CPSC 340: Machine Learning and Data Mining

Deep Learning

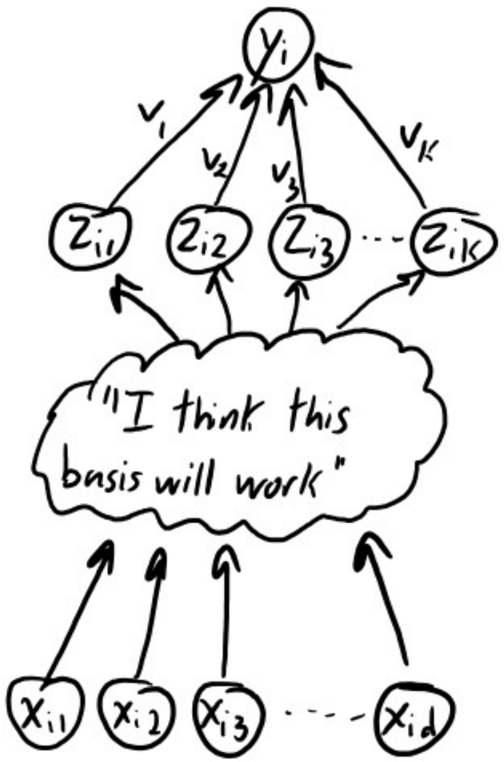
Admin

- Course surveys
  - Please fill them out
  - We care deeply about your education, so we take them very seriously
  - You will be able to evaluate the class overall, and then Prof. Schmidt and I separately
  - As always, please remember we’re real people, so both praise and constructive criticism feedback are great. Please avoid personal, hurtful, or unconstructive negative comments. Tone matters!

Last Time

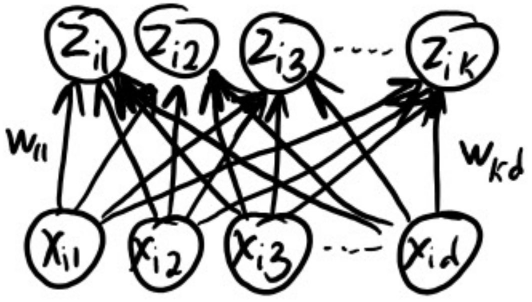
Supervised Learning Roadmap

Hand-engineered features:

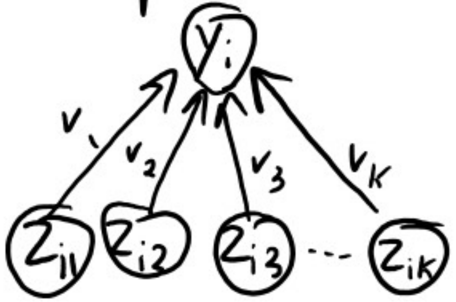


Requires domain knowledge and can be time-consuming

Learn a latent-factor model:

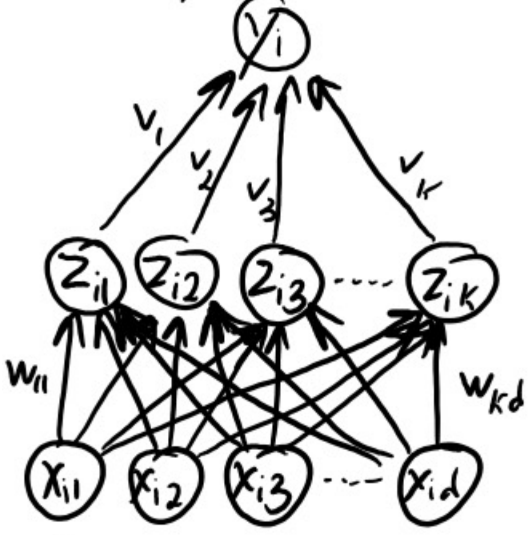


Use latent features in supervized model:



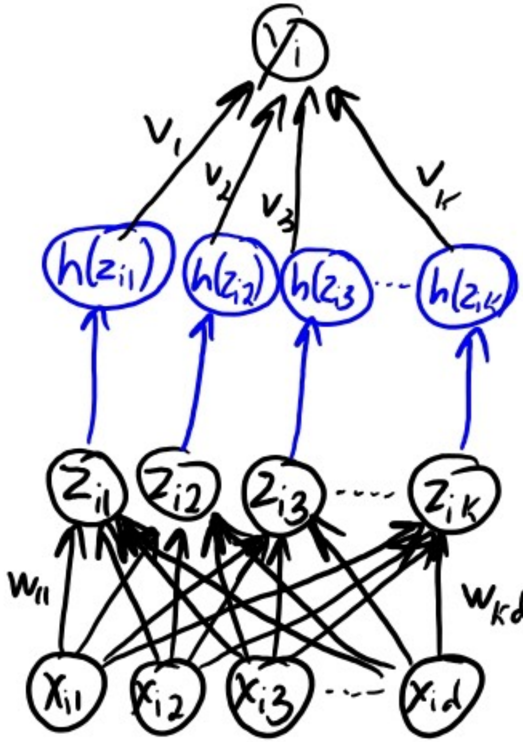
Good representation of x_i might be bad for predicting y_i

Learn 'w' and 'W' together:



But still gives a linear model.

Neural network:

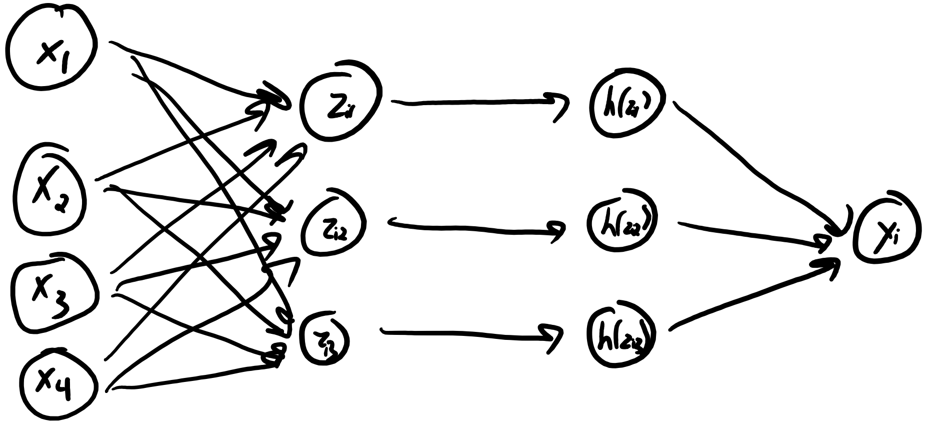


Extra non-linear transformation 'h'.

Regression vs. Binary Classification

- For regression problems, our prediction (ignoring biases) is:

$$\hat{y}_i = v^T h(W x_i)$$



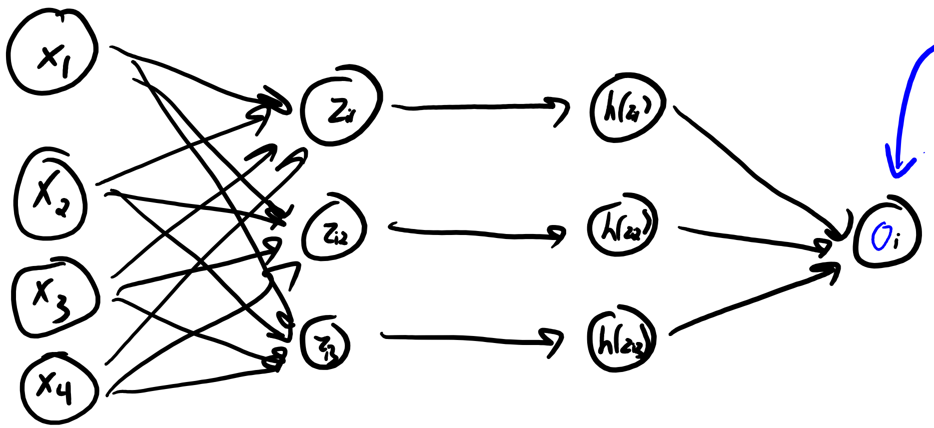
- And we might train to minimize the squared residual:

$$f(W, v) = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(v^T h(Wx_i) - y_i)^2$$

Regression vs. Binary Classification

- For binary classification problems, our prediction is:

$$o_i = v^T h(Wx_i) \qquad \hat{y}_i = \text{sign}(o_i)$$



- And we might train to minimize the logistic loss:

$$f(W,v) = \sum_{i=1}^{n} \log(1+\exp(-y_i o_i)) = \sum_{i=1}^{n} \log(1+\exp(-y_i v^T h(Wx_i)))$$

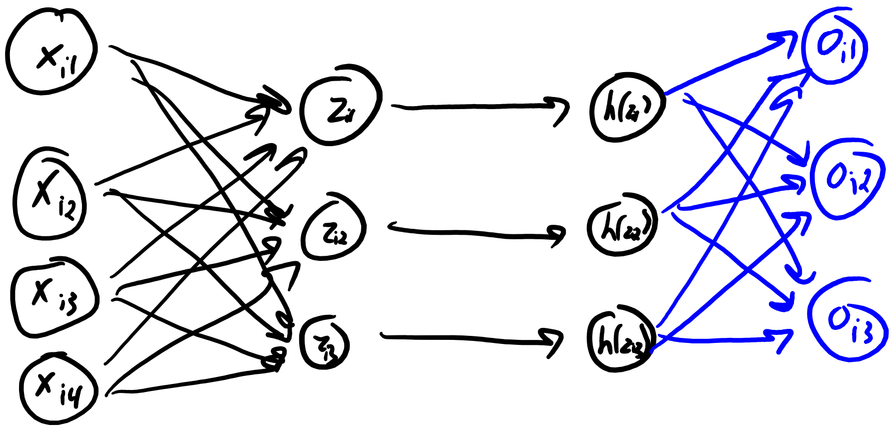
  – This is like logistic regression with learned features.

$$p(y_i \mid W, v, x_i) = \frac{1}{1+\exp(-y_i \underbrace{v^T h(Wx_i)}_{o_i})}$$

Use a sigmoid on output to get a probability

Neural Network for Multi-Class Classification

- Multi-class classification a neural network:
  - Input is connected to a hidden layer (same as regression and binary case).
  - Hidden layer is connected to multiple output units (one for each label.).



$o_{i1} = v_1^T h(Wx_i)$

$o_{i2} = v_2^T h(Wx_i)$

$o_{i3} = v_3^T h(Wx_i)$

Now have a matrix of parameters:

$$V = \begin{bmatrix} — & v_1^T & — \\ — & v_2^T & — \\ & \vdots & \\ — & v_{k'}^T & — \end{bmatrix}$$

$\ell \times k$

number of classes

number of hidden units

  - We can predict by maximizing o_{ic} over all 'c'.
  - We can convert to probabilities for each class using softmax to the o_{ic} values:

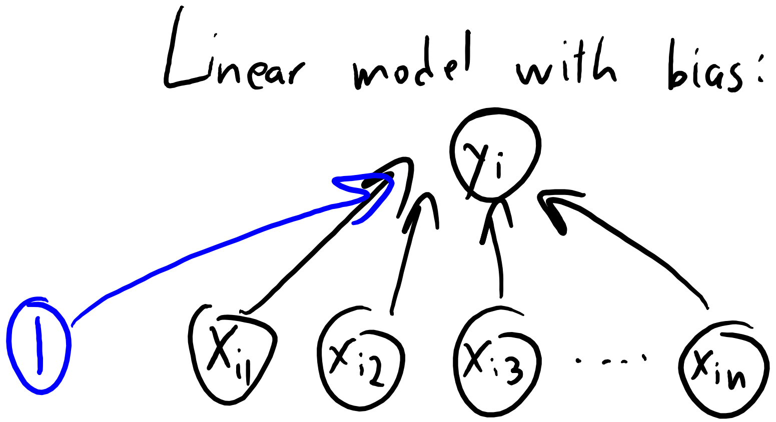
$$\frac{\exp(o_{ic})}{\sum_{c'=1}^{k'} \exp(o_{ic'})}$$

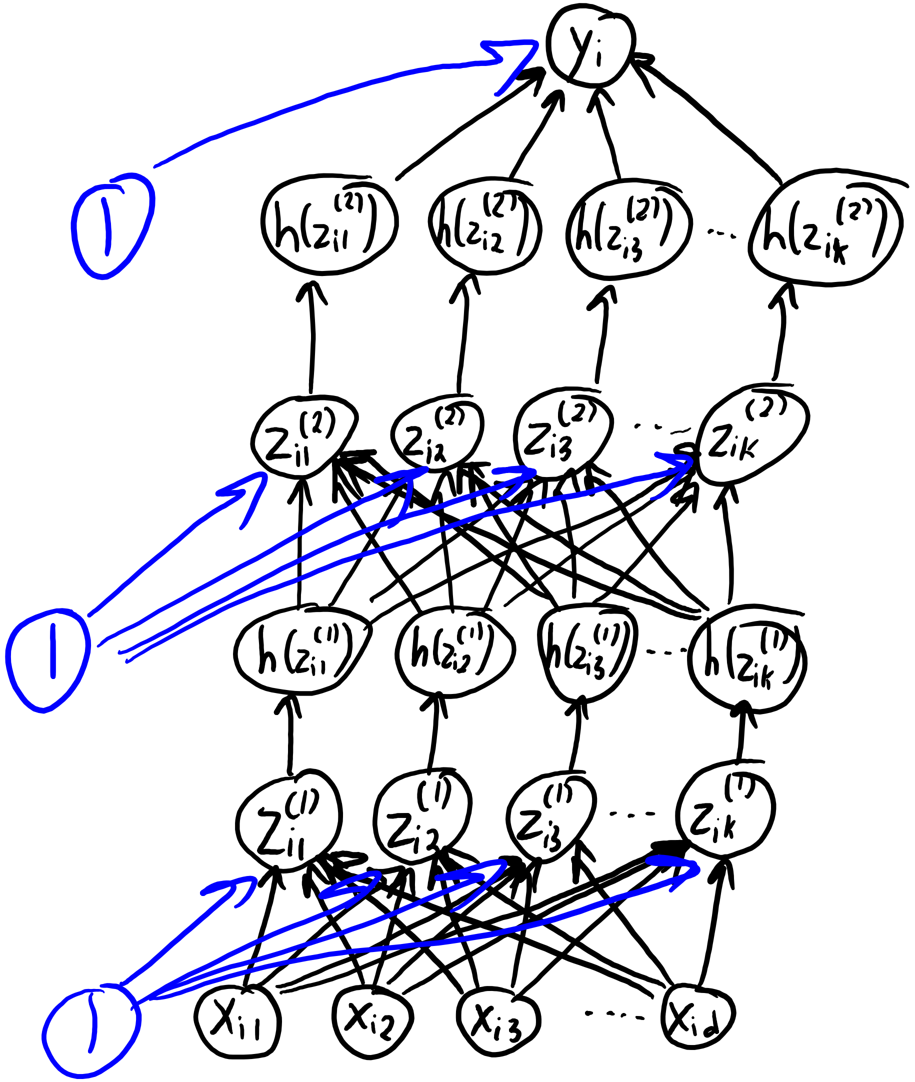
  - We train by minimizing negative log of this probability (softmax loss, summed across examples).
  - Notice that we changed tasks by only changing last layer (and loss function).

aka "cross entropy"(good intuitive explanation here)

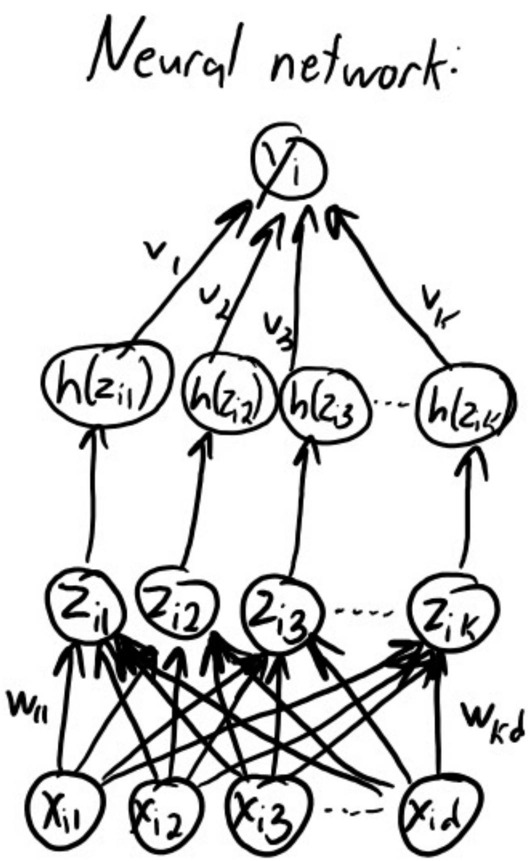
Adding Bias Variables

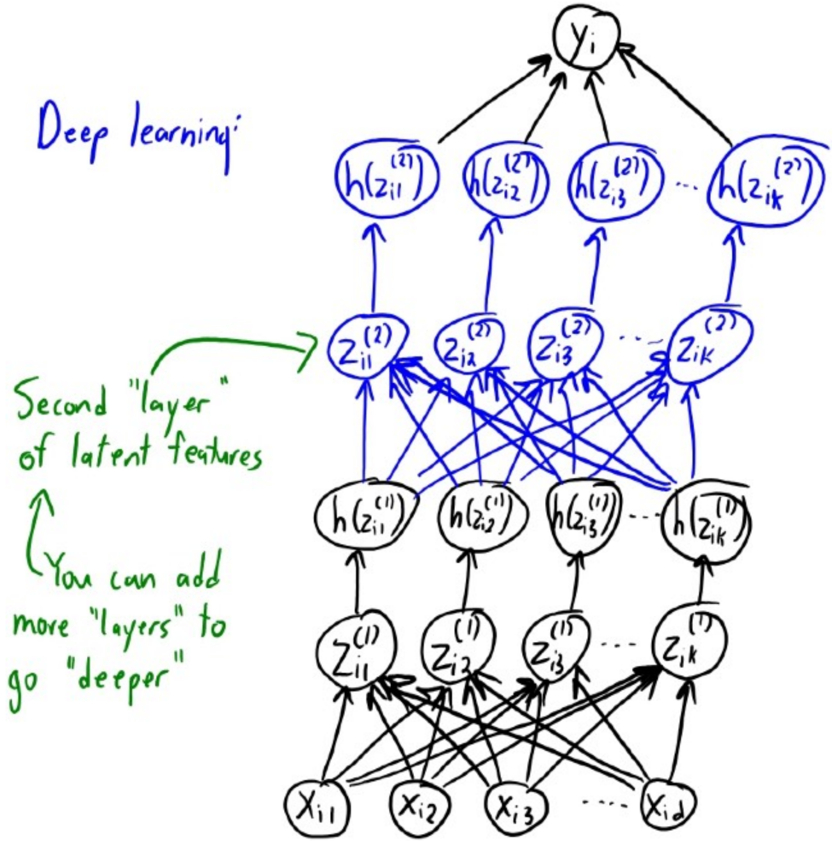
- We typically add a bias to each layer:

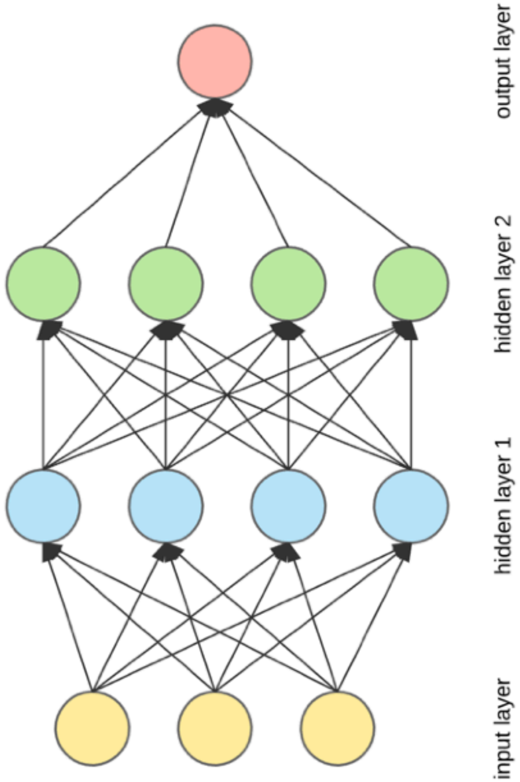




Deep Learning





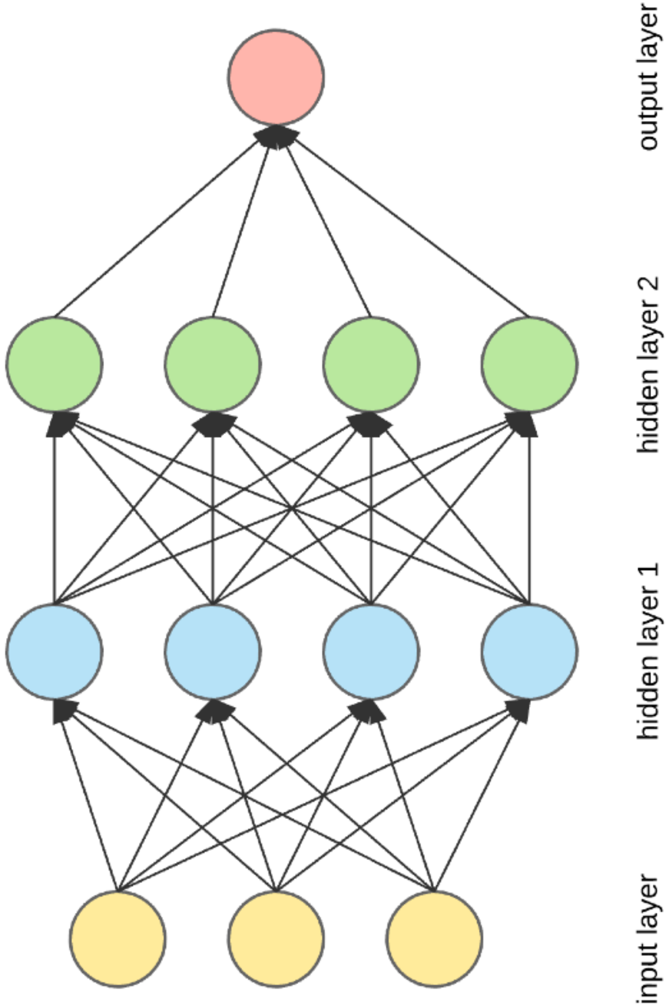


Deep Learning Terminology

- "layer" = number of layers of weights
  (numWs + V)

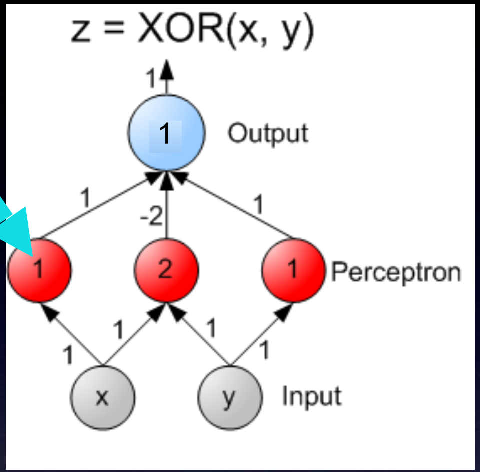
- "hidden layer" = number of activation layers (Zs)
  (not including inputs)

- do not assume people are consistent with this language

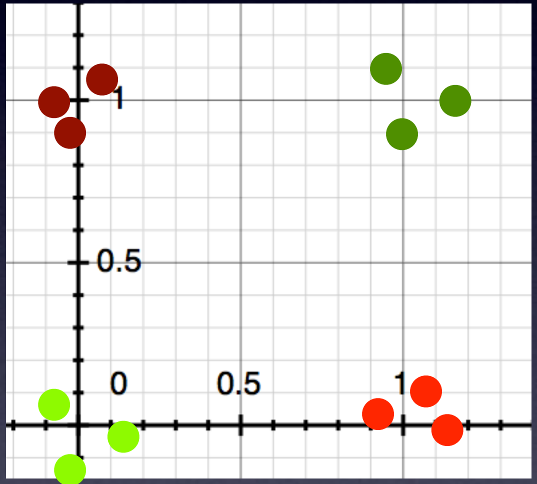


Neural Networks

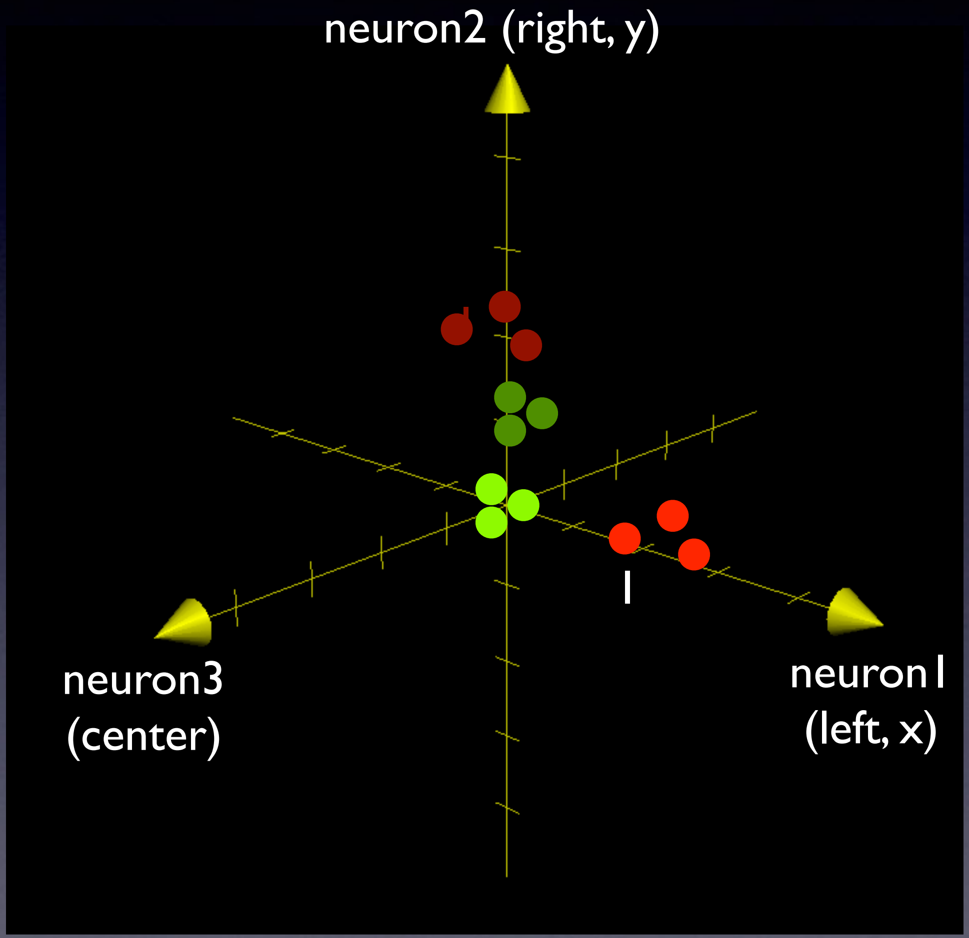
Outputs 1 if >= number in node
Else: 0



2D



Non-Linear Mapping



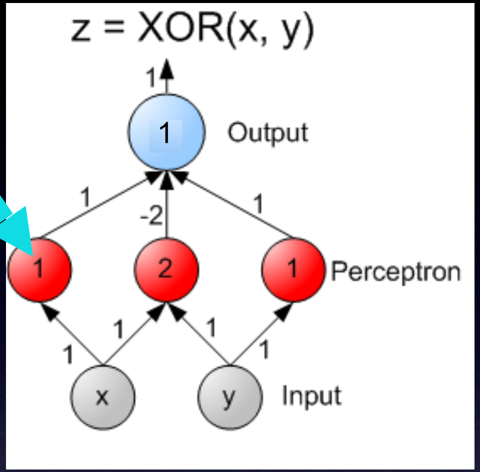
3D

Outputs of each node

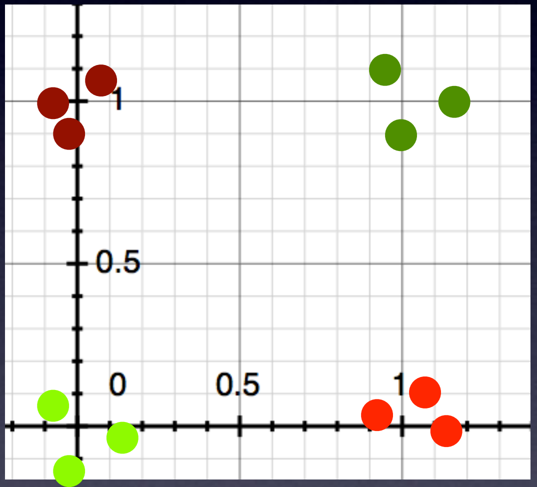
	x	y	Left	Center	Right	Output
● (green)	1	1				
● (red)	1	0				
● (dark red)	0	1				
● (light green)	0	0				

Neural Networks

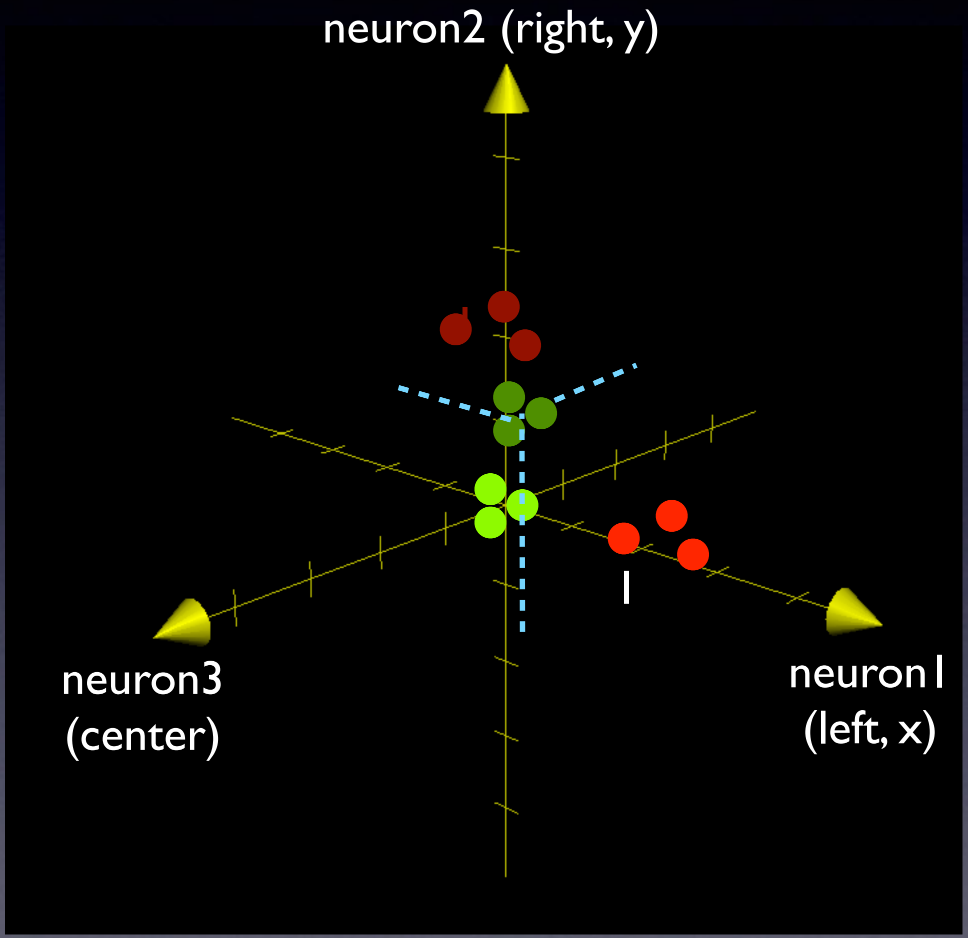
Outputs 1 if >= number in node
Else: 0



2D



Non-Linear Mapping



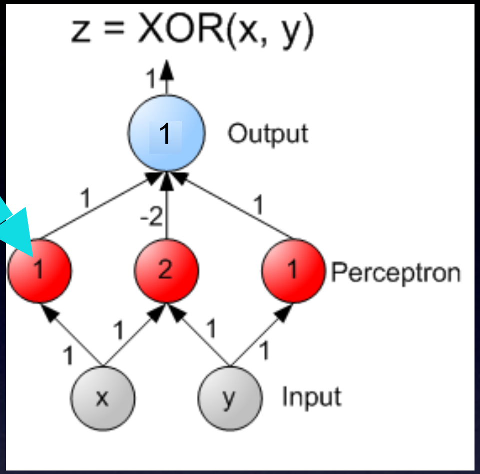
3D

Outputs of each node

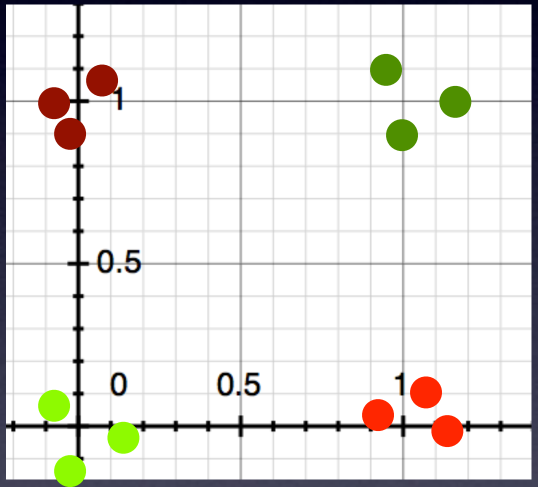
x	y	Left	Center	Right	Output
1	1	1	1	1	0
1	0				
0	1				
0	0				

Neural Networks

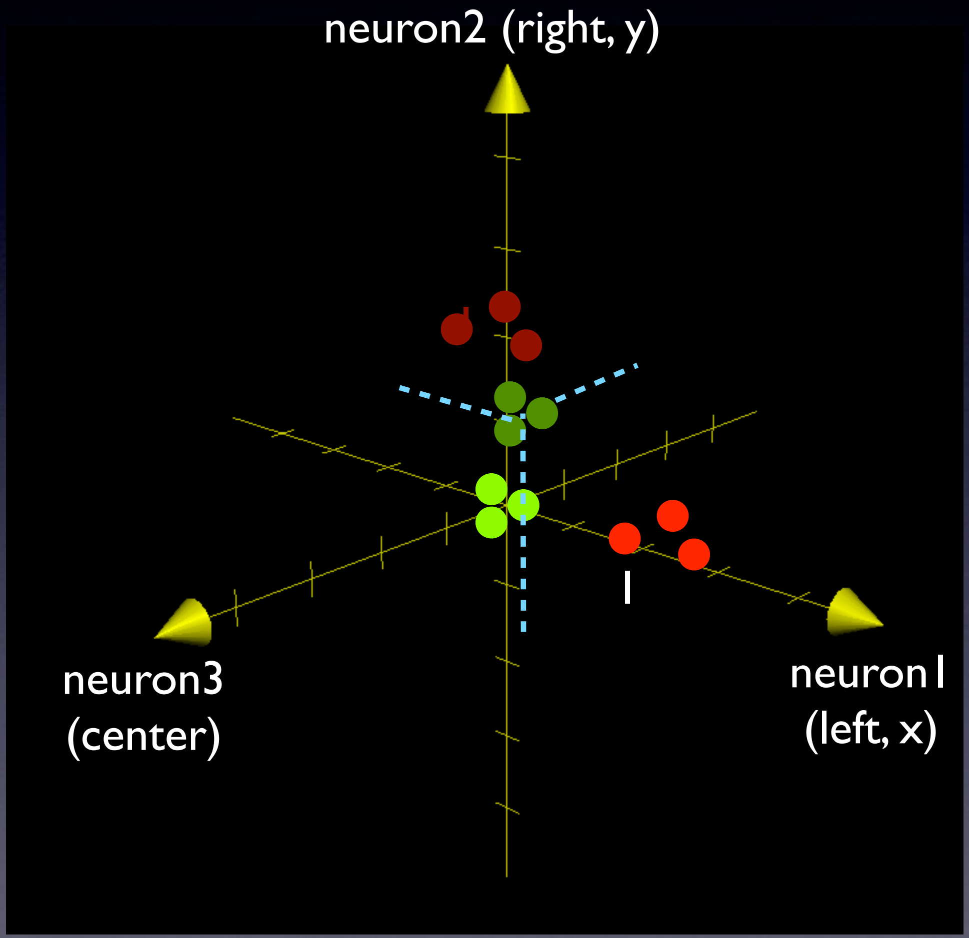
Outputs 1 if >= number in node
Else: 0



2D



Non-Linear Mapping



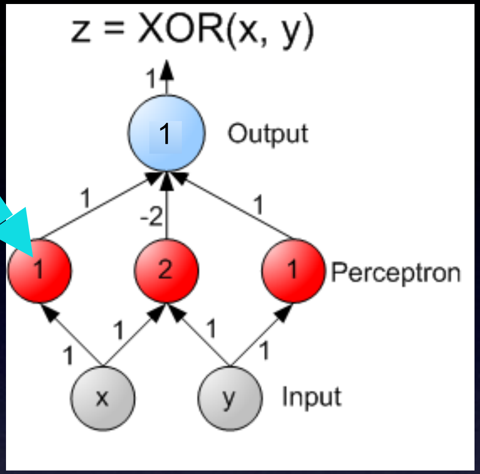
3D

Outputs of each node

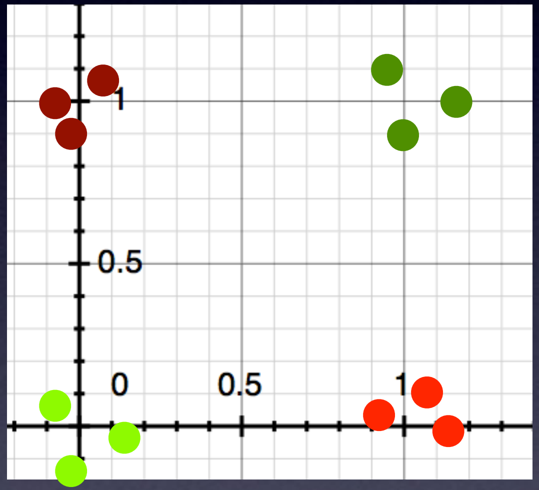
	x	y	Left	Center	Right	Output
● (dark green)	1	1	1	1	1	0
● (red)	1	0	1	0	0	1
● (dark red)	0	1				
● (light green)	0	0				

Neural Networks

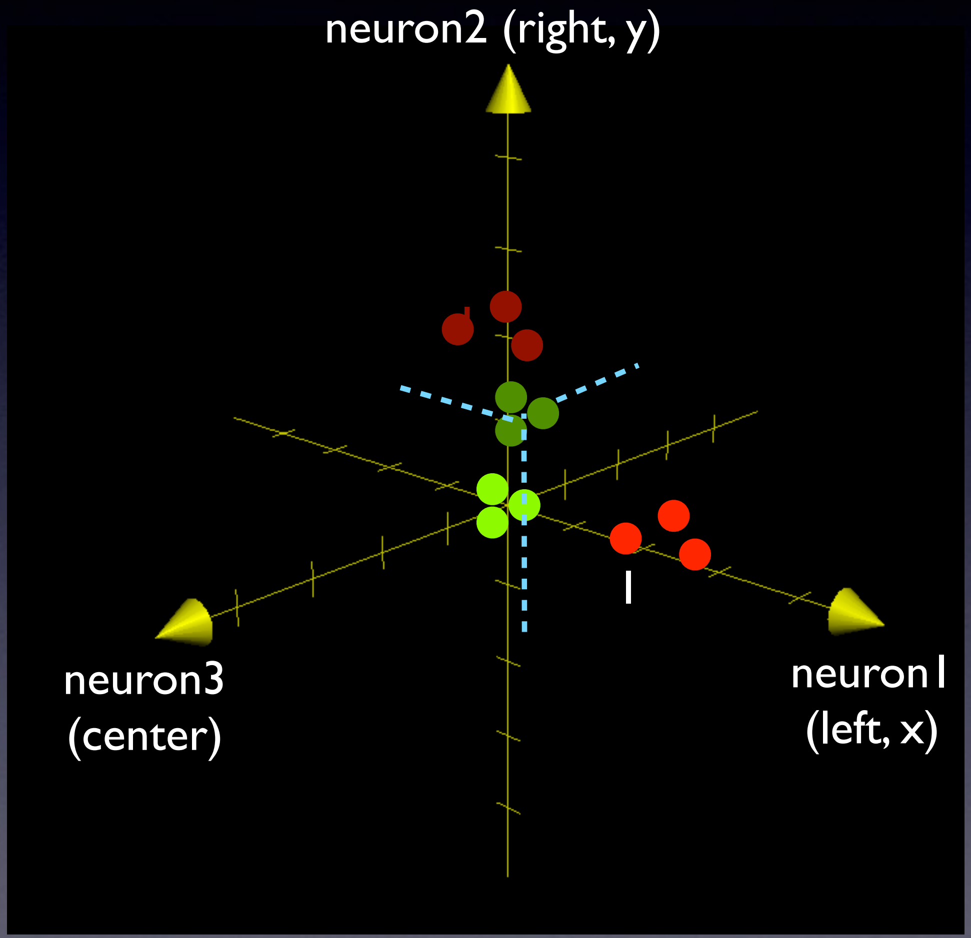
Outputs 1 if >= number in node
Else: 0



2D



Non-Linear Mapping



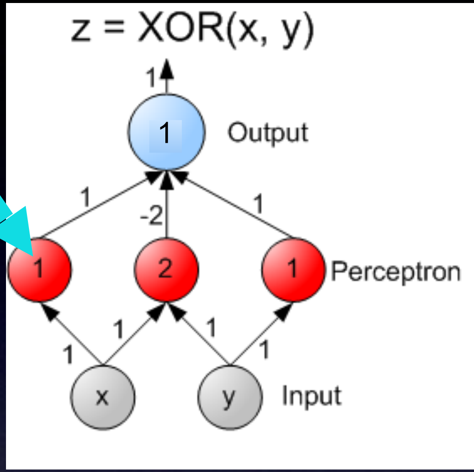
3D

Outputs of each node

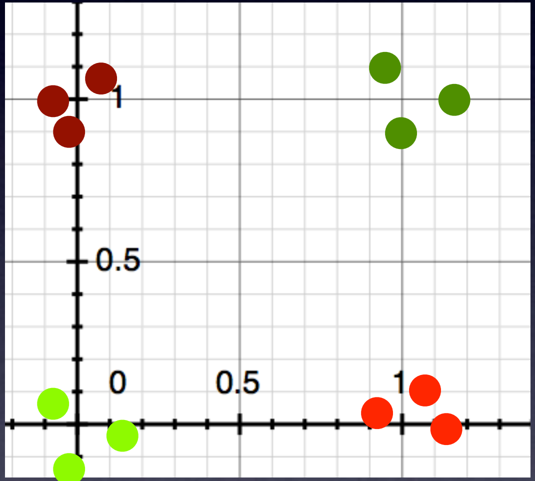
x	y	Left	Center	Right	Output
1	1	1	1	1	0
1	0	1	0	0	1
0	1	0	0	1	1
0	0				

Neural Networks

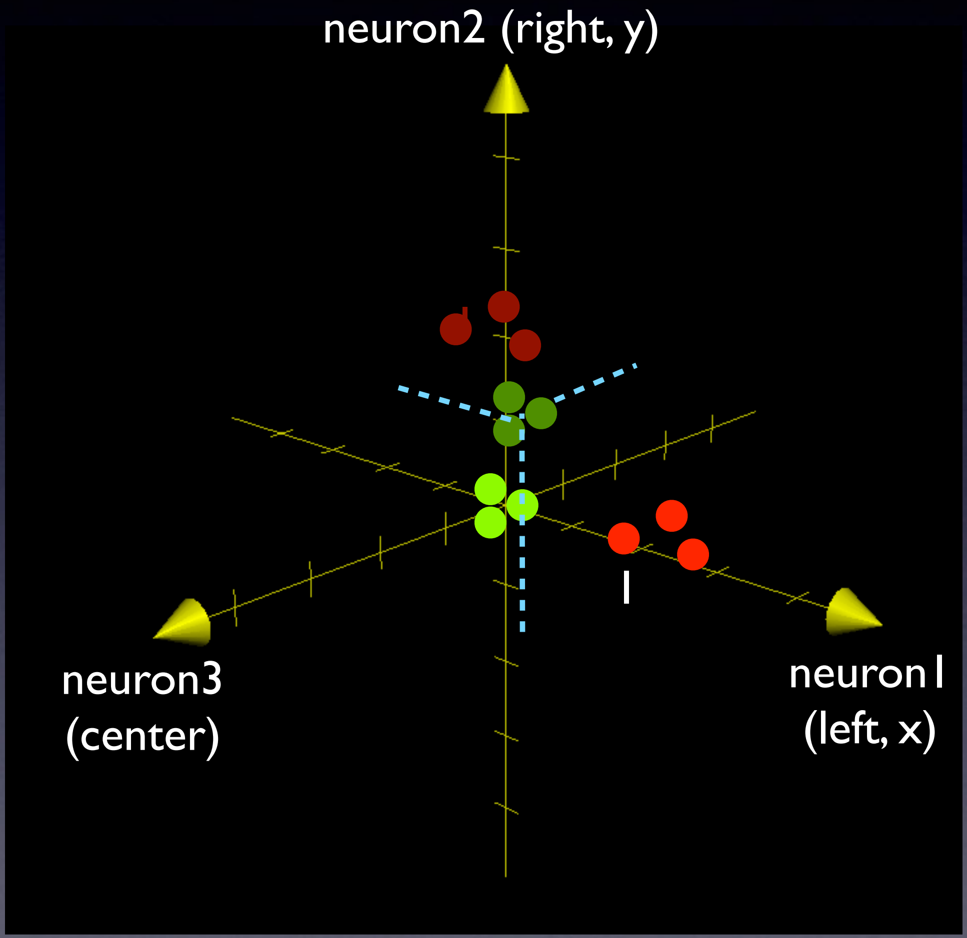
Outputs 1 if >= number in node
Else: 0



2D



Non-Linear Mapping

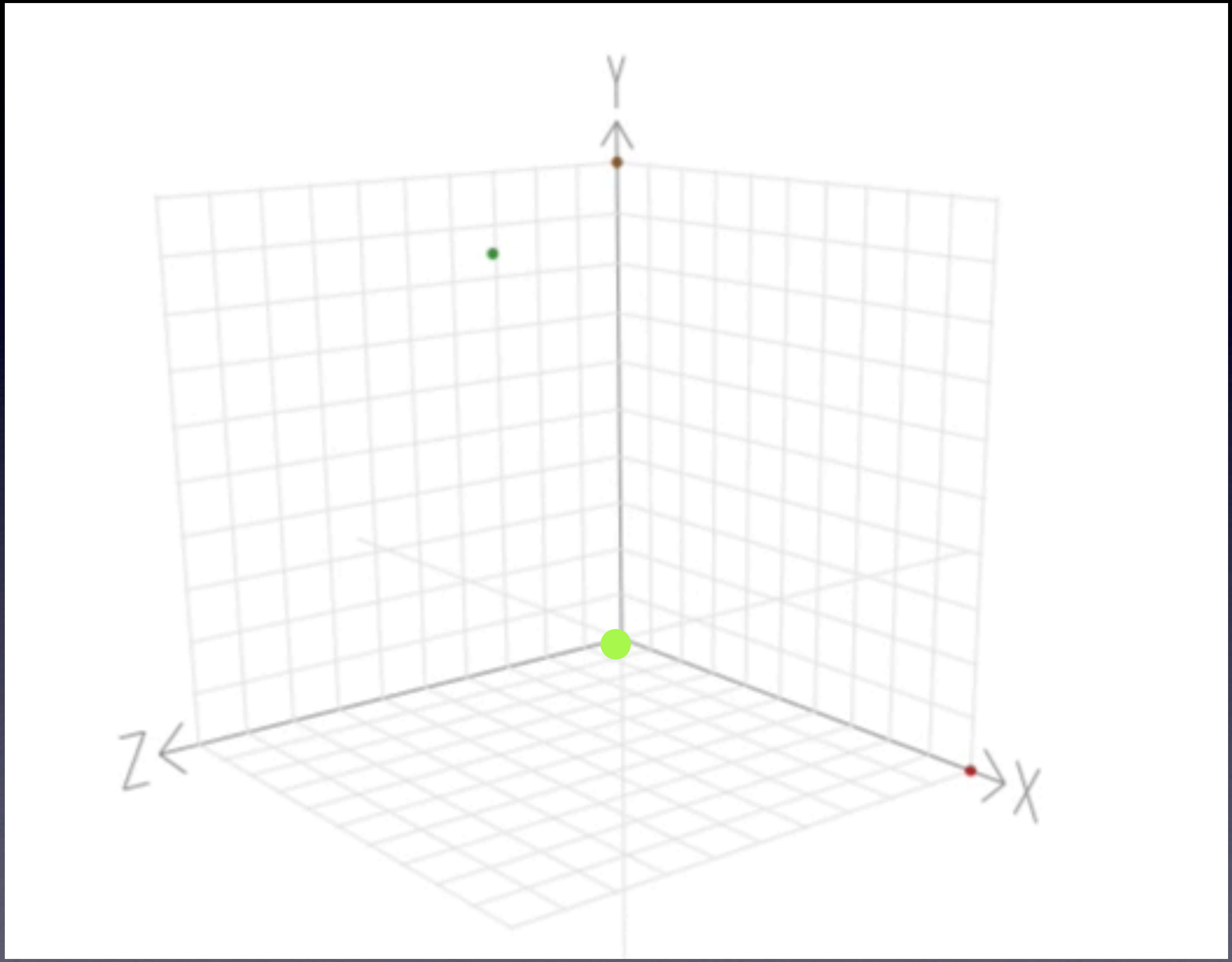


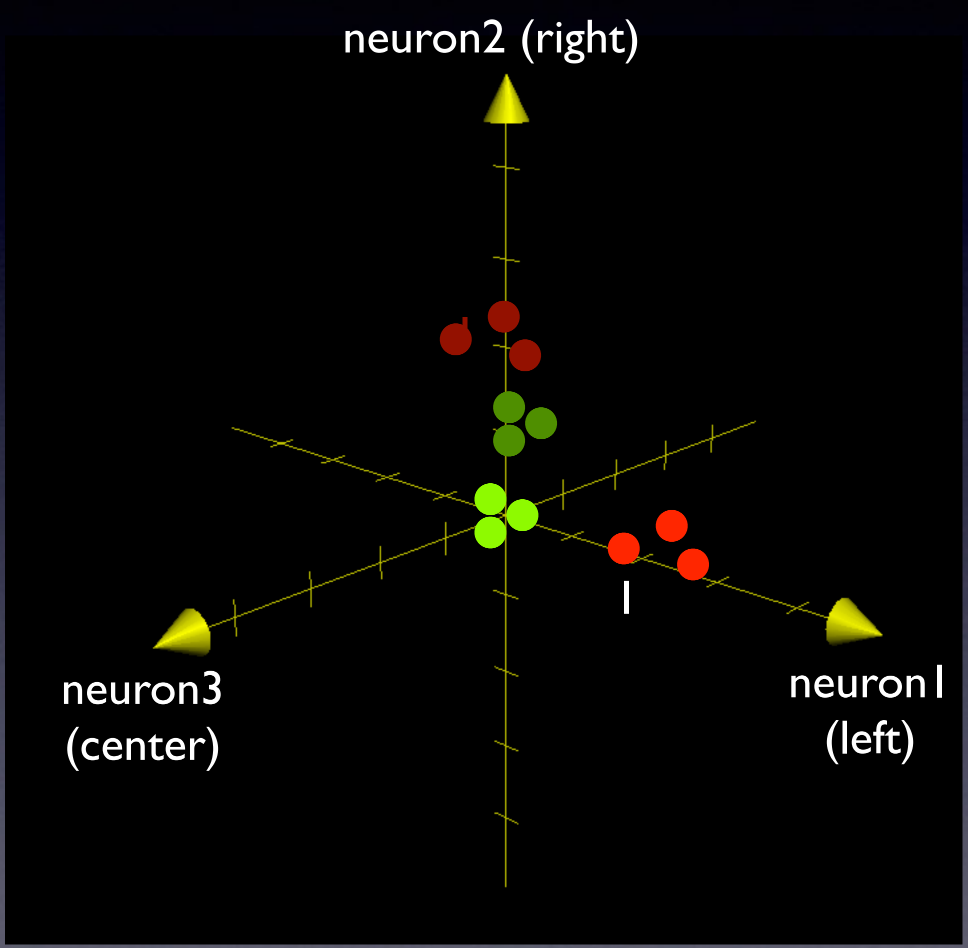
3D

Outputs of each node

x	y	Left	Center	Right	Output
1	1	1	1	1	0
1	0	1	0	0	1
0	1	0	0	1	1
0	0	0	0	0	0

Neural Networks





3D

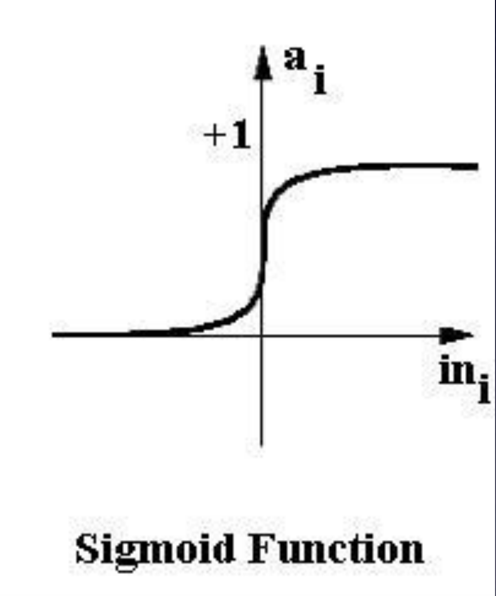
Neural Networks

- Point of XOR was not that you need k>d
    - Instead
        - an example of how that can make things easier
        - and how a transformation can make things linearly separable
- Cover's theorem: "The probability that classes are linearly separable increases when the features are nonlinearly mapped to a higher dimensional feature space." [Coover 1965]
- The output layer requires linear separability.
- The purpose of the hidden layers is to make the problem linearly separable!
- Multi-layer networks thus allow "non-linear regression"

From: http://130.236.96.13/edu/courses/TBMI26/pdfs/lectures/le5.pdf

Neural Networks

- Multi-layer networks thus allow “non-linear regression”



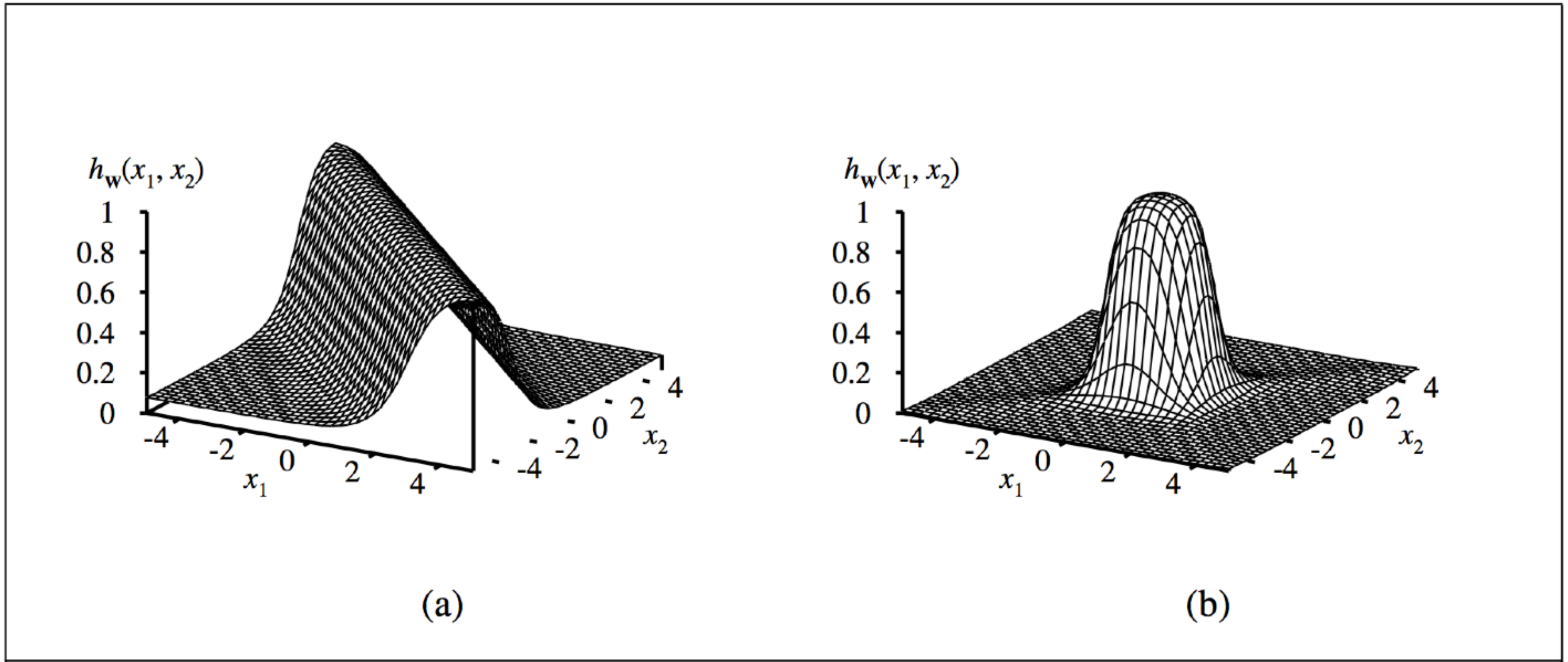


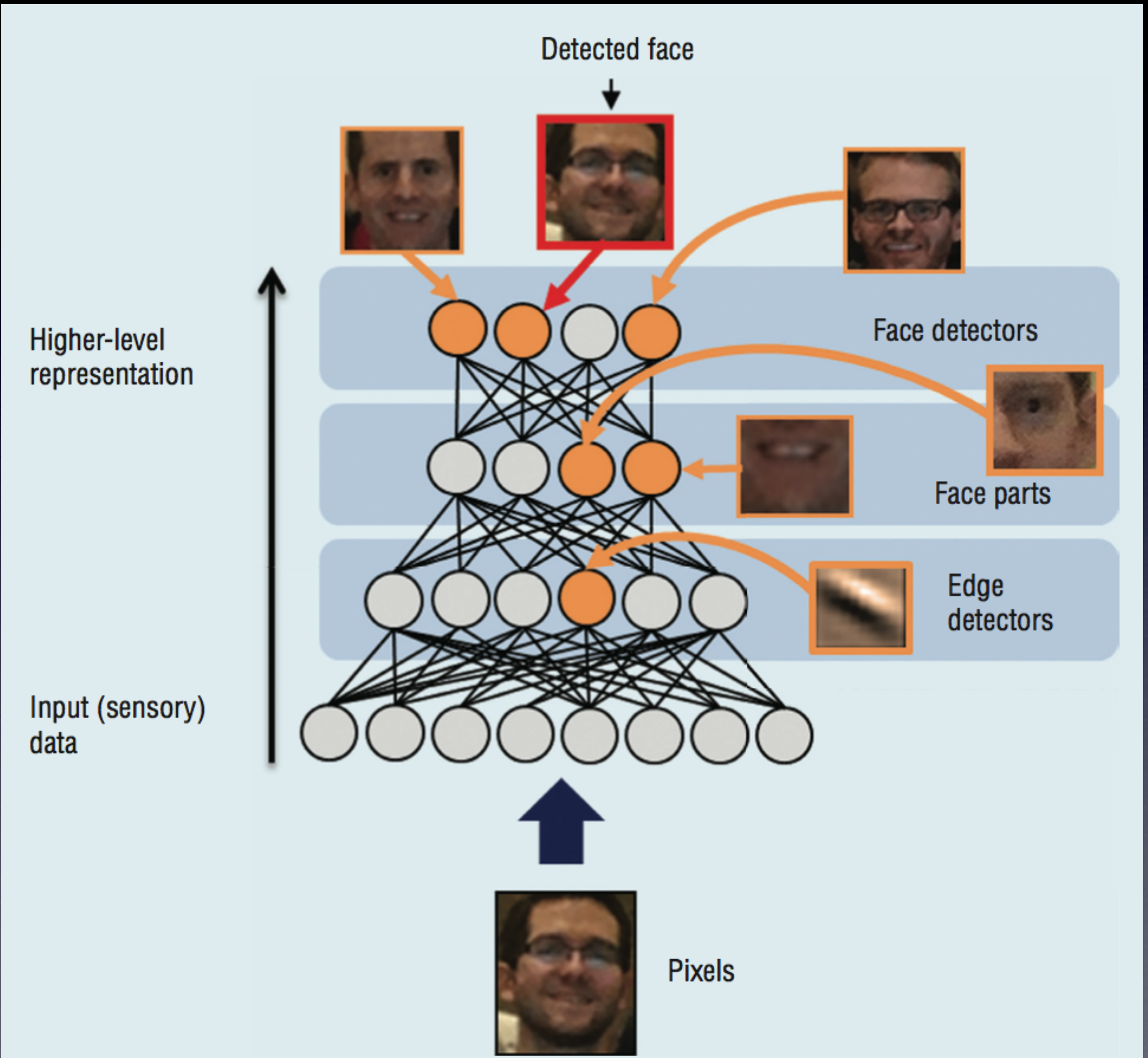
Figure 18.23 (a) The result of combining two opposite-facing soft threshold functions* to produce a ridge. (b) The result of combining two ridges to produce a bump.

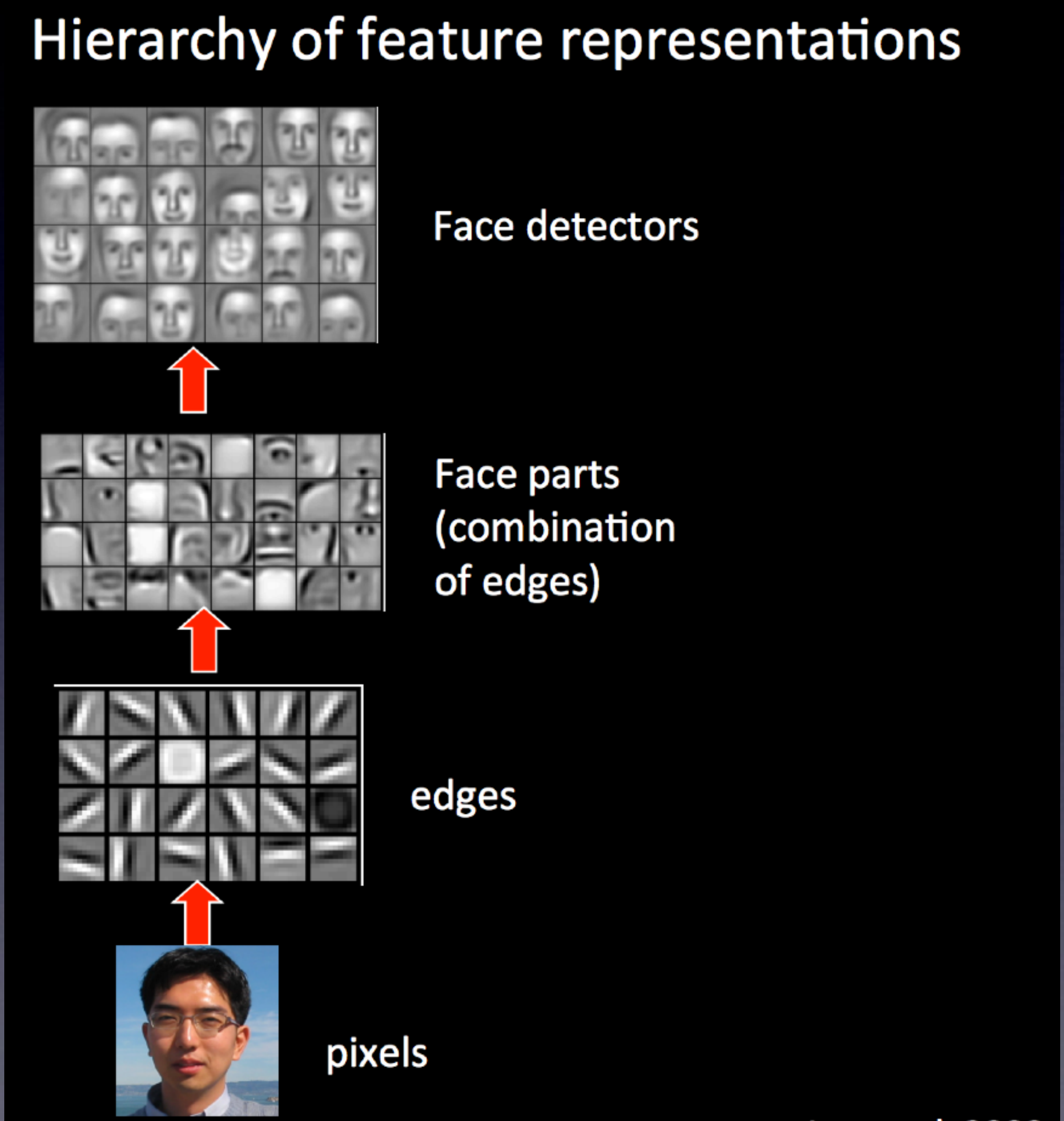
* and a threshold

Neural Networks

- Multi-layer networks thus allow “non-linear regression”
- Single hidden layer (often very large):
  - can represent any continuous function
- Two hidden layers:
  - can represent any discontinuous function

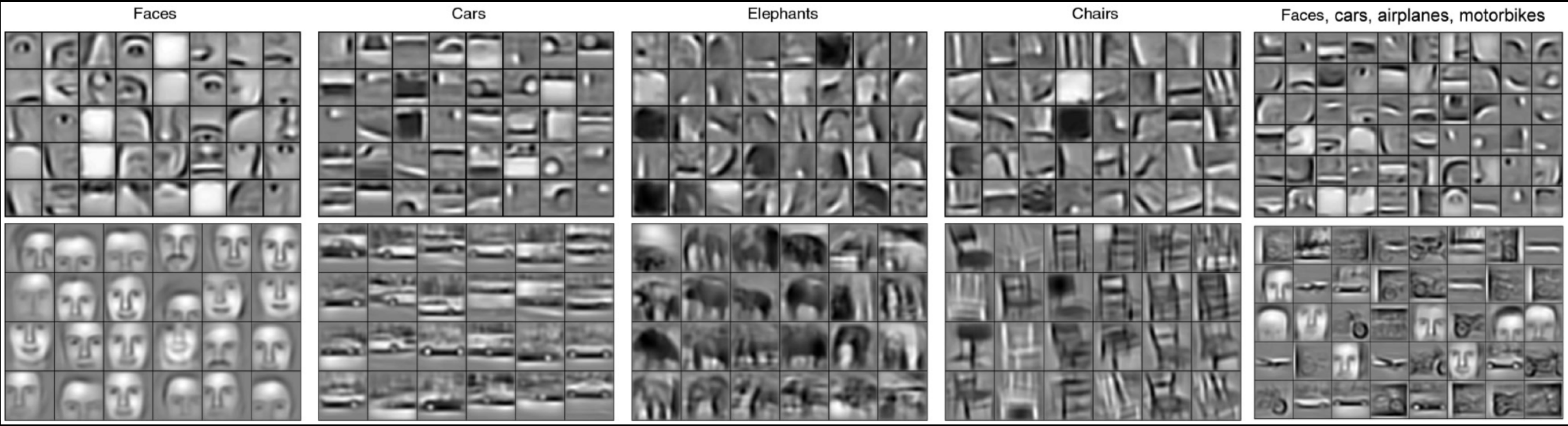
Hierarchically composed feature representations





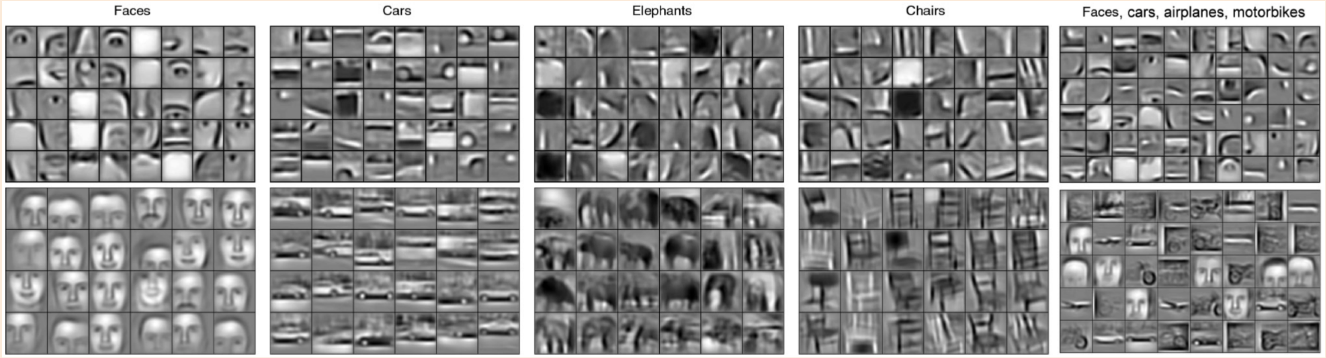
Lee et al, 2009.

Learning features relevant to the data



Lee et al. 2007

Learning features relevant to the data



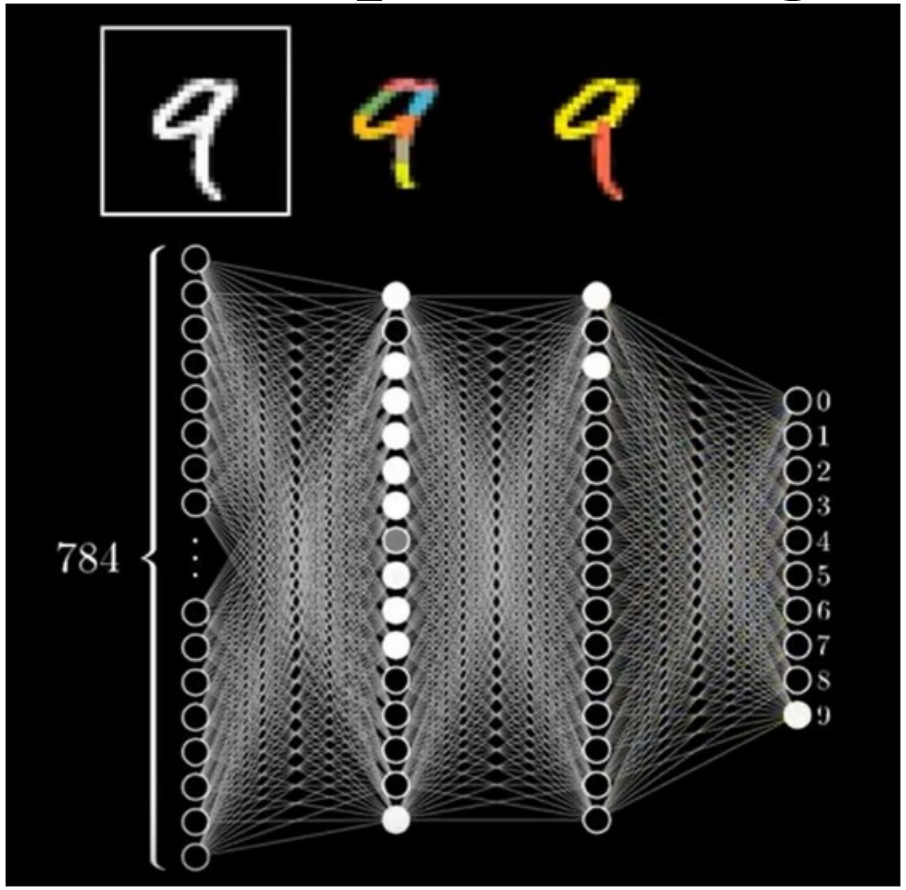
- I give a whole talk on work my colleagues and I have done in visualizing deep neural networks that I think you will enjoy
  - Deep Learning Overview & Visualizing What Deep Neural Networks Learn
  - https://www.youtube.com/watch?v=3Ip9eN5JE2A

Lehman, Clune, & Risi 2015

Lee et al. 2007

“Hierarchies of Parts” Motivation for Deep Learning

- Each “neuron” might recognize a “part” of a digit.
  - “Deeper” neurons might recognize combinations of parts .
  - Represent complex objects as hierarchical combinations of re -useable parts (a simple “grammar”).

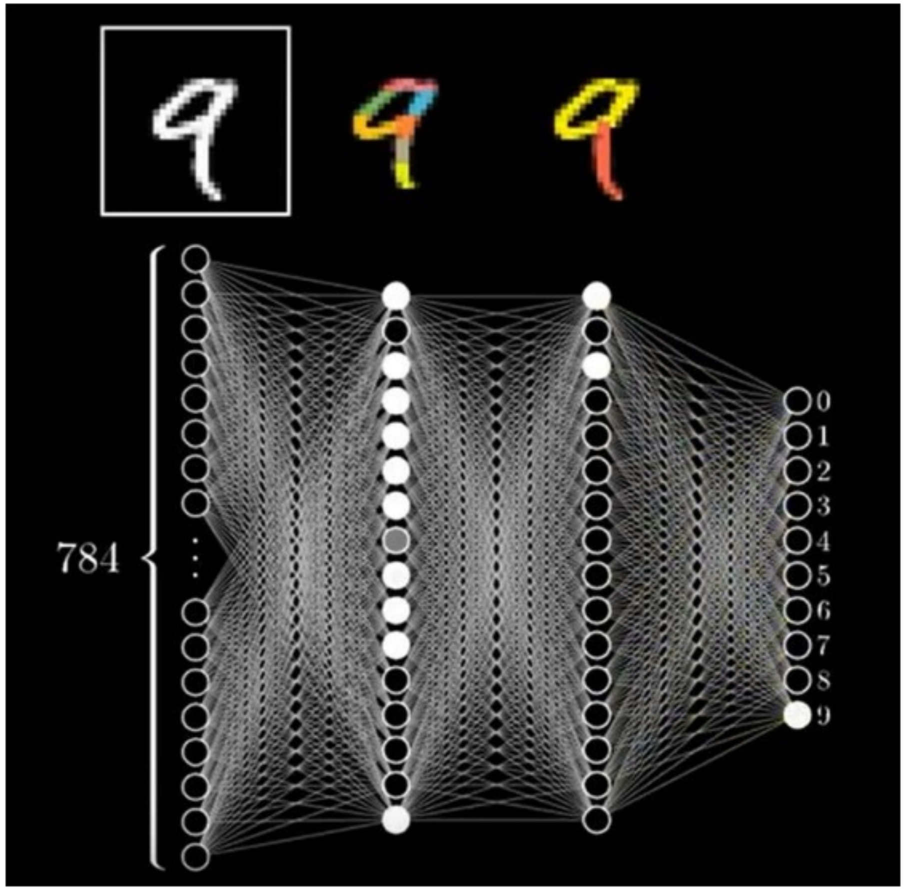


- Watch the full video here:
  - https://www.youtube.com/watch?v=aircAruvnKk

- Theory:
  - 1 big -enough hidden layer already gives universal approximation.
  - But some functions require exponentially -fewer parameters to approximate with more layers (can fight curse of dimensionality).

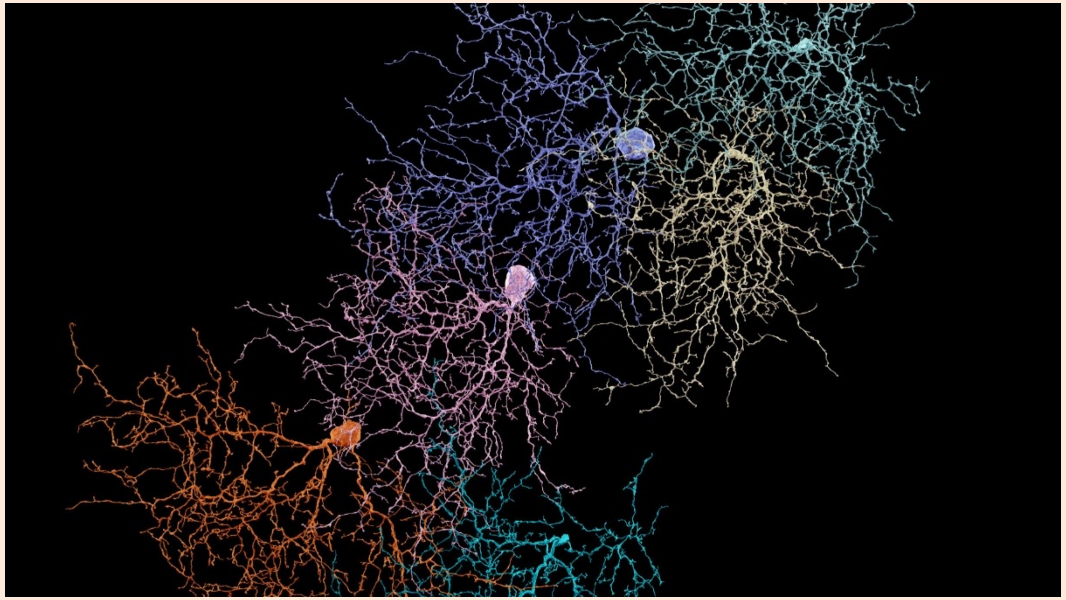
Deep Learning Terminology

- "layer" = number of layers of weights (numWs + V)
- "hidden layer" = number of activation layers (Zs) (not including inputs,)
- Everyone: use both terms to describe this net



Why Multiple Layers?

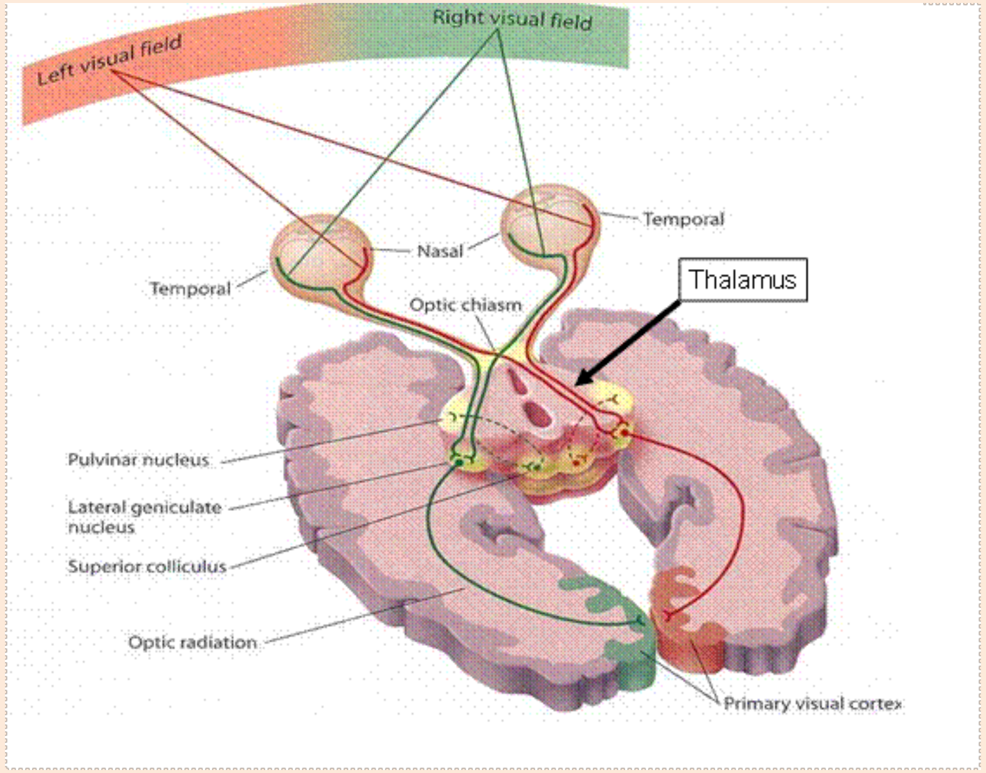
- Historically, deep learning was motivated by "connectionist" ideas:
  - Brain consists of network of highly-connected simple units.
    - Same units repeated in various places.
    - Computations are done in parallel.
    - Information is stored in distributed way.
    - Learning comes from updating of connection strengths.
    - One learning algorithm used everywhere.

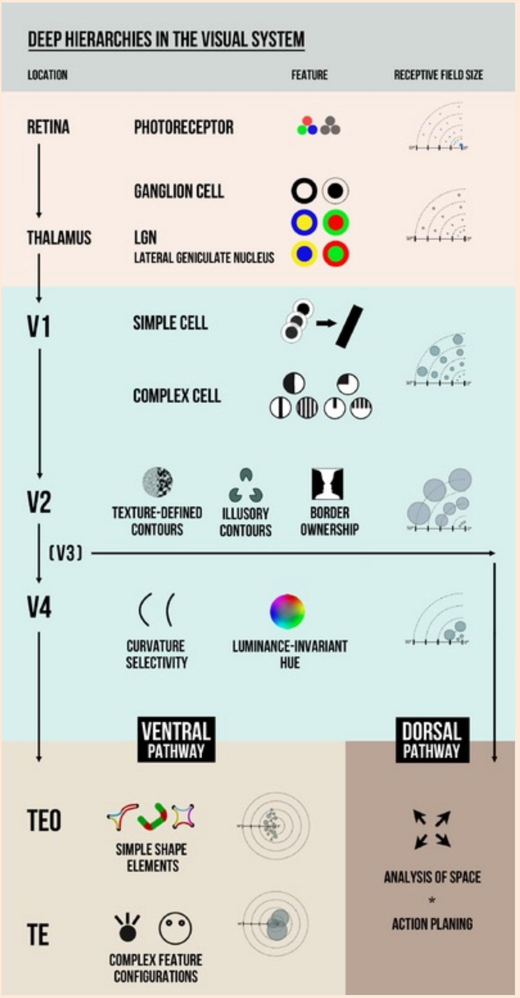


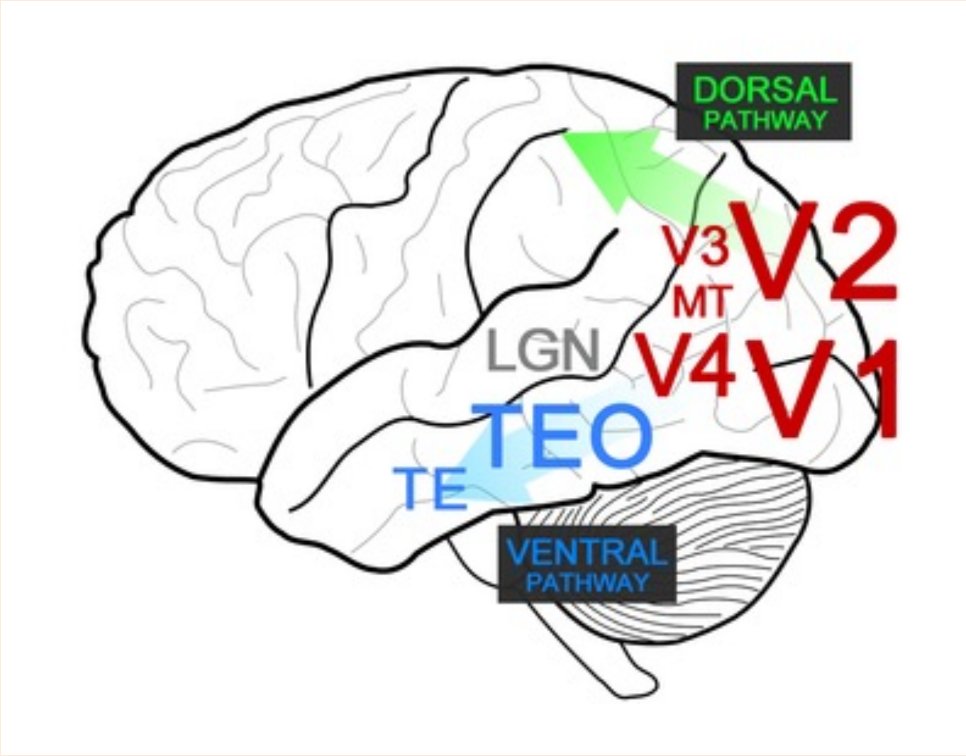
https://www.nytimes.com/2015/01/11/magazine/sebastian-seungs-quest-to-map-the-human-brain.html

Why Multiple Layers?

- And theories on the hierarchical organization of the visual system:



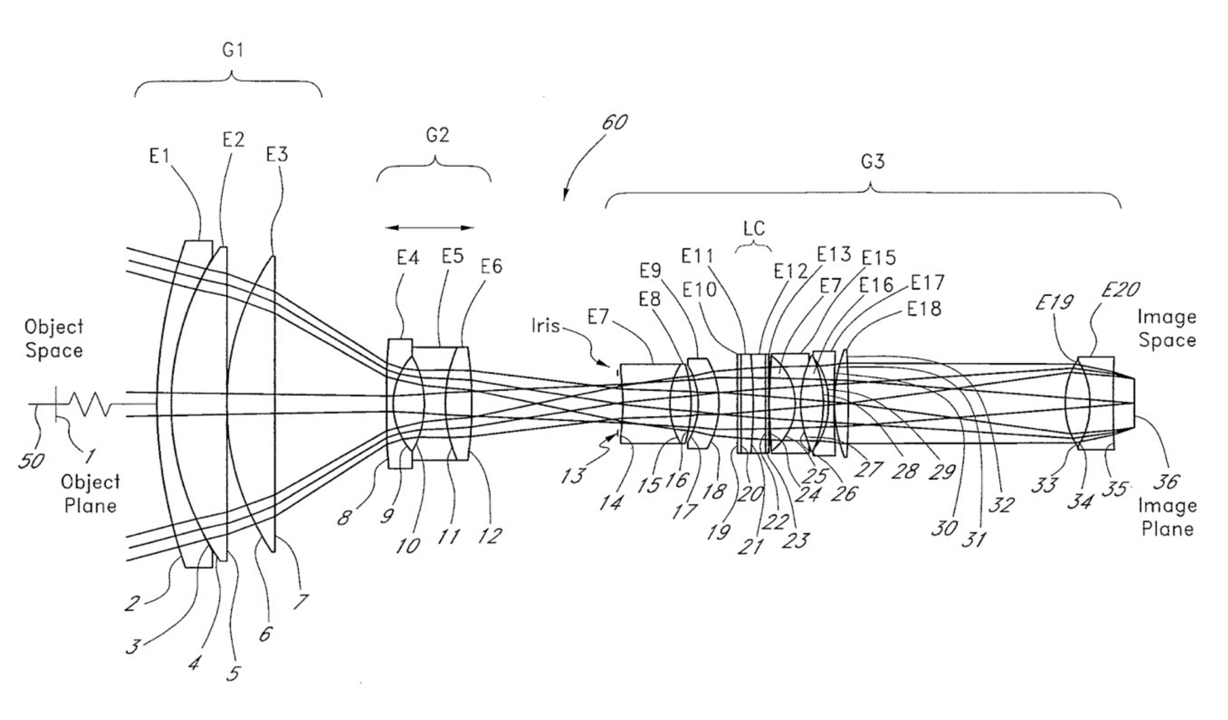


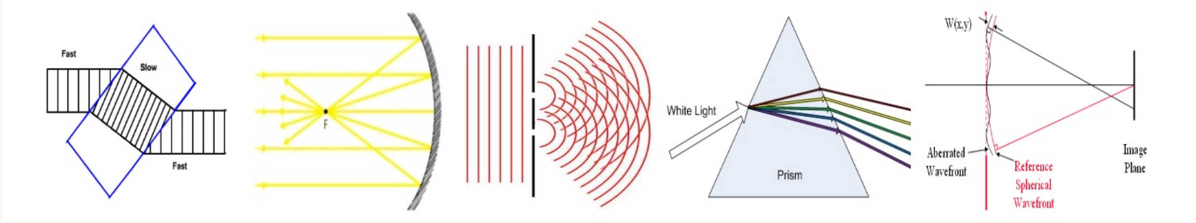


http://www.strokenetwork.org/newsletter/articles/vision.htm
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing

Why Multiple Layers?

- The idea of multi-layer designs appears in engineering too:
  - Deep hierarchies in camera design:





http://www.argmin.net/2018/01/25/optics/

Why Multiple Layers?

- There are also mathematical motivations for using multiple layers:
  - 1 layer gives us a universal approximator.
    - But this layer might need to be huge.
  - With deep networks:
    - Some functions can be approximated with exponentially-fewer parameters.
      - Compared to a network with 1 hidden layer.
    - So deep networks may need fewer parameters than “shallow but wide” networks.
      - And hence may need less data to train.
- Empirical motivation for using multiple layers:
  - In many domains deep networks have led to unprecedented performance.

Deep Learning

Linear model:

$$\hat{y}_i = w^T x_i$$

Neural network with 1 hidden layer:

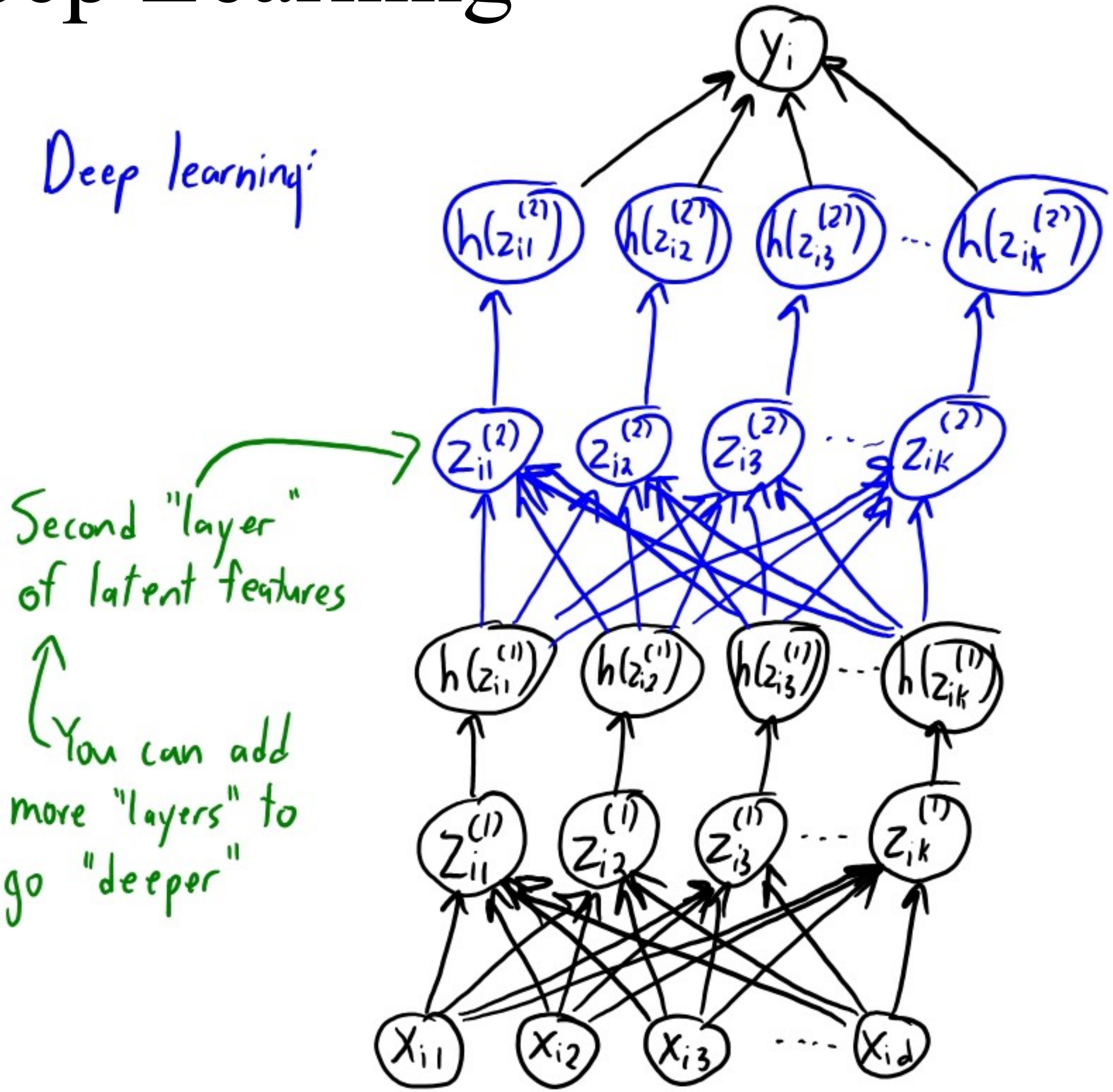
$$\hat{y}_i = v^T h(\underbrace{W x_i}_{z_i})$$

Neural network with 2 hidden layers:

$$\hat{y}_i = v^T h(\underbrace{W^{(2)} h(\underbrace{W^{(1)} x_i}_{z_i^{(1)}})}_{z_i^{(2)}})$$

Neural network with 3 hidden layers

$$\hat{y}_i = v^T h(\underbrace{W^{(3)} h(\underbrace{W^{(2)} h(\underbrace{W^{(1)} x_i}_{z_i^{(1)}})}_{z_i^{(2)}})}_{z_i^{(3)}})$$



Summary

- Neural networks learn features z_i for supervised learning.
- Sigmoid function avoids degeneracy by introducing non-linearity.
  – Universal approximator with large - enough 'k'.
- Biological motivation for (deep) neural networks.
- Deep learning considers neural networks with many hidden layers.
  – Can more- efficiently represent some functions.
- Unprecedented performance on difficult pattern recognition tasks.

bonus!

Multiple Word Prototypes

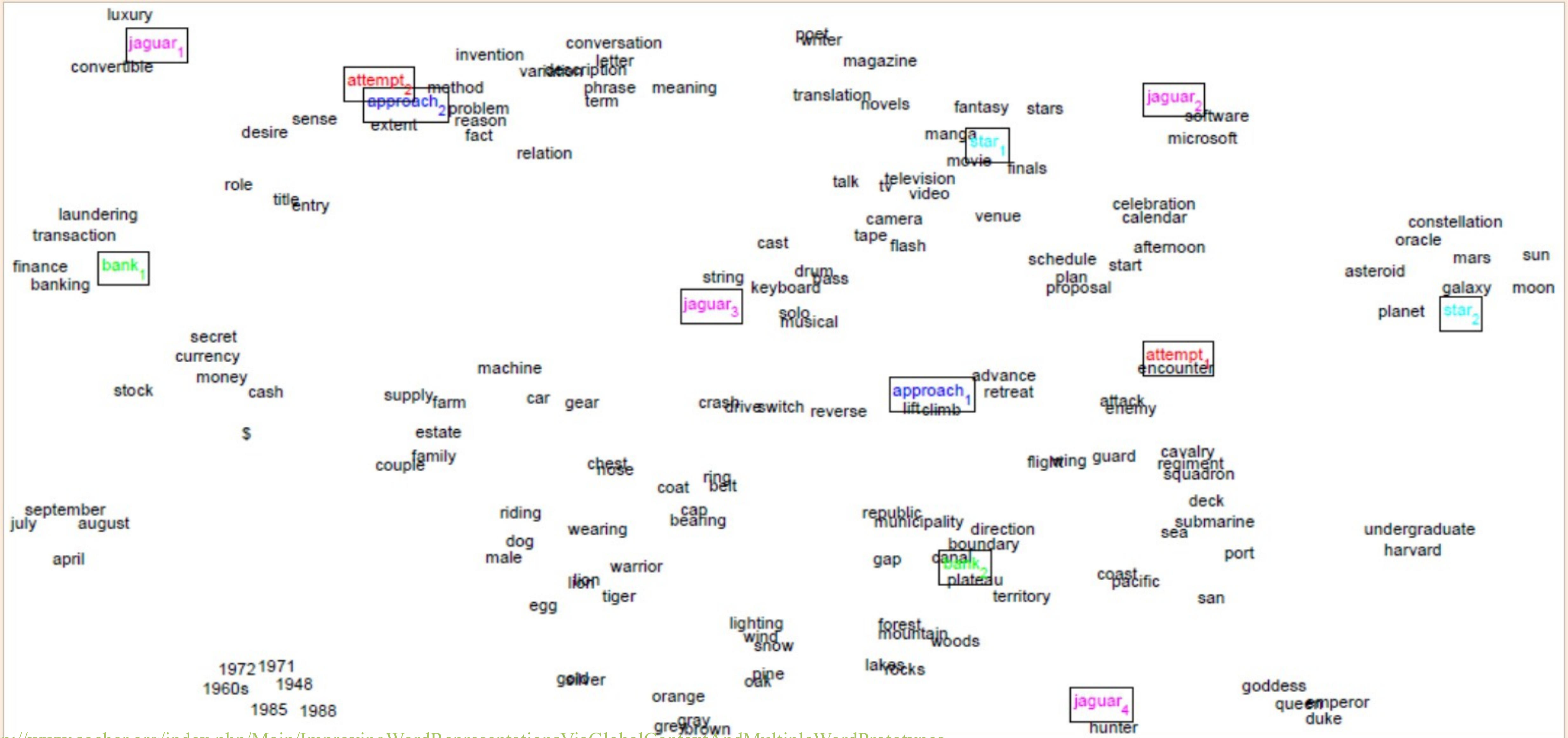
- What about homonyms and polysemy?
  - The word vectors would need to account for all meanings.

- More recent approaches:
  - Try to cluster the different contexts where words appear.
  - Use different vectors for different contexts .

$$X_{jaguar} = \begin{array}{c} \begin{bmatrix} \cdots \\ \cdots \\ \end{bmatrix} \end{array} \begin{array}{l} z_{j1} \\ z_{j2} \\ z_{j3} \end{array}$$

bonus!

Multiple Word Prototypes



http://www.socher.org/index.php/Main/ImprovingWordRepresentationsViaGlobalContextAndMultipleWordPrototypes

Why $z_i = Wx_i$?

bonus!

- In PCA we had that the optimal $Z = XW^T(WW^T)^{-1}$.
- If W had normalized+orthogonal rows, $Z = XW^T$ (since $WW^T = I$).
  - So $z_i = Wx_i$ in this normalized+orthogonal case.

- Why we would use $z_i = Wx_i$ in neural networks?
  - We didn't enforce normalization or orthogonality.

- Well, the value $W^T(WW^T)^{-1}$ is just "some matrix".
  - You can think of neural networks as just directly learning this matrix.

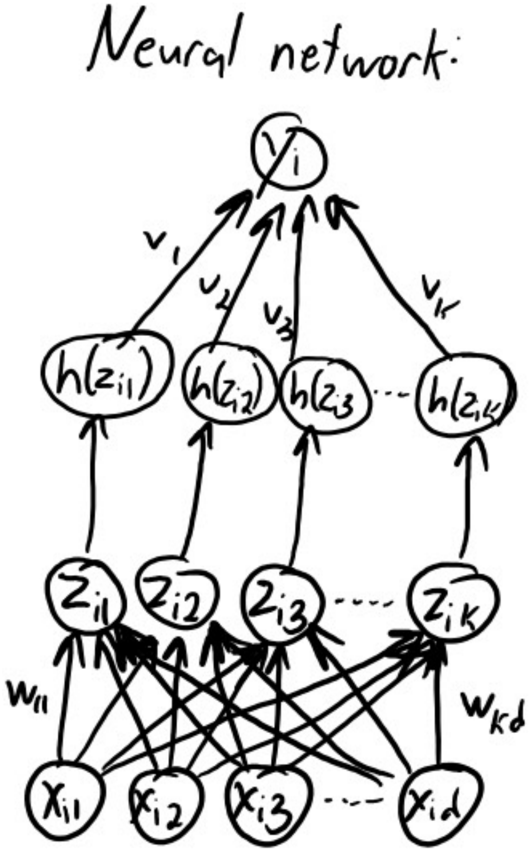
(pause)

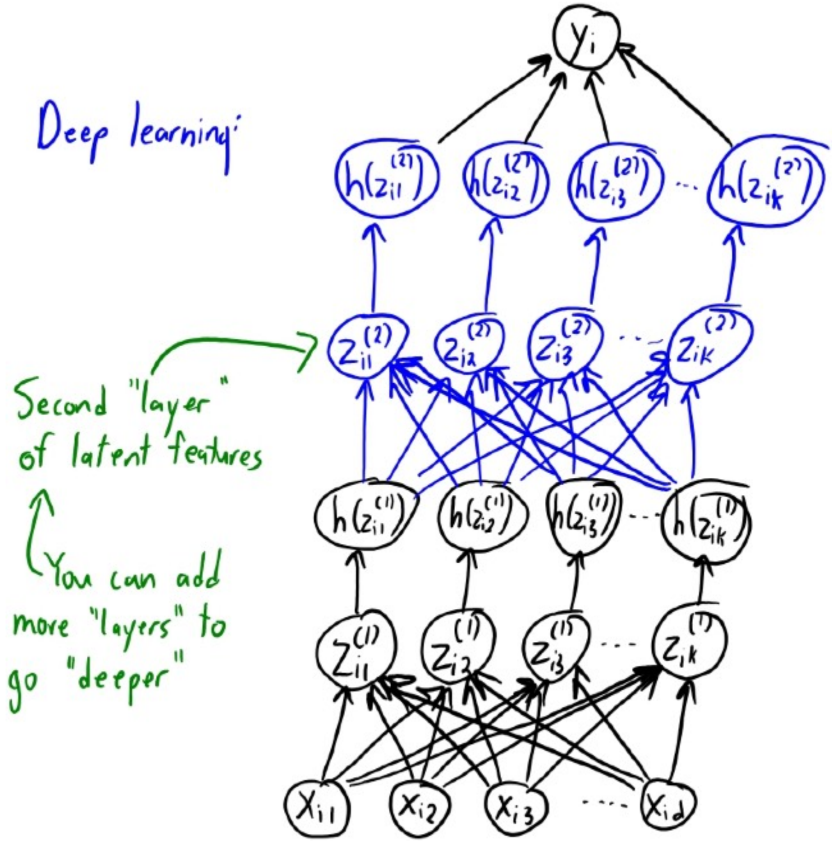
CPSC 340: Machine Learning and Data Mining

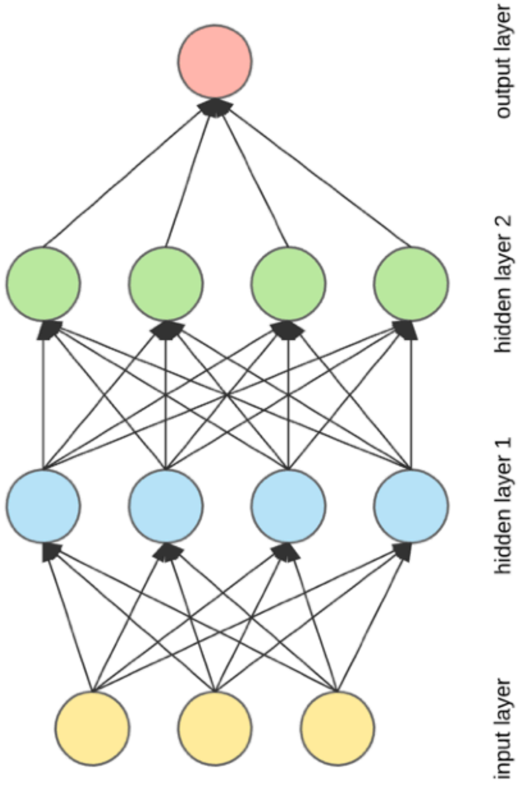
Deep Learning

Recap of Last Time

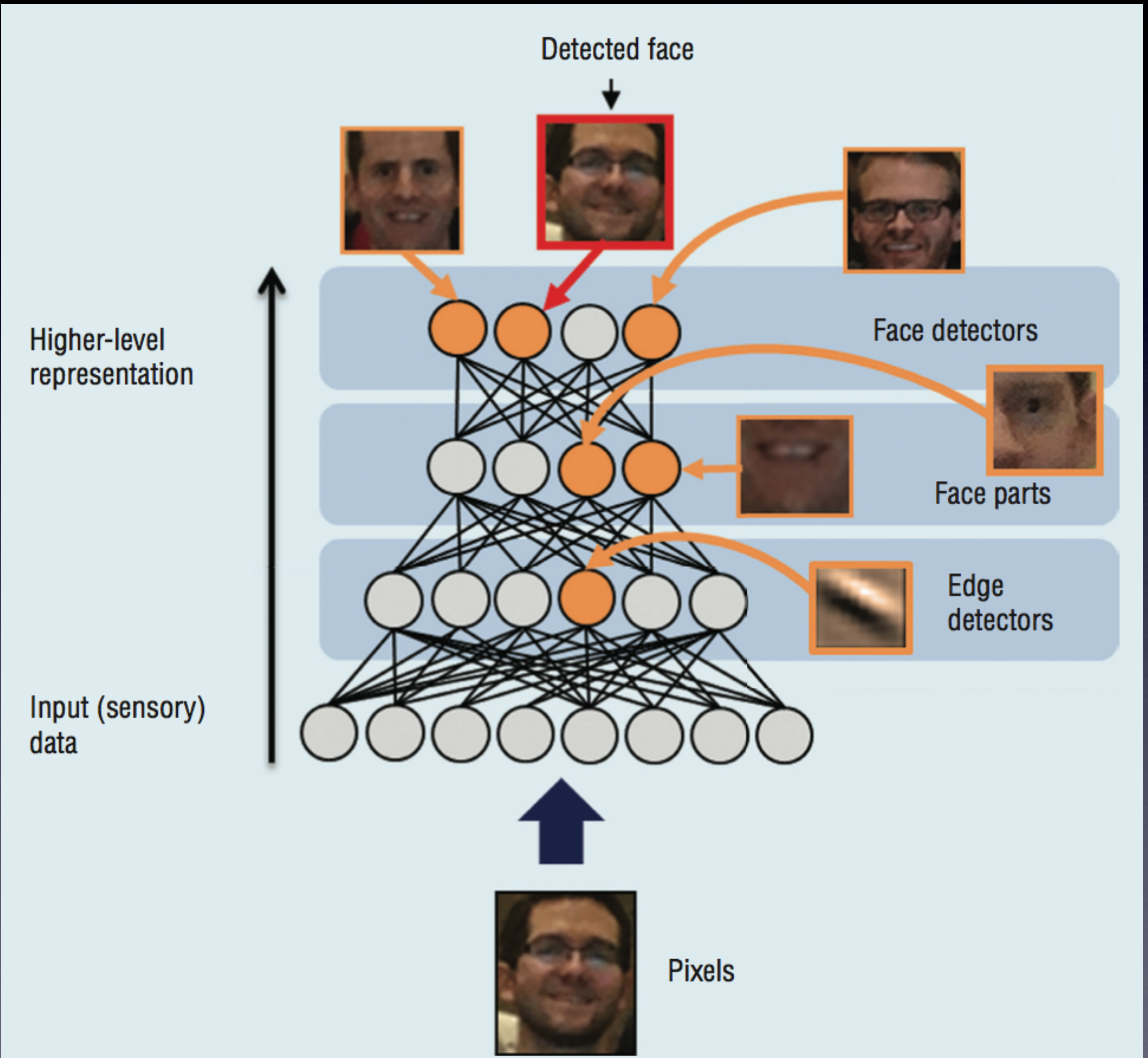
Deep Learning

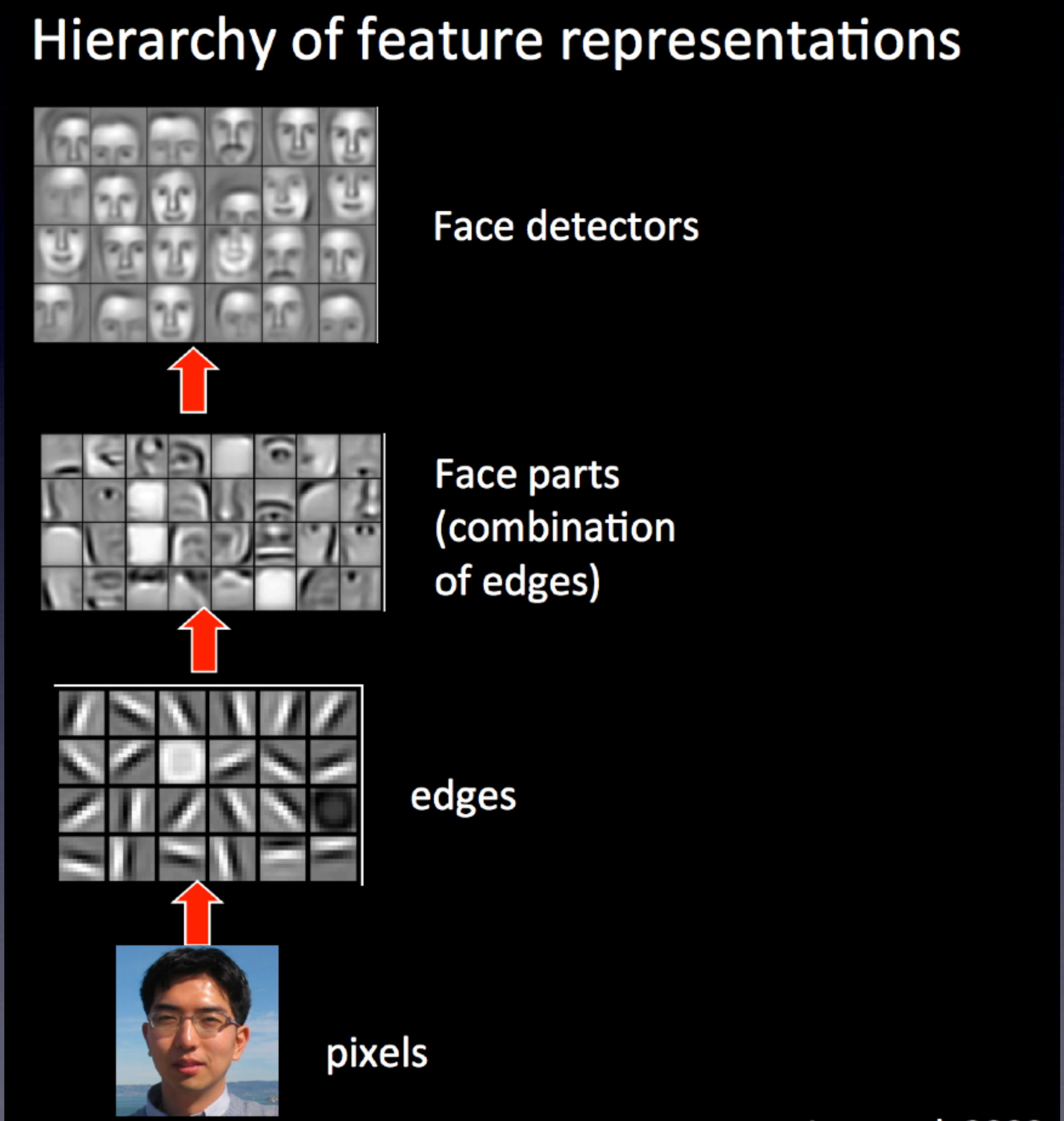






Hierarchically composed feature representations

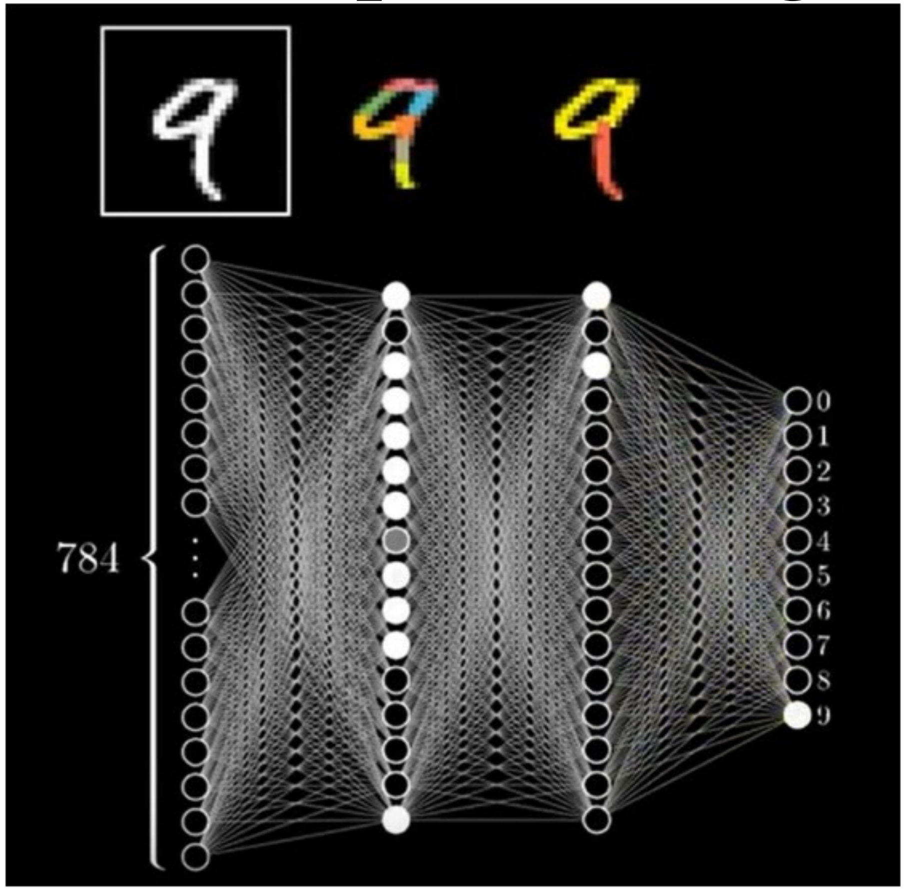




Lee et al, 2009.

“Hierarchies of Parts” Motivation for Deep Learning

- Each “neuron” might recognize a “part” of a digit.
  - “Deeper” neurons might recognize combinations of parts .
  - Represent complex objects as hierarchical combinations of re -useable parts (a simple “grammar”).



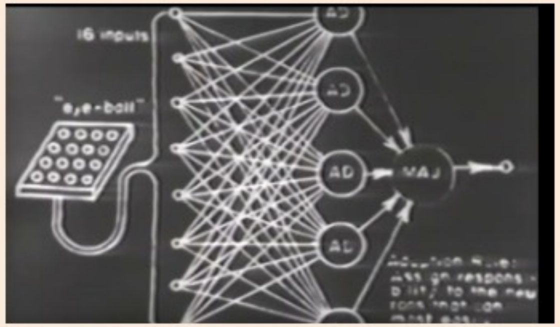
- Watch the full video here:
  - https://www.youtube.com/watch?v=aircAruvnKk

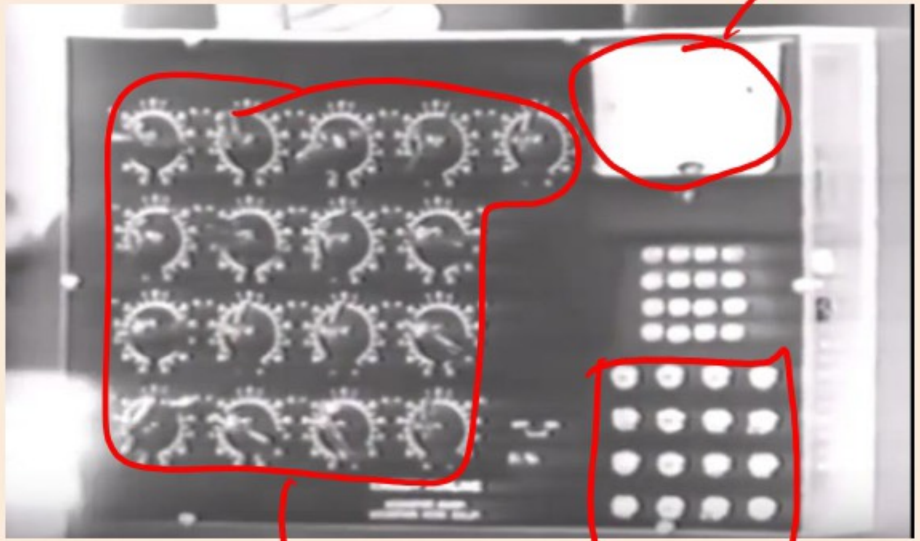
- Theory:
  - 1 big -enough hidden layer already gives universal approximation.
  - But some functions require exponentially -fewer parameters to approximate with more layers (can fight curse of dimensionality).

bonus!

ML and Deep Learning History

- 1950 and 1960s: Initial excitement.
  - Perceptron : linear classifier and stochastic gradient (roughly).
  - "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." New York Times (1958).
    - https://www.youtube.com/watch?v=IEFRtz68m-8
  - Object recognition assigned to students as a summer project





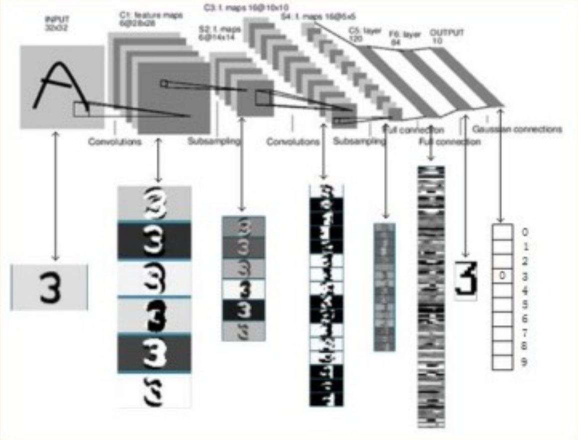
- Then drop in popularity:
  - Quickly realized limitations of linear models .

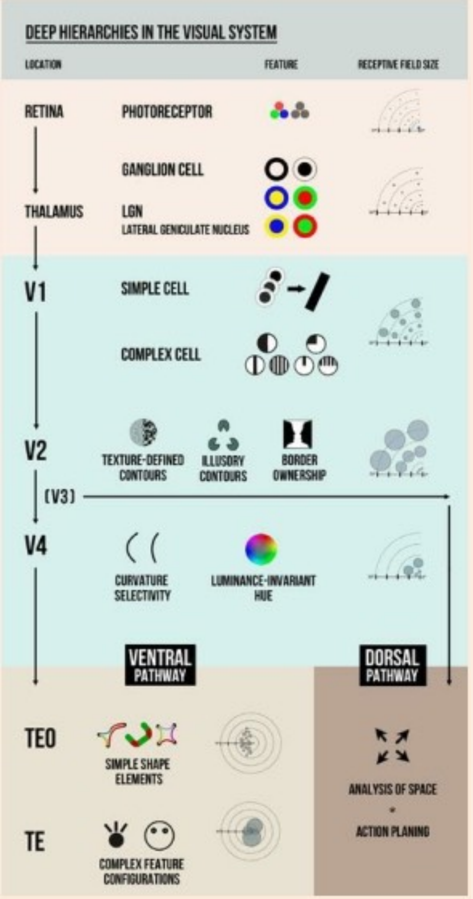
https://mitpress.mit.edu/books/perceptrons/

bonus!

ML and Deep Learning History

- 1970 and 1980s: Connectionism (brain -inspired ML)
  - Want “connected networks of simple units ”.
    - Use parallel computation and distributed representations.
  - Adding hidden layers z_i increases expressive power.
    - With 1 layer and enough sigmoid units, a universal approximator.
  - Success in optical character recognition.





https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing
http://www.datarobot.com/blog/a -primer -on-deep-learning/
http://blog.csdn.net/strint/article/details/44163869

bonus!

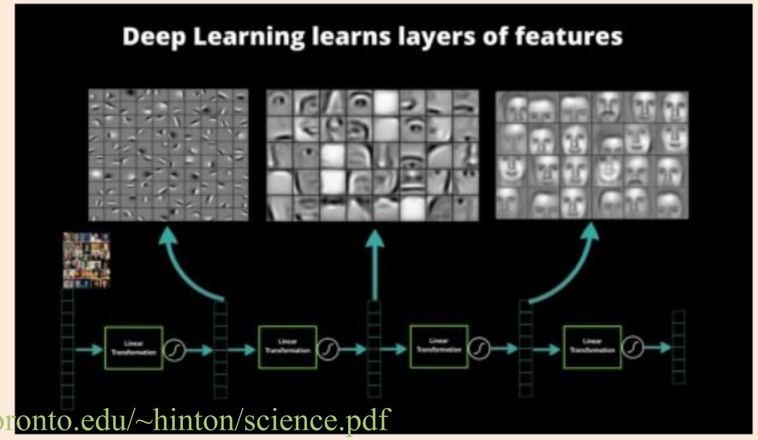
ML and Deep Learning History

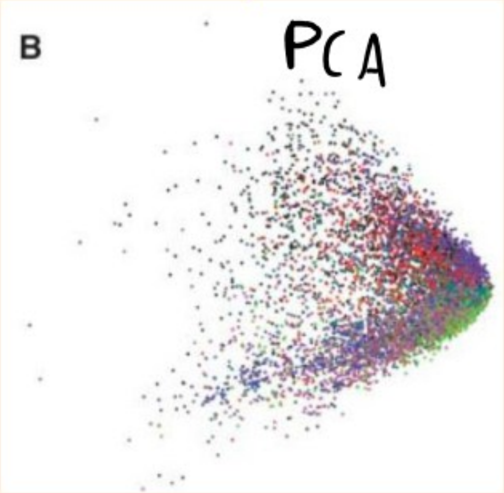
- 1990s and early -2000s: drop in popularity.
  - It proved really difficult to get multi - layer models working robustly.
  - We obtained similar performance with simpler models:
    - Rise in popularity of logistic regression and SVMs with regularization and kernels .
  - Lots of internet successes (spam filtering, web search, recommendation).
  - ML moved closer to other fields like numerical optimization and statistics.

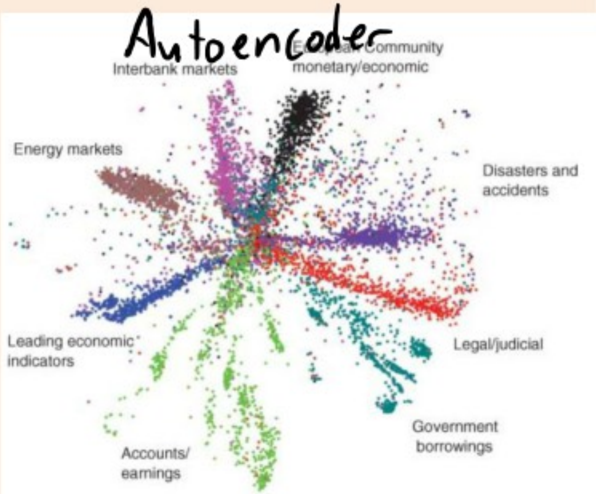
bonus!

ML and Deep Learning History

- Late 2000s: push to revive connectionism as "deep learning ".
  - Canadian Institute For Advanced Research (CIFAR) NCAP program:
    - "Neural Computation and Adaptive Perception".
    - Led by Geoff Hinton, Yann LeCun , and Yoshua Bengio
    - Unsupervised successes: "deep belief networks" and "autoencoders".
    - Could be used to initialize deep neural networks.





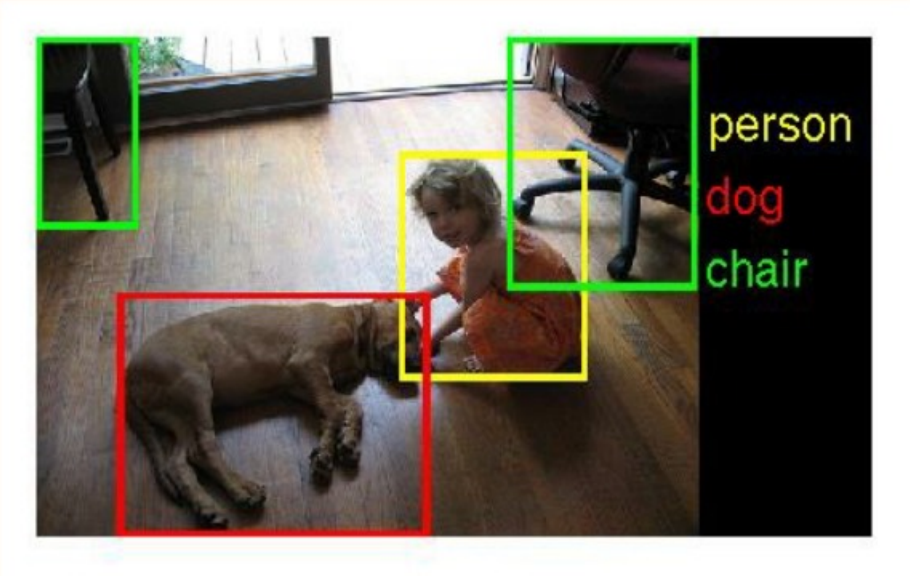


https://www.cs.toronto.edu/~hinton/science.pdf

bonus!

2010s: DEEP LEARNING!!!

- Bigger datasets, bigger models, parallel computing (GPUs/clusters).
  - And some tweaks to the models from the 1980s.
- Huge improvements in automatic speech recognition (2009).
  - All phones now have deep learning.
- Huge improvements in computer vision (2012).
  - Changed computer vision field almost instantly.
  - Now is how most CV (and AI) is done



http://www.image - net.org/challenges/LSVRC/2014/

bonus!

2010s: DEEP LEARNING!!!

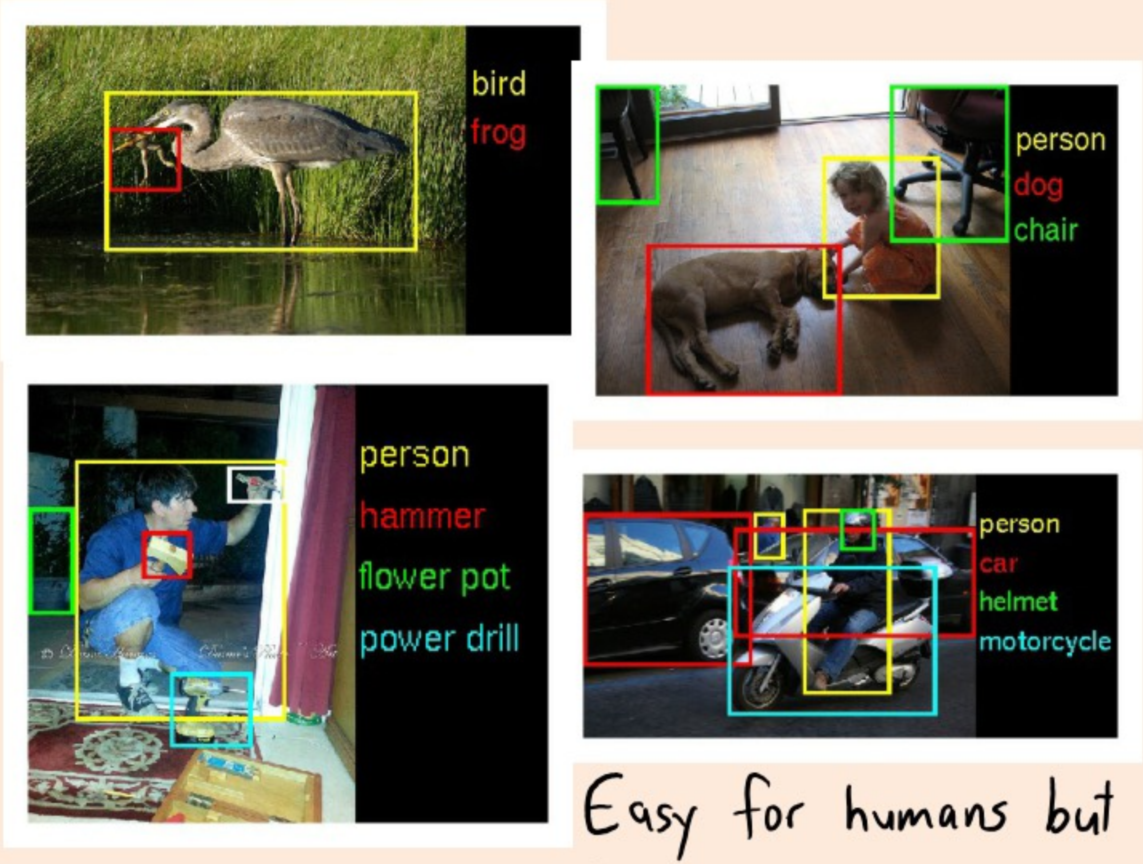
- Media hype:
  - "How many computers to identify a cat? 16,000"
    New York Times (2012).
  - "Why Facebook is teaching its machines to think like humans"
    Wired (2013).
  - "What is 'deep learning' and why should businesses care?"
    Forbes (2013).
  - "Computer eyesight gets a lot more accurate"
    New York Times (2014).

- 2015: huge improvement in language understanding

bonus!

ImageNet Challenge

- Millions of labeled images, 1000 object classes.



Easy for humans but <u>hard</u> for computers.

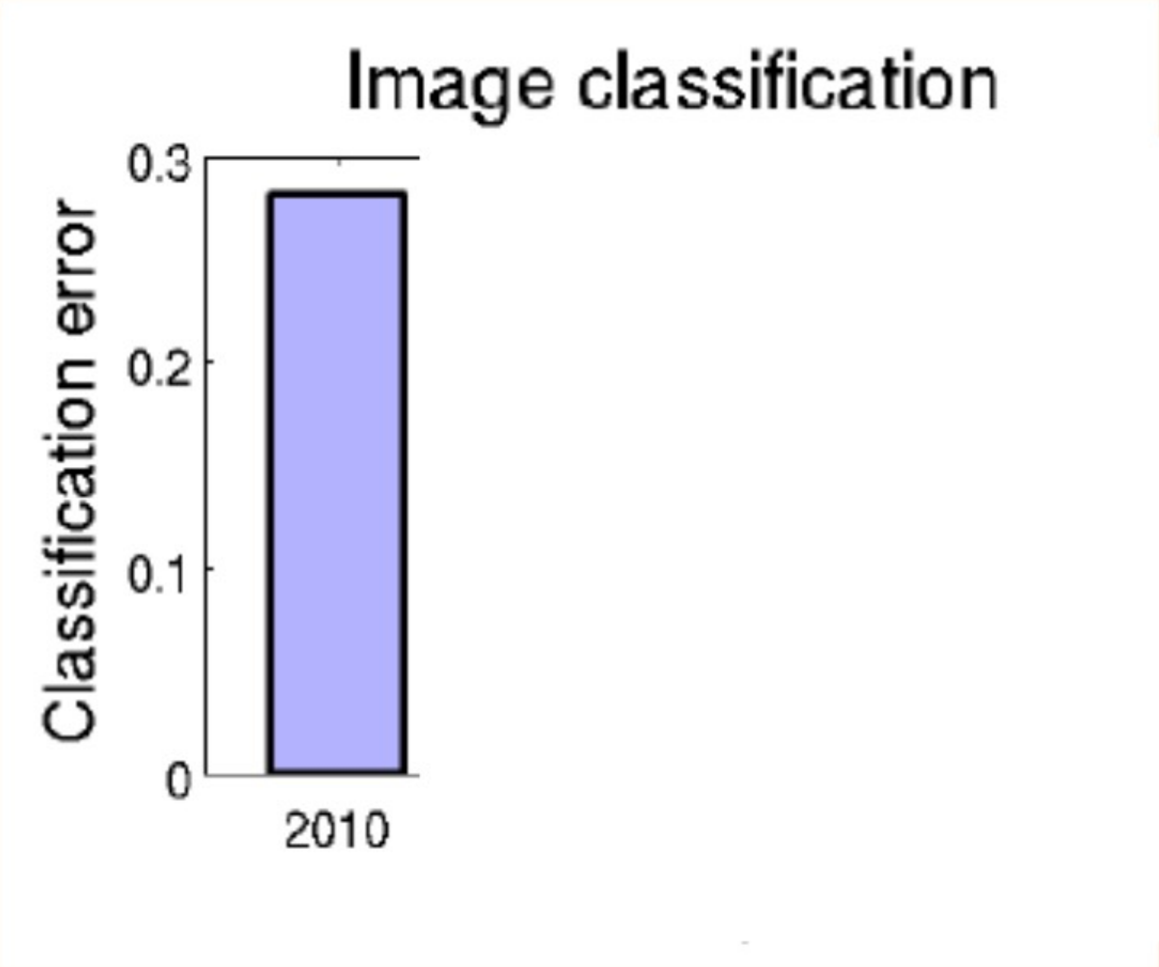
http://www.image -net.org/challenges/LSVRC/2014/

bonus!

ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

Syberian Husky Canadian Husky



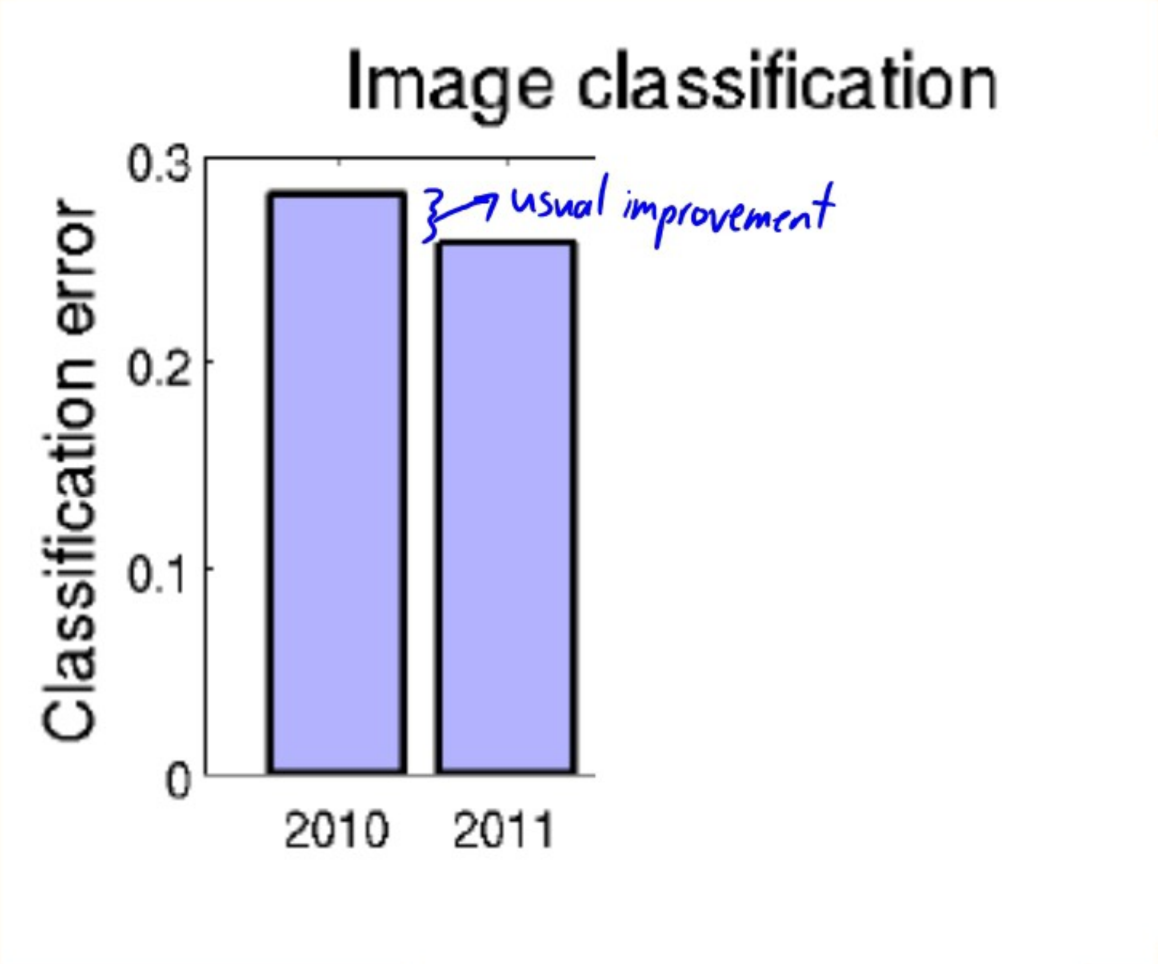
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

bonus!

ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

Syberian Husky Canadian Husky



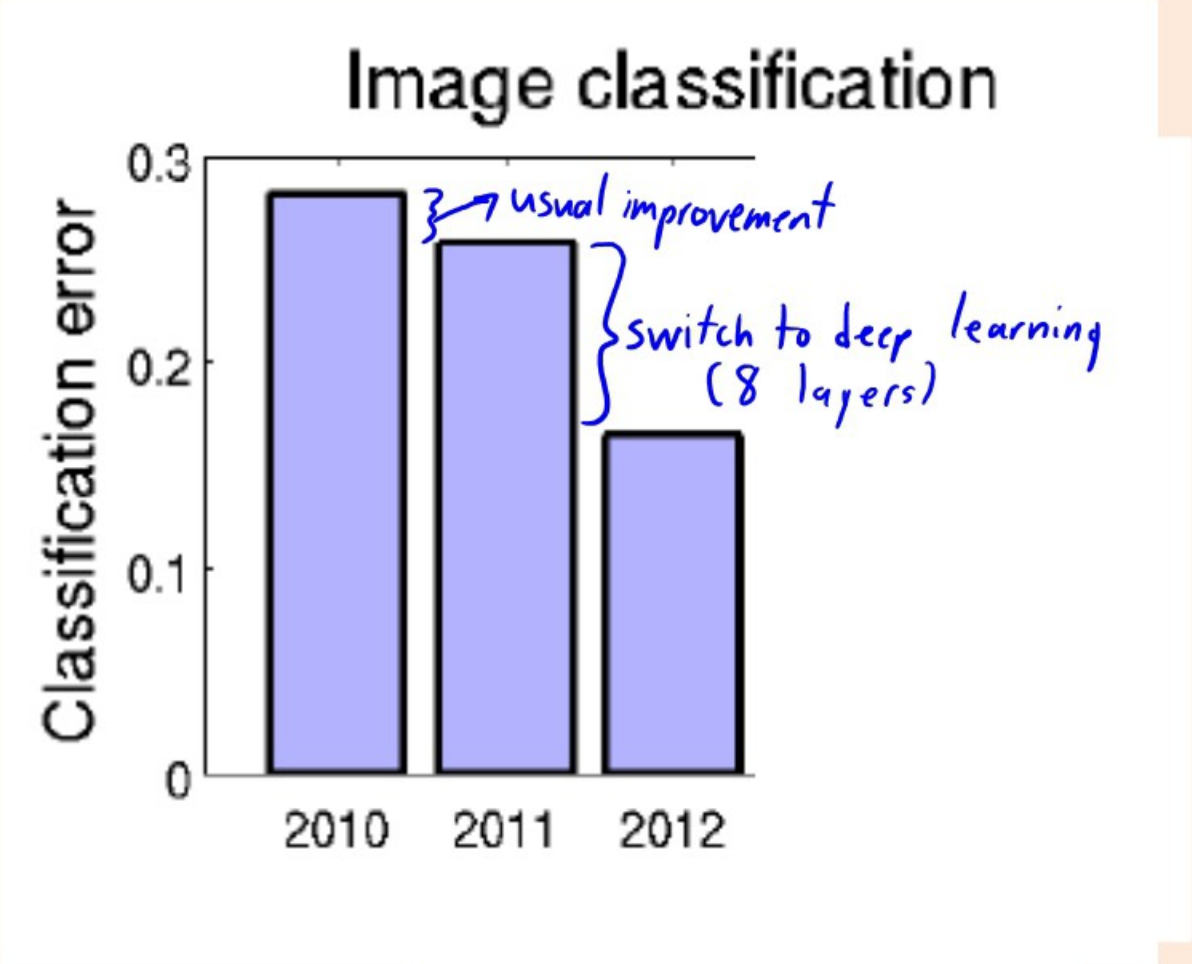
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

bonus!

ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

Syberian Husky   Canadian Husky



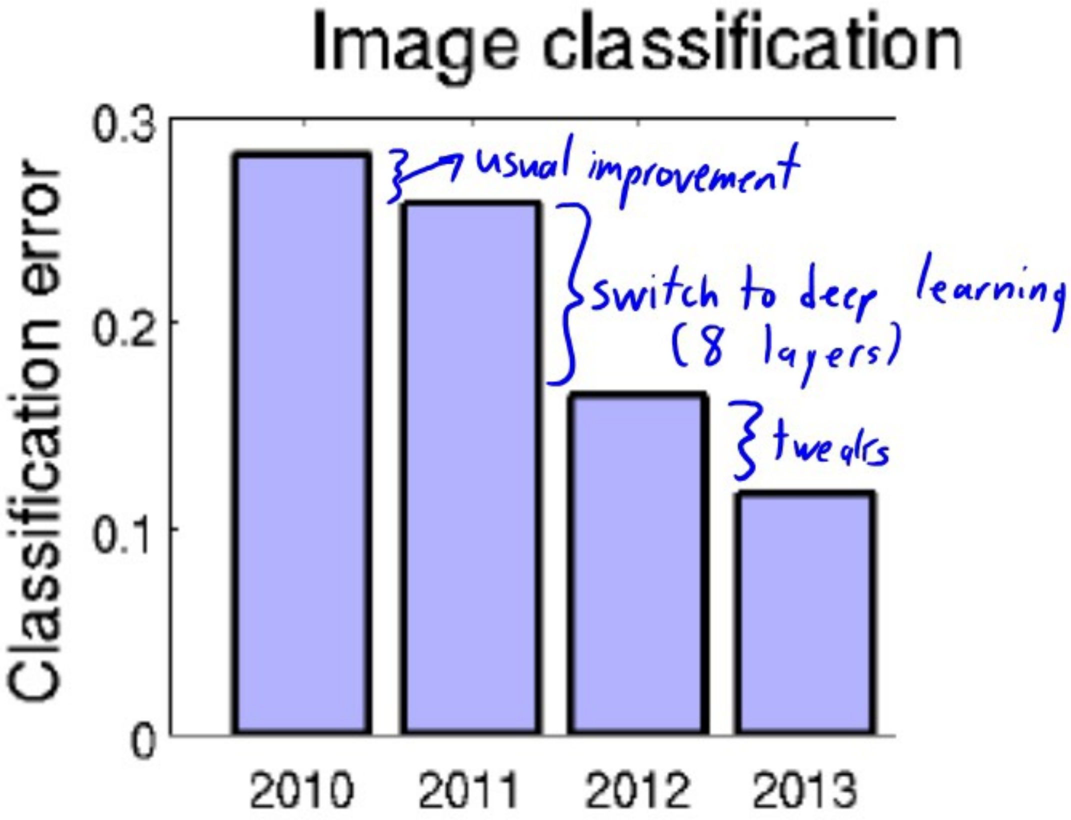
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

bonus!

ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

Syberian Husky Canadian Husky

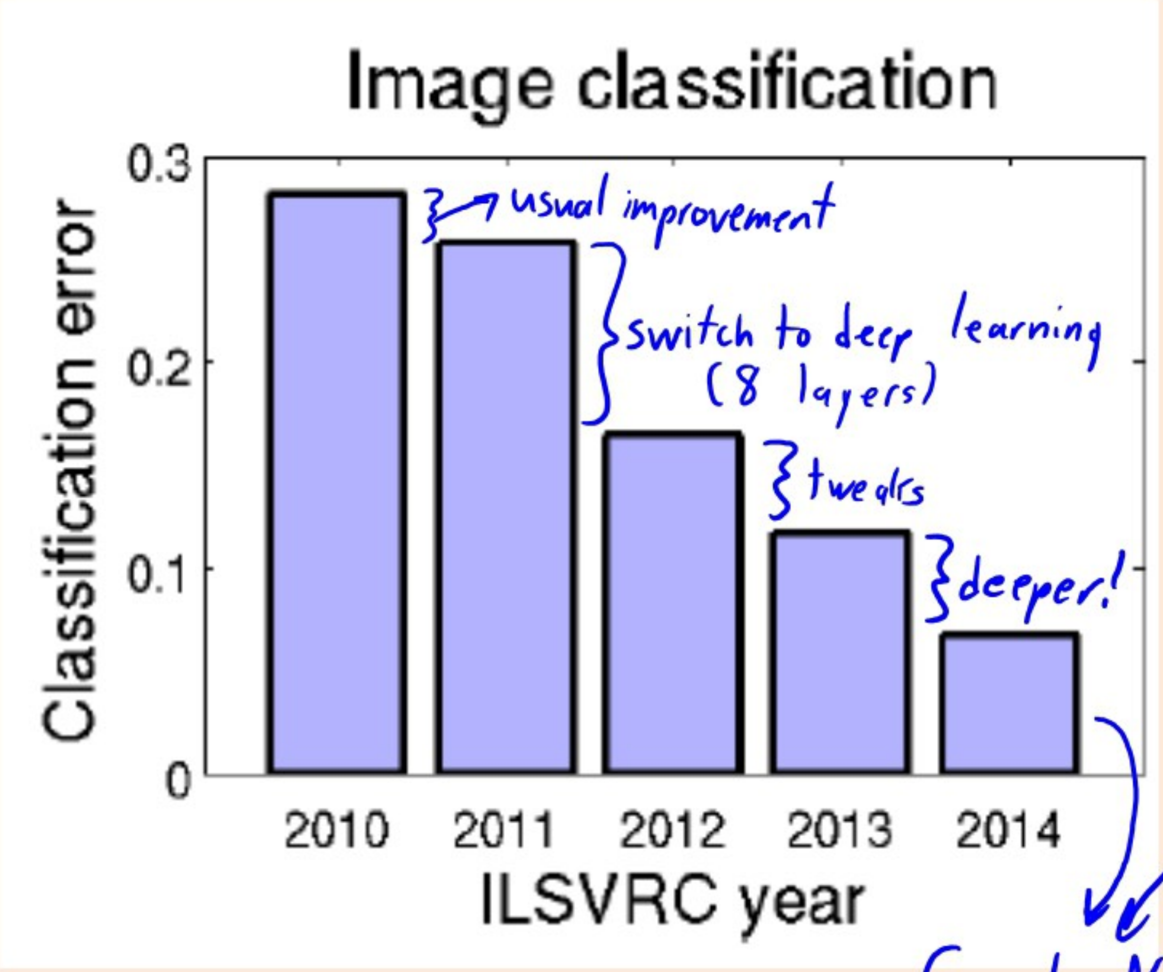


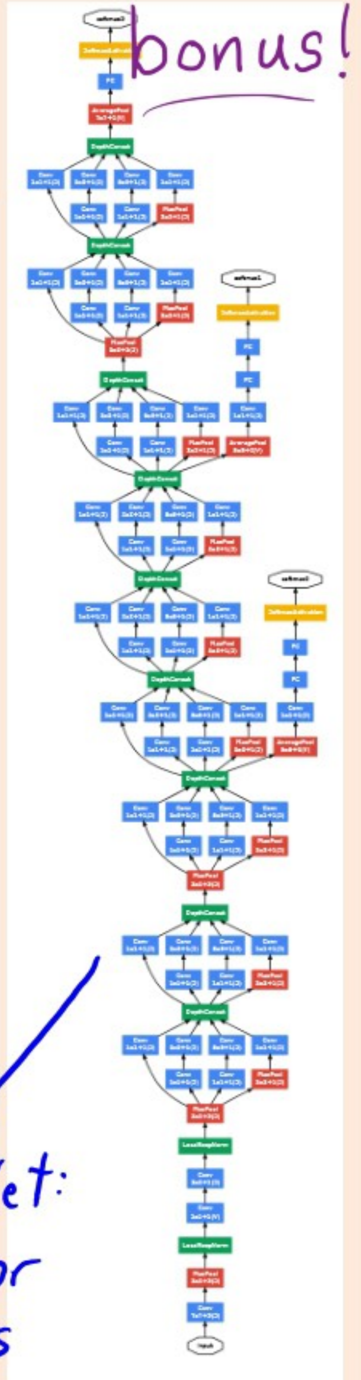
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

Syberian Husky Canadian Husky





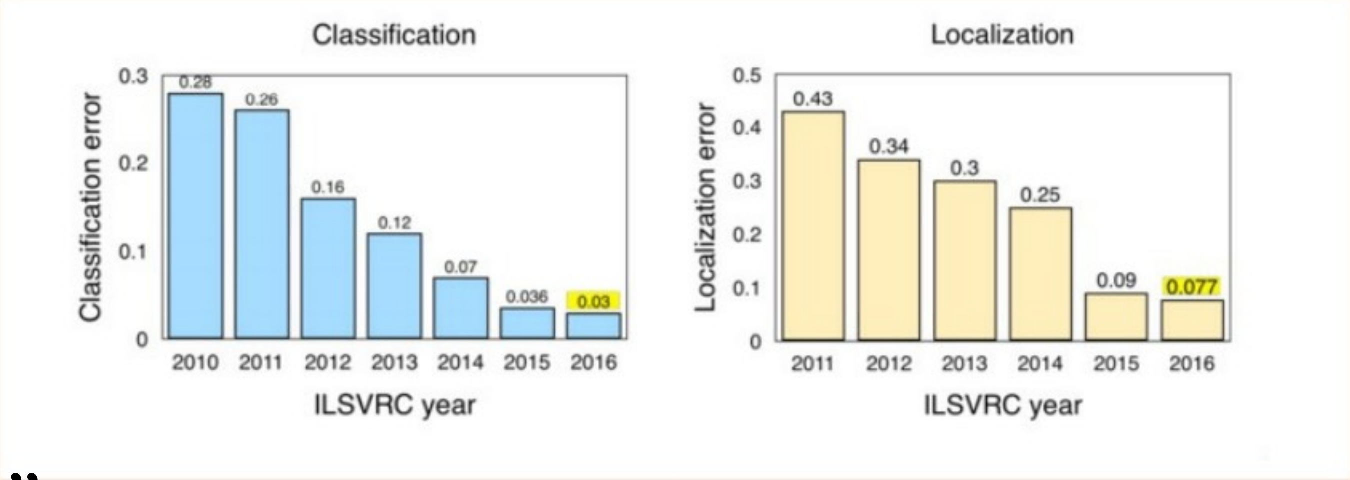
GoogLeNet:
6.7% error
22 layers

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

bonus!

ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.
- 2015: Won by Microsoft Asia
  - 3.6% error.
  - 152 layers, introduced "ResNets".
  - Also won "localization" (finding location of objects in images).
- 2016: Chinese University of Hong Kong:
  - Ensembles of previous winners and other existing methods.
- 2017: fewer entries, organizers decided this would be last year.



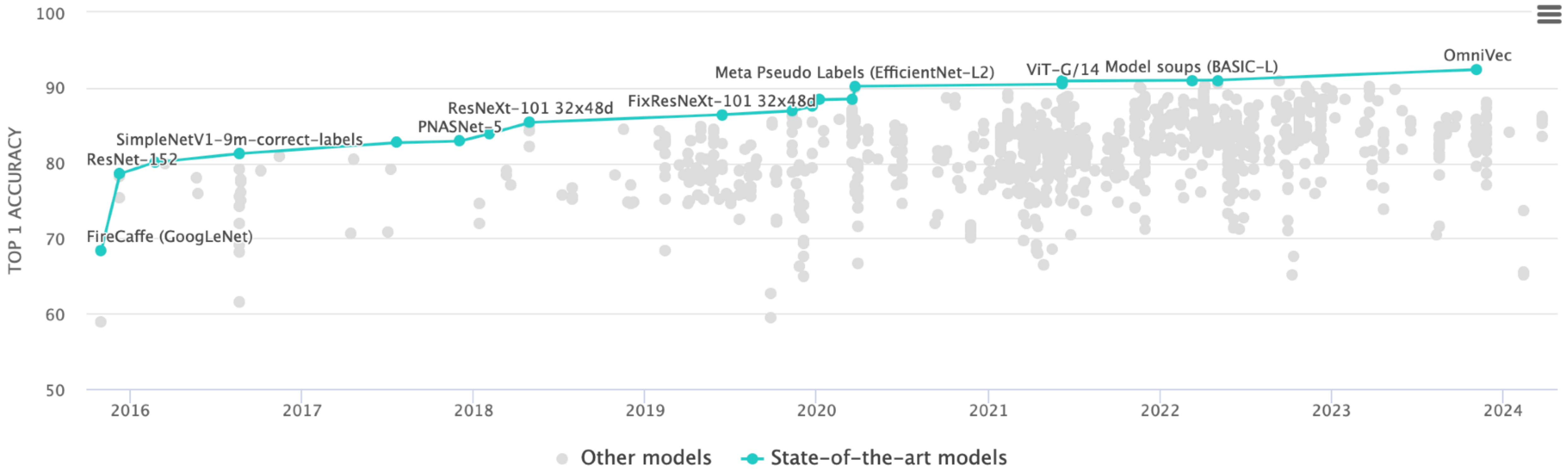
http://www.themtank.org/a-year-in-computer-vision

bonus!

Image Classification on ImageNet

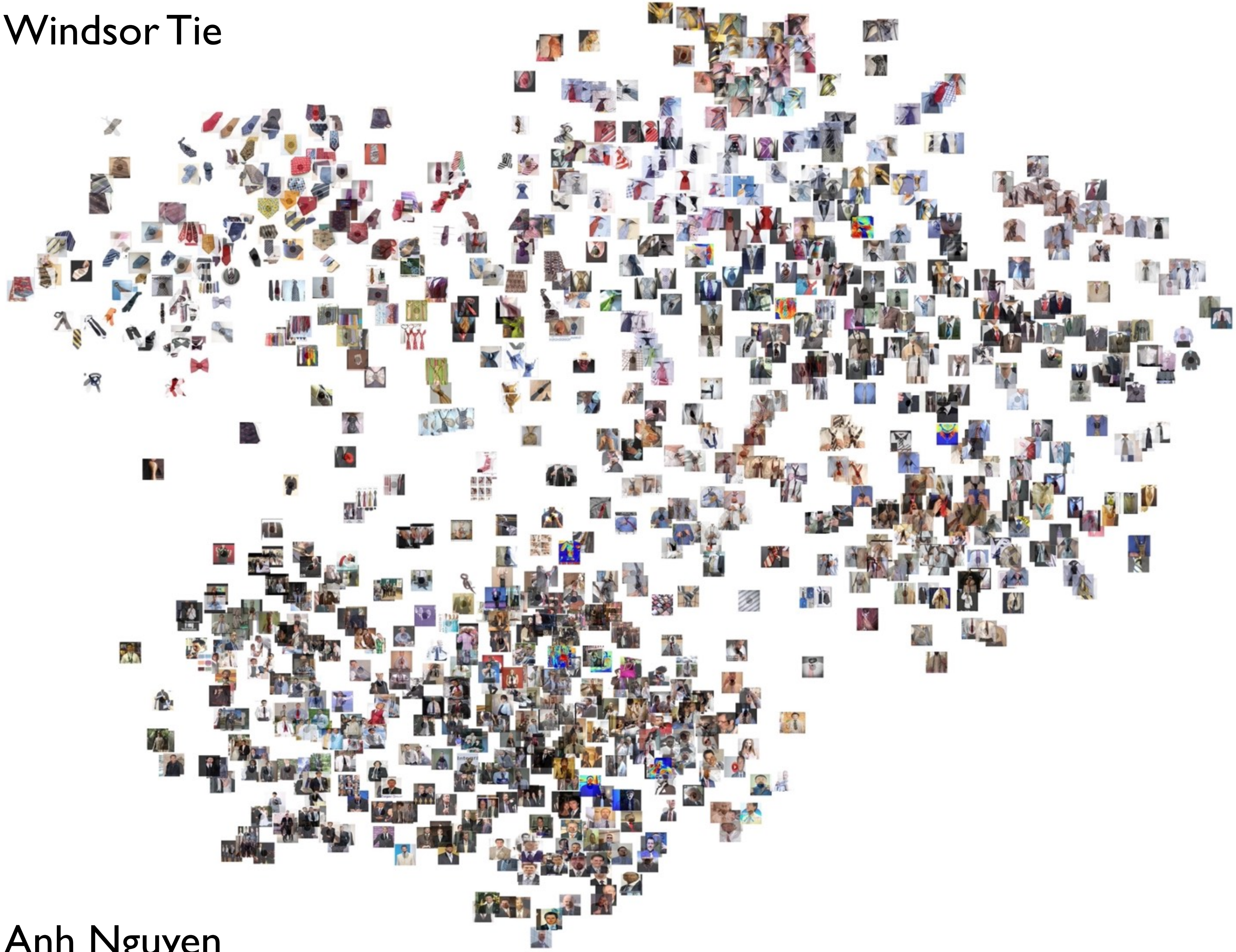
Leaderboard   Dataset

View [Top 1 Accuracy] by [Date] for [All models]



https://paperswithcode.com /sota /image -classification-on-imagenet

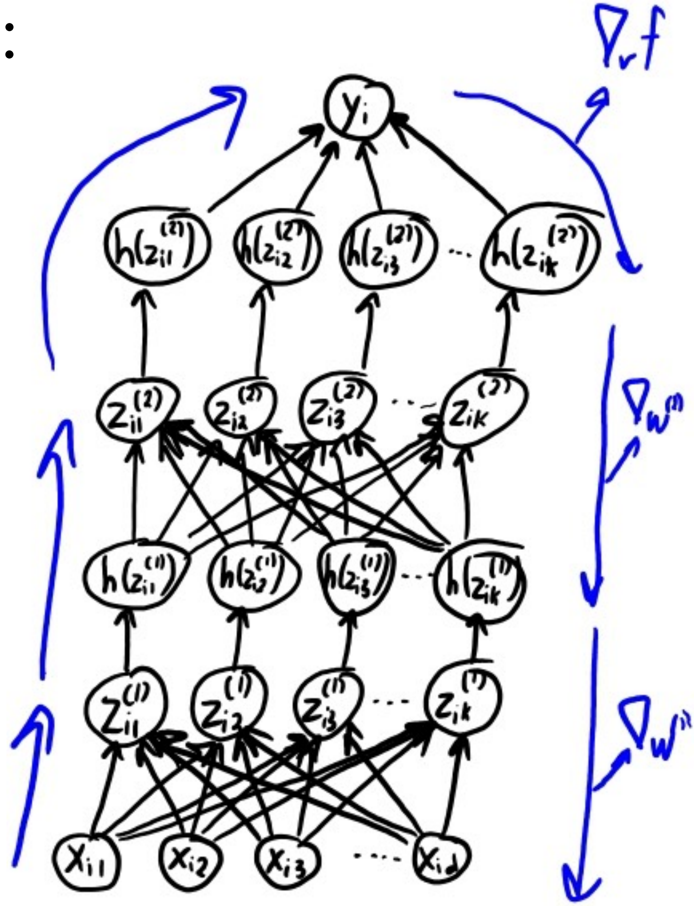
Windsor Tie



Anh Nguyen

Backpropagation

- Overview of how we compute neural network gradient:
  - Forward propagation :
    - Compute $z_{i\,(1)}$ from x_i .
    - Compute $z_{i\,(2)}$ from $z_{i\,(1)}$.
    - …
    - Compute Y_hat$_i$ from $z_{i\,(m)}$, and use this to compute error.
  - Backpropagation :
    - Compute gradient with respect to regression weights 'v'.
    - Compute gradient with respect to $z_{i\,(m)}$ weights $W_{(m)}$.
    - Compute gradient with respect to $z_{i\,(m-1)}$ weights $W_{(m-1)}$.
    - …
    - Compute gradient with respect to $z_{i\,(1)}$ weights $W_{(1)}$.



- "Backpropagation" is the chain rule plus some bookkeeping for speed.

bonus!

Backpropagation

- Instead of the next few bonus slides, I HIGHLY recommend watching this video from former UBC master's student Andrej Karpathy (of OpenAI, former director of AI and Autopilot Vision at Tesla)
  - https://www.youtube.com/watch?v =i94OvYb6noo

bonus!

$$\hat{y}_i \xleftarrow{v} h(z_i^{(3)}) \leftarrow z_i^{(3)} \xleftarrow{W^{(3)}} h(z_i^{(2)}) \leftarrow z_i^{(2)} \xleftarrow{W^{(2)}} h(z_i^{(1)}) \leftarrow z_i^{(1)} \xleftarrow{W^{(1)}} x_i$$

Backpropagation

- Let's illustrate backpropagation in a simple setting:
  - 1 training example, 3 hidden layers, 1 hidden "unit" in layer.

$$f(W^{(1)}, W^{(2)}, W^{(3)}, v) = \frac{1}{2}\underbrace{(\hat{y}_i - y_i)}_{r}^2 \quad \text{where} \quad \hat{y}_i = vh(W^{(3)}h(W^{(2)}h(W^{(1)}x_i)))$$

$$\frac{\partial f}{\partial v} = r\,h(W^{(3)}h(W^{(2)}h(W^{(1)}x_i))) = r\,h(z_i^{(3)})$$

$$\frac{\partial f}{\partial W^{(3)}} = r\,v\,h'(W^{(3)}h(W^{(2)}h(W^{(1)}x_i)))\,h(W^{(2)}h(W^{(1)}x_i)) = r\,v\,h'(z_i^{(3)})\,h(z_i^{(2)})$$

bonus!

Backpropagation

- Let's illustrate backpropagation in a simple setting:
  - 1 training example, 3 hidden layers, 1 hidden "unit" in layer.

$$f(W^{(1)}, W^{(2)}, W^{(3)}, v) = \frac{1}{2}\underbrace{(\hat{y}_i - y_i)}_{r}^2 \quad \text{where} \quad \hat{y}_i = vh(W^{(3)}h(W^{(2)}h(W^{(1)}x_i)))$$

$$\frac{\partial f}{\partial v} = r\,h(W^{(3)}h(W^{(2)}h(W^{(1)}x_i))) = r\,h(z_i^{(3)})$$

$$\frac{\partial f}{\partial W^{(3)}} = r\,v\,h'(W^{(3)}h(W^{(2)}h(W^{(1)}x_i)))h(W^{(2)}h(W^{(1)}x_i)) = \underbrace{r\,v\,h'(z_i^{(3)})}_{r^{(3)}}h(z_i^{(2)})$$

$$\frac{\partial f}{\partial W^{(2)}} = r\,v\,h'(W^{(3)}h(W^{(2)}h(W^{(1)}x_i)))W^{(3)}h'(W^{(2)}h(W^{(1)}x_i))h(W^{(1)}x_i) = \underbrace{r^{(3)}W^{(3)}h'(z_i^{(2)})}_{r^{(2)}}h(z_i^{(1)})$$

$$\frac{\partial f}{\partial W^{(1)}} = r\,v\,h'(W^{(3)}h(W^{(2)}h(W^{(1)}x_i)))W^{(3)}h'(W^{(2)}(W^{(1)}x_i))W^{(2)}h'(W^{(1)}x_i)x_i = r^{(2)}W^{(2)}h'(z_i^{(1)})x_i$$

bonus!

Backpropagation

- Let's illustrate backpropagation in a simple setting:
  - 1 training example, 3 hidden layers, 1 hidden "unit" in layer.

$$\frac{\partial f}{\partial v} = r\,h(z_i^{(3)})$$

$$\frac{\partial f}{\partial W^{(3)}} = r\,v h'(z_i^{(3)})h(z_i^{(2)})$$

$$\frac{\partial f}{\partial W^{(2)}} = r^{(3)} W^{(3)} h'(z_i^{(2)})h(z_i^{(1)})$$

$$\frac{\partial f}{\partial W^{(1)}} = r^{(2)} W^{(2)} h'(z_i^{(1)})x_i$$

$$\frac{\partial f}{\partial v_c} = r\,h(z_{ic}^{(3)})$$

$$\frac{\partial f}{\partial W^{(3)}_{c'c}} = \underbrace{r\,v_c h'(z_{ic'}^{(3)})}\,h(z_{ic}^{(2)})$$

$$\frac{\partial f}{\partial W^{(2)}_{c'c}} = \underbrace{\left[\sum_{c''=1}^{k} r_{c''}^{(3)} W_{cc''}^{(3)}\right] h'(z_{ic'}^{(2)})}\, h(z_{ic}^{(1)})$$

$$\frac{\partial f}{\partial W^{(1)}_{cj}} = \left[\sum_{c''=1}^{k} r_{c''}^{(2)} W_{c''c}^{(2)}\right] h'(z_{ic}^{(1)})x_j$$

  - Only the first 'r' changes if you use a different loss.
  - With multiple hidden units, you get extra sums.
    - Efficient if you store the sums rather than computing from scratch.

Backpropagation

- We’ve made backprop details bonus material
- Do you need to know how to do this?
  - Exact details are probably not vital (there are many implementations).
  - “Automatic differentiation ” is now standard and has same cost.
  - But understanding basic idea helps you know what can go wrong.
    - Or give hints about what to do when you run out of memory.
  - See discussion by a neural network expert (Andrej!)
    - https://karpathy.medium.com/yes-you-should-understand-backprop-e2f06eab496b

Andrej Karpathy

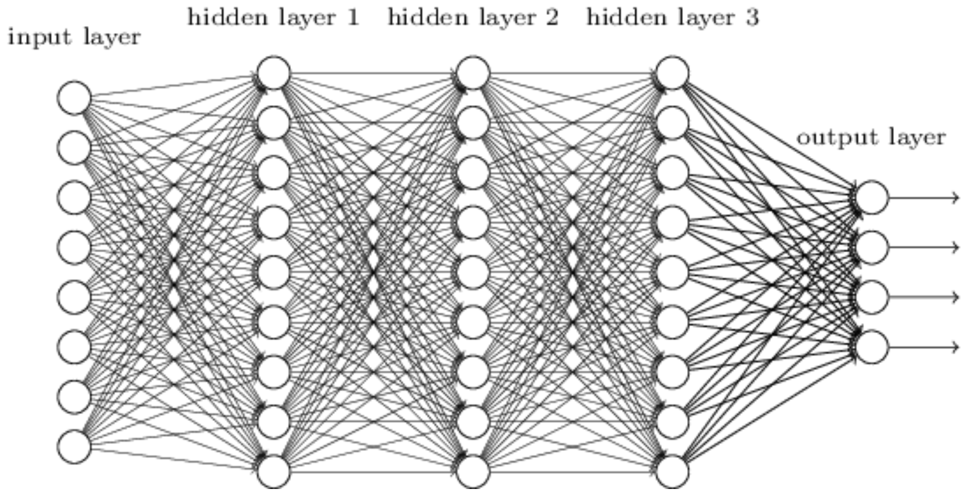
Dec 19, 2016 · 7 min read · Listen

Yes you should understand backprop

When we offered CS231n (Deep Learning class) at Stanford, we intentionally designed the programming assignments to include explicit calculations involved in backpropagation on the lowest level. The students had to

Backpropagation

- You should know cost of backpropagation:
  - Forward pass dominated by matrix multiplications by $W^{(1)}$, $W^{(2)}$, $W^{(3)}$, and 'v'.
    - If have 'm' layers and all z_i have 'k' elements, cost would be $O(dk + mk^2)$.
  - Backward pass has same cost as forward pass.



Next

- Finish discussion of how to train deep neural networks
  - algorithms, tips, and tricks, and miscellaneous key info