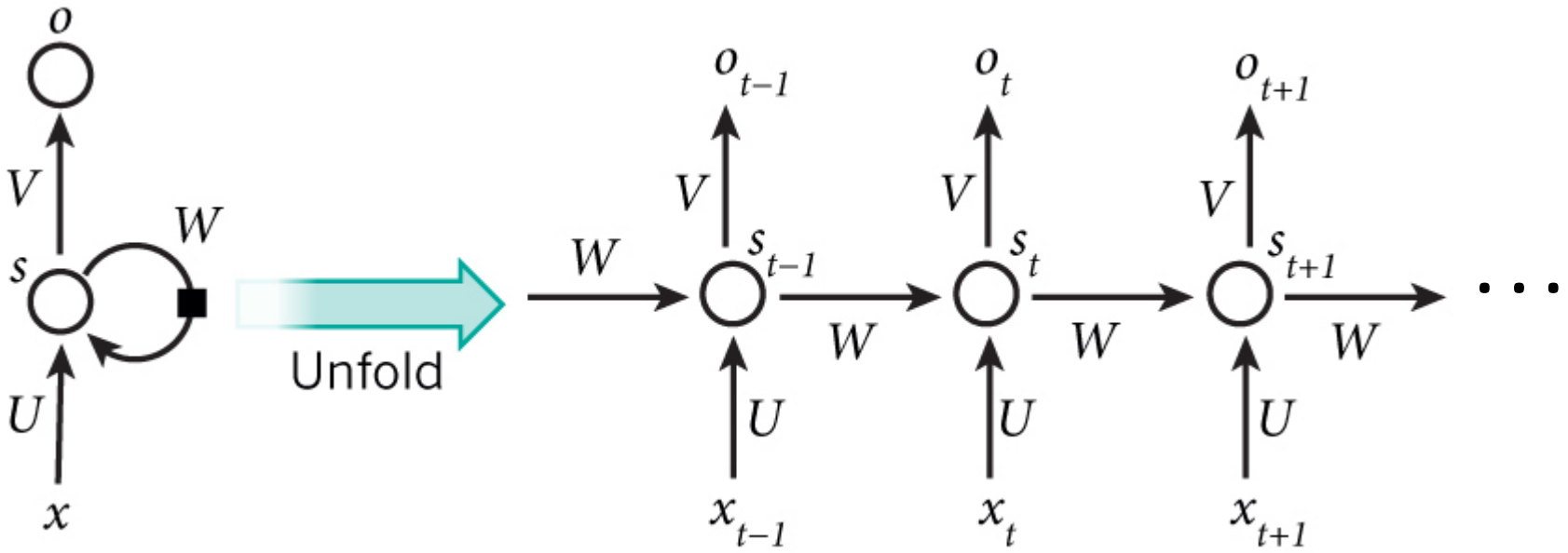
CNN Uses

- 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 predictions or prior values…

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

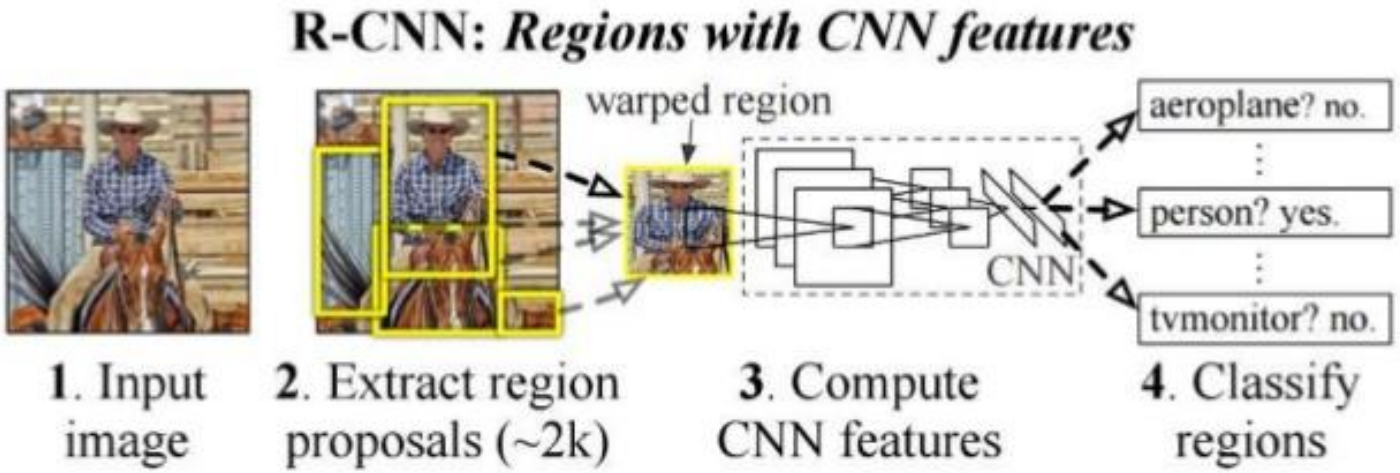
http://blog.otoro.net/2015/12/28/recurrent-net-dreams-up-fake-chinese-characters-in-vector-format-with-tensorflow/

https://twitter.com/DeepDrumpf 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

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

Recent Advances in CNNs

- Ford bought Argo for $1B, a self driving car company.
- Increase in NN-financial traders:
  - 2000: ~600 Goldman-Sachs desk traders.
  - 2017: 2 traders, supported by 200 computer engineers.
- Near realtime smoke/particle dynamics predictions.
  - https://www.youtube.com/watch?v=9pXubxPsIJw
- Ultra Low Power 7mm deep learning chip:
  - http://cubeworks.us/
- Debunking the myth of predicting gender from Iris:
  - https://arxiv.org/abs/1702.01304v1
- Automatic Handgun Detection in Videos:
  - https://arxiv.org/abs/1702.05147v1

Natural Language Processing with NNs

- Before, we would have to wrangle our data features to try to represent the *context* of words to get at the meaning.
  - We would have to look at pairs of words, triplets or words, etc…
- Really, this is not how humans work.
  - If I know the words ‘Vice President’ and ‘People’, and I know that ‘HR’ is related to ‘People’, then I should be able to intuit what ‘VP of People’ means.
- Remember that at all times, we are relying on ‘encoding’ a word. This means that we have an input string or token, and somehow we arrive at a numerical vector.
  - One hot encoding is an example.
  - But what if I could make the encodings for ‘people’ and ‘HR’ be close to each other?

Word2Vec

- Word2Vec is a popular way to represent language.
- We create a model that tries to quantify the meaning of words in numerical quantities.
- This is accomplished by learning a word as it's used in specific contexts (i.e., consider the surrounding words).
- There are two ways to create these word vectors:
  - (1) Create a neural network to predict 1 word based on surrounding words.
  - (2) Create a neural network to predict surrounding words based on 1 word input.

Word2Vec – Model 1

- (1) Create a neural network to predict 1 word based on surrounding words.
- CBOW Model

"The quick brown fox jumped …"

[input1, input2]=>output

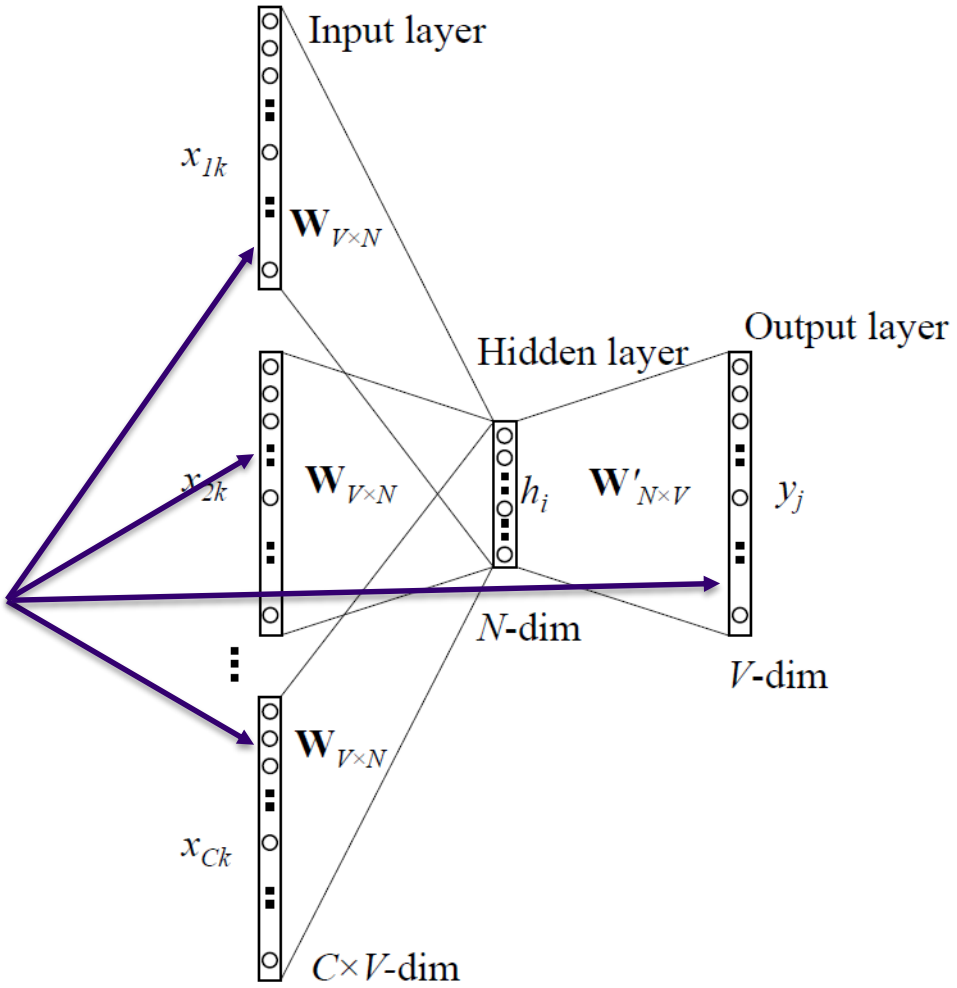
[the, brown]=> quick

[quick, fox]=>brown

[brown, jumped]=>fox

…

Occurrence or TF-IDF Vectors from Corpus



Word2Vec

- (2) Create a neural network to predict surrounding words based on 1 word input.
- SkipGram Model

"The quick brown fox jumped …"

Input=>[output1, output2]

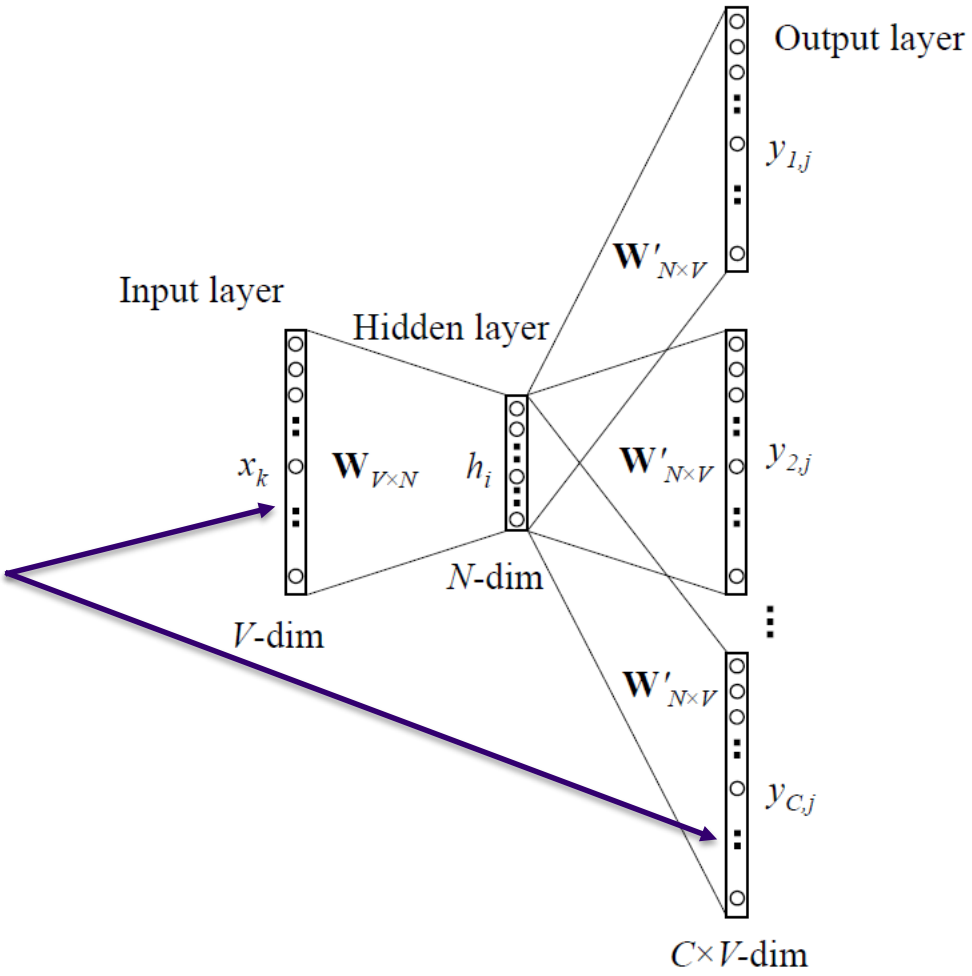
quick=> [the, brown]

brown=> [quick, fox]

fox=> [brown, jumped]

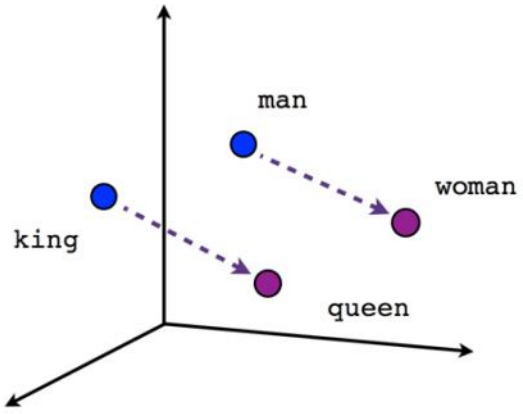
…

Occurrence or TF-IDF Vectors from Corpus

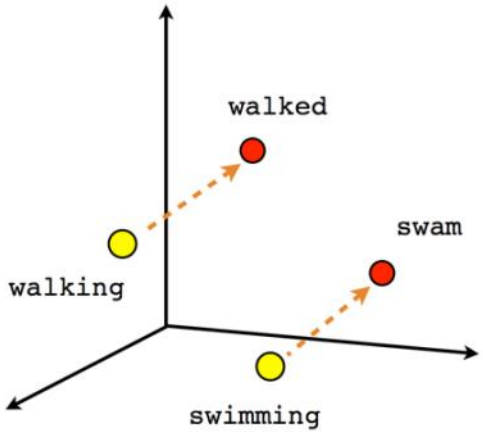


Word2Vec

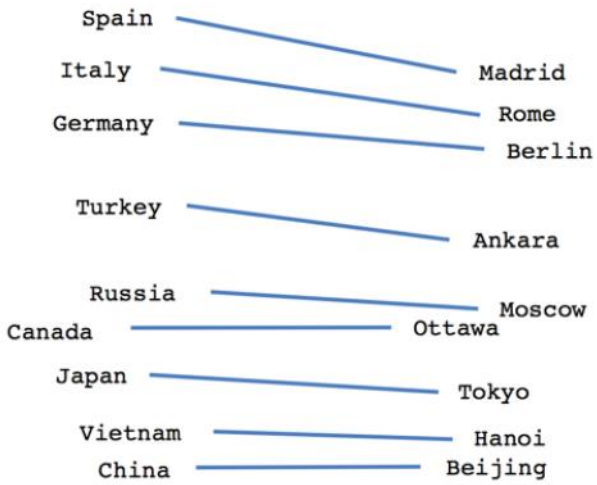
- In general, CBOW (predicts one word from surroundings) smoothens the predictions. This is better for smaller datasets.
- The Skipgram trains to predict context (predicts multiple words from one word) and this performs better on larger datasets.
- Which ever one we choose, the hidden layer represents learned numerical vectors of textual information from the text. We consider these as the learned vectors.



Male-Female

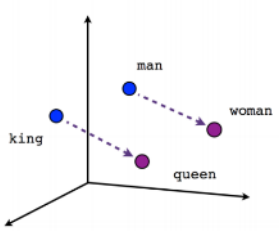


Verb tense

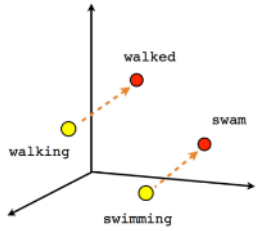


Country-Capital

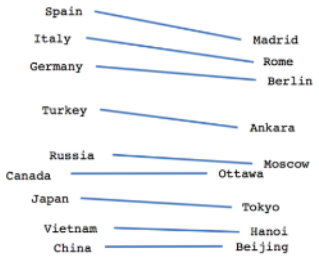
Doc2Vec



Male-Female



Verb tense

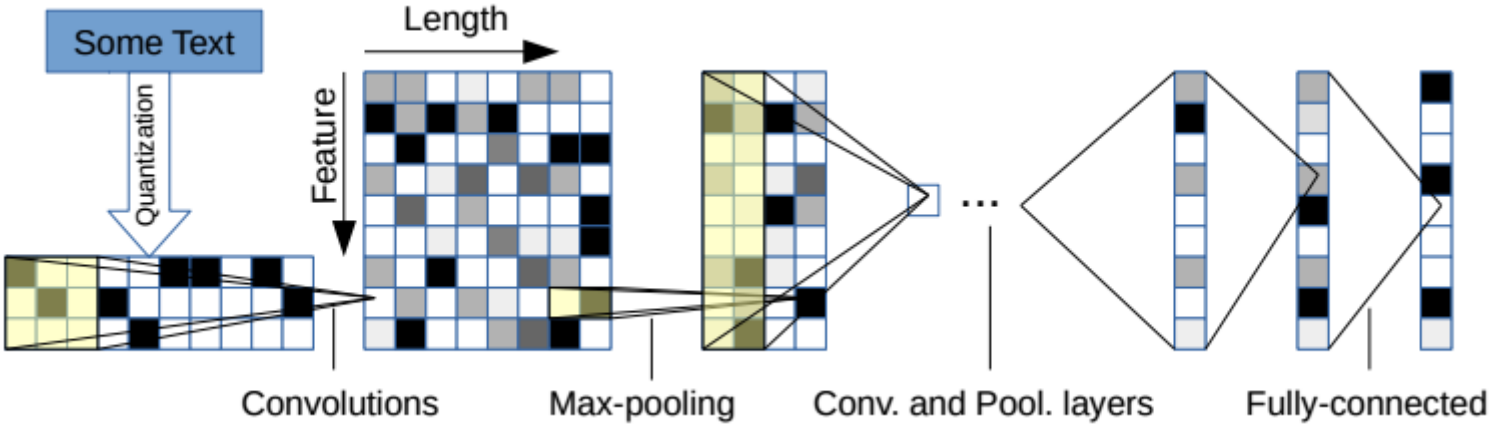


Country-Capital

- We now have a learned vector for each word. We can now use these vectors in any trainable model we wish.
- But what if our entries are combinations of words, like sentences, paragraphs, or tweets of variable length?
- We have to combine this Word2Vec information into something called Doc2Vec.
- There are many methods to converting an array of Word2Vec vectors into one Doc2Vec vector.
  - Average all words
  - Weighted average by TF-IDF score
  - Cluster vectors and use the average of k-clusters
  - …

CNN for Text (2015)

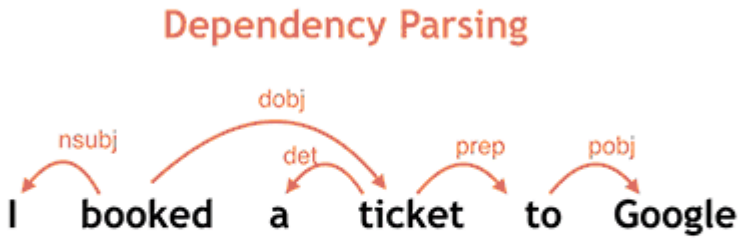
- Zhang, X., et. al. released a paper (http://arxiv.org/abs/1509.01626) that showed great results for treating strings as 1-D numerical vectors.
- Ways to get strings into numerical vectors:
  - Word level one-hot encoding.
  - Word2Vec encoding or similar
  - Character level one-hot encoding.
- Can't resize vectors like pictures, so instead we pick a max length and pad many entries with zeros.



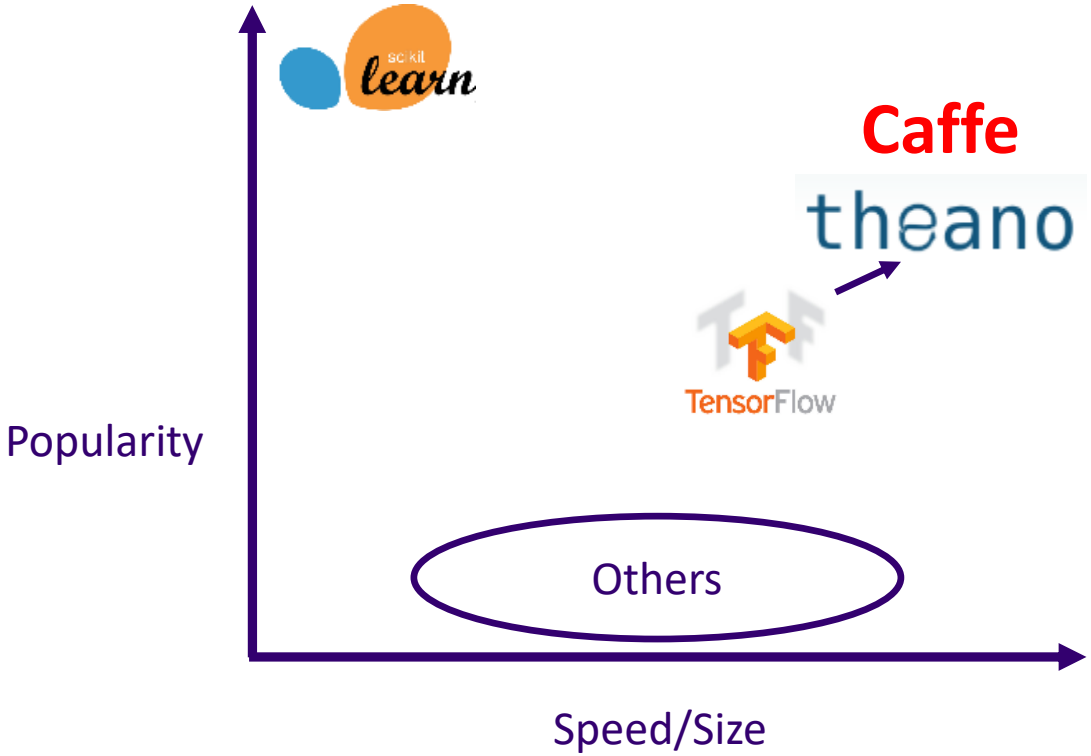
- Like many NN results, depend heavily on dataset and architecture.
- Paper shows good results for low # of classes (ratings or sentiment).

Parsey McParseface

- Google released a model called ‘Syntaxnet’ in May 2016, based on Tensorflow.
- Syntaxnet is arguably one of the best part-of-speech parsers available.
- They trained an English version, called ‘Parsey McParseface’.
- This is a free and open source model to use.



Software For Neural Networks



I prefer TensorFlow…

Why Tensorflow?

- Tensorflow is also very portable and flexible!
  - Virtually no code change to go from CPU-> GPU or…

TensorFlow

Linux CPU	Linux GPU PIP	Mac OS CPU	Android
build passing	build running	build passing	build passing

The Possibilities are Endless

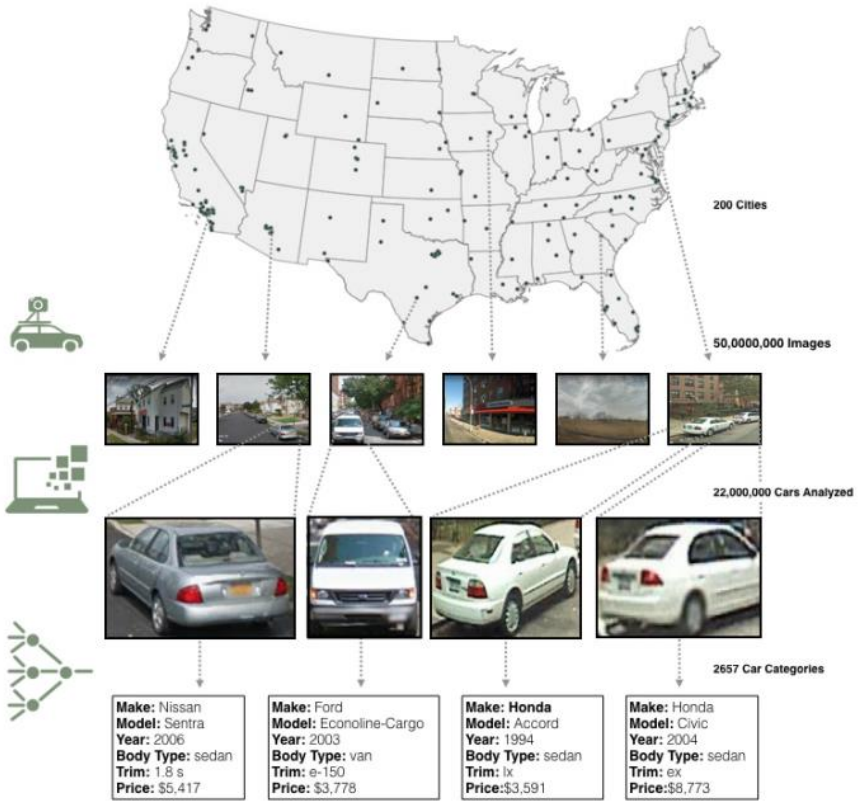
- Predict age from facial photo - https://how-old.net
- Predict dog breed from photo - https://www.what-dog.net
- Inside a self driving car brain : https://www.youtube.com/watch?v=ZJMtDRbqH40
- Memory networks: https://arxiv.org/pdf/1410.3916.pdf
  - Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.
  - Frodo went back to the Shire. Bilbo travelled to the Grey-havens.
  - Where is the ring? A: Mount-Doom
  - Where is Bilbo now? A: Grey-havens
- "Synthia"- Virtual driving school for self driving cars:
  - http://www.gizmag.com/synthia-dataset-self-driving-cars/43895/
- Solving hand-written chalk board problems in real time:
  - http://www.willforfang.com/computer-vision/2016/4/9/artificial-intelligence-for-handwritten-mathematical-expression-evaluation

- Real Time Face Swap: https://www.youtube.com/watch?v=ohmajJTcpNk

(a) (b) (c)

Figure 1: (a) The input image. (b) The result of face swapping with Nicolas Cage using our method. (c) The result of a manual face swap (source: http://niccageaseveryone.blogspot.com).

- Predicting Demographics from Streetview Images
  - https://arxiv.org/abs/1702.06683



- Image to Image CNNs:
  - Input image (night image), output a day image of same scene.
  - Input image (face w/ no beard), output face w/ beard.
  - https://arxiv.org/pdf/1703.00848.pdf

- Playing AND explanation of video games:
- https://arxiv.org/pdf/1702.07826.pdf

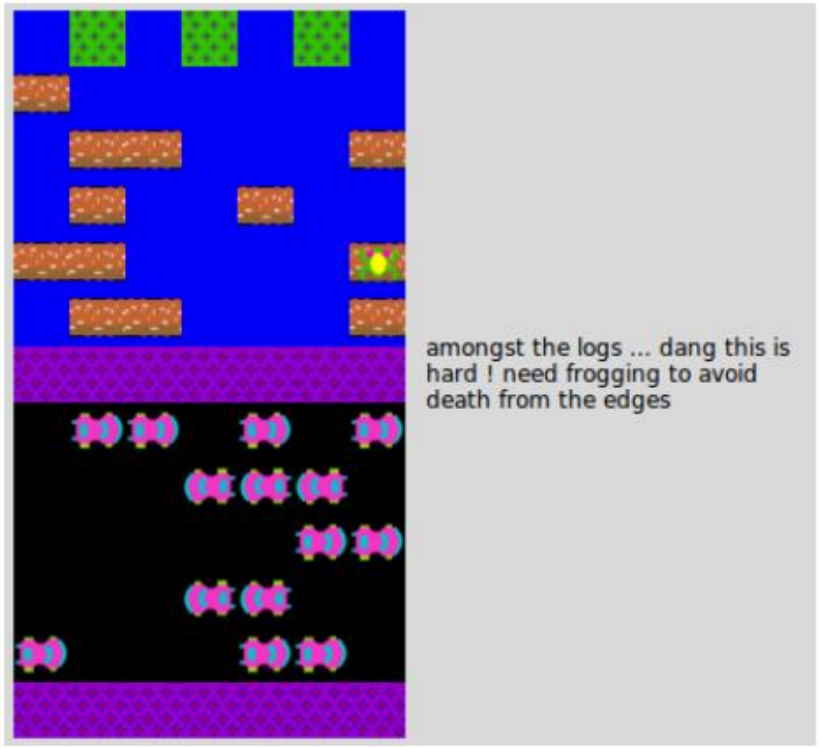


Figure 4. The Frogger agent rationalizing its actions.

- ‘DeepCoder: Learning to Write Programs’
  - https://openreview.net/pdf?id=ByldLrqlx
  - Good summary: http://smerity.com/articles/2017/deepcoder_and_ai_hype.html
- “Beating the World’s Best at Super Smash Bros.”
  - https://arxiv.org/abs/1702.06230
  - Video of algo playing against itself: https://www.youtube.com/watch?v=vWjr3338AHQ
- Edge to Image: Neural network that translates drawing to real-looking pictures.
  - https://affinelayer.com/pixsrv/index.html

More Resources

- https://nmarkou.blogspot.com.cy/2017/02/the-black-magic-of-deep-learning-tips.html
- http://playground.tensorflow.org
- 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: (2016-Present)
- 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

<Insert Name Here>-Neural Net

- Last Month (June-July 2016):
  - YodaNN: http://arxiv.org/abs/1606.05487
  - CMS-RCNN: http://arxiv.org/abs/1606.05413
  - Zoneout RNN: https://arxiv.org/abs/1606.01305
  - Pixel CNN: http://arxiv.org/abs/1606.05328
  - Memory CNN: http://arxiv.org/abs/1606.05262
  - DISCO Nets: http://arxiv.org/abs/1606.02556
- This Year (2016):
  - Stochastic Depth Net: https://arxiv.org/abs/1603.09382
  - Squeeze Net: https://arxiv.org/abs/1602.07360
  - Binary Nets: http://arxiv.org/abs/1602.02830
- Last Year:
  - PoseNet: http://arxiv.org/abs/1505.07427
  - YOLO Nets: http://arxiv.org/abs/1506.02640
  - Style Net: http://arxiv.org/abs/1508.06576
  - Residual Net: https://arxiv.org/abs/1512.03385
  - And so many more…
- What's Your Architecture?

How to Keep ML Current

- Machine learning advances happen more and more frequently.
  - Can't rely on journals that take multiple years to publish.
- Twitter
- http://arxiv.org/
  - http://www.arxiv-sanity.com/
- New idea: Jack Clark, AI Journalist
  - Import-AI Newsletter: https://jack-clark.net/import-ai/
  - https://docs.google.com/spreadsheets/d/1xej5Nca2xUUtrZ1GCyPjFMqI9ZgNq_OhgnTxOOMQ2js/
- Others?