

CPSC 340: Machine Learning and Data Mining

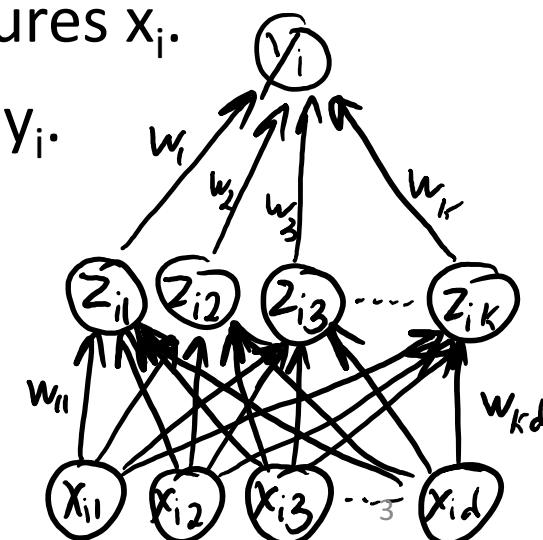
Neural Networks

Admin

- Assignment 4:
 - Solutions are posted
- Assignment 5:
 - Due a week from today (Friday March 31)
- Assignment 6:
 - Will be due 2 weeks from today (Friday April 6)
- Final exam:
 - April 25, 8:30am, ESB 1013
 - Covers Assignments 1-6.
 - Final from last year is posted on GitHub.
 - Closed-book except 1-page double-sided cheat sheet (same as midterm)

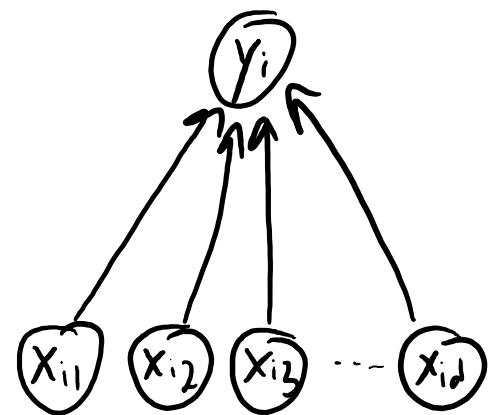
Supervised Learning Roadmap

- 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.



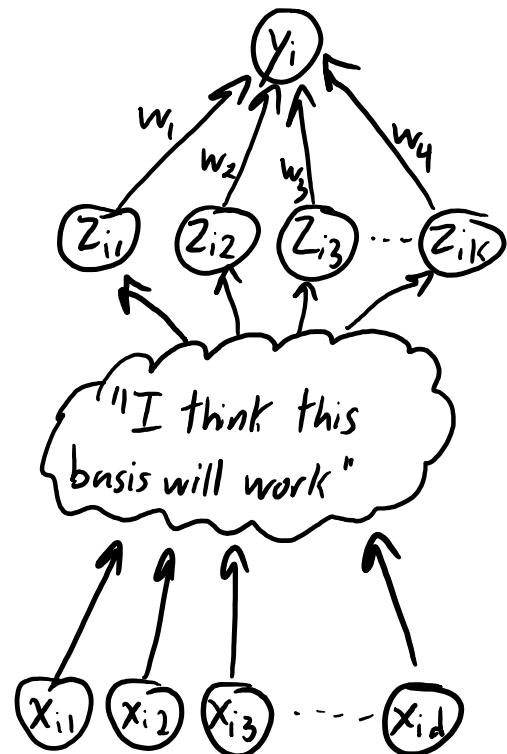
Supervised Learning Roadmap

Part 1: x_i is the basis

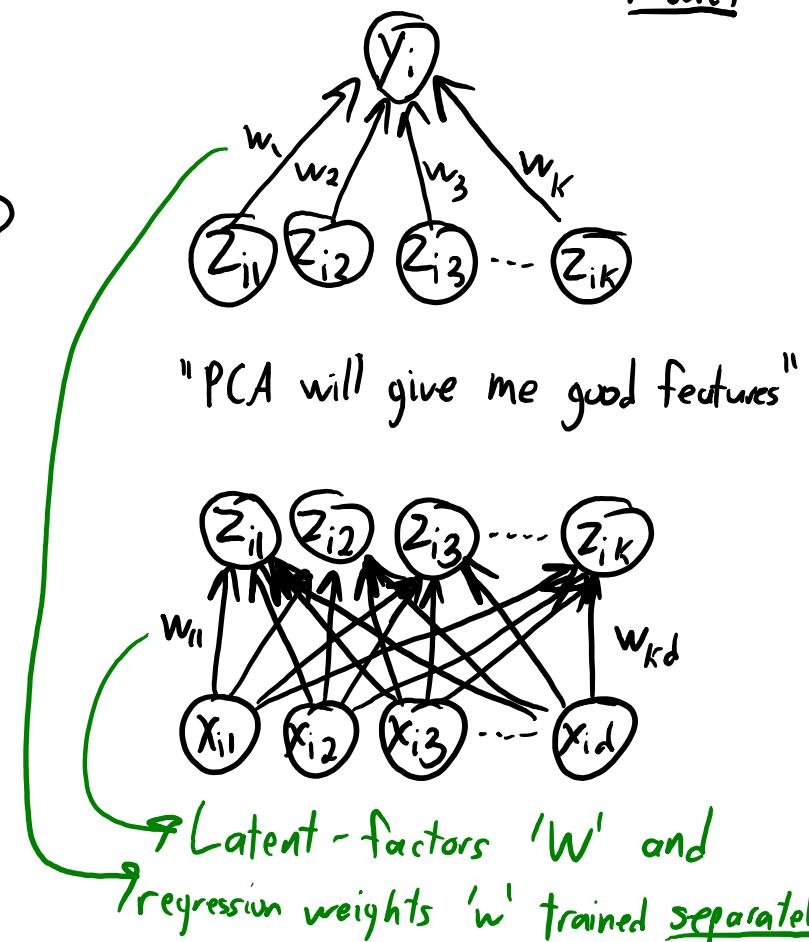


"These are the features"

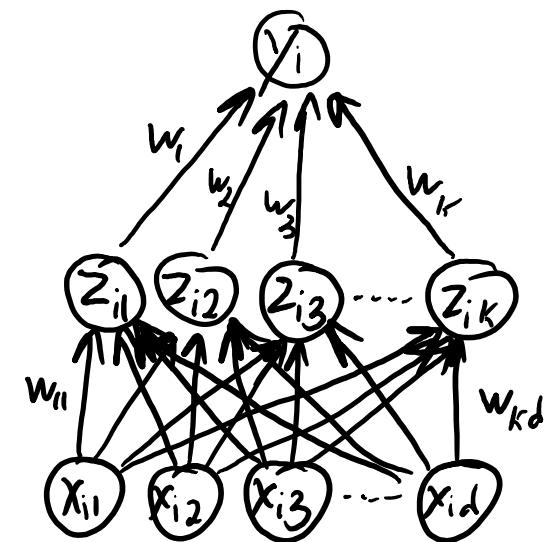
Part 3: Change of basis



Part 4: basis from latent-factor model



Part 5: Neural networks



Learn ' W ' and ' w ' together: learn features that are good for supervised task.

Linear-Linear Model

- Natural choice: linear latent-factor model with linear regression.

Use features from latent-factor model: $z_i = Wx_i$

Make predictions using a linear model: $y_i = w^T z_i$

- We want to train 'W' and 'w' jointly, so we could minimize:

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

linear regression
with z_i as features z_i come from
latent-factor model

- But this is just a linear model:

$$y_i = w^T z_i = w^T (Wx_i) = (\underbrace{w^T W}_{\text{some vector } v}) x_i = v^T x_i$$

5

Introducing Non-Linearity

- To increase flexibility, something needs to be non-linear.
- Typical choice: transform z_i by non-linear function ‘ h ’.

$$y_i = w^T z_i \quad \text{with} \quad z_i = h(w_{x_i})$$

- Common choice for ‘ h ’: applying sigmoid function element-wise:

$$z_{ic} = \frac{1}{1 + \exp(-w_c^T x_i)}$$

- We can now (roughly) think of the z_{ic} as binary features we learn.
- This is called a “multi-layer perceptron” or a “neural network”.

Notation for Neural Networks

We have our usual supervised learning notation:

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

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

We have our latent features:

$$Z = \begin{bmatrix} z_1^\top \\ z_2^\top \\ \vdots \\ z_n^\top \end{bmatrix}$$

$n \times K$

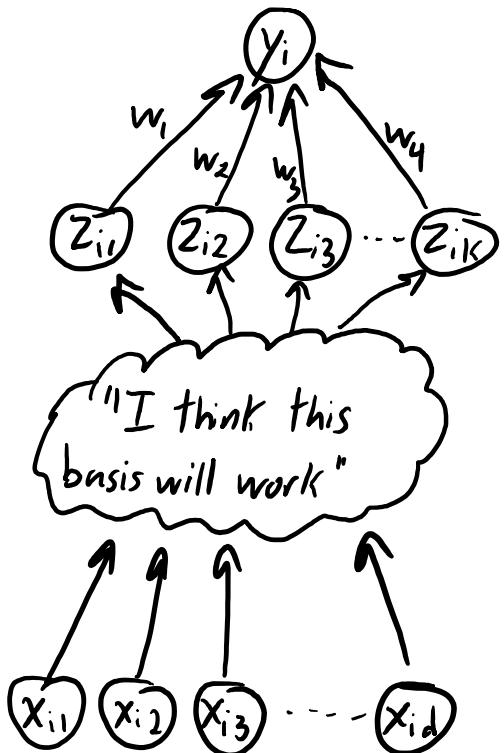
We have two sets of parameters:

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix} \quad W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_K \end{bmatrix}$$

$K \times 1$ $K \times d$

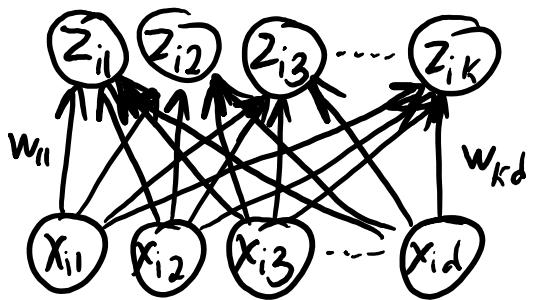
Supervised Learning Roadmap

Hand-engineered features:

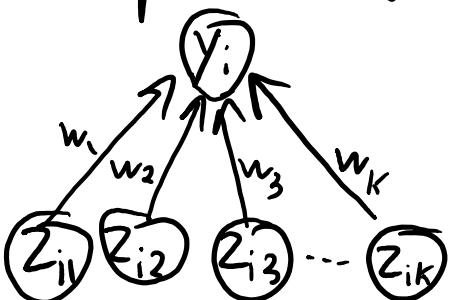


Requires domain knowledge
and can be time-consuming

Learn a latent-factor model:

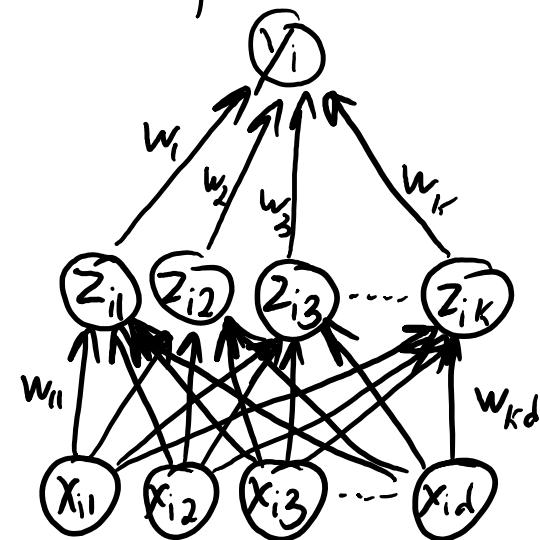


Use latent features
in supervised model:



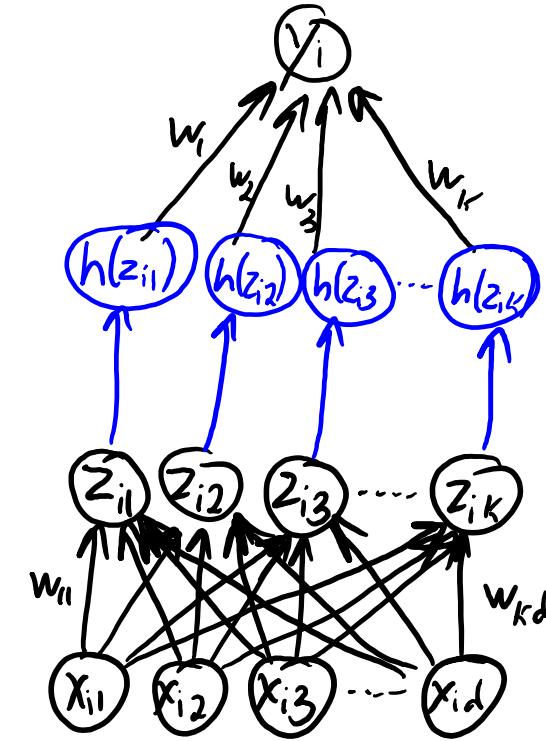
Good representation of
 x_i might be bad for predicting y_i

Learn ' w ' and ' W '
together:



But still gives a
linear model.

Neural network:



Extra non-linear
transformation ' h '!

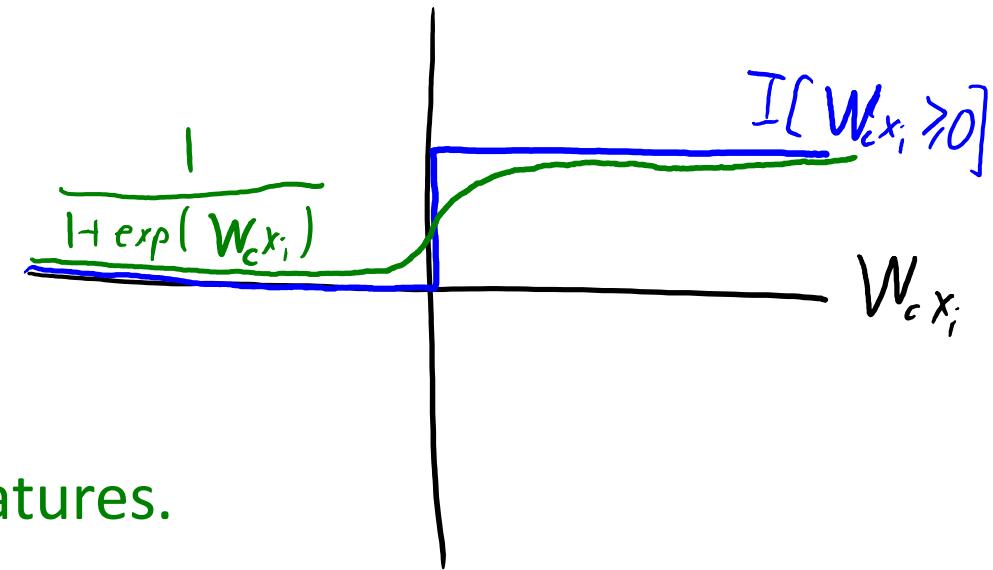
Why Sigmoid?

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

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

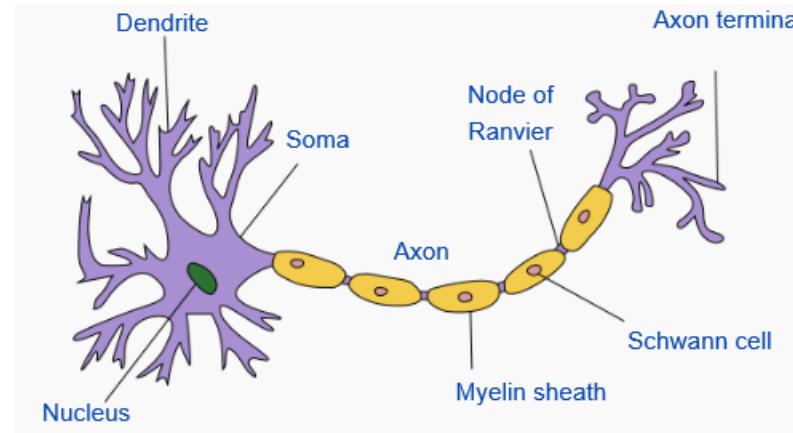
$$= \begin{cases} 1 & \text{if } W_c x_i \geq 0 \\ 0 & \text{if } W_c x_i < 0 \end{cases}$$

- 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.

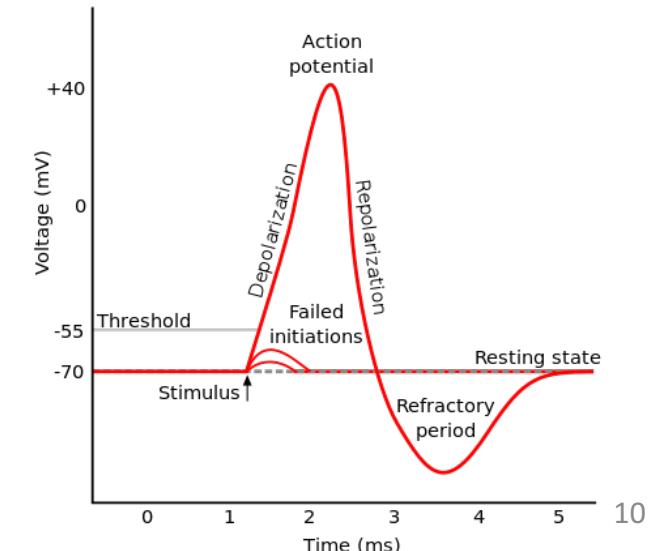


Why “Neural Network”?

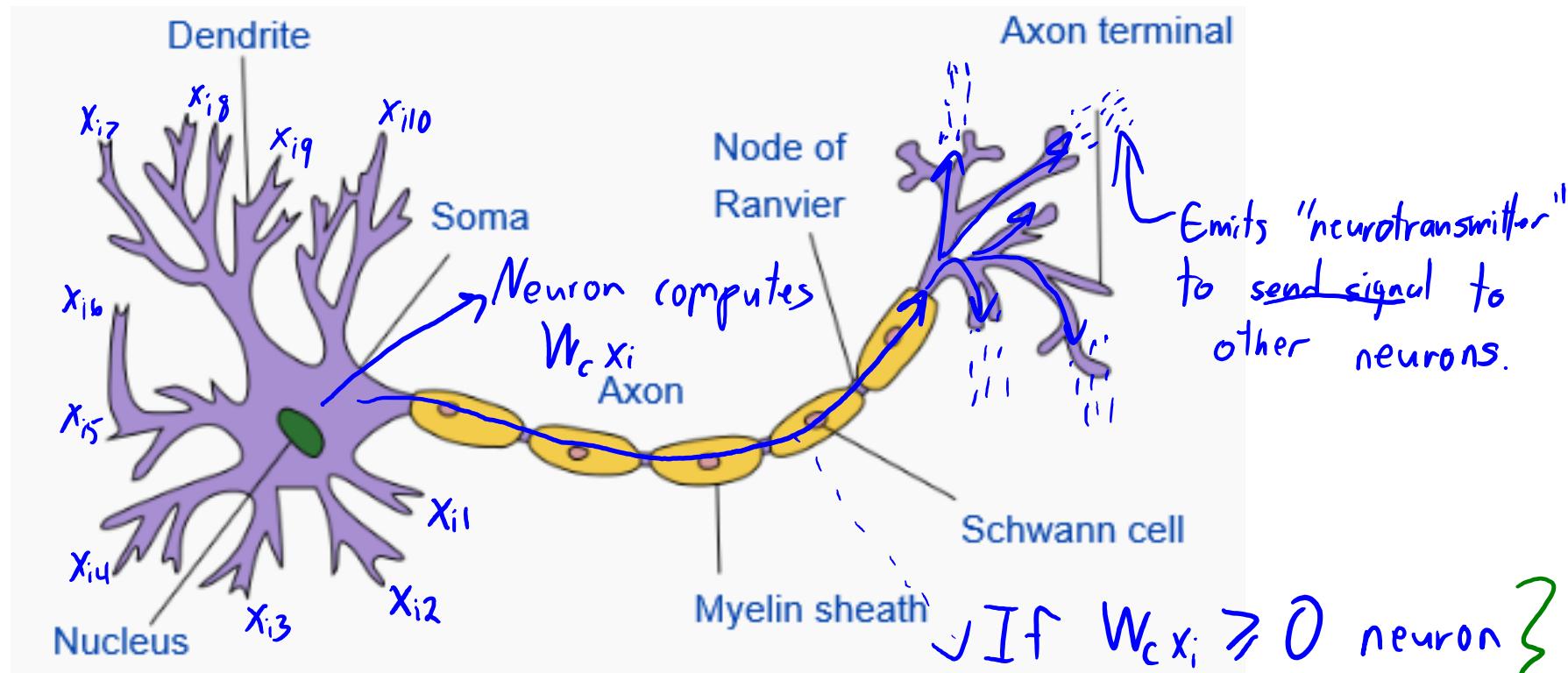
- Cartoon of “typical” neuron:



- 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):



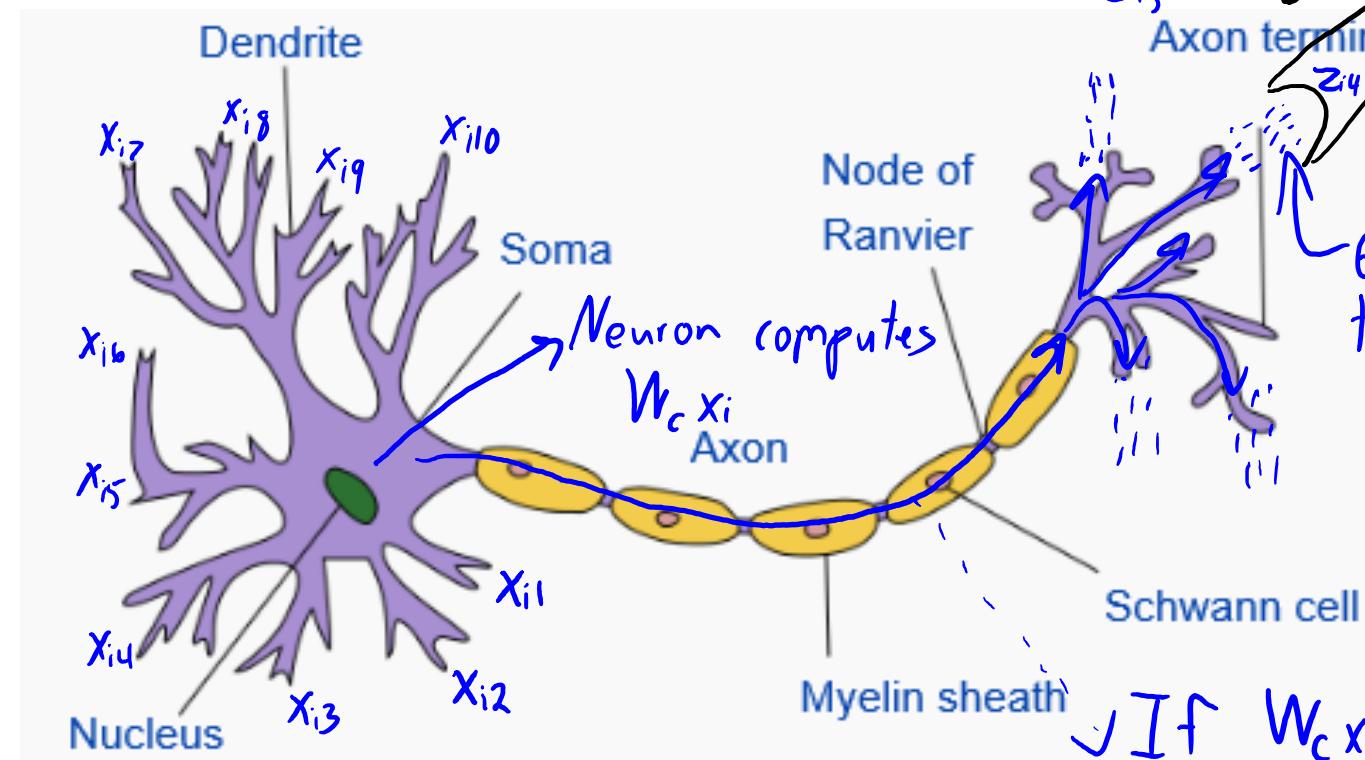
Why “Neural Network”?



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

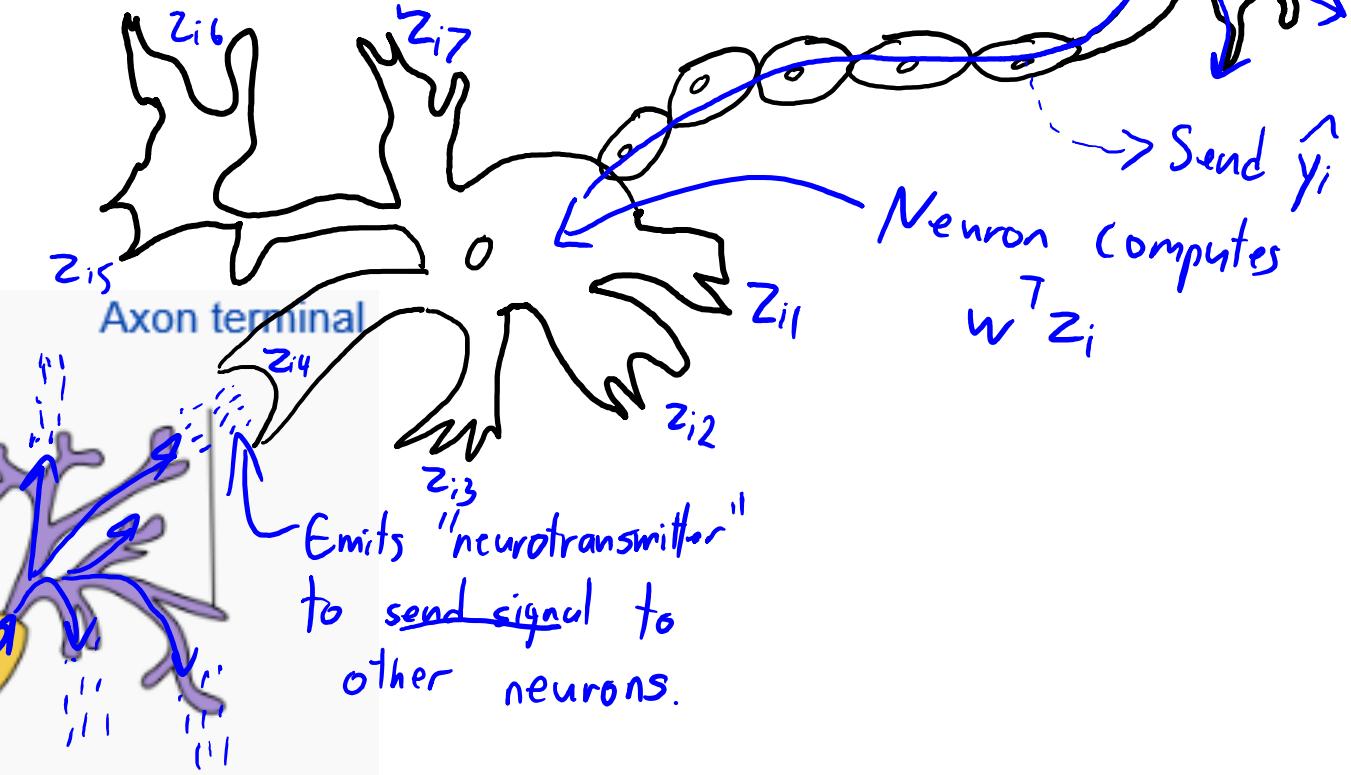
We approximate binary
signal with $\frac{1}{1 + \exp(-W_c x_i)}$

Why “Neural Network”?

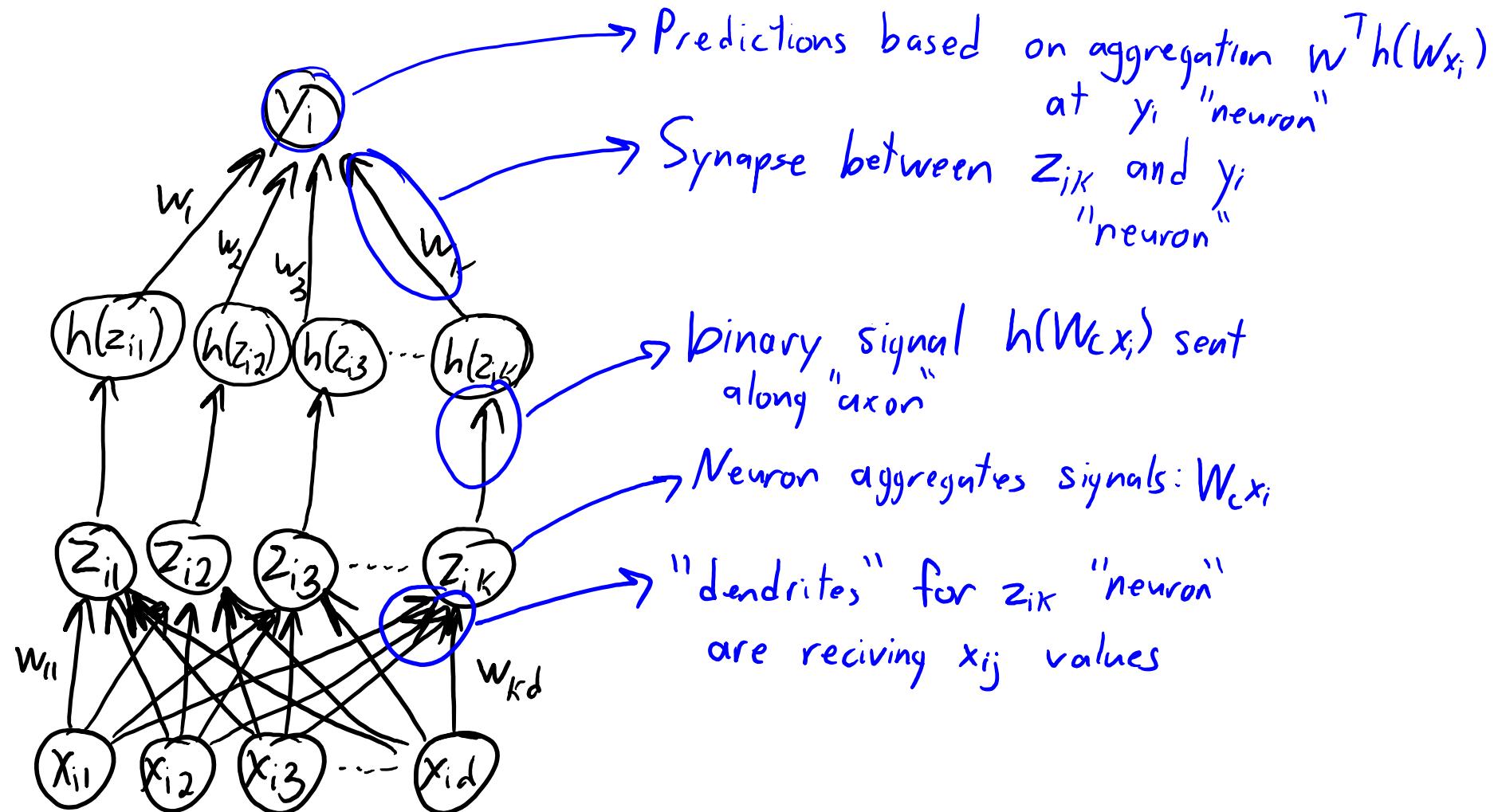


If $W_c x_i \geq 0$ neuron
Sends signal along axon.

We approximate binary
signal with $\frac{1}{1 + \exp(-12W_c x_i)}$

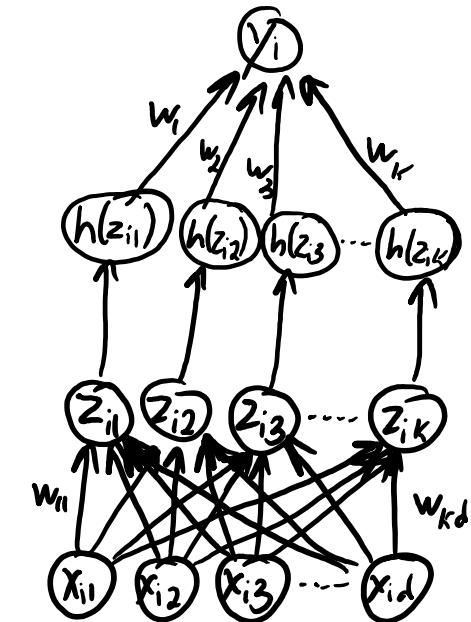


Why “Neural Network”?

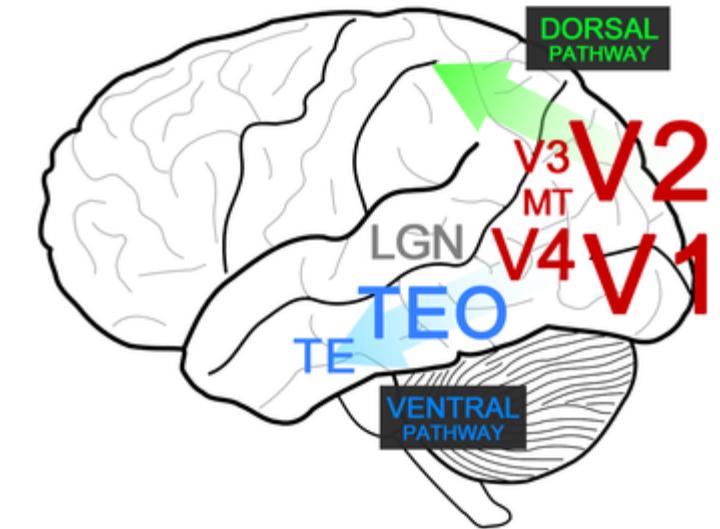
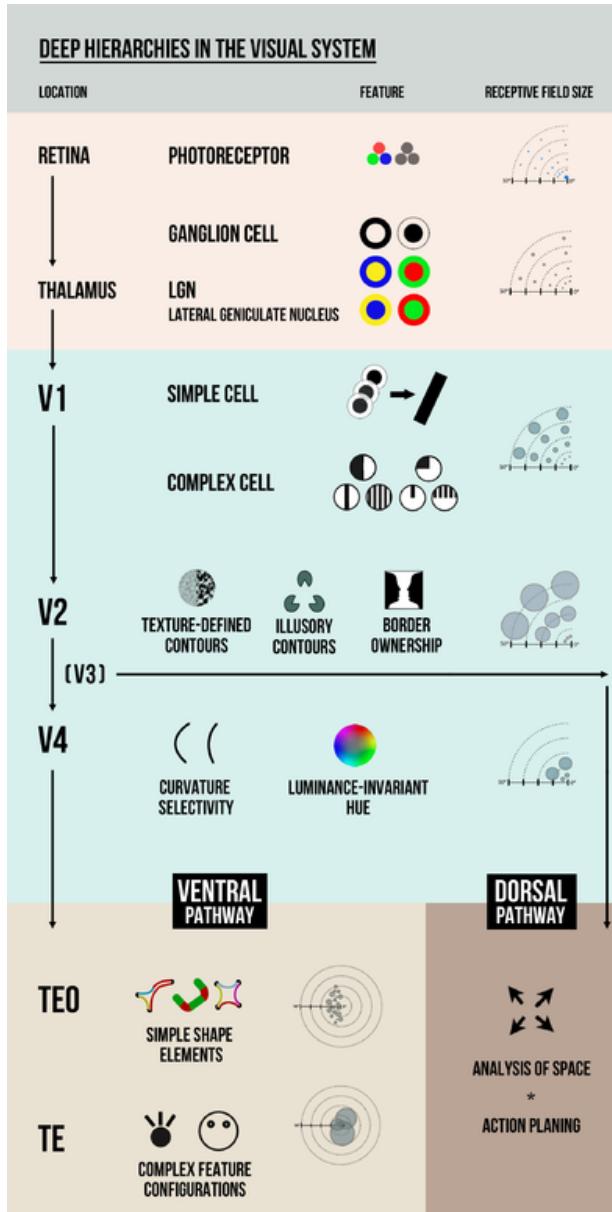
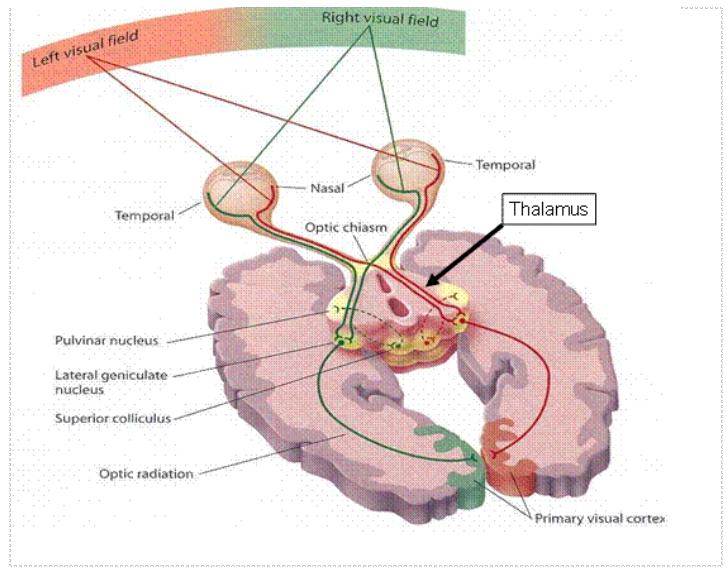


“Artificial” Neural Nets vs. “Real” Networks Nets

- 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 a neuron?**
 - Brain is highly **organized** (e.g., substructures and cortical columns).
 - Connection **structure changes**.
 - **Different types** of neurotransmitters.

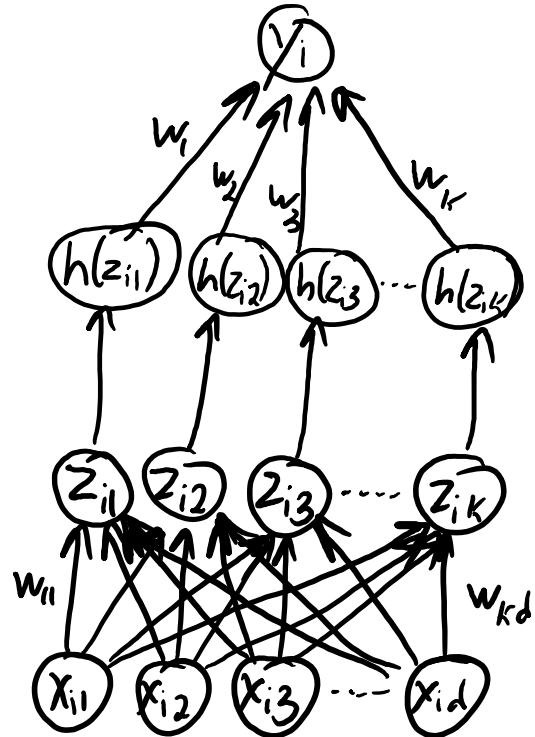


Deep Hierarchies in the Brain



Deep Learning

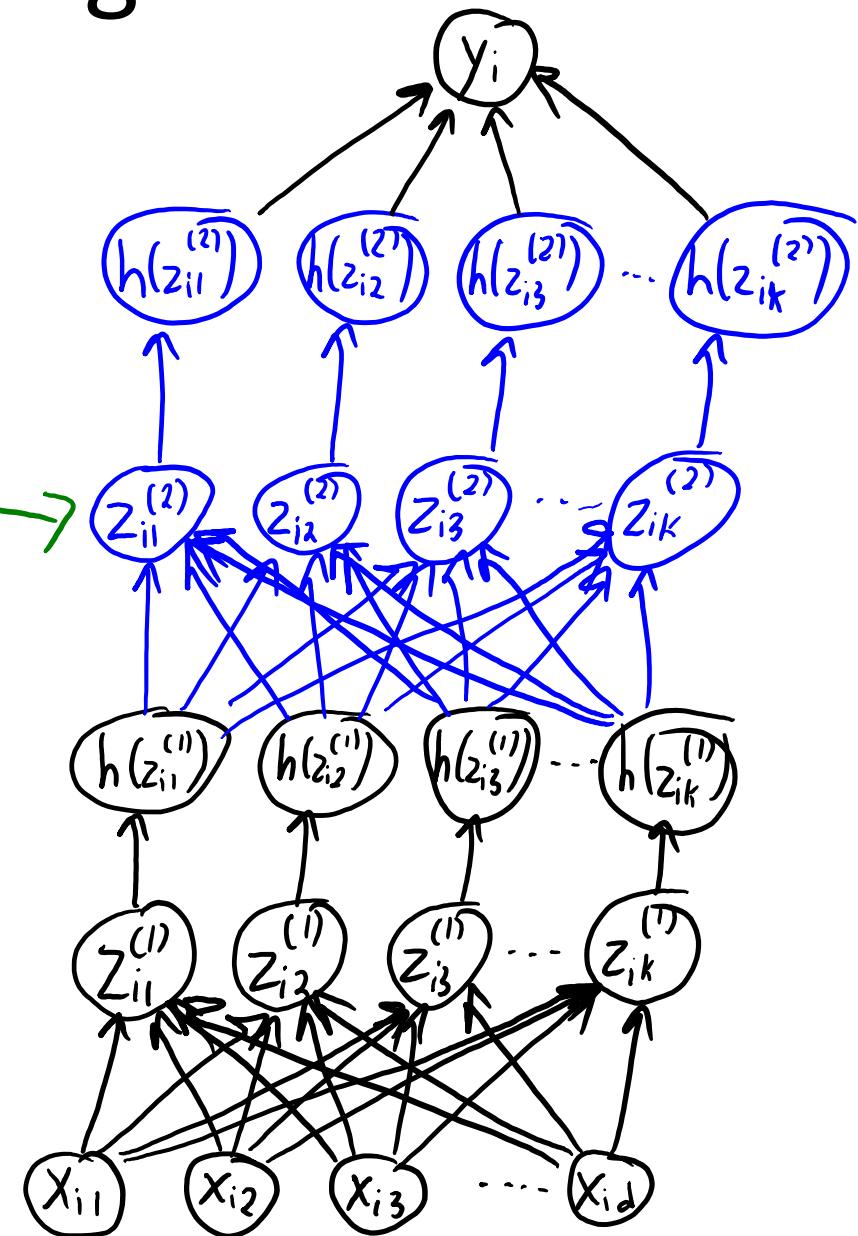
Neural network:



Deep learning:

Second "layer" of latent features

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



Deep Learning

Linear model:

$$y_i = w^\top x_i$$

Neural network with 1 hidden layer:

$$y_i = w^\top h(Wx_i)$$

Neural network with 2 hidden layers:

$$y_i = w^\top h(W^{(2)} h(W^{(1)} x_i))$$

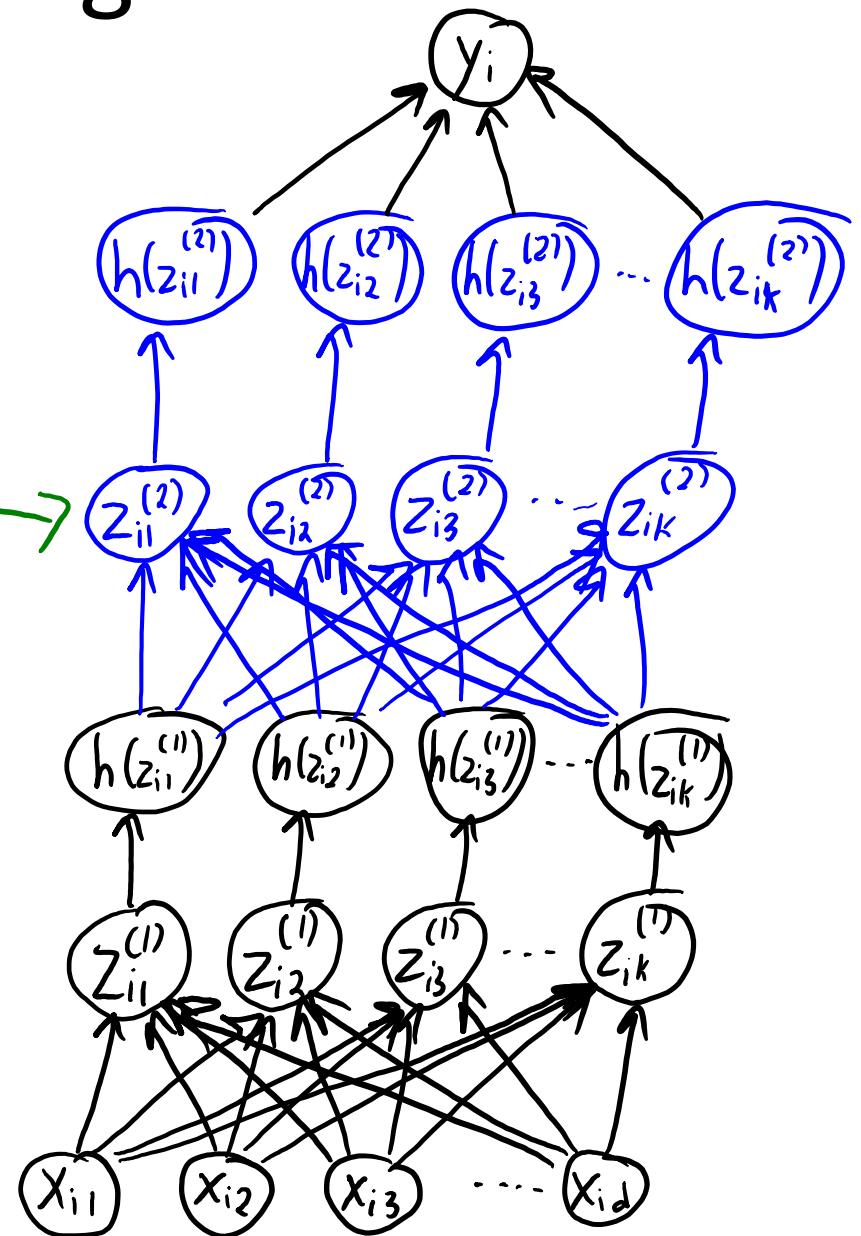
Neural network with 3 hidden layers

$$y_i = w^\top h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i)))$$

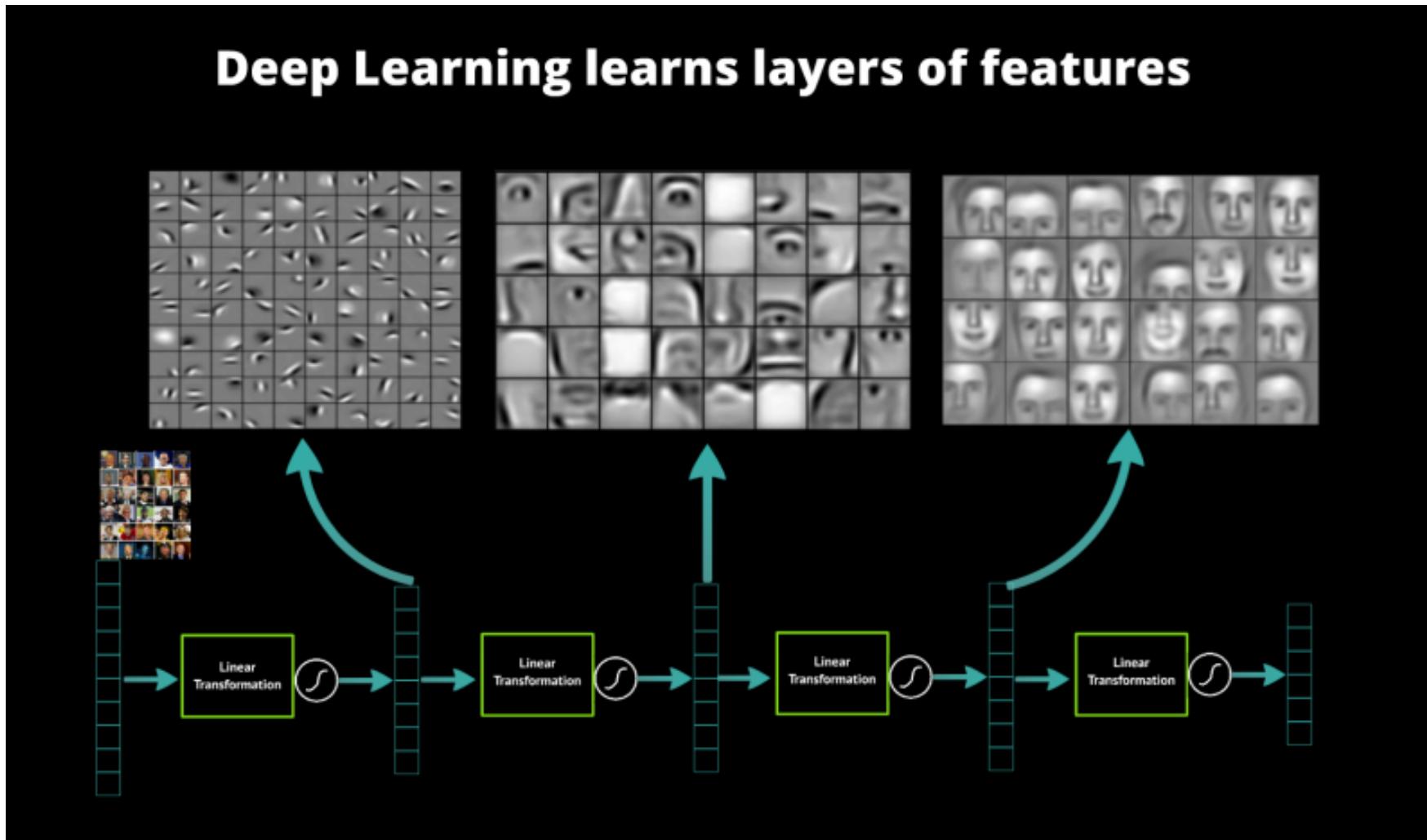
Deep learning:

Second "layer" of latent features

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

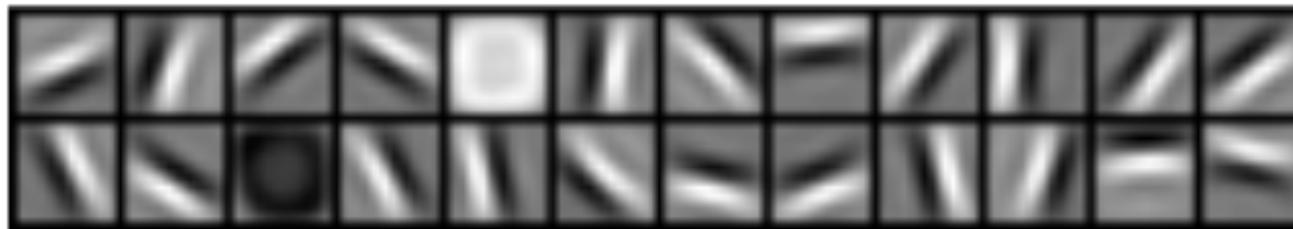


Cool Picture Motivation for Deep Learning

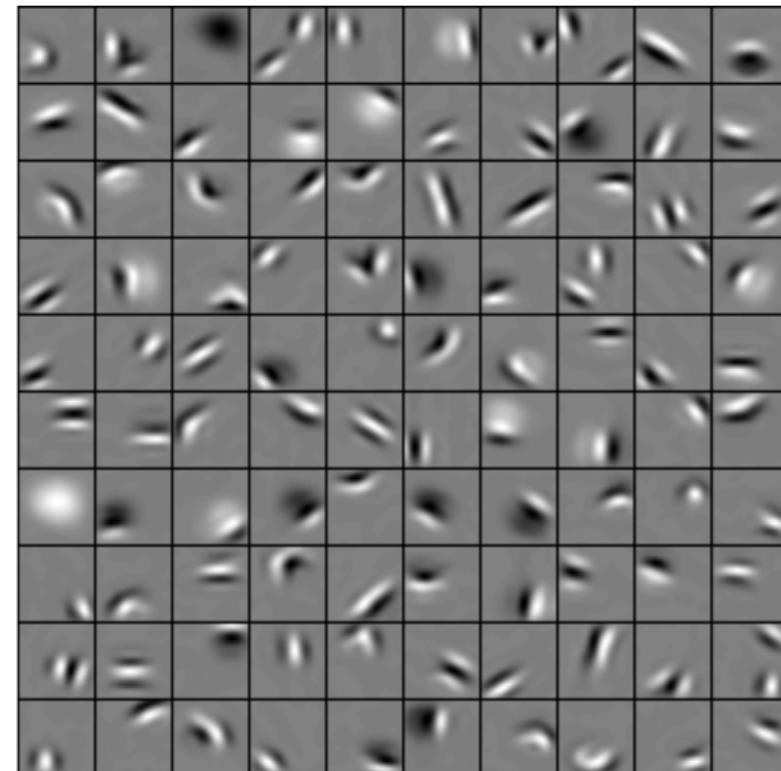


Cool Picture Motivation for Deep Learning

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

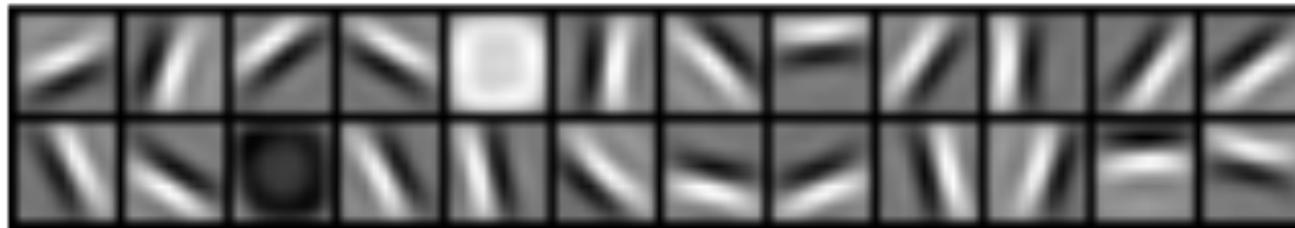


- Attempt to visualize second layer:
 - Corners, angles, surface boundaries?
- Models require many tricks to work.
 - We'll discuss these next time.



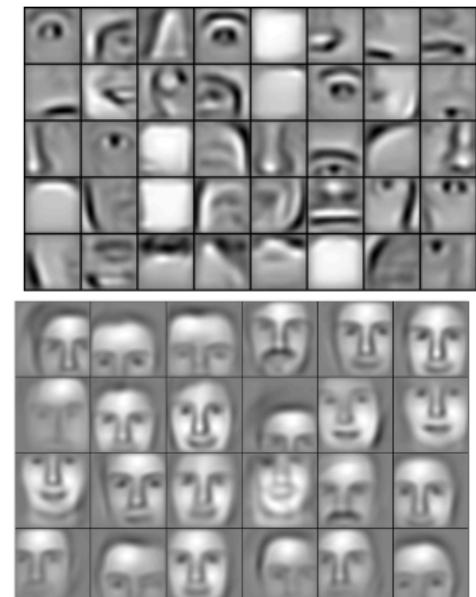
Cool Picture Motivation for Deep Learning

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



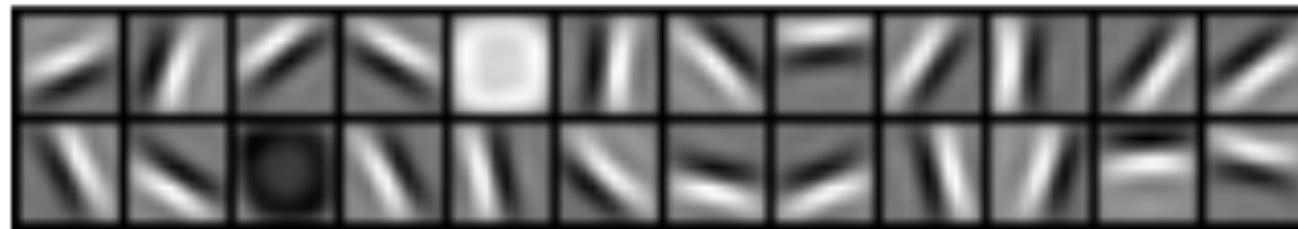
- Visualization of second and third layers trained on specific objects:

faces



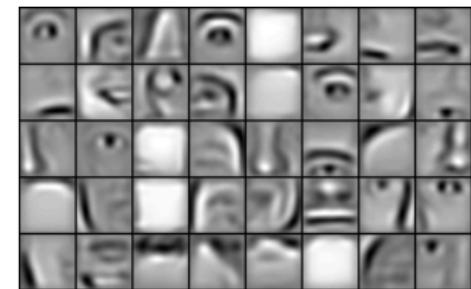
Cool Picture Motivation for Deep Learning

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



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

faces

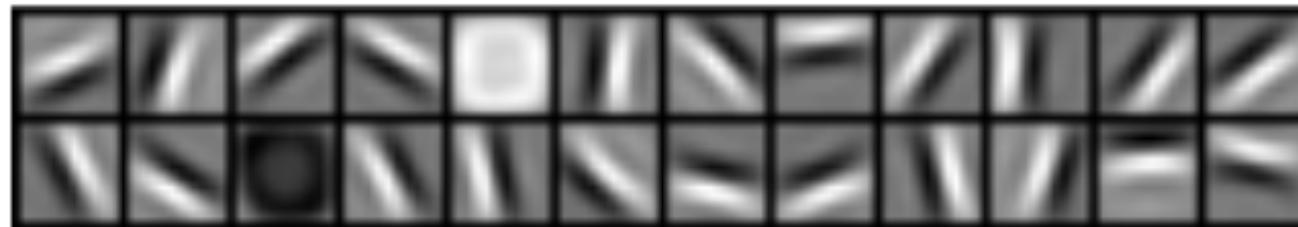


cars



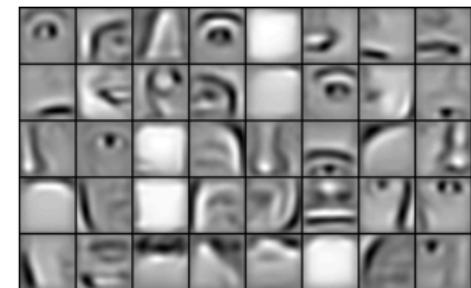
Cool Picture Motivation for Deep Learning

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



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

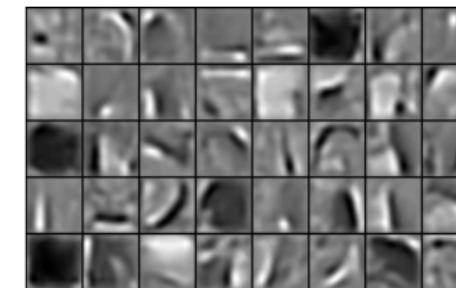
faces



cars

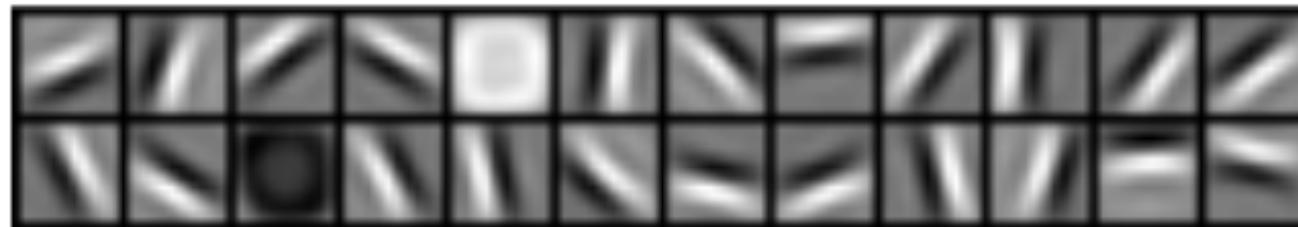


elephants



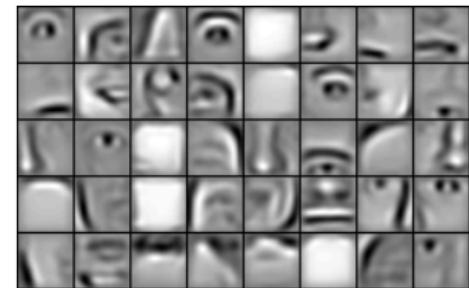
Cool Picture Motivation for Deep Learning

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



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

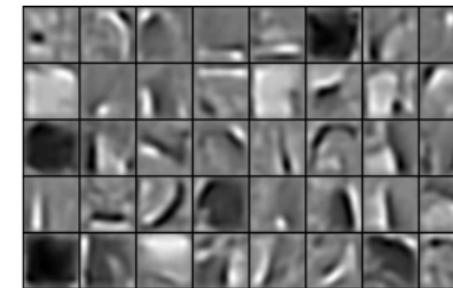
faces



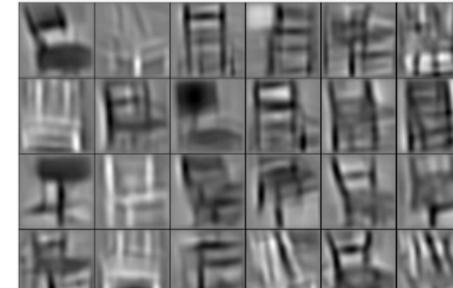
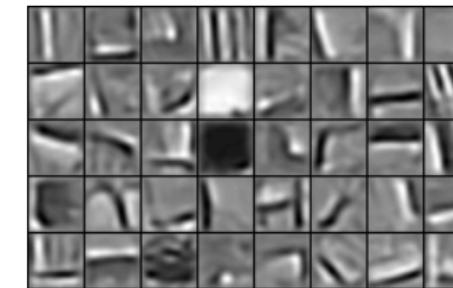
cars



elephants

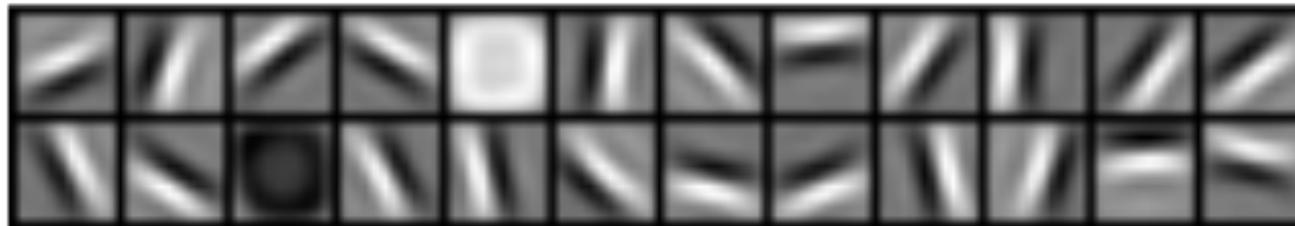


chairs



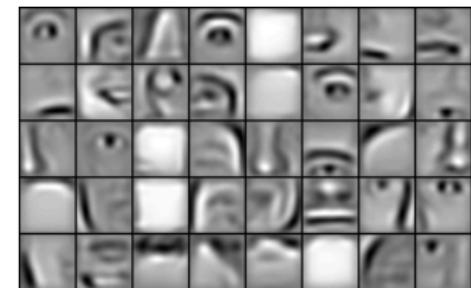
Cool Picture Motivation for Deep Learning

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



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

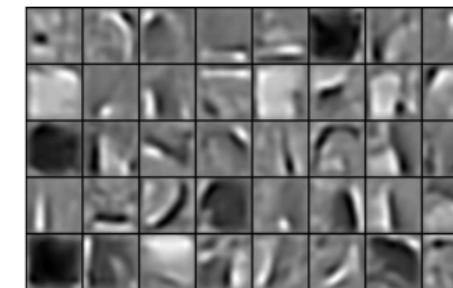
faces



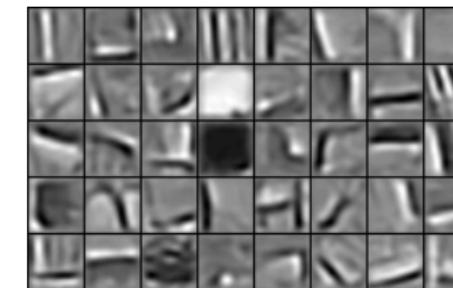
cars



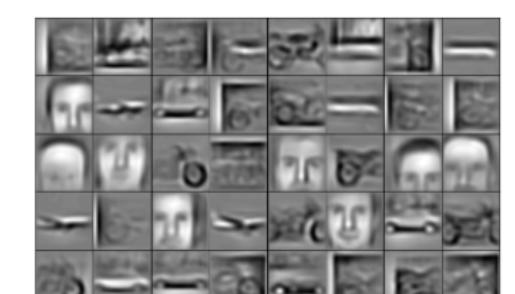
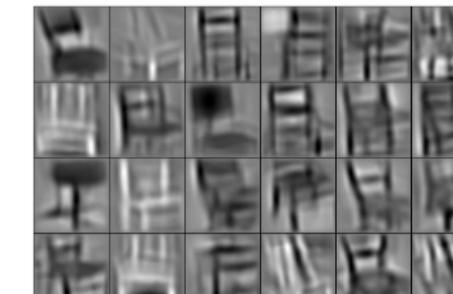
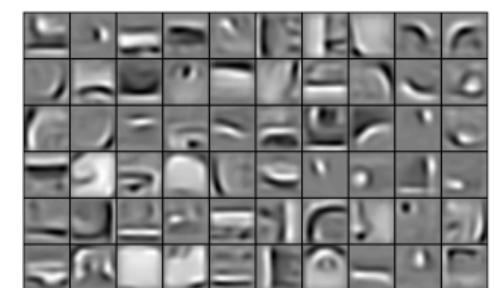
elephants

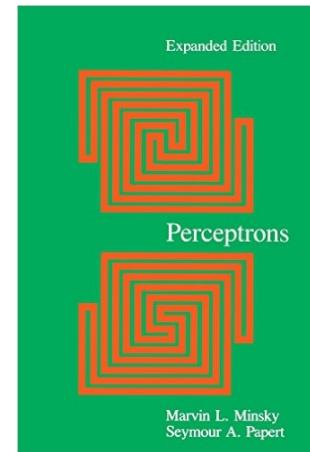


chairs



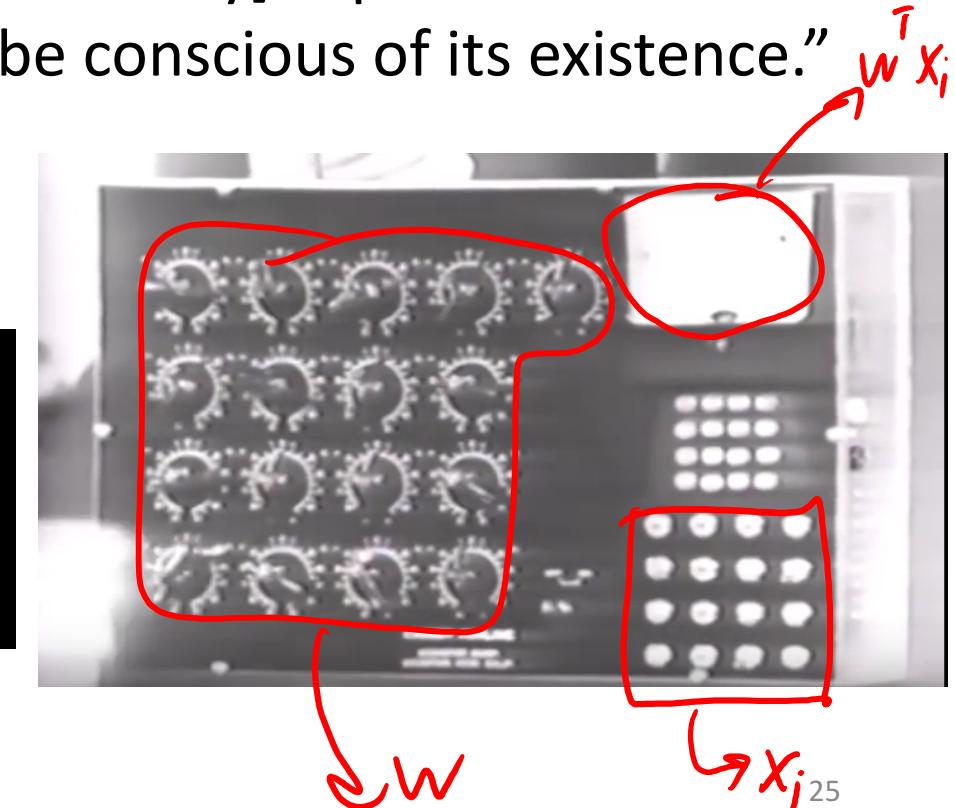
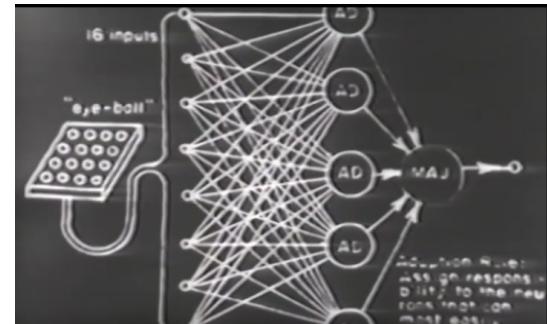
faces, cars, airplanes, motorbikes





ML and Deep Learning History

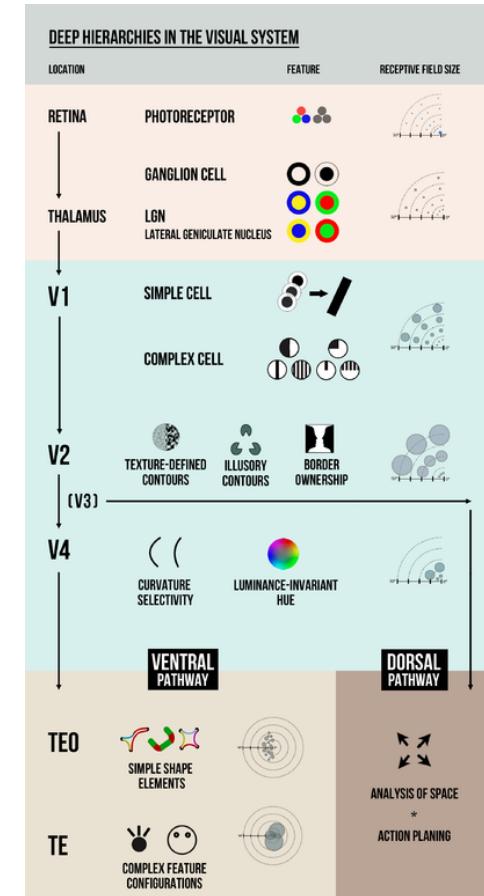
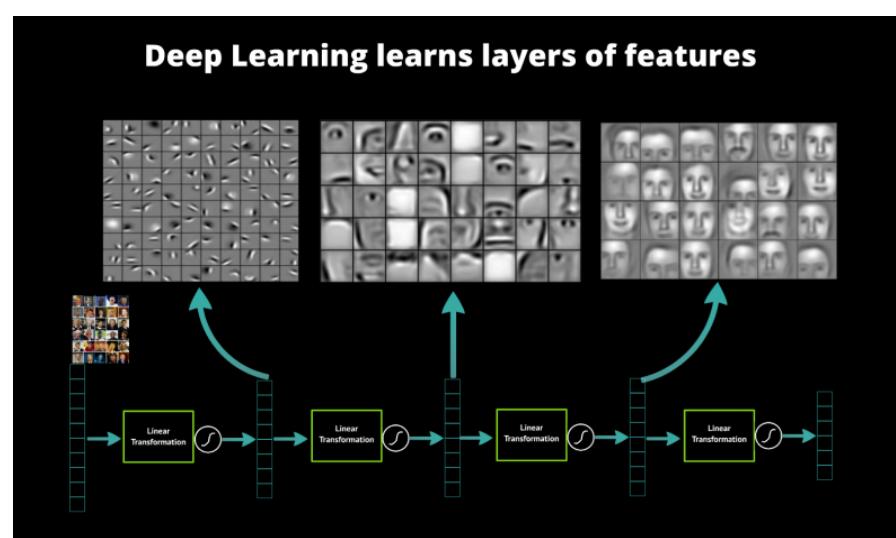
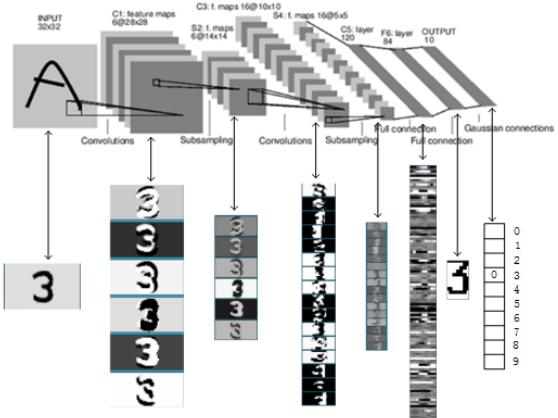
- 1950 and 1960s: Initial excitement.
 - **Perceptron**: linear classifier and stochastic gradient (roughly).
 - “the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.” New York Times (1958).
 - <https://www.youtube.com/watch?v=IEFRtz68m-8>



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

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.

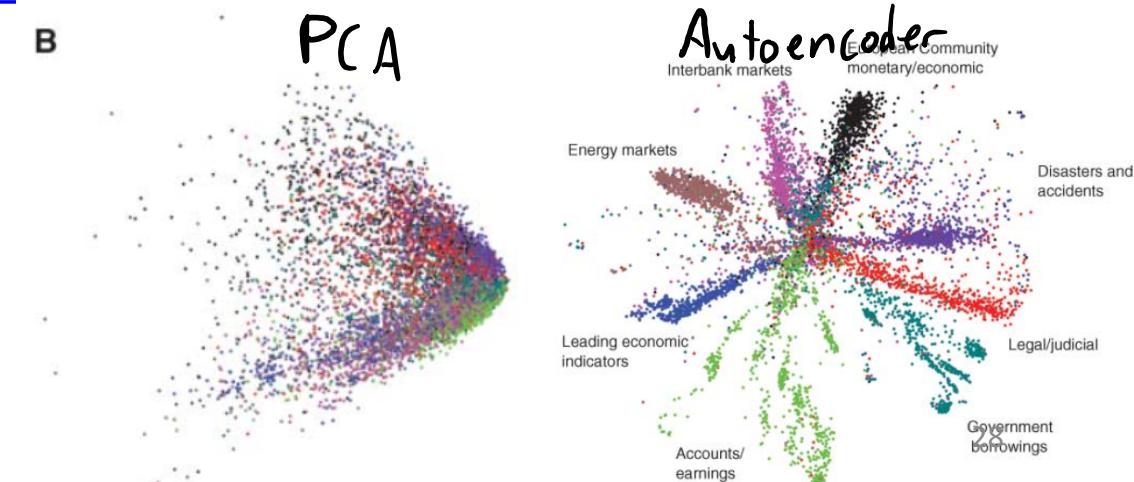


ML and Deep Learning History

- 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.

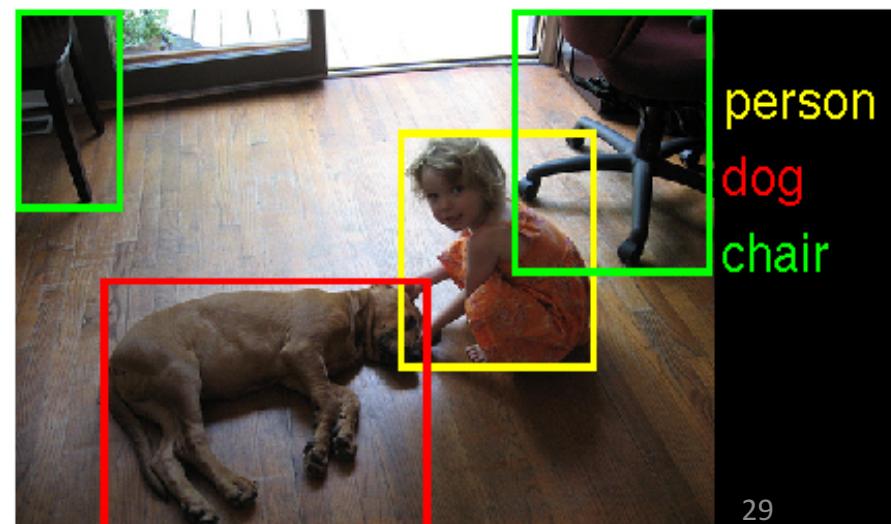
ML and Deep Learning History

- 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>



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.
 - This is now finding its way into products.



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.

Summary

- Neural networks learn features for supervised learning
- Sigmoid function avoids degeneracy by introducing non-linearity.
- Biological motivation for (deep) neural networks.
- Deep learning considers neural networks with many hidden layers.
- Unprecedented performance on difficult pattern recognition tasks.
- Next time:
 - How deep learners fight the fundamental trade-off.