CPSC 340: Machine Learning and Data Mining

Neural Networks

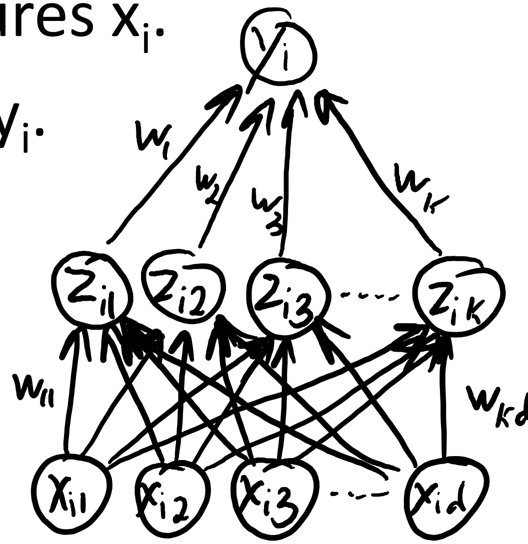
Fall 2016

Admin

- Assignment 4:
  - 3 late days to hand in Monday.
- Assignment 5:
  - Out, due next Friday.
- Assignment 6:
  - Out, due last day of class.
- Final:
  - December 12 (8:30am – HEBB 100)
  - Covers Assignments 1-6.
  - Final from last year and list of topics will be posted.
  - Closed-book, cheat sheet: 4-pages each double-sided.

Supervised Learning Roadmap

- Part 1: "Direct" Supervised Learning.
  - We learned parameters 'w' based on the original features x_i and target y_i.
- Part 3: Change of Basis.
  - We learned parameters 'w' based on a change of basis z_i and target y_i.
- Part 4: Latent-Factor Models.
  - We learned parameters 'W' for basis z_i based on only on features x_i.
  - You can then learn 'w' based on change of basis z_i and target y_i.
- Part 5: Neural Networks.
  - Jointly learn 'W' and 'w' based on x_i and y_i.
  - **Learn basis z_i that is good for supervised learning.**



Neural Networks: Introducing Non-Linearity

- Natural choice of neural network regression objective would be:

$$f(w,W) = \frac{1}{2}\sum_{i=1}^{n}(w^T \underbrace{z_i}_{Wx_i} - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(w^T(Wx_i) - y_i)^2$$

- But we saw last time this gives a linear model.

$w^T W$ is a vector so equivalent to just having $w^T x_i$ for some 'w'.

- Typical fix is to introduce non-linearity 'h':

$$f(w,W) = \frac{1}{2}\sum_{i=1}^{n}(w^T \underbrace{h(Wx_i)}_{z_i} - y_i)^2$$ where 'h' has 'd' inputs and 'k' outputs.

- Most common choice of 'h' is sigmoid applied to elements of Wx_i.

$$z_{ic} = \frac{1}{1 + \exp(-W_c x_i)}$$

Notation for Neural Networks

We have our usual supervised learning notation:

$$X = \begin{bmatrix} — & x_1^T & — \\ — & x_2^T & — \\ & \vdots & \\ — & x_n^T & — \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$n \times d$ $\qquad$ $n \times 1$

We have our latent features:

$$Z = \begin{bmatrix} — & z_1^T & — \\ — & z_2^T & — \\ & \vdots & \\ — & z_n^T & — \end{bmatrix}$$

$n \times k$

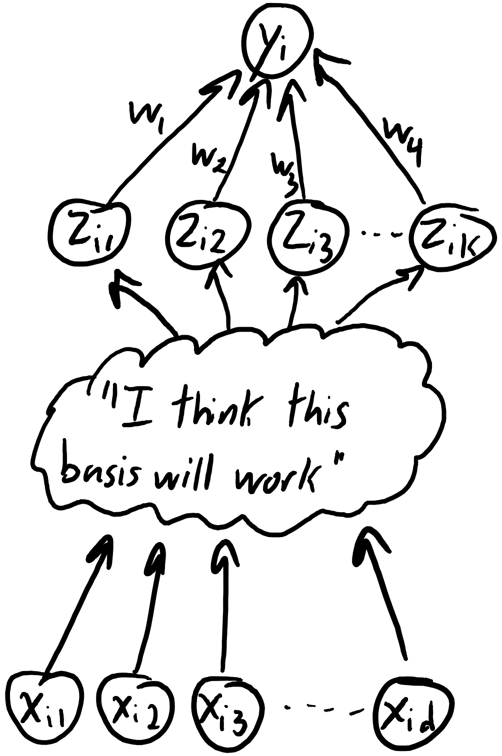
We have <u>two</u> sets of parameters:

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix} \quad W = \begin{bmatrix} — & W_1 & — \\ — & W_2 & — \\ & \vdots & \\ — & W_k & — \end{bmatrix}$$

$k \times 1$ $\qquad$ $k \times d$

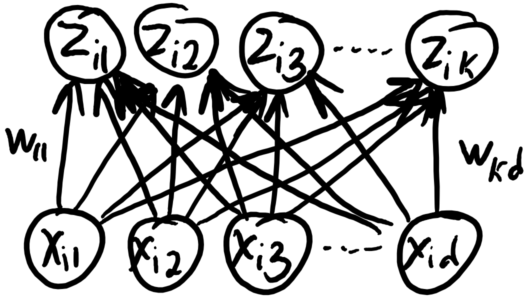
Supervised Learning Roadmap

Hand-engineered features:

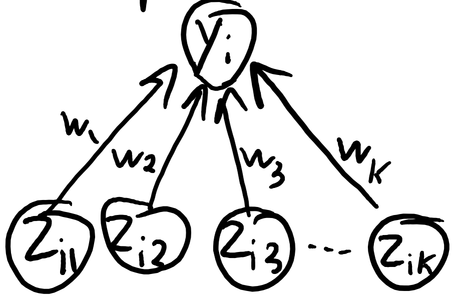


Requires domain knowledge and can be time-consuming

Learn a latent-factor model:

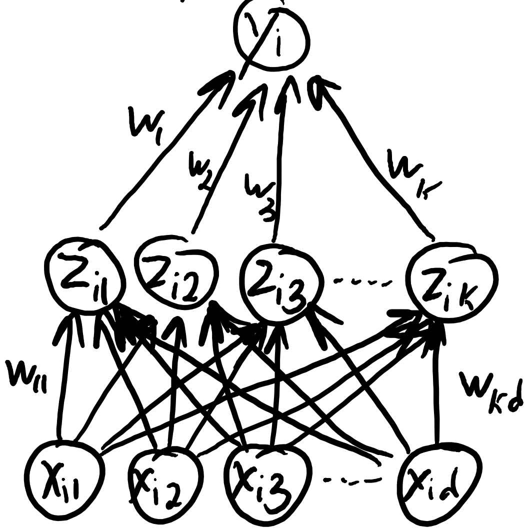


Use latent features in supervized model:



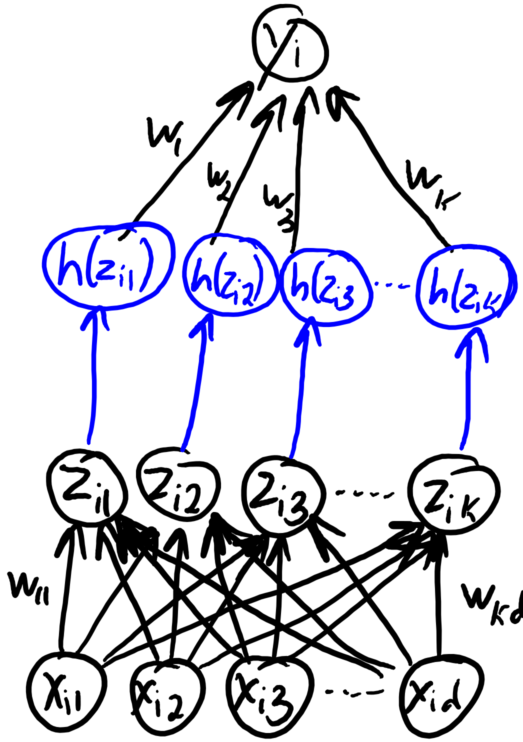
Good representation of x_i might be bad for predicting y_i

Learn 'w' and 'W' together:



But still gives a <u>linear</u> model.

Neural network:



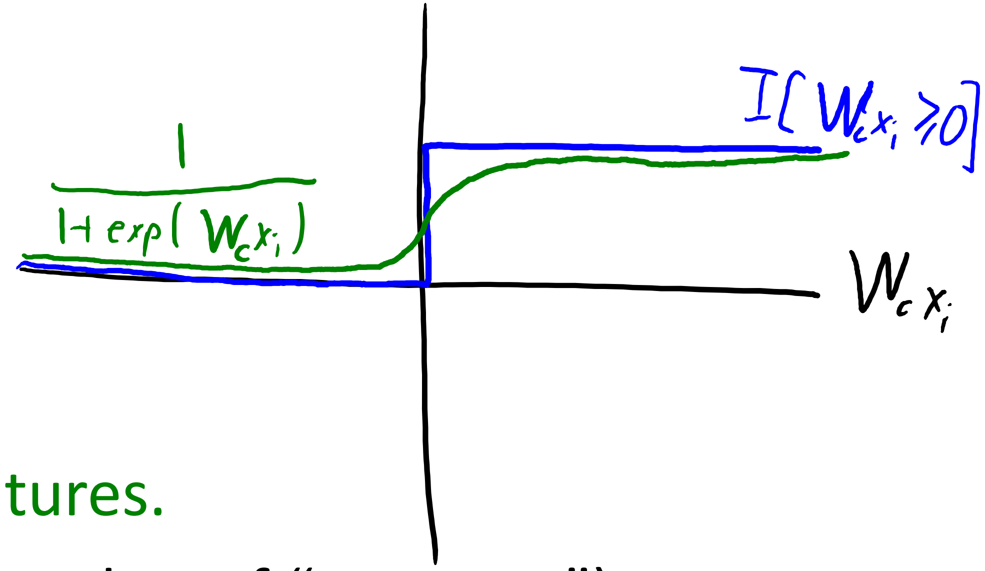
Extra <u>non-linear</u> transformation 'h'.

Why Sigmoid?

- Consider setting ‘h’ to define binary features z_i using:

$$z_i = I[W_c x_i \geq 0]$$

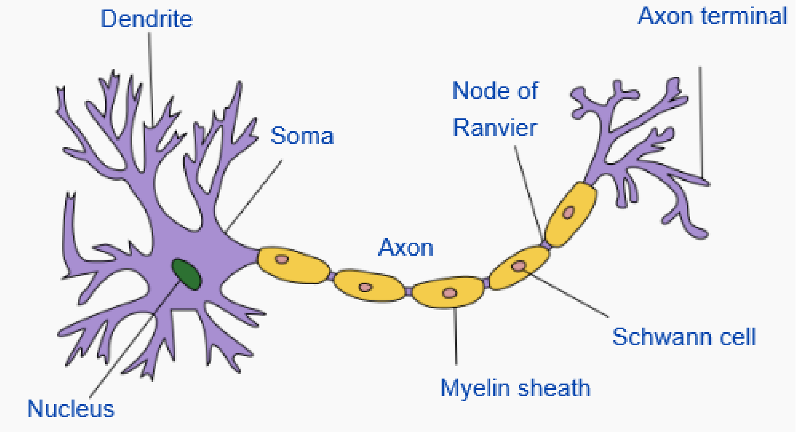
$$= \begin{cases} 1 & \text{if } W_c x_i \geq 0 \\ 0 & \text{if } W_c x_i < 0 \end{cases}$$



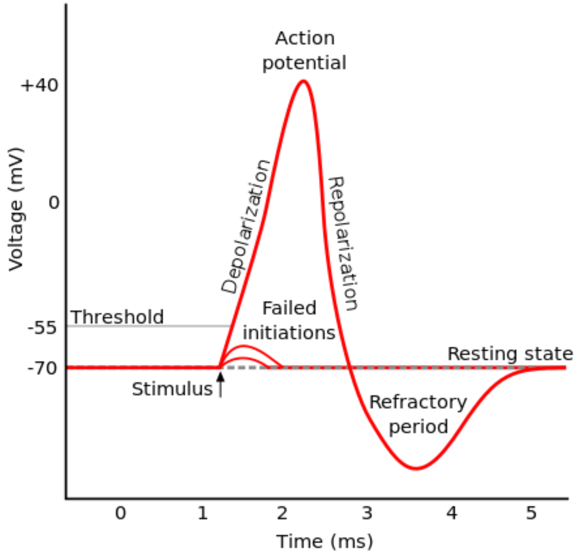
  - Vector $z_i = h(Wx_i)$ can be viewed as binary features.
  - z_i can take 2^k possible values (combinatorial number of “concepts”).
- But non-differentiable and discontinuous so hard to optimize.
- Sigmoid is a smooth approximation to these binary features.

Why “Neural Network”?

- Cartoon of “typical” neuron:

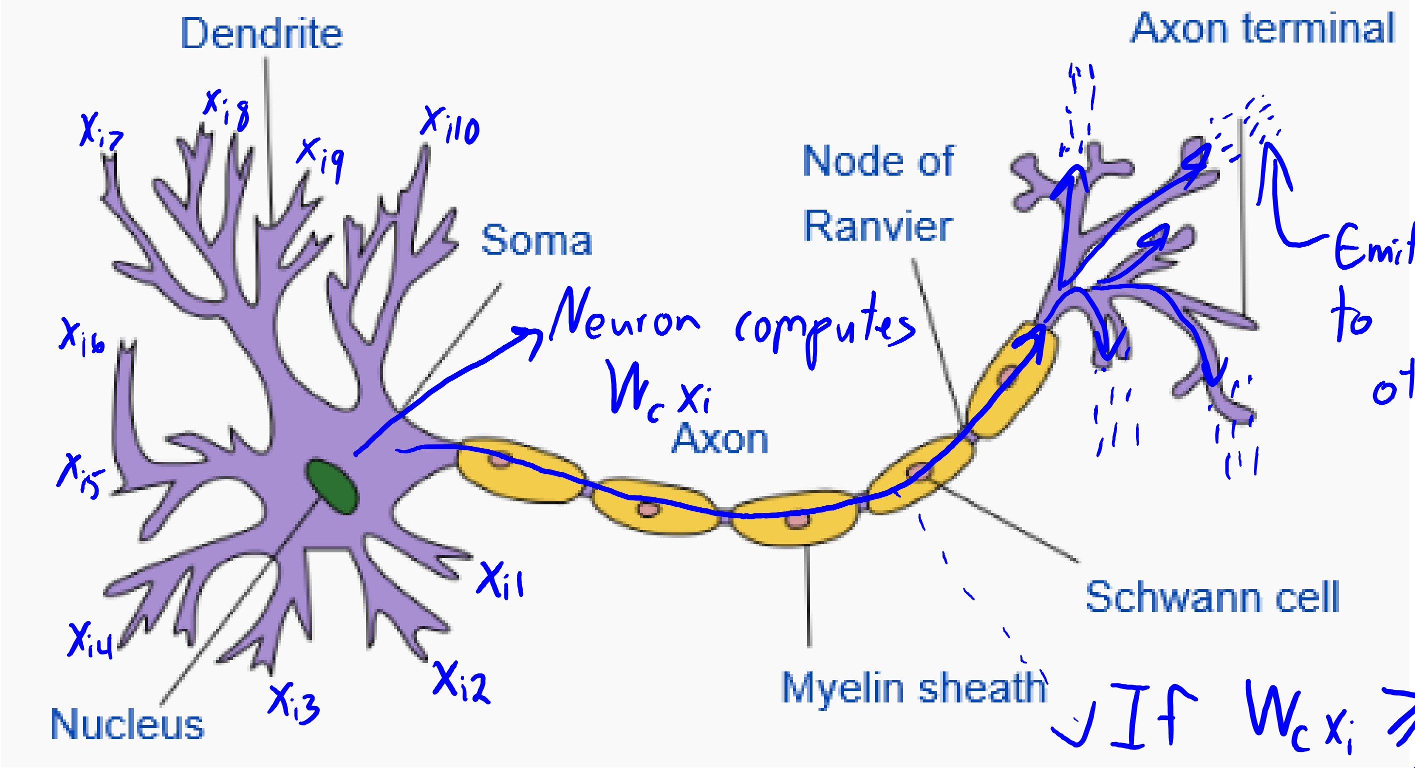


- Neuron has many “dendrites”, which take an input signal.
- Neuron has a single “axon”, which sends an output signal.
- With the right input to dendrites:
  – “Action potential” along axon (like a binary signal):



https://en.wikipedia.org/wiki/Neuron
https://en.wikipedia.org/wiki/Action_potential

Why "Neural Network"?



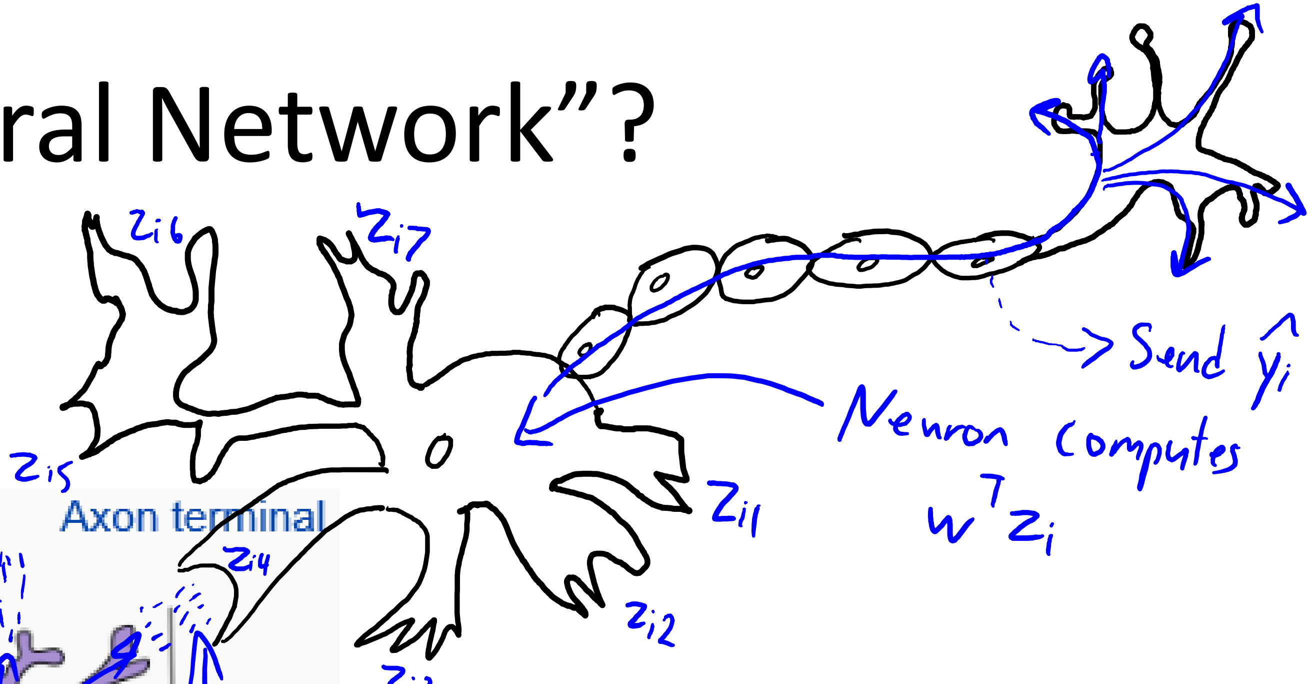
Emits "neurotransmitter" to send signal to other neurons.

If $W_c x_i \geq 0$ neuron sends signal along axon.

We approximate binary signal with $\dfrac{1}{1+\exp(-W_c x_i)}$

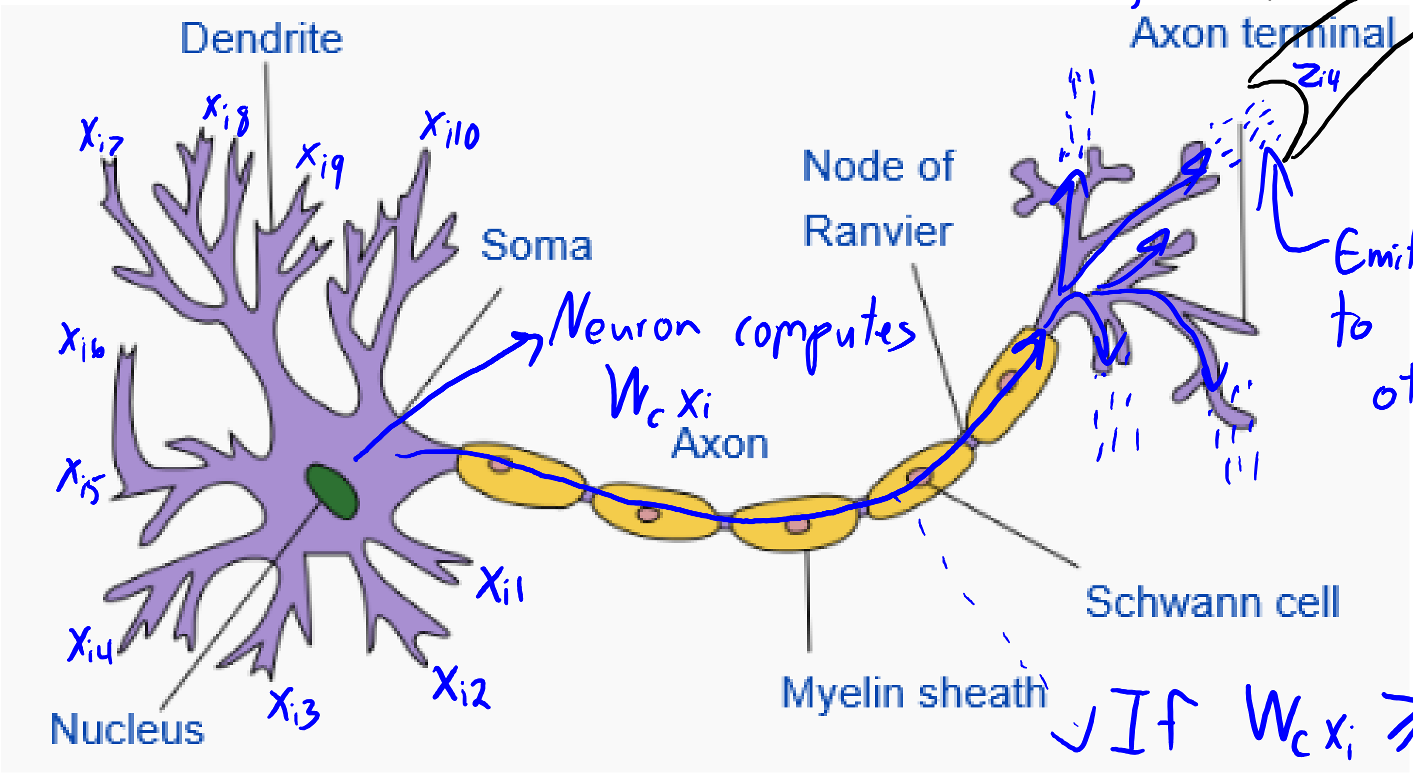
https://en.wikipedia.org/wiki/Neuron

Why "Neural Network"?



Send $\hat{y}_i$

Neuron computes $w^T z_i$



Neuron computes $W_c x_i$

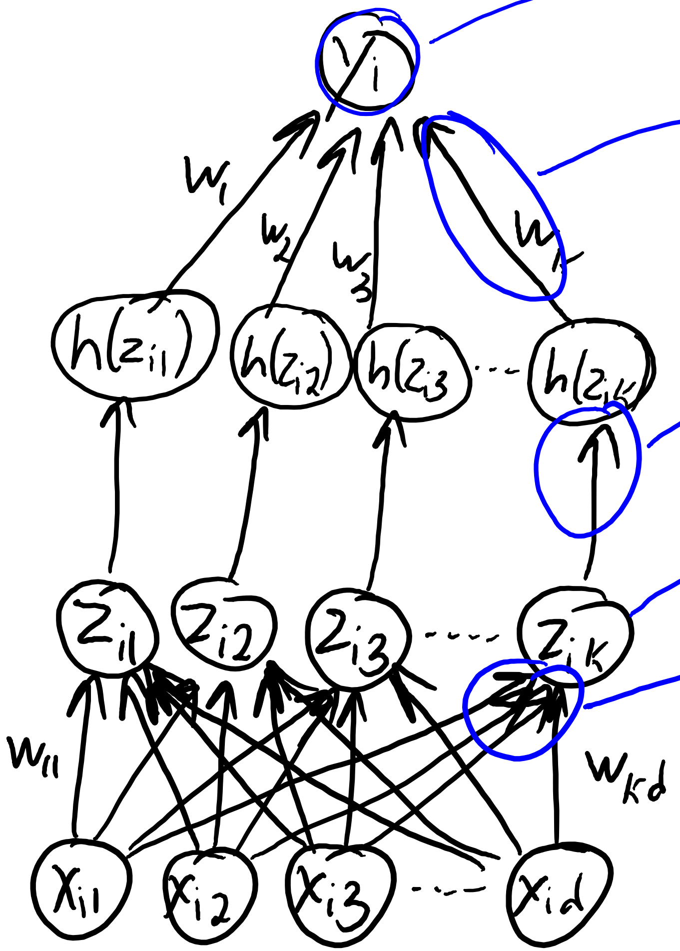
Emits "neurotransmitter" to send signal to other neurons.

If $W_c x_i \geq 0$ neuron sends signal along axon.

We approximate binary signal with $\frac{1}{1+\exp(-W_c x_i)}$

https://en.wikipedia.org/wiki/Neuron

Why "Neural Network"?



Predictions based on aggregation $w^T h(Wx_i)$ at y_i "neuron"

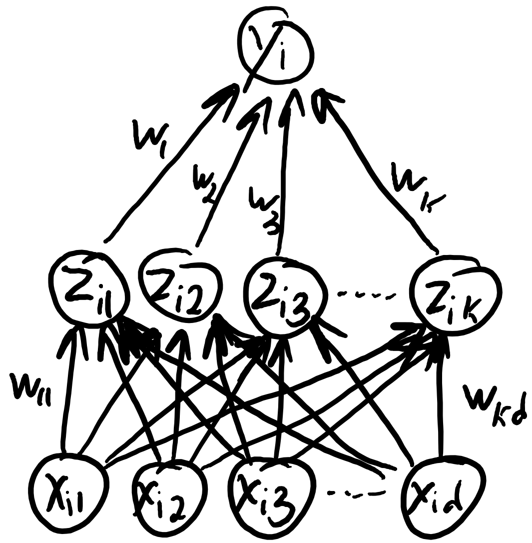
Synapse between z_{ik} and y_i "neuron"

binary signal $h(W_c x_i)$ sent along "axon"

Neuron aggregates signals: $W_c x_i$

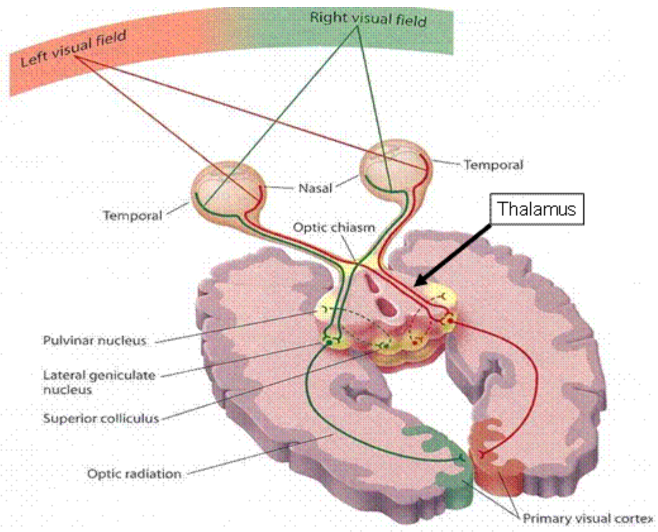
"dendrites" for z_{ik} "neuron" are reciving x_{ij} values

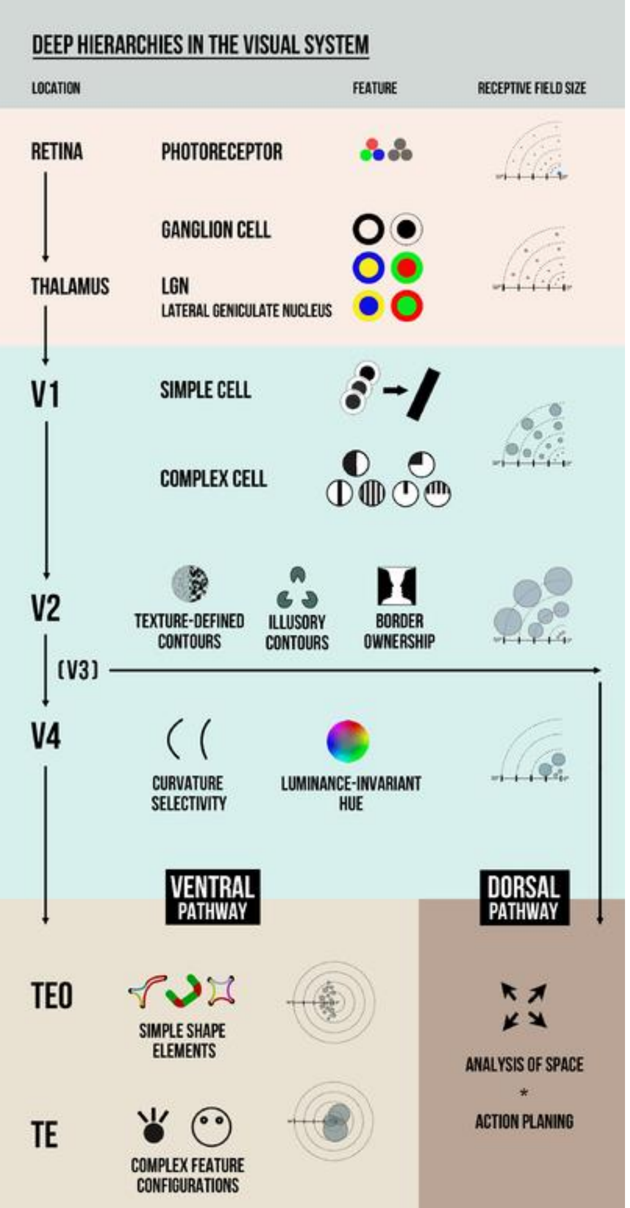
“Artificial” Neural Nets vs. “Real” Networks Nets

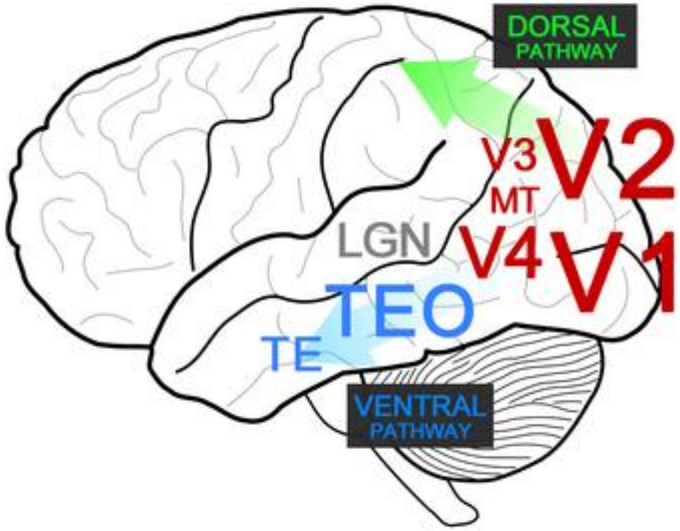


- Artificial neural network:
  - x_i is measurement of the world.
  - z_i is internal representation of world.
  - y_i is output of neuron for classification/regression.
- Real neural networks are more complicated:
  - Timing of action potentials seems to be important.
    - “Rate coding”: frequency of action potentials simulates continuous output.
  - Neural networks don’t reflect sparsity of action potentials.
  - How much computation is done inside neuron?
  - Brain is highly organized (e.g., substructures and cortical columns).
  - Connection structure changes.
  - Different types of neurotransmitters.

Deep Hierarchies in the Brain

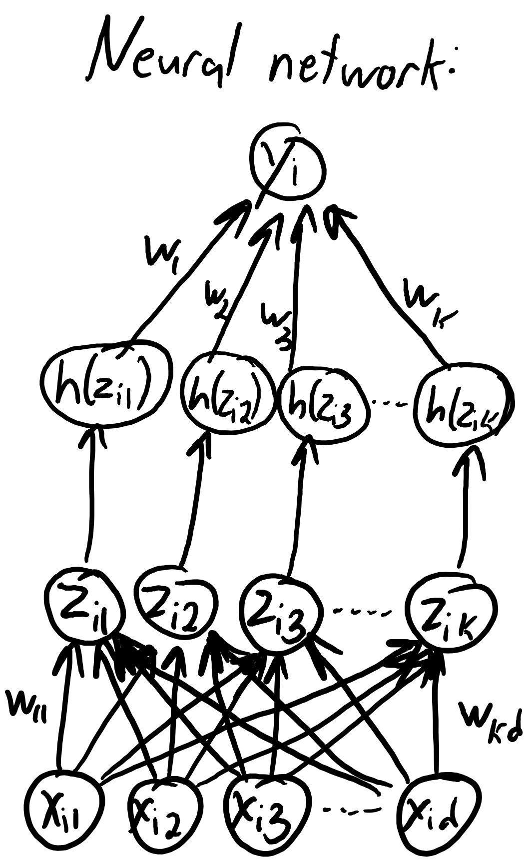


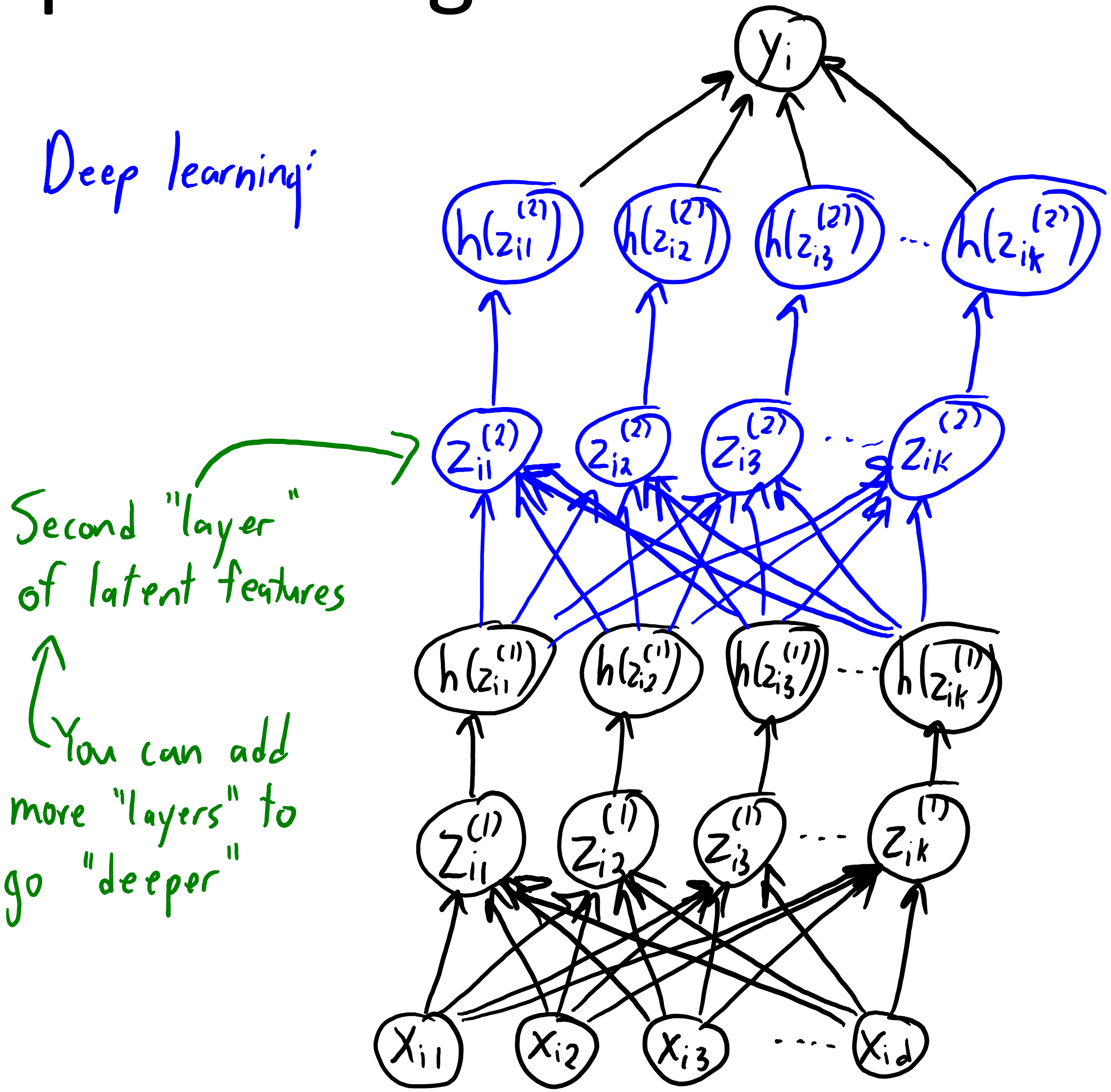




http://www.strokenetwork.org/newsletter/articles/vision.htm
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing

Deep Learning





Deep Learning

Linear model:

$$y_i = w^T x_i$$

Neural network with 1 hidden layer:

$$y_i = w^T h(\underbrace{W x_i}_{z_i})$$

Neural network with 2 hidden layers:

$$y_i = w^T h(\underbrace{W^{(2)} h(\underbrace{W^{(1)} x_i}_{z_i^{(1)}})}_{z_i^{(2)}})$$

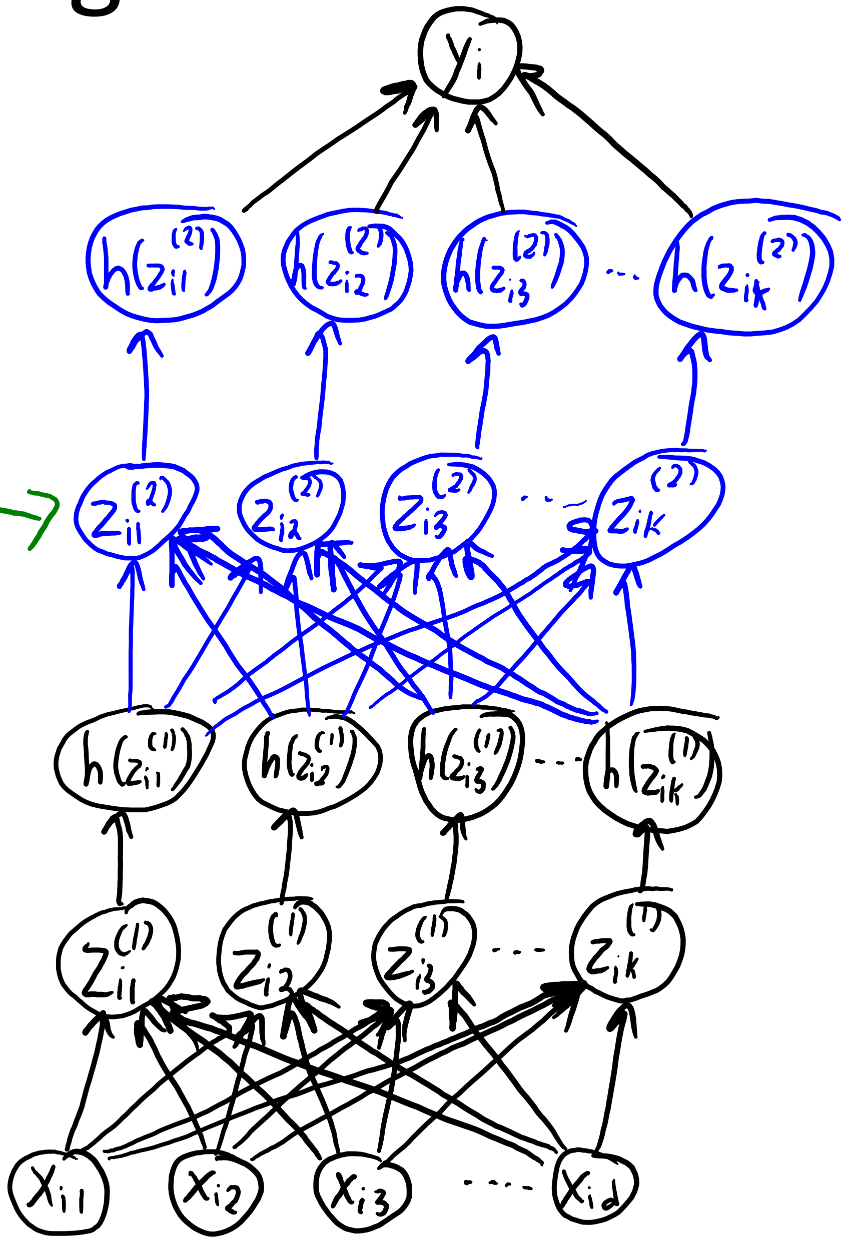
Neural network with 3 hidden layers

$$y_i = w^T h(\underbrace{W^{(3)} h(\underbrace{W^{(2)} h(\underbrace{W^{(1)} x_i}_{z_i^{(1)}})}_{z_i^{(2)}})}_{z_i^{(3)}})$$

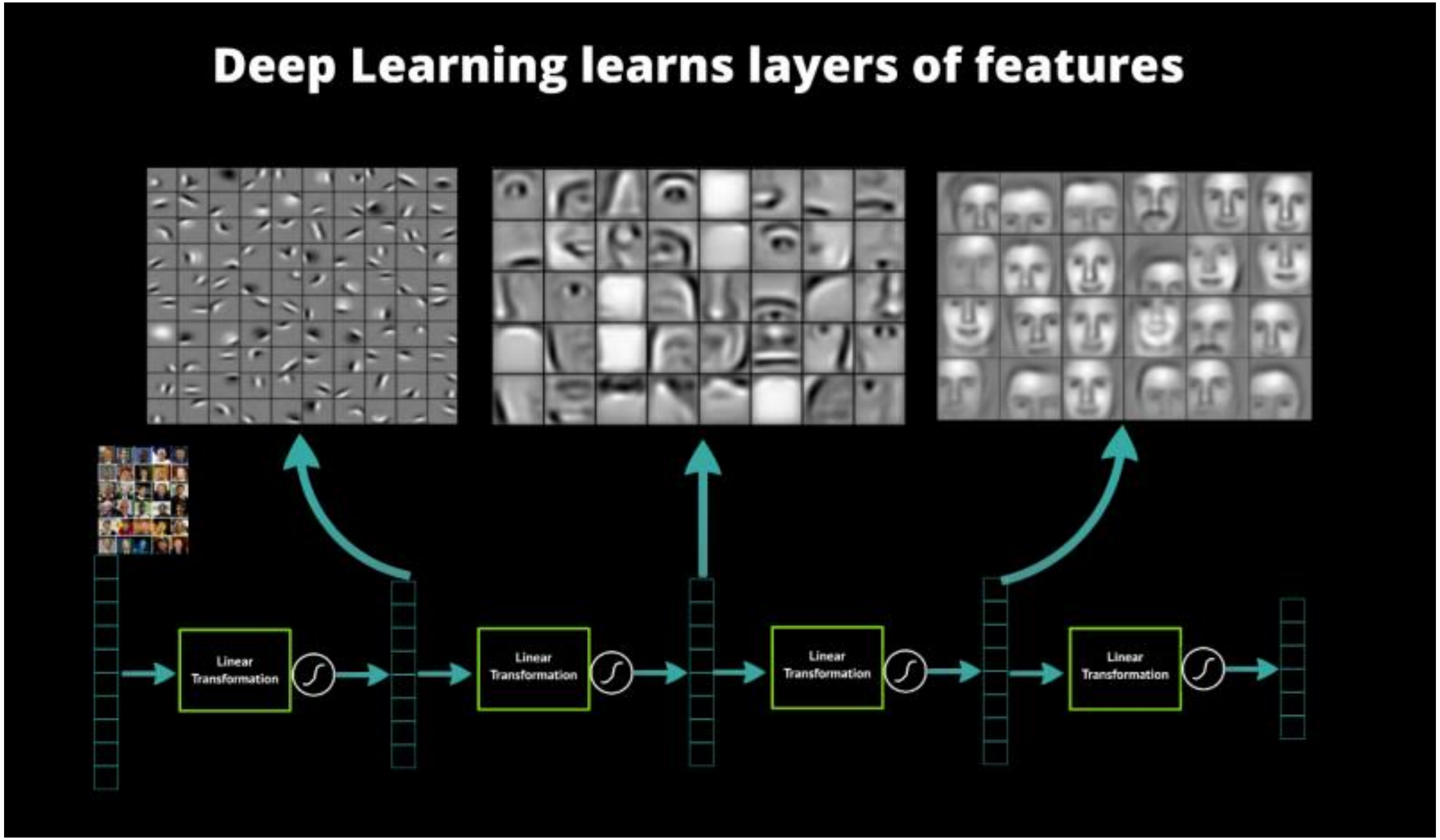
Deep learning:

Second "layer" of latent features

You can add more "layers" to go "deeper"



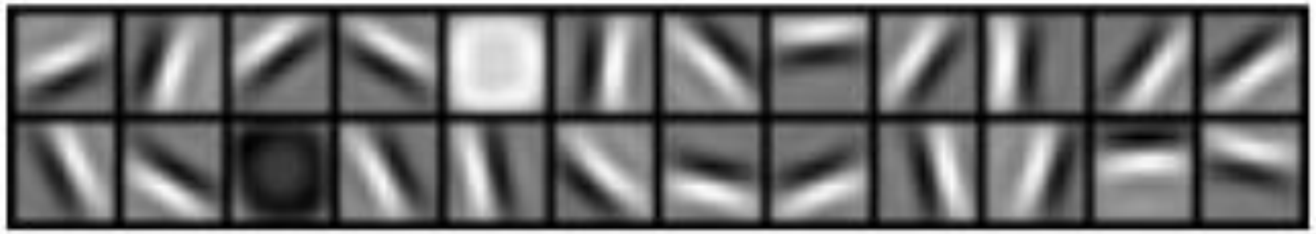
Cool Picture Motivation for Deep Learning



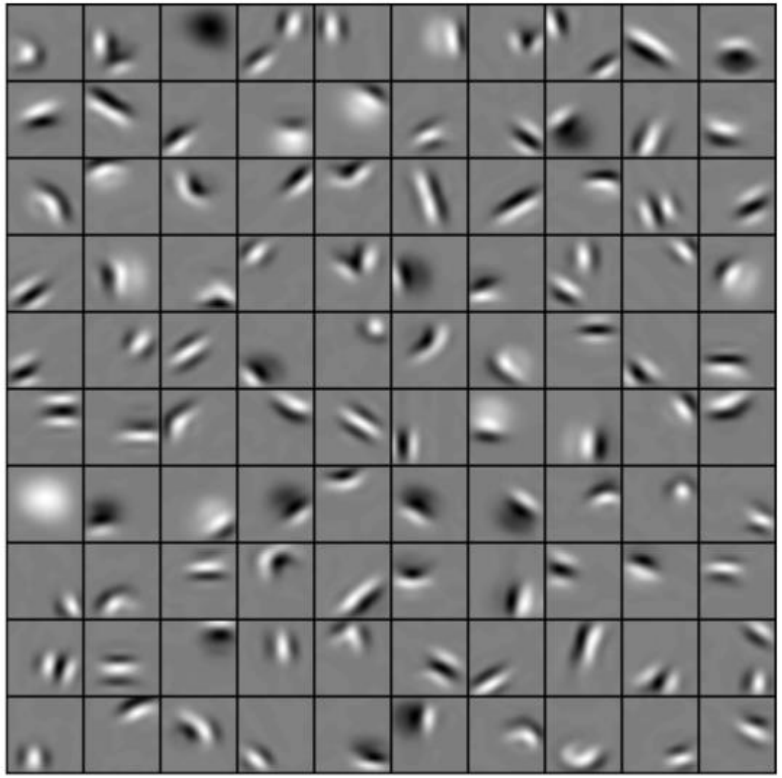
http://www.datarobot.com/blog/a-primer-on-deep-learning/

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



- Attempt to visualize second layer:
  - Corners, angles, surface boundaries?

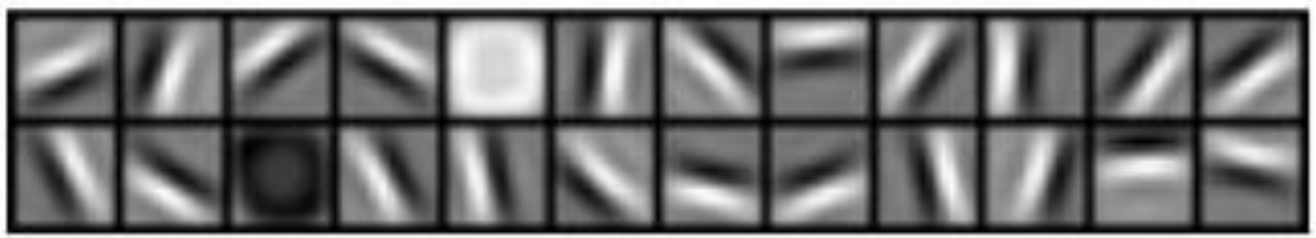


- Models require many tricks to work.
  - We'll discuss these next time.

http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



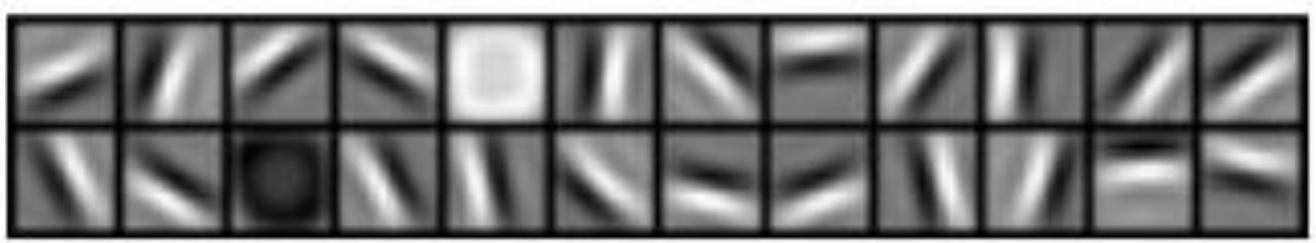
- Visualization of second and third layers trained on specific objects:

faces

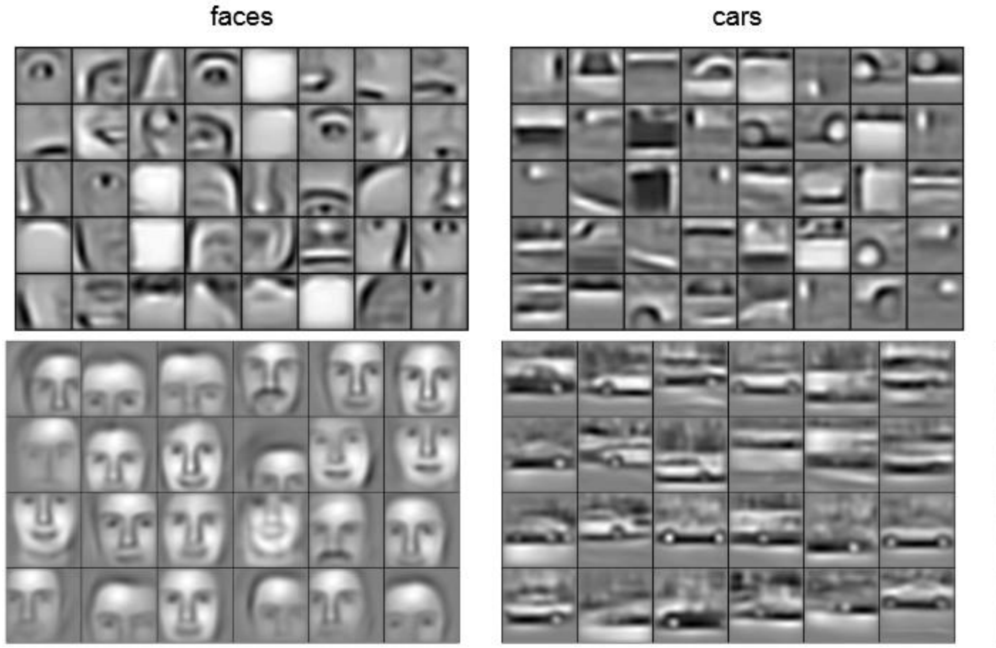
http://www.cs.toronto.edu/~rgrosse

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



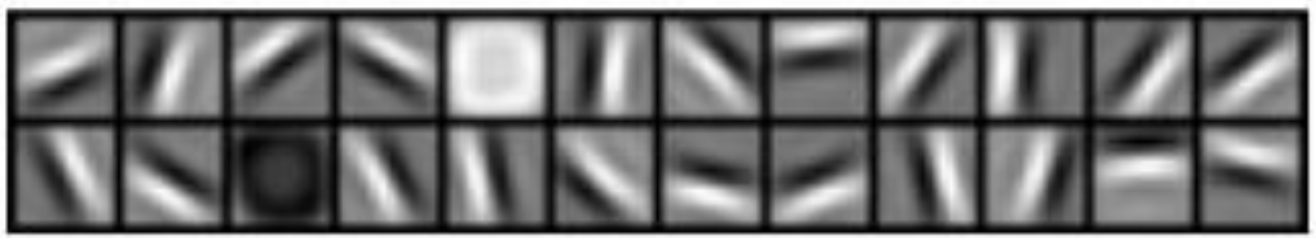
- Visualization of second and third layers trained on specific objects:



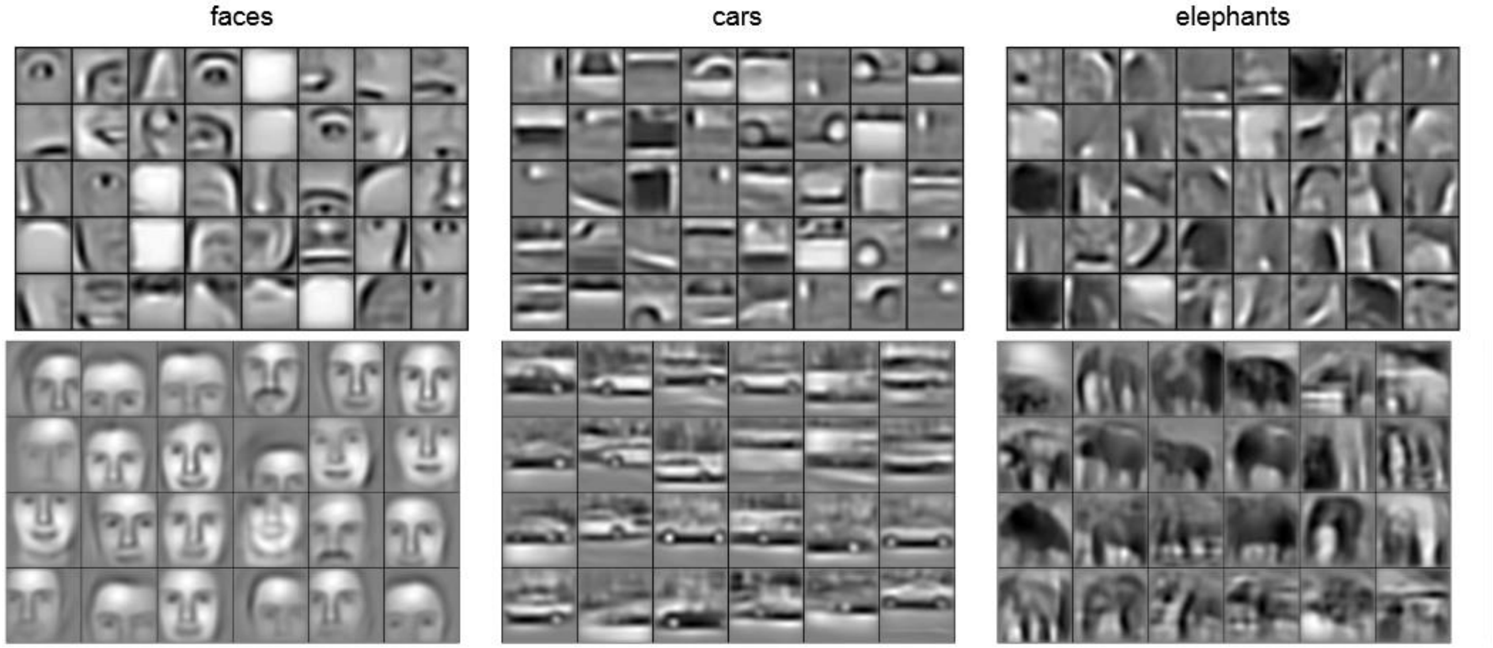
http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



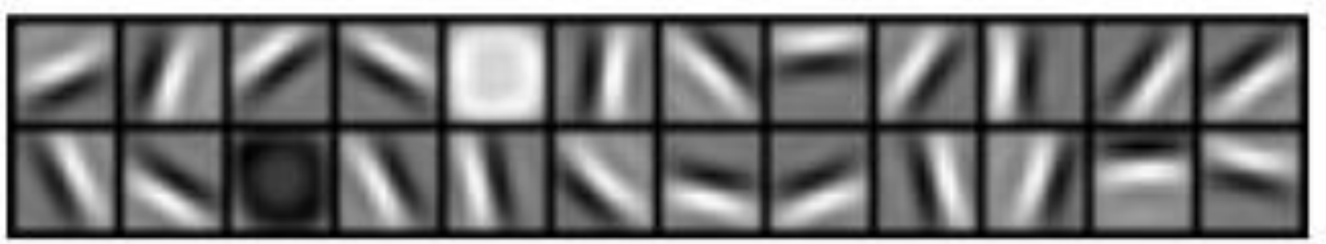
- Visualization of second and third layers trained on specific objects:



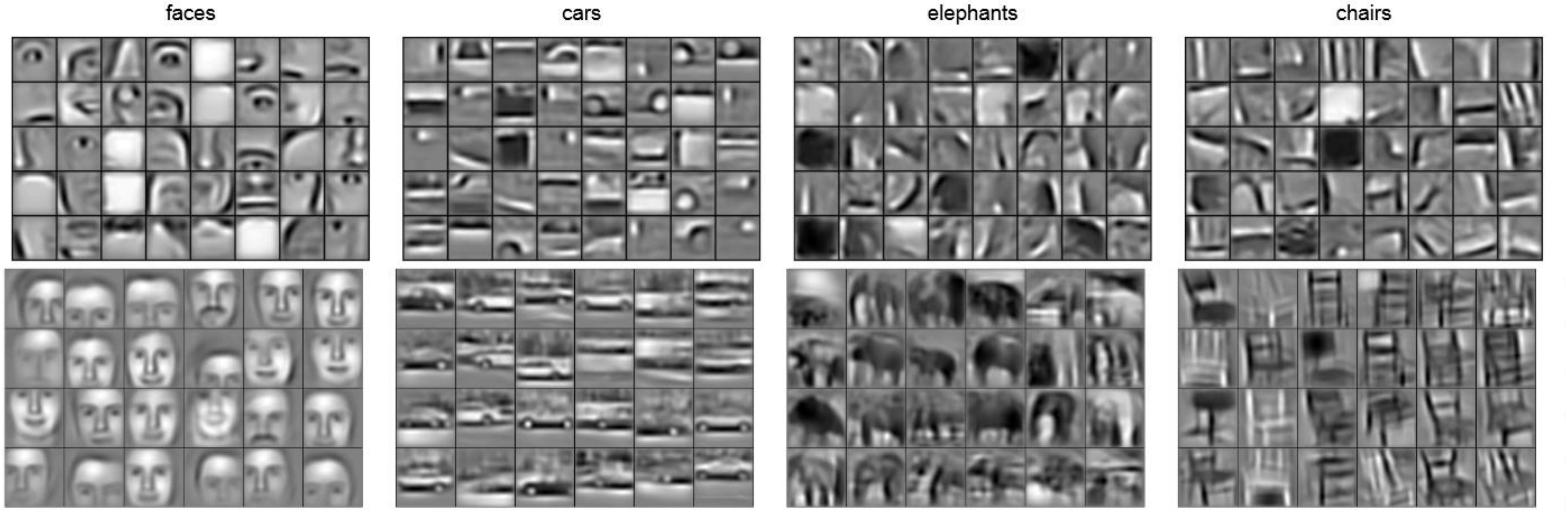
http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:



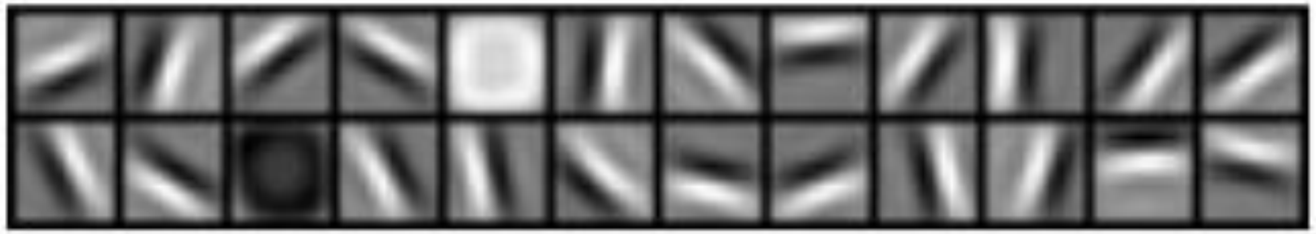
- Visualization of second and third layers trained on specific objects:



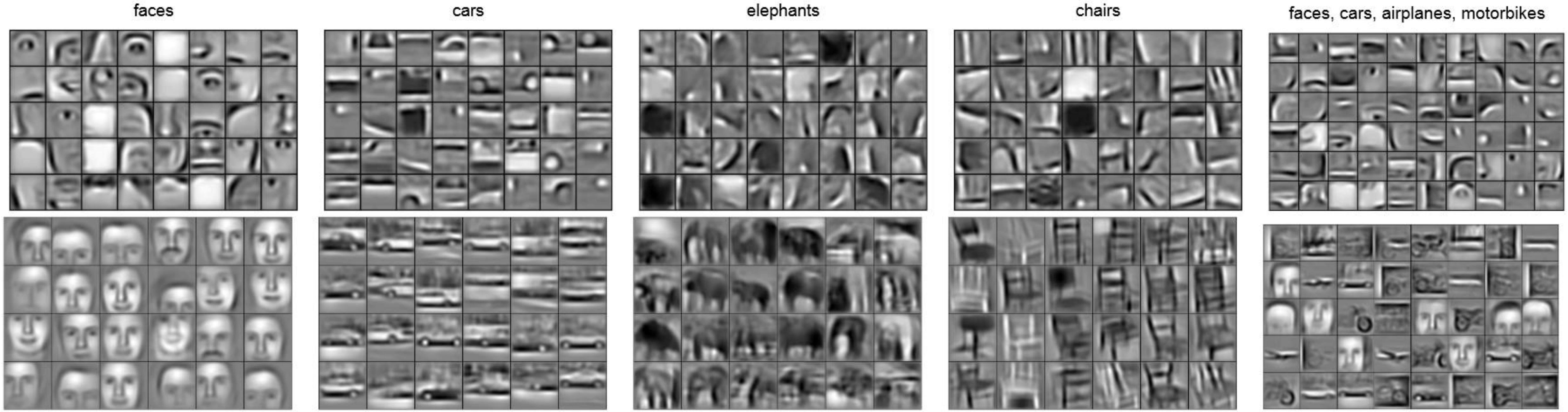
http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

Cool Picture Motivation for Deep Learning

- First layer of z_i trained on 10 by 10 image patches:

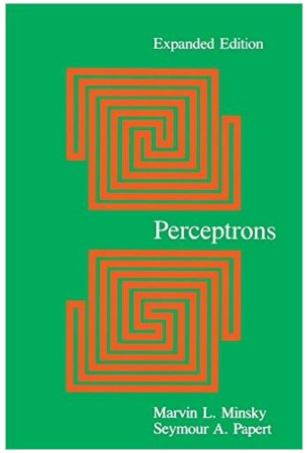


- Visualization of second and third layers trained on specific objects:



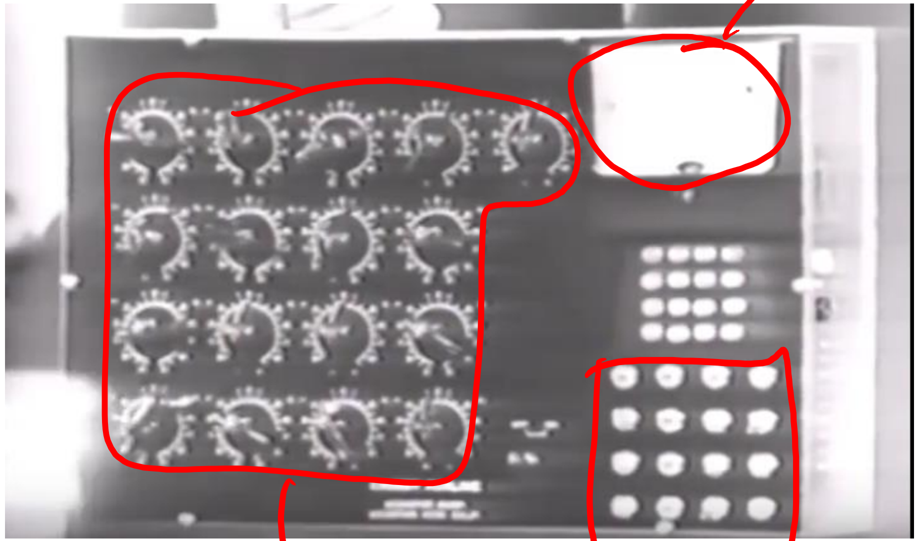
http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

ML and Deep Learning History

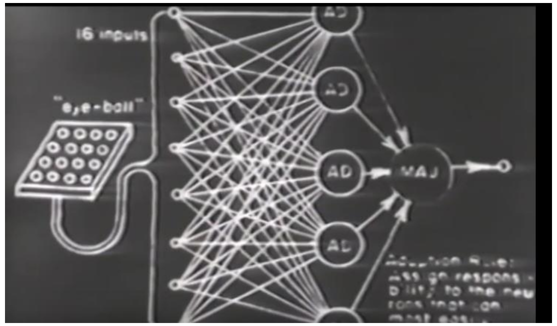


- 1950 and 1960s: Initial excitement.
  - Perceptron: linear classifier and stochastic gradient (roughly).
  - "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." New York Times (1958).
    - https://www.youtube.com/watch?v=IEFRtz68m-8

$w^T x_i$



w x_i

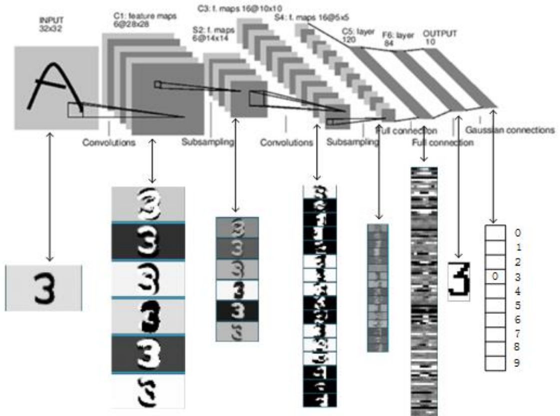


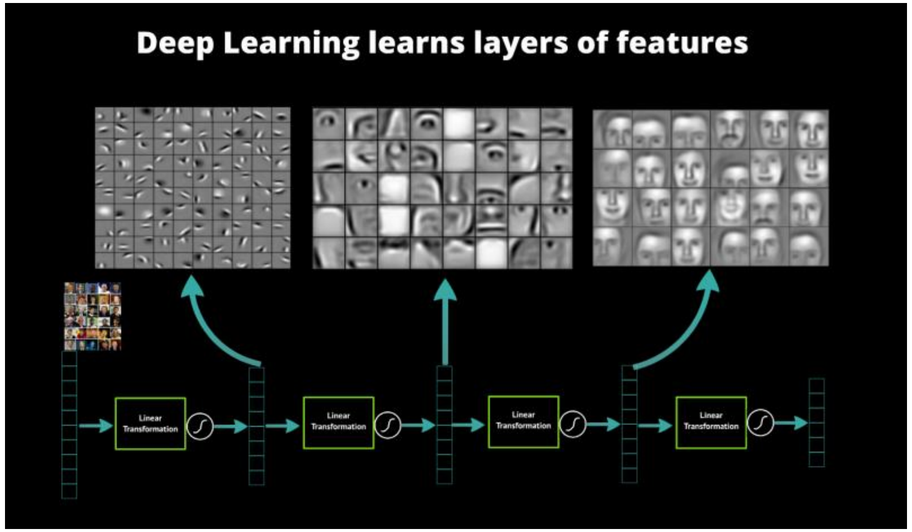
- Then drop in popularity:
  - Quickly realized limitations of linear models.

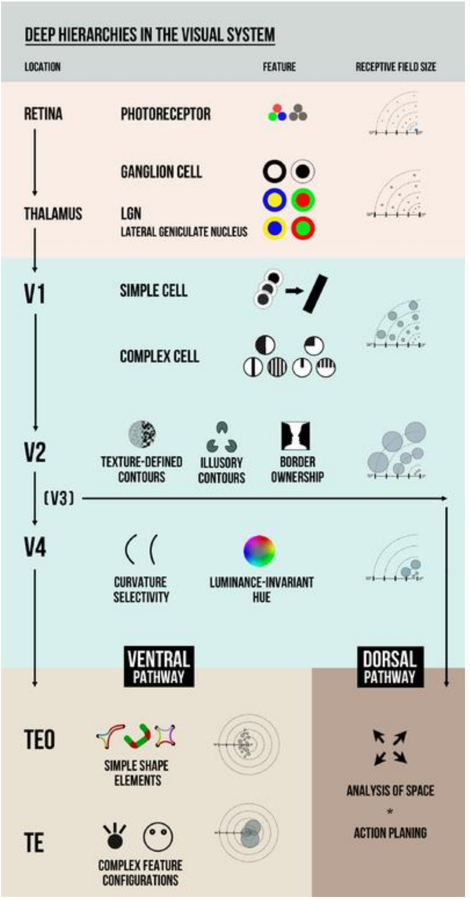
https://mitpress.mit.edu/books/perceptrons/

ML and Deep Learning History

- 1970 and 1980s: Connectionism (brain-inspired ML)
  - 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.







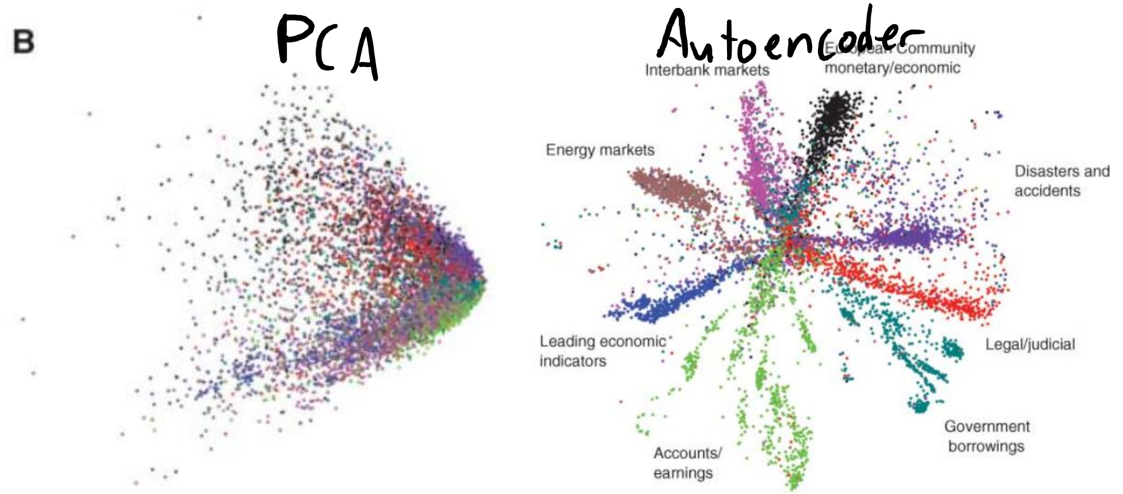
https://en.wikibooks.org/wiki/Sensory_Systems/Visual_Signal_Processing
http://www.datarobot.com/blog/a-primer-on-deep-learning/
http://blog.csdn.net/strint/article/details/44163869

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.
  - ML moved closer to other fields (CPSC 540):
    - Numerical optimization.
    - Probabilistic graphical models.
    - Bayesian methods.

ML and Deep Learning History

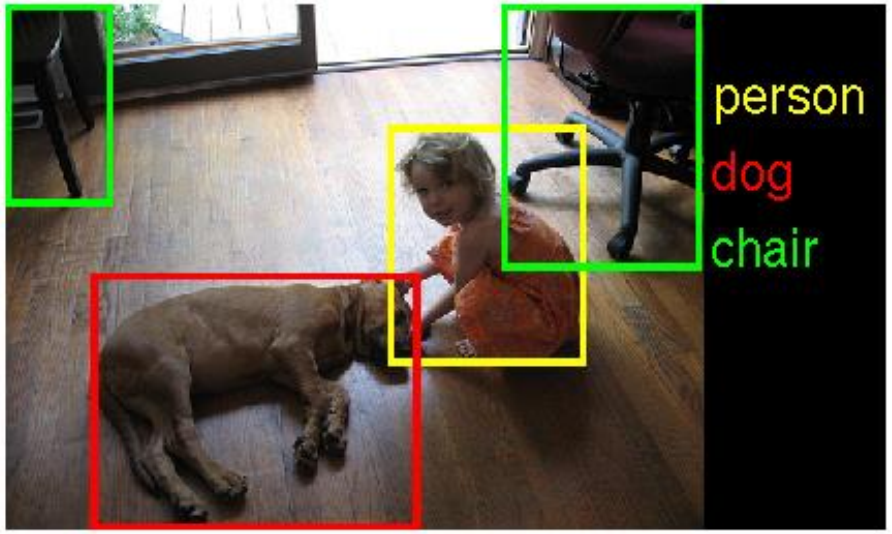
- Late 2000s: push to revive connectionism as "deep learning".
  - Canadian Institute For Advanced Research (CIFAR) NCAP program:
    - "Neural Computation and Adaptive Perception".
    - Led by Geoff Hinton, Yann LeCun, and Yoshua Bengio ("Canadian mafia").
  - Unsupervised successes: "deep belief networks" and "autoencoders".
    - Could be used to initialize deep neural networks.
    - https://www.youtube.com/watch?v=KuPai0ogiHk



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

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 instantl
  - This is now finding its way into products.



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

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?"

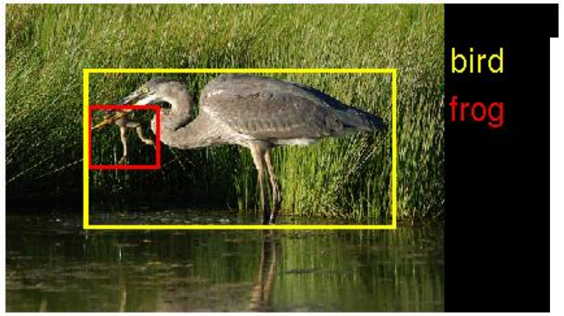
    Forbes (2013).
  - "Computer eyesight gets a lot more accurate"

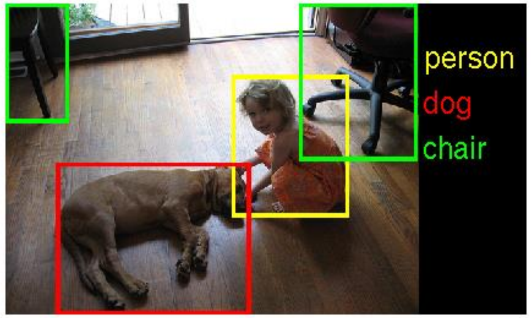
    New York Times (2014).

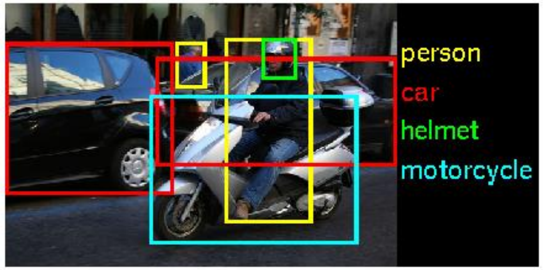
- 2015: huge improvement in language understanding.

ImageNet Challenge

- Millions of labeled images, 1000 object classes.







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

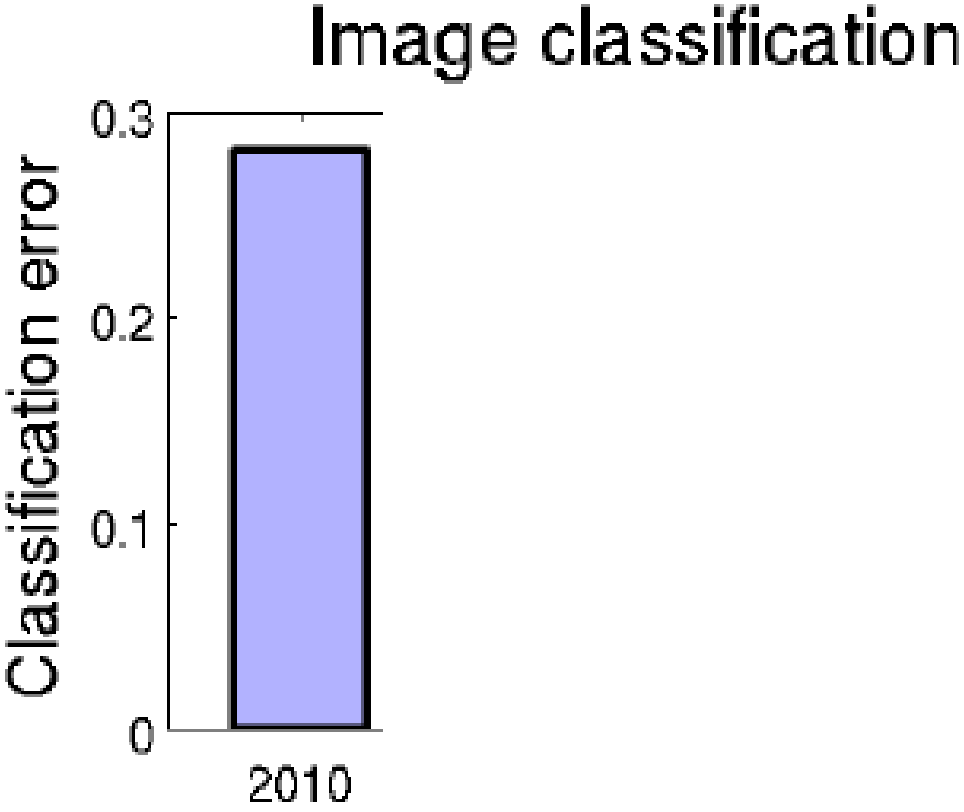
http://www.image-net.org/challenges/LSVRC/2014/

ImageNet Challenge

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

(a) Siberian husky

(b) Eskimo dog

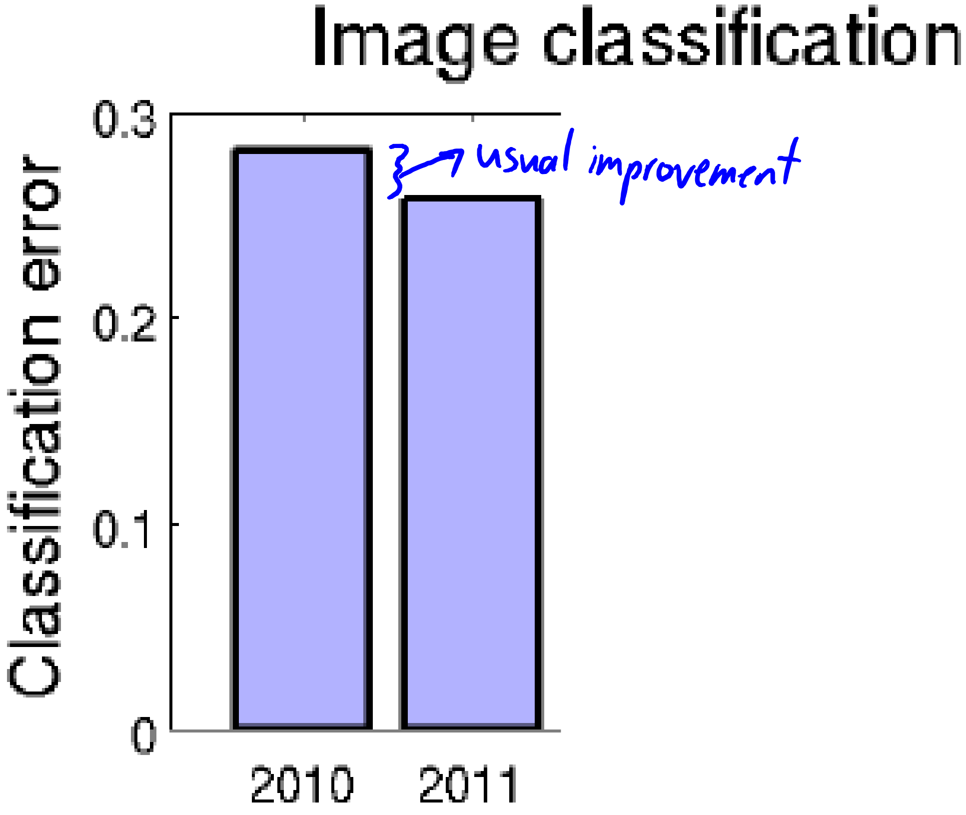


https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

ImageNet Challenge

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

(a) Siberian husky (b) Eskimo dog

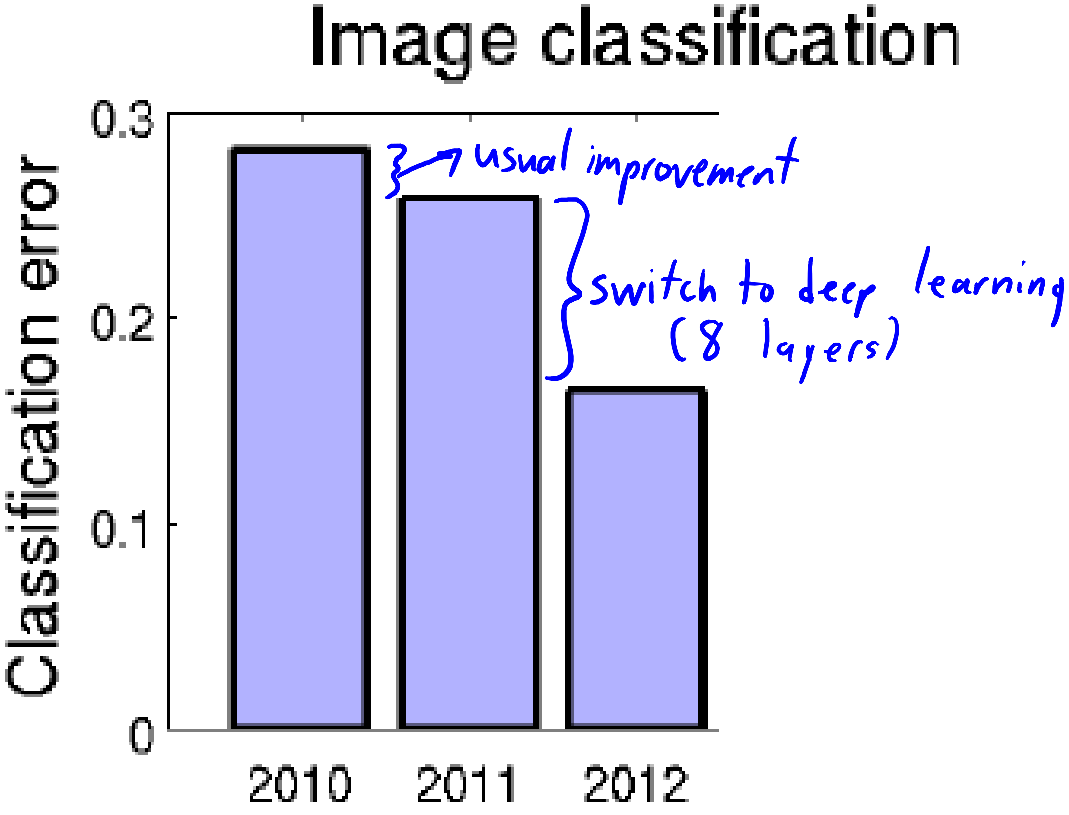


https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

ImageNet Challenge

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

(a) Siberian husky (b) Eskimo dog

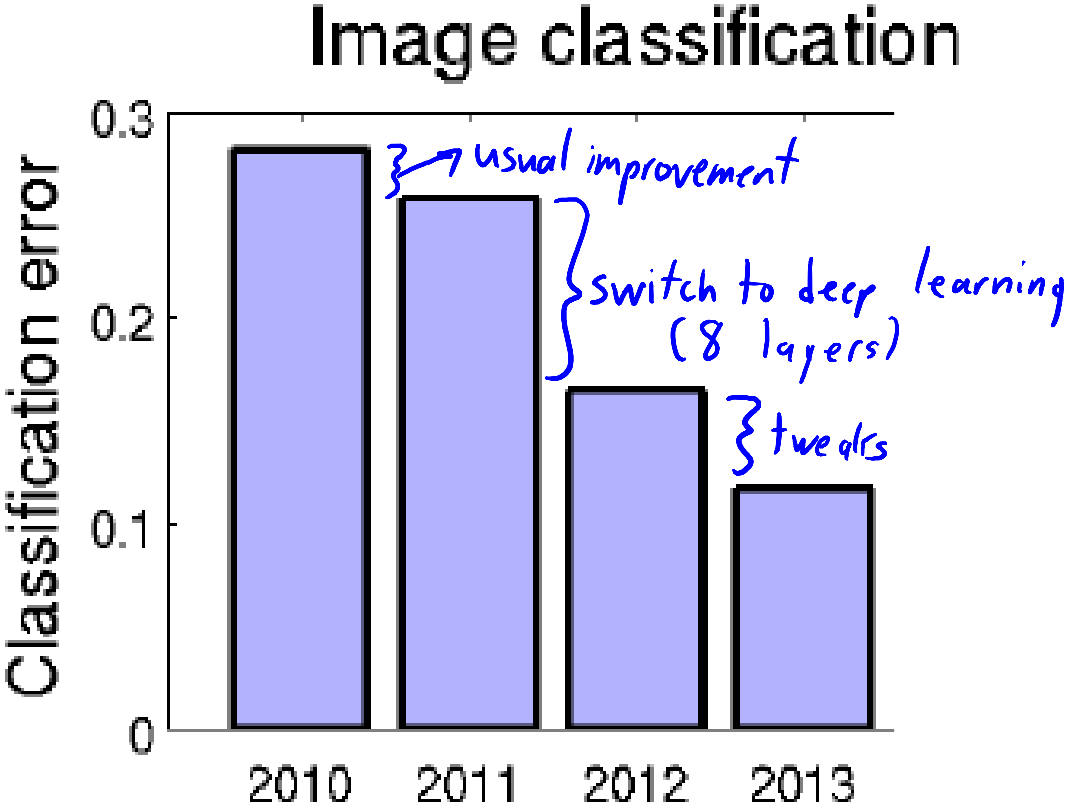


https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

ImageNet Challenge

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

(a) Siberian husky (b) Eskimo dog

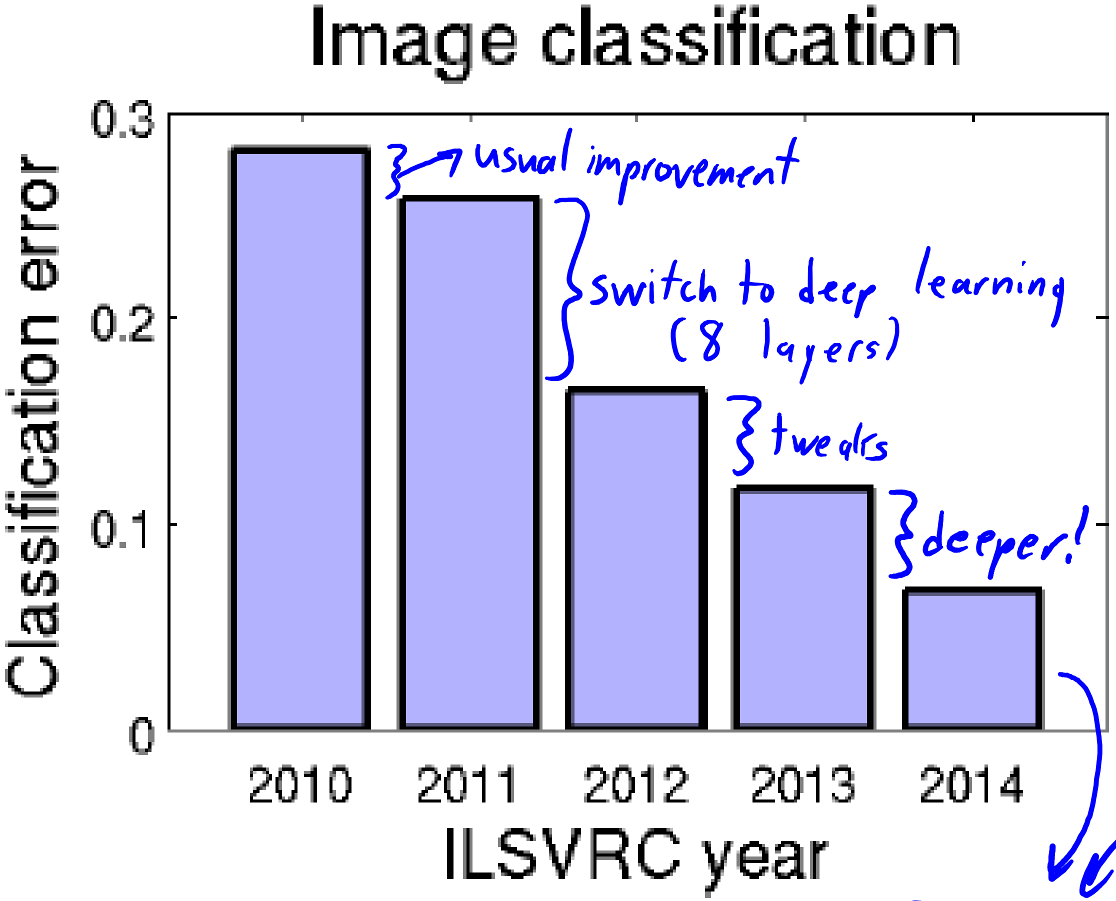


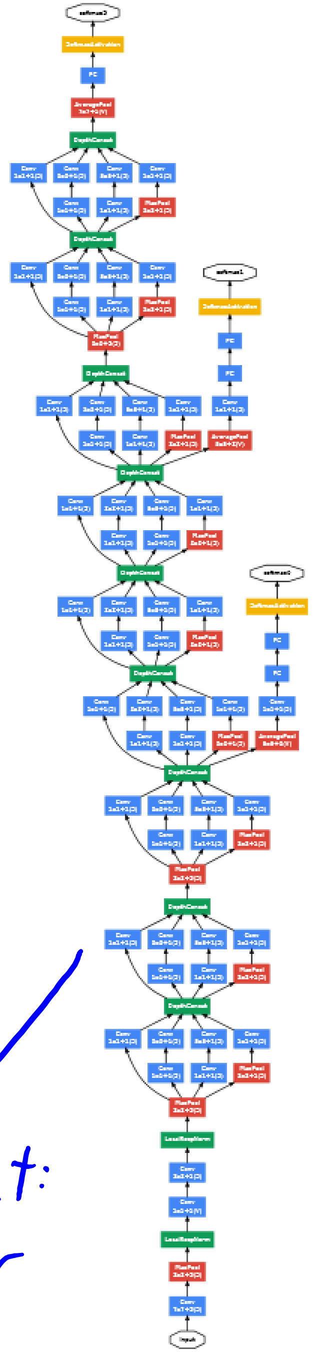
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
http://arxiv.org/pdf/1409.0575v3.pdf
http://arxiv.org/pdf/1409.4842v1.pdf

ImageNet Challenge

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

(a) Siberian husky (b) Eskimo dog





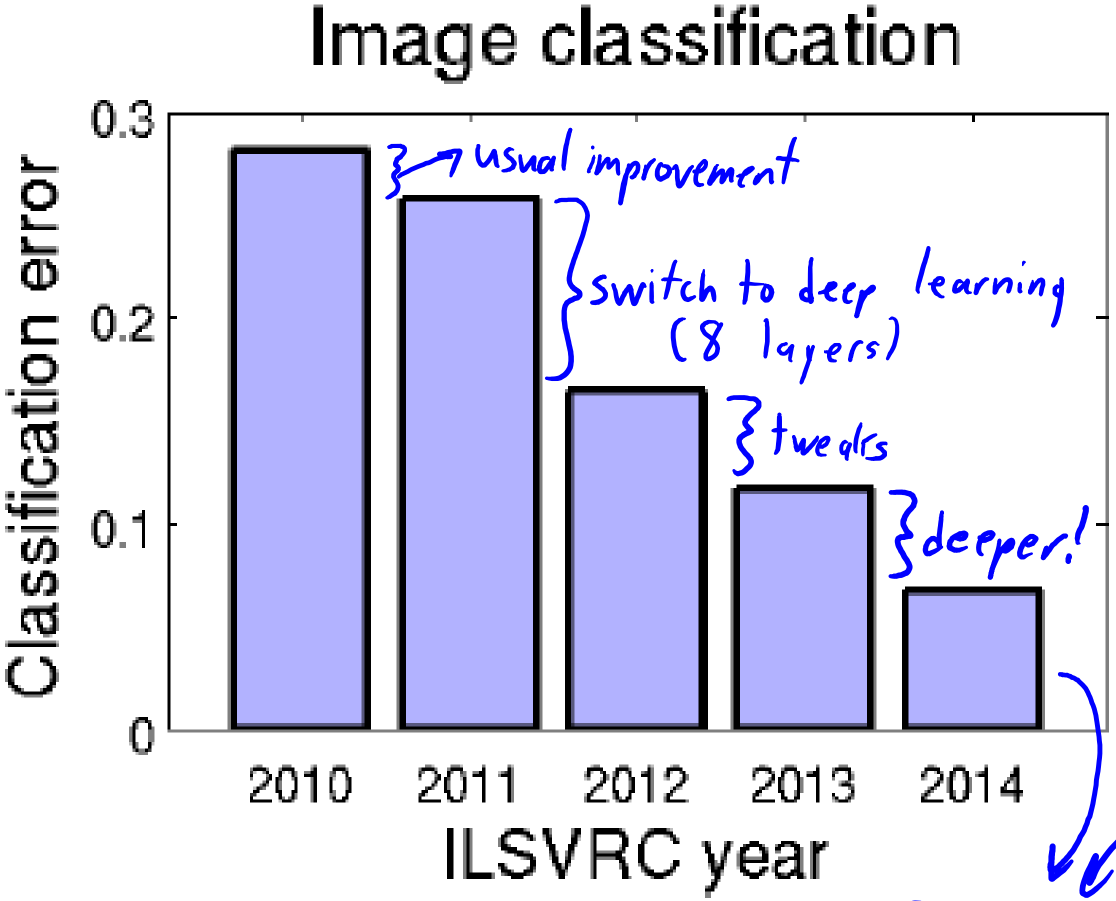
GoogLeNet:
6.7% error
22 layers

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

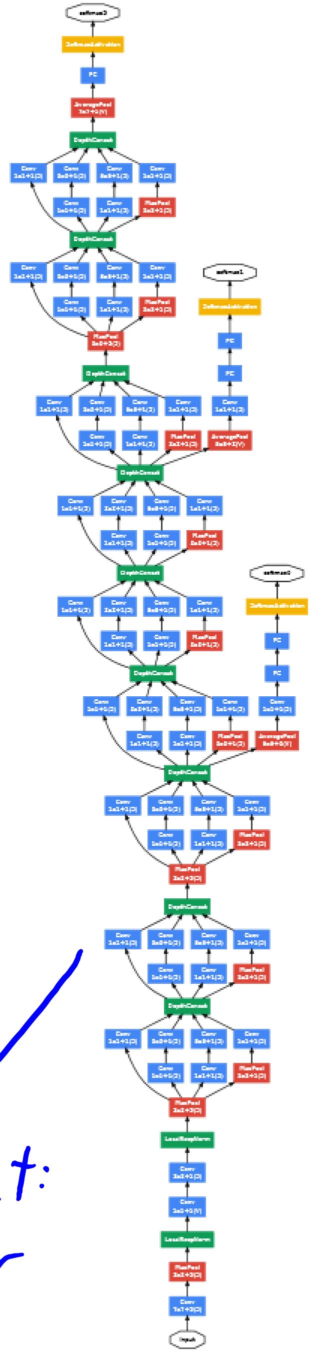
ImageNet Challenge

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

(a) Siberian husky (b) Eskimo dog



GoogLeNet:
6.7% error
22 layers



- 2015 winner: Microsoft
  - 3.6% error.
  - 152 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

Adding a Bias Variable

Remembe that in linear models we may non-zero y-intercept:

$$y_i = w^T x_i + \beta$$

For neural networks we could have explicit bias:

$$y_i = w^T h(Wx_i) + \beta$$

We can just use $y_i = w^T x_i$ if we fix $x_{ij} = 1$ for all 'i' for some 'j'.

(constant x_{ij} value)

Or we could set $W_c = 0$ for one row ~~column~~ W_c of 'W'.

$$\frac{1}{1+\exp(-W_c x_i)} = \frac{1}{1+\exp(0)} = 1/2$$

(constant z_{ic} value)

Artificial Neural Networks

- With squared loss, our objective function is:

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

- Usual training procedure: stochastic gradient.
  - Compute gradient of random example ‘i’, update both ‘w’ and ‘W’.
  - Highly non-convex and can be difficult to tune.

- Computing the gradient is known as “backpropagation”.

Backpropagation

- Consider the loss for a single example:

$$f(w, W) = \frac{1}{2}\Big(\sum_{c=1}^{k} \underbrace{w_c}_{\text{Element 'c' of 'w'}} h(\underbrace{W_c}_{\text{Row 'c' of } W}\, x_i) - y_i\Big)^2$$

- Derivative with respect to 'w_c':

$$\frac{\partial}{\partial w_c}[f(w,W)] = \Big(\overbrace{\underbrace{\sum_{c=1}^{k} w_c h(W_c x_i)}_{\text{derivative with respect to } w_c} - y_i}^{\text{From squared loss}}\Big) h(W_c x_i)$$

- Derivative with respect to 'W_{cj}'

$$\frac{\partial}{\partial W_{cj}}[f(w,W)] = \Big(\underbrace{\sum_{c=1}^{k} w_c h(W_c x_i)}_{\text{derivative with respect to } W_c x_i} - y_i\Big) \underbrace{w_c h'(\overbrace{W_c x_i}^{\text{derivative with respect to } W_{cj}})}\, x_{ij}$$

Backpropagation

- Notice repeated calculations in gradients:

$$\frac{\partial}{\partial w_c}\left[f(w,W)\right] = \underbrace{\left(\sum_{c=1}^{k} w_c h(W_c x_i) - y_i\right)}_{r_i} h(W_c x_i)$$

$$= r_i h(W_c x_i)$$

same r_i for all 'c'

$$\frac{\partial}{\partial W_{cj}}\left[f(w,W)\right] = \underbrace{\left(\sum_{c=1}^{k} w_c h(W_c x_i) - y_i\right)}_{r_i} \underbrace{w_c h'(W_c x_i)}_{v_c} x_{ij}$$

$$= r_i v_c x_{ij}$$

same v_c for all 'j'

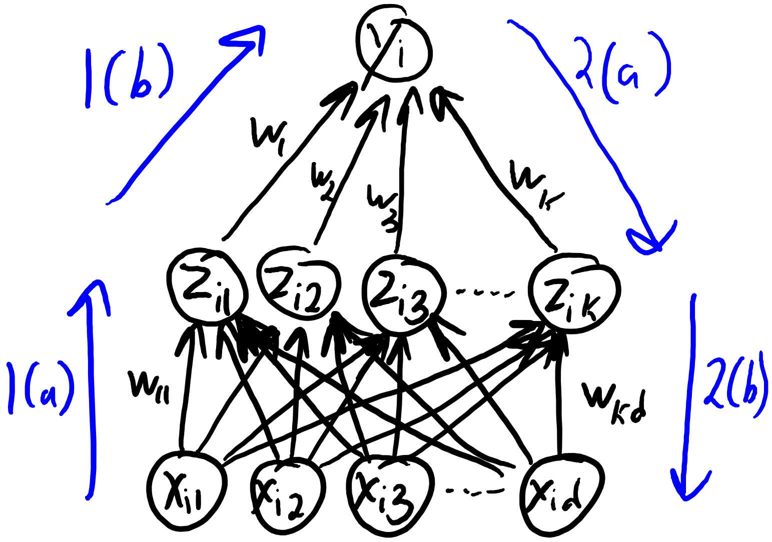
same r_i for all 'c')

Backpropagation

- Calculation of gradient is split into two phases:

1. "Forward" pass
   (a) Compute $h(W_c x_i)$ for all 'c'
   (b) Compute residual $r_i = \left(\sum_{c=1}^{k} w_c h(W_c x_i) - y_i\right)$

2. "Backprogation"
   (a) Compute $\frac{\partial f}{\partial w_c} = r_i h(W_c x_i)$ for all 'c'
   (b) Compute $v_c = w_c h'(W_c x_i)$ for all 'c'
   (c) Compute $\frac{\partial f}{\partial W_{cj}} = r_i v_c x_{ij}$ for all 'c' and 'j'



Summary

- Biological motivation for (deep) neural networks.
- Deep learning considers neural networks with many hidden layers.
- Unprecedented performance on difficult pattern recognition tasks.
- Backpropagation computes neural network gradient via chain rule.

- Next time:
  - How deep learners fight the fundamental trade-off.