

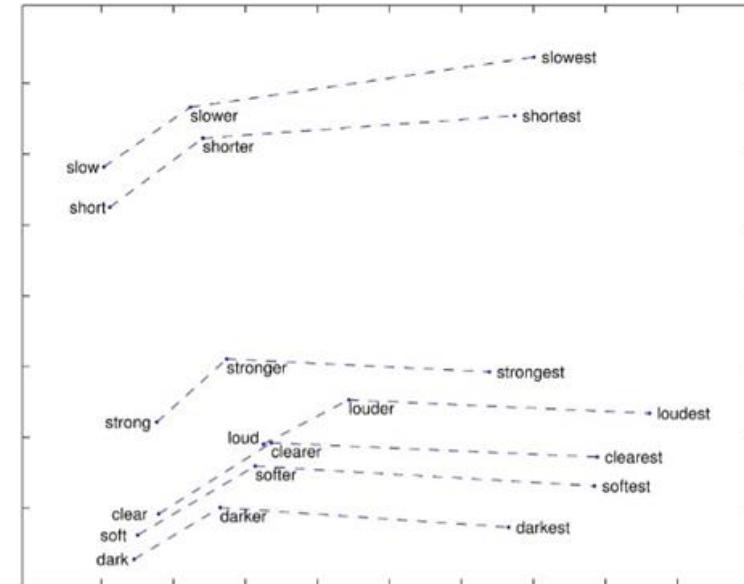
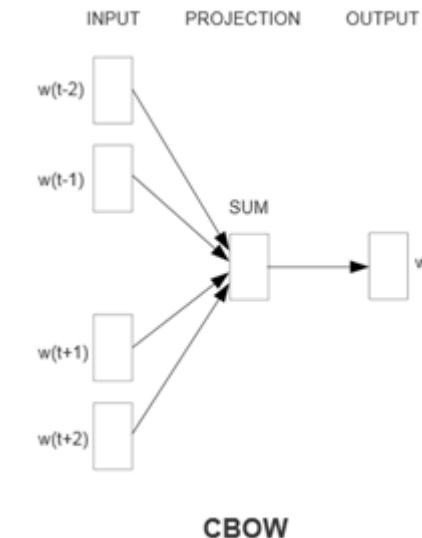
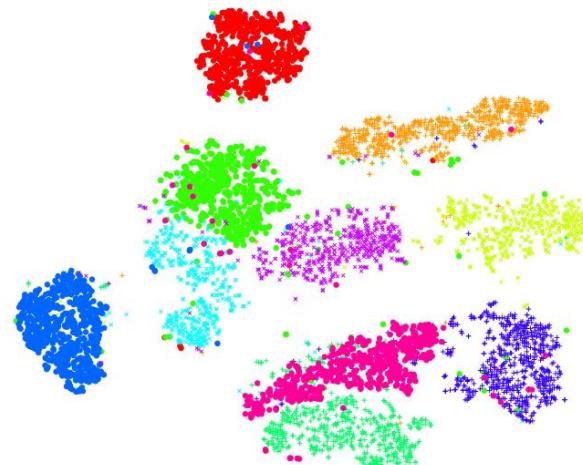
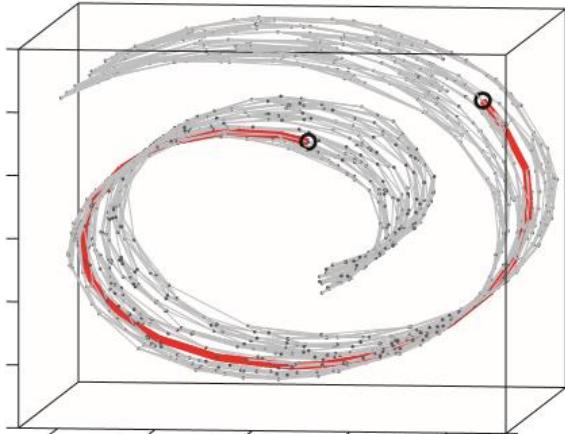
CPSC 340: Machine Learning and Data Mining

Deep Learning

Fall 2019

Last Time: Multi-Dimensional Scaling

- Modern multi-dimensional scaling (MDS) methods:
 - ISOMAP uses geodesic distance in data manifold.
 - T-SNE tends to reveal clusters and manifold structures.
 - Word2vec gives continuous alternative to bag of words.



Word2Vec

- Subtracting word vectors to find related vectors.

Table 8: Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Table 8 shows words that follow various relationships. We follow the approach described above: the relationship is defined by subtracting two word vectors, and the result is added to another word. Thus for example, $\text{Paris} - \text{France} + \text{Italy} = \text{Rome}$. As it can be seen, accuracy is quite good, although

- Word vectors for 157 languages [here](#).

End of Part 4: Key Concepts

- We discussed **linear latent-factor models**:

$$\begin{aligned} f(W, z) &= \sum_{i=1}^n \sum_{j=1}^k (\langle w_j^T z_i \rangle - x_{ij})^2 \\ &= \sum_{i=1}^n \|W^T z_i - x_i\|^2 \\ &= \|Z^T W - X\|_F^2 \end{aligned}$$

- Represent 'X' as linear combination of **latent factors 'w_c'**.
 - **Latent features 'z_i'** give a lower-dimensional version of each 'x_i'.
 - When k=1, finds **direction that minimizes squared orthogonal distance**.
- Applications:
 - Outlier detection, dimensionality reduction, data compression, features for linear models, visualization, factor discovery, filling in missing entries.

End of Part 4: Key Concepts

- We discussed linear latent-factor models:

$$f(W, z) = \sum_{i=1}^n \sum_{j=1}^d (\langle w_j^T z_i \rangle - x_{ij})^2$$

- Principal component analysis (PCA):

- Often uses orthogonal factors and fits them sequentially (via SVD).

- Non-negative matrix factorization:

- Uses non-negative factors giving sparsity.
 - Can be minimized with projected gradient.

- Many variations are possible:

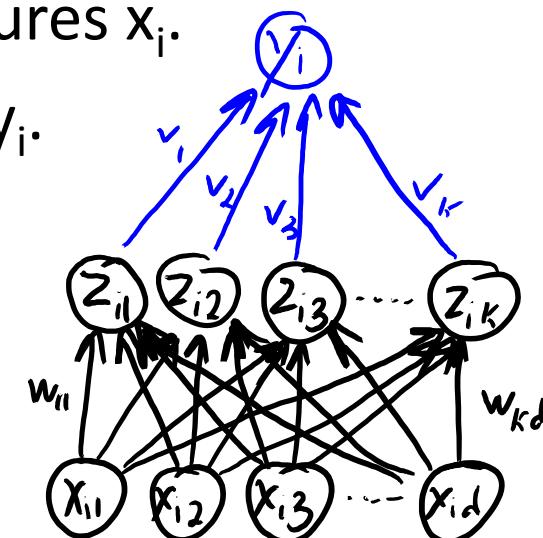
- Different regularizers (sparse coding) or loss functions (robust/binary PCA).
 - Missing values (recommender systems) or change of basis (kernel PCA).

End of Part 4: Key Concepts

- We discussed multi-dimensional scaling (MDS):
 - Non-parametric method for high-dimensional data visualization.
 - Tries to match distance/similarity in high-/low-dimensions.
 - “Gradient descent on scatterplot points”.
- Main challenge in MDS methods is “crowding” effect:
 - Methods focus on large distances and lose local structure.
- Common solutions:
 - Sammon mapping: use weighted cost function.
 - ISOMAP: approximate geodesic distance using via shortest paths in graph.
 - T-SNE: give up on large distances and focus on neighbour distances.
- Word2vec is a recent MDS method giving better “word features”.

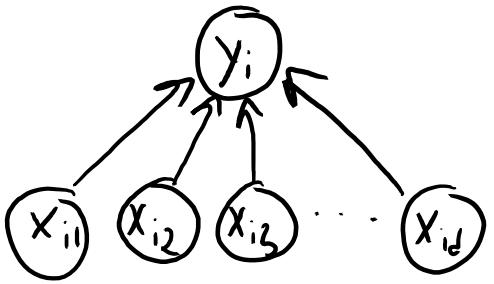
Supervised Learning Roadmap

- 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 ‘v’ 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 ‘v’ based on change of basis z_i and target y_i .
- Part 5: Neural Networks.
 - Jointly learn ‘W’ and ‘v’ based on x_i and y_i .
 - Learn basis z_i that is good for supervised learning.

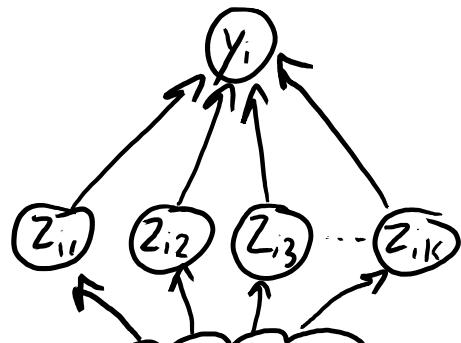


A Graphical Summary of CPSC 340 Parts 1-5

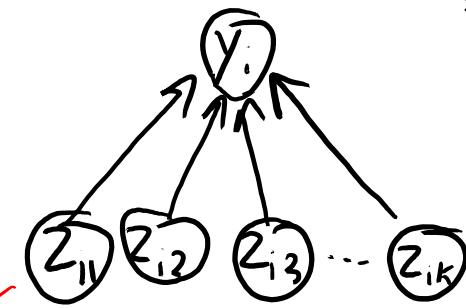
Part 1: "I have features x_i "



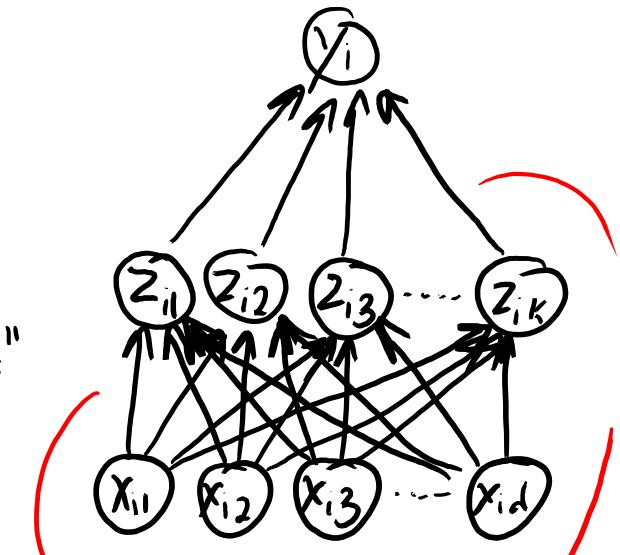
Part 3: Change of basis



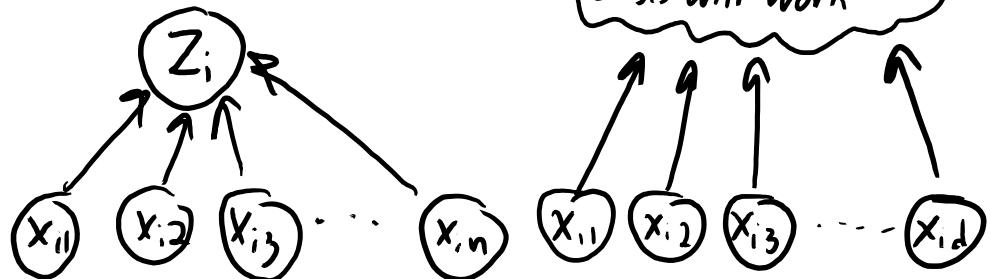
Part 4: basis from latent-factor model



Part 5: Neural networks



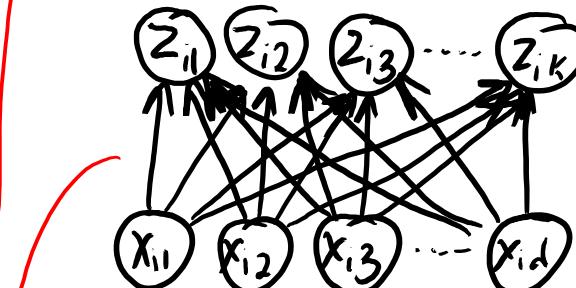
Part 2: "What is the group of x_i ?"



Trained separately

"What are the 'parts' of x_i ?"

"PCA will give me good features"



Learn features and classifier at the same time.

Notation for Neural Networks

We have our usual supervised learning notation:

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

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

We have our latent features:

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

$n \times K$

We have two sets of parameters:

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_K \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}$$

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

The diagram illustrates a neural network layer. On the left, inputs X are shown as a vector of n features ($n \times d$). In the middle, latent features Z are shown as a vector of n features ($n \times K$). On the right, two sets of parameters are shown: V (a vector of K latent feature vectors, $K \times 1$) and W (a matrix of K weight vectors, $K \times d$). Red curly braces group V and W . Red arrows point from these groups to a neural network diagram. The network diagram shows X as input nodes at the bottom, Z as hidden nodes in the middle, and V and W as parameters. The output node y_i is connected to all nodes in Z via V , and each node in Z is connected to all nodes in X via W .

Linear-Linear Model

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

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

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

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

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

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

- But this is just a linear model:

$$y_i = v^T z_i = v^T (Wx_i) = (\underbrace{v^T W}_{\text{some vector } 'w'}) x_i = w^T x_i$$

Introducing Non-Linearity

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

$$z_i = Wx_i \quad y_i = v^T h(z_i)$$

- Here the function ‘ h ’ transforms ‘ k ’ inputs to ‘ k ’ outputs.
- Common choice for ‘ h ’: applying sigmoid function element-wise:

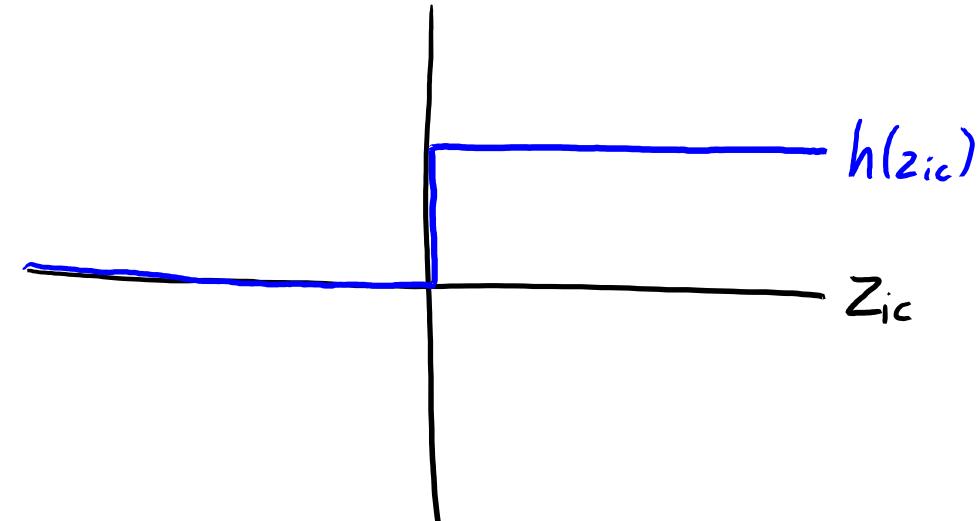
$$h(z_{ic}) = \frac{1}{1 + \exp(-z_{ic})}$$

- So this takes the z_{ic} in $(-\infty, \infty)$ and maps it to $(0, 1)$.
- This is called a “multi-layer perceptron” or a “neural network”.

Why Sigmoid?

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

$$h(z_{ic}) = \begin{cases} 1 & \text{if } z_{ic} \geq 0 \\ 0 & \text{if } z_{ic} < 0 \end{cases}$$



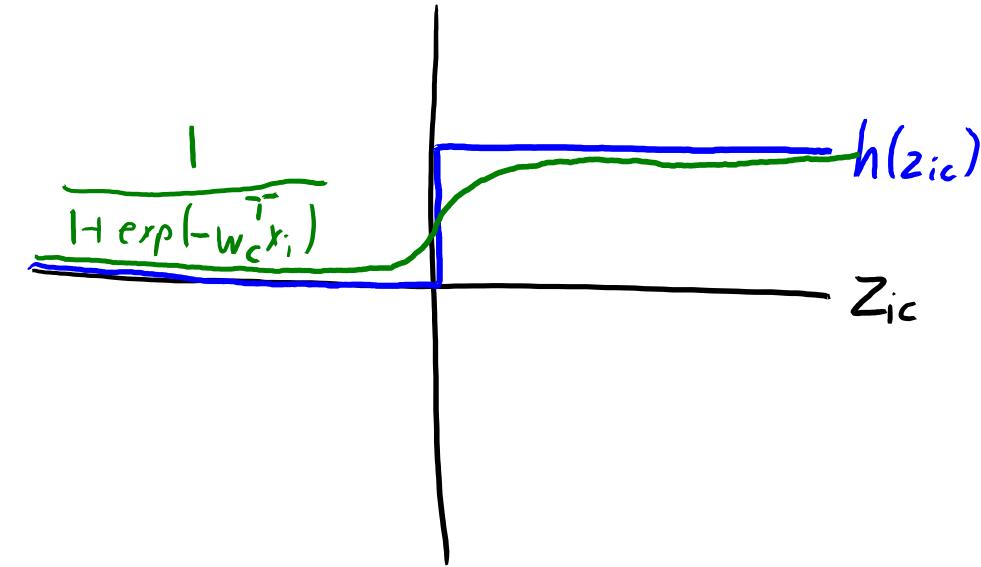
- Each $h(z_i)$ can be viewed as binary feature.
 - “You either have this ‘part’ or you don’t have it.”
- We can make 2^k objects by all the possible “part combinations”.

Motivation: Pixels vs. Parts												
• We could represent other digits as different combinations of “parts”:												
3	= 1	-	+ 1	1	+ 1	-	+ 1	1	+ 0	1	+ 0	2
5	= 1	-	+ 0	1	+ 1	-	+ 1	1	+ 0	1	+ 1	3
8	= 1	-	+ 1	1	+ 1	-	+ 1	1	+ 1	1	+ 1	4

Why Sigmoid?

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

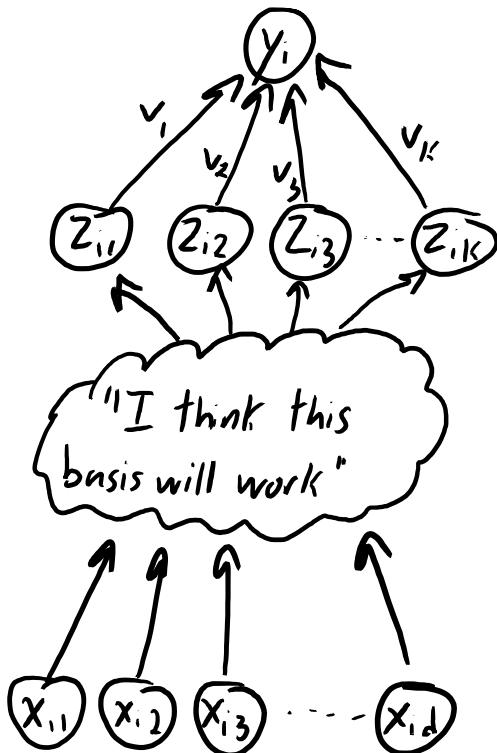
$$h(z_{ic}) = \begin{cases} 1 & \text{if } z_{ic} \geq 0 \\ 0 & \text{if } z_{ic} < 0 \end{cases}$$



- Each $h(z_i)$ can be viewed as binary feature.
 - “You either have this ‘part’ or you don’t have it.”
- We can make 2^k objects by all the possible “part combinations”.
- But this is hard to optimize (**non-differentiable/discontinuous**).
- Sigmoid is a smooth approximation to these binary features.

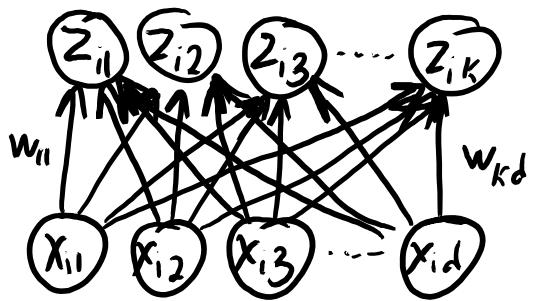
Supervised Learning Roadmap

Hand-engineered features

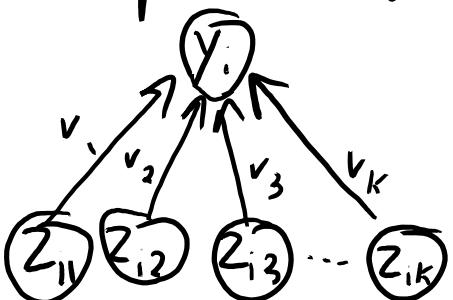


Requires domain knowledge
and can be time-consuming

Learn a latent-factor model:

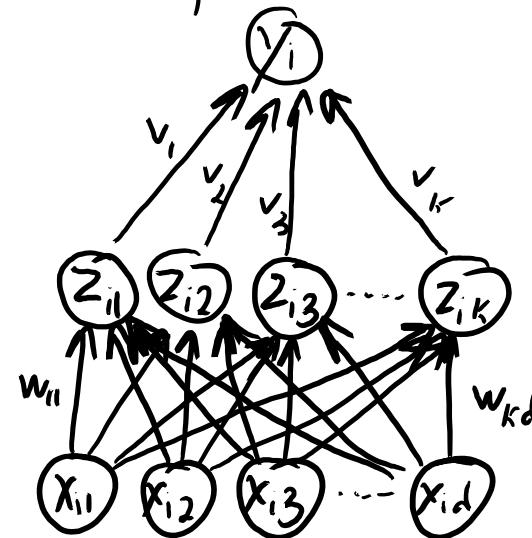


Use latent features
in supervised model:



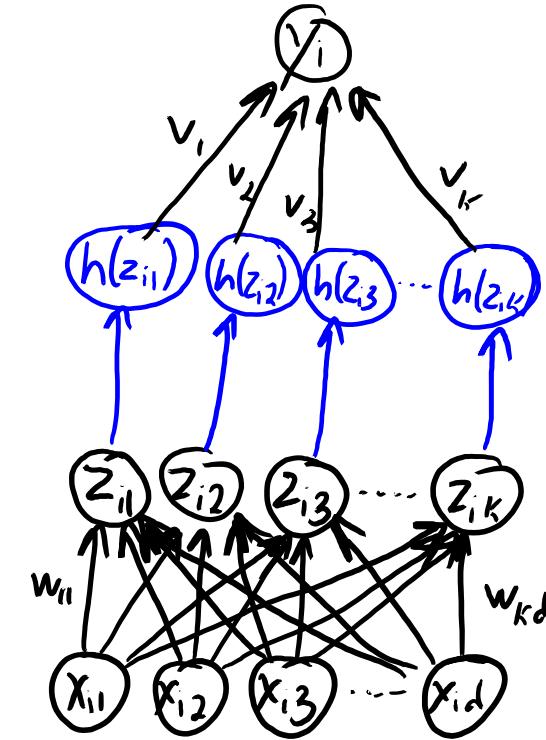
Good representation of
 x_i might be bad for predicting y_i

Learn ' w ' and ' W '
together.



But still gives a
linear model.

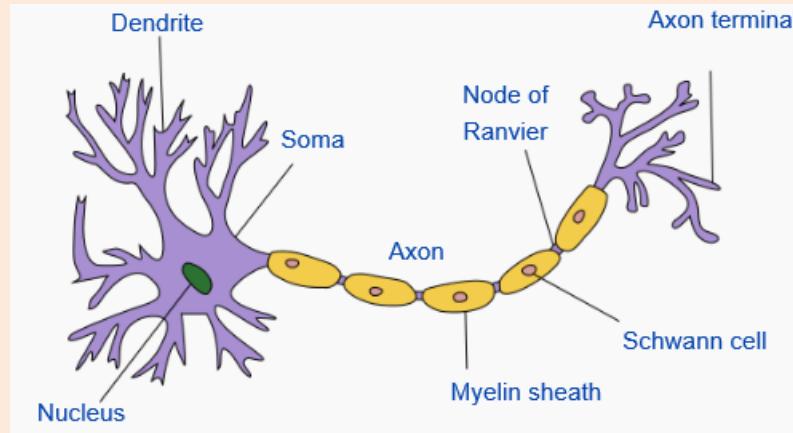
Neural network.



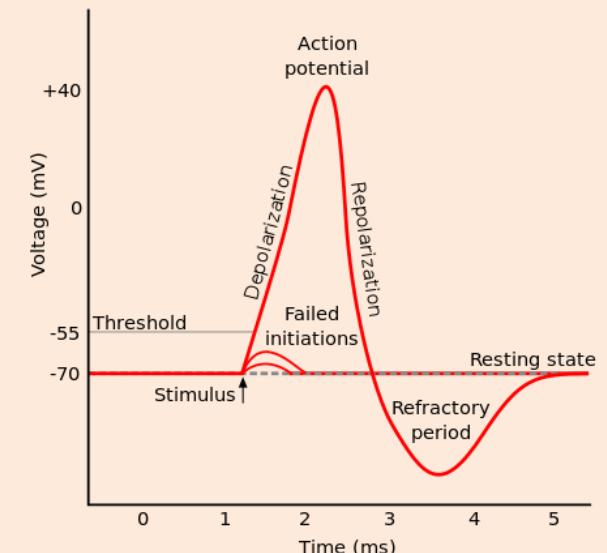
Extra non-linear
transformation ' h '

Why “Neural Network”?

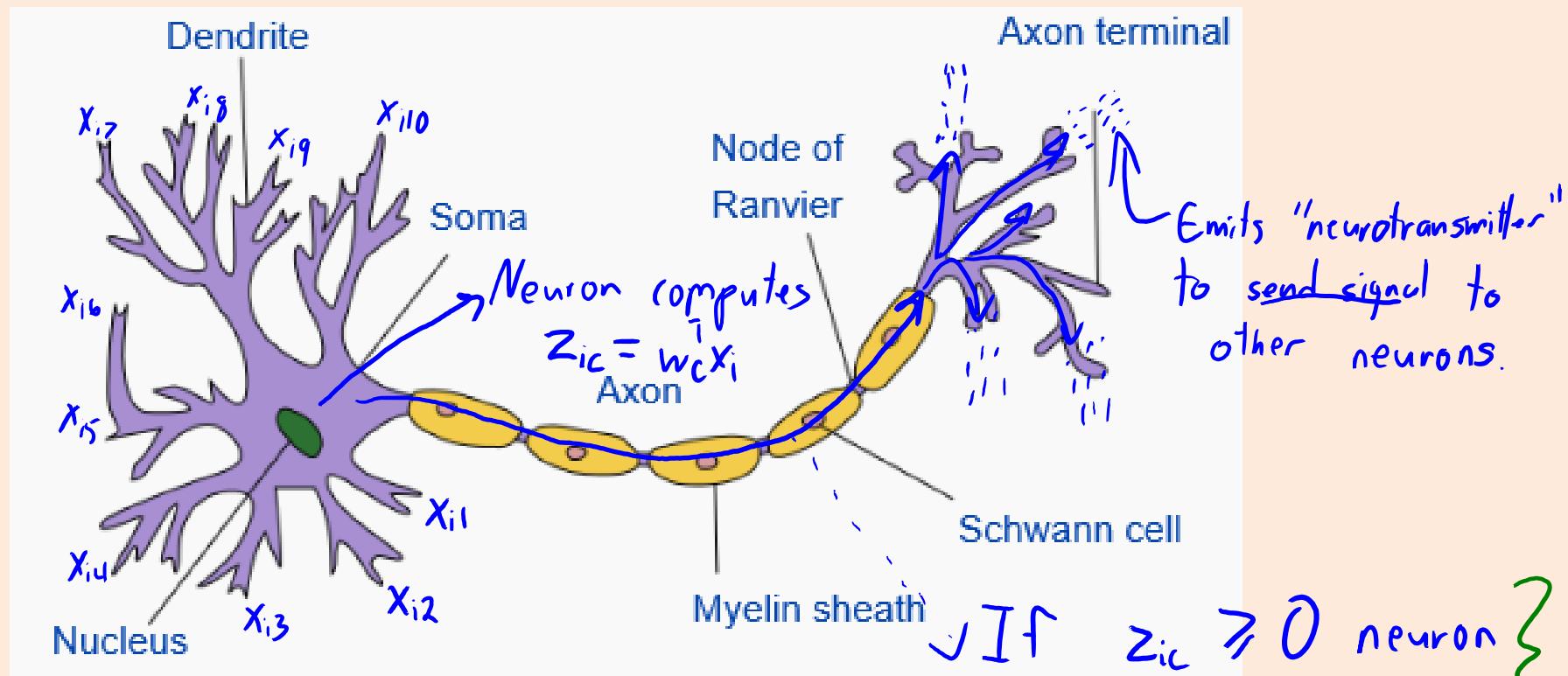
- Cartoon of “typical” neuron:



- 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):

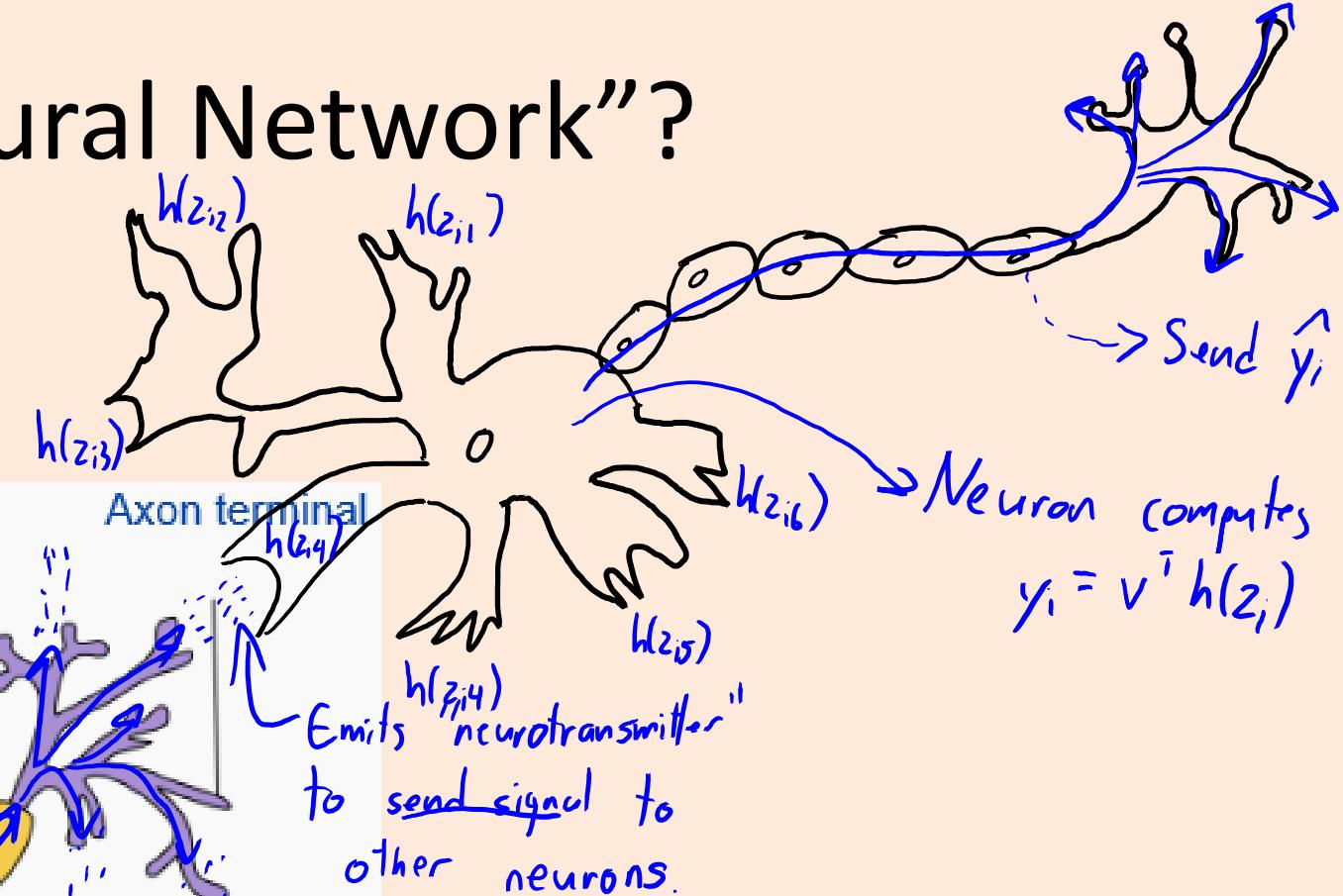
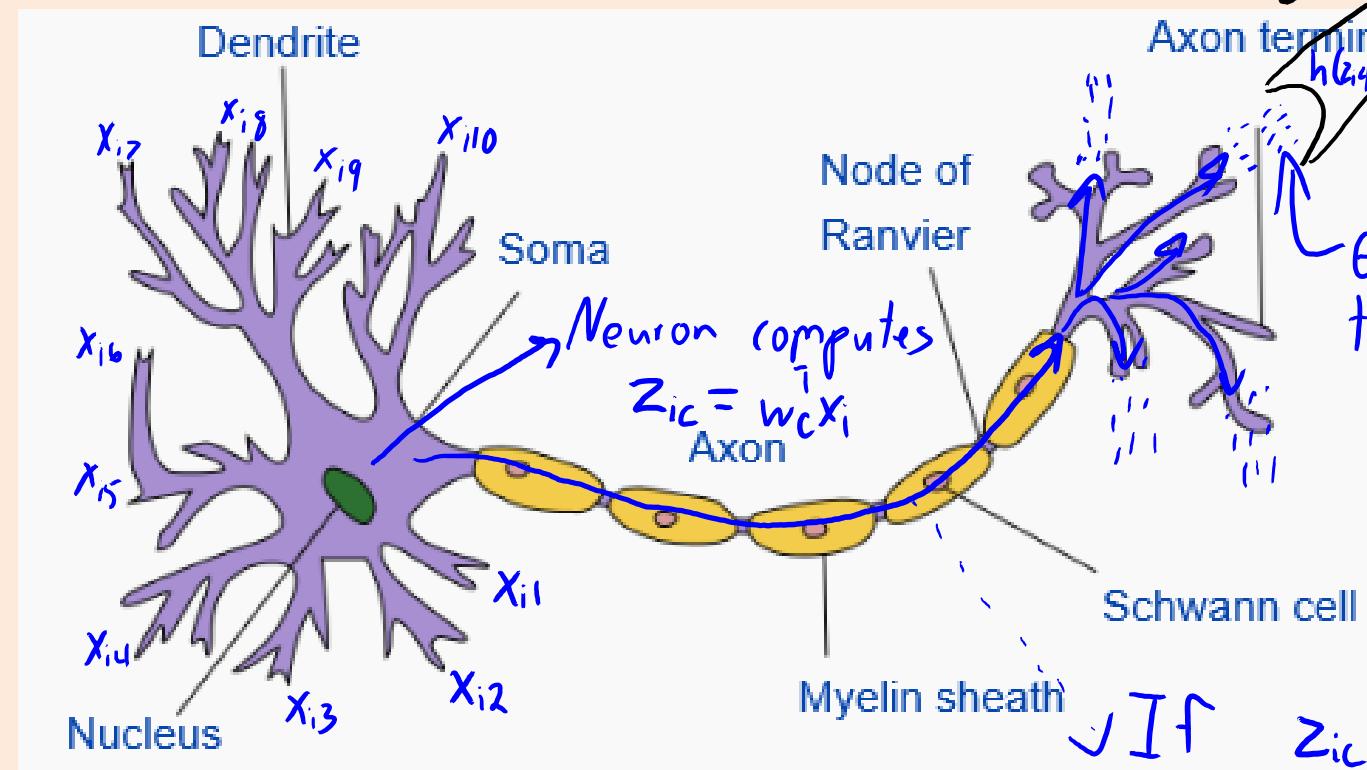


Why “Neural Network”?



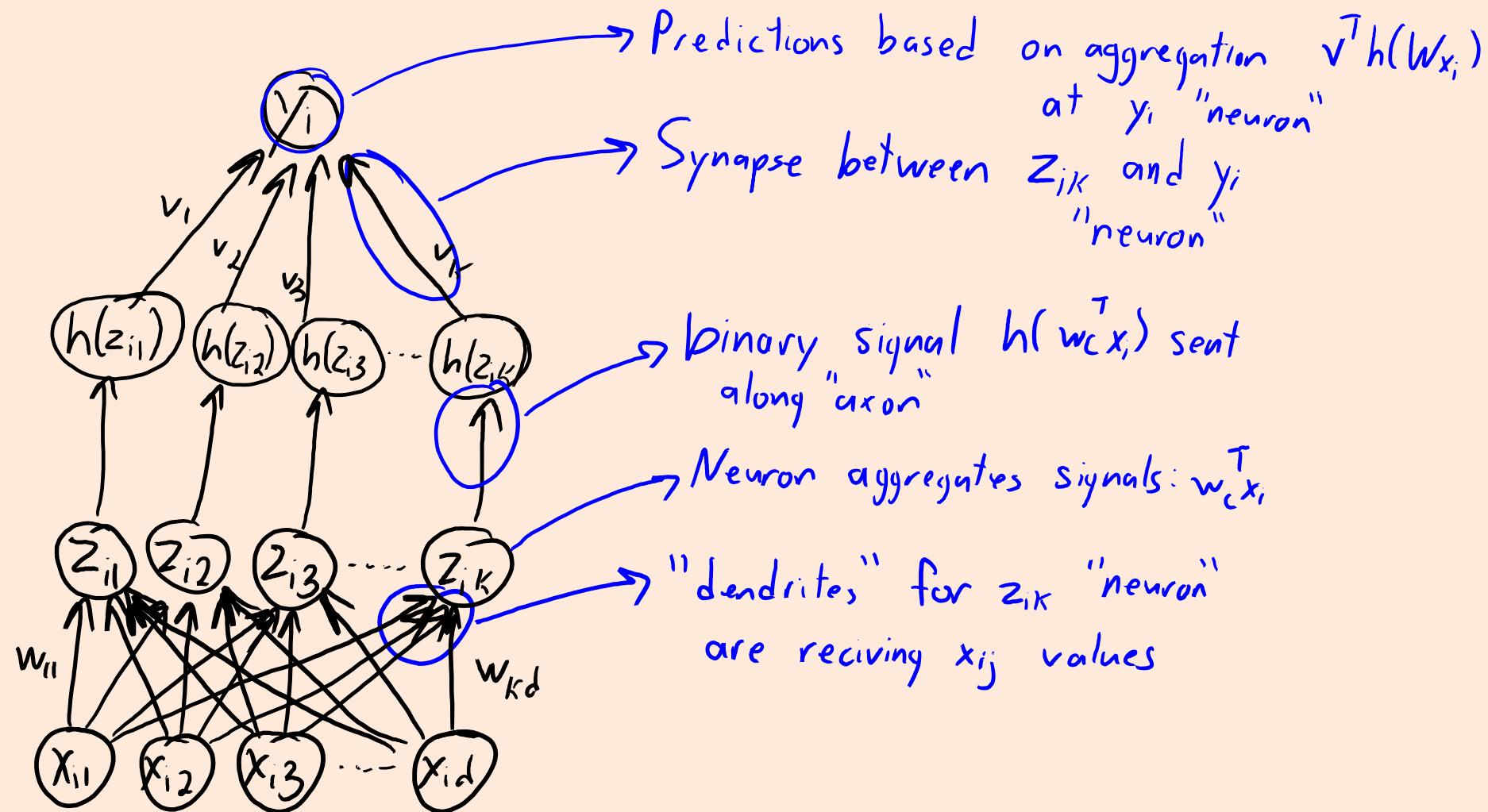
We approximate binary signal with $\frac{1}{1 + \exp(-Z_{ic})}$

Why “Neural Network”?



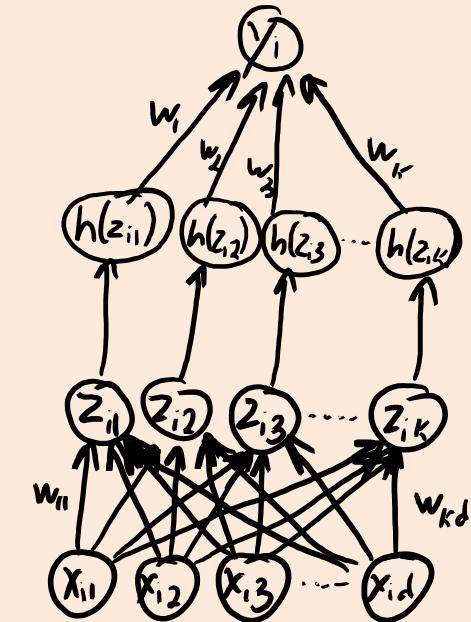
We approximate binary signal with $\frac{1}{1 + \exp(-z_{ic})}$

Why “Neural Network”?



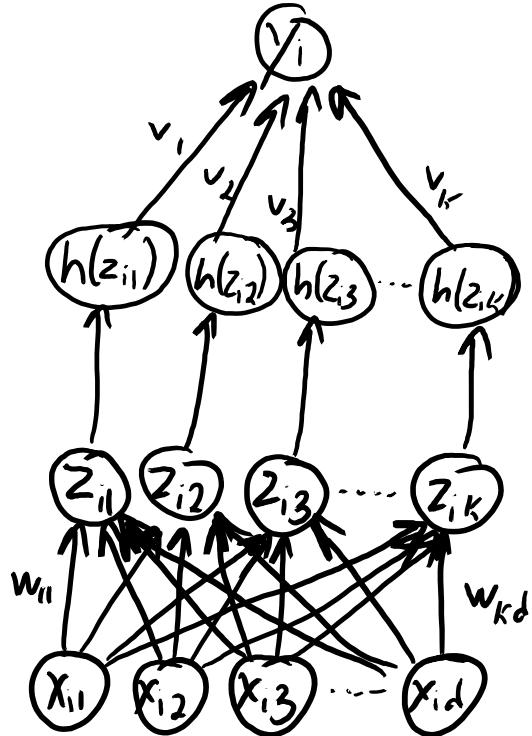
“Artificial” Neural Nets vs. “Real” Networks Nets

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



Deep Learning

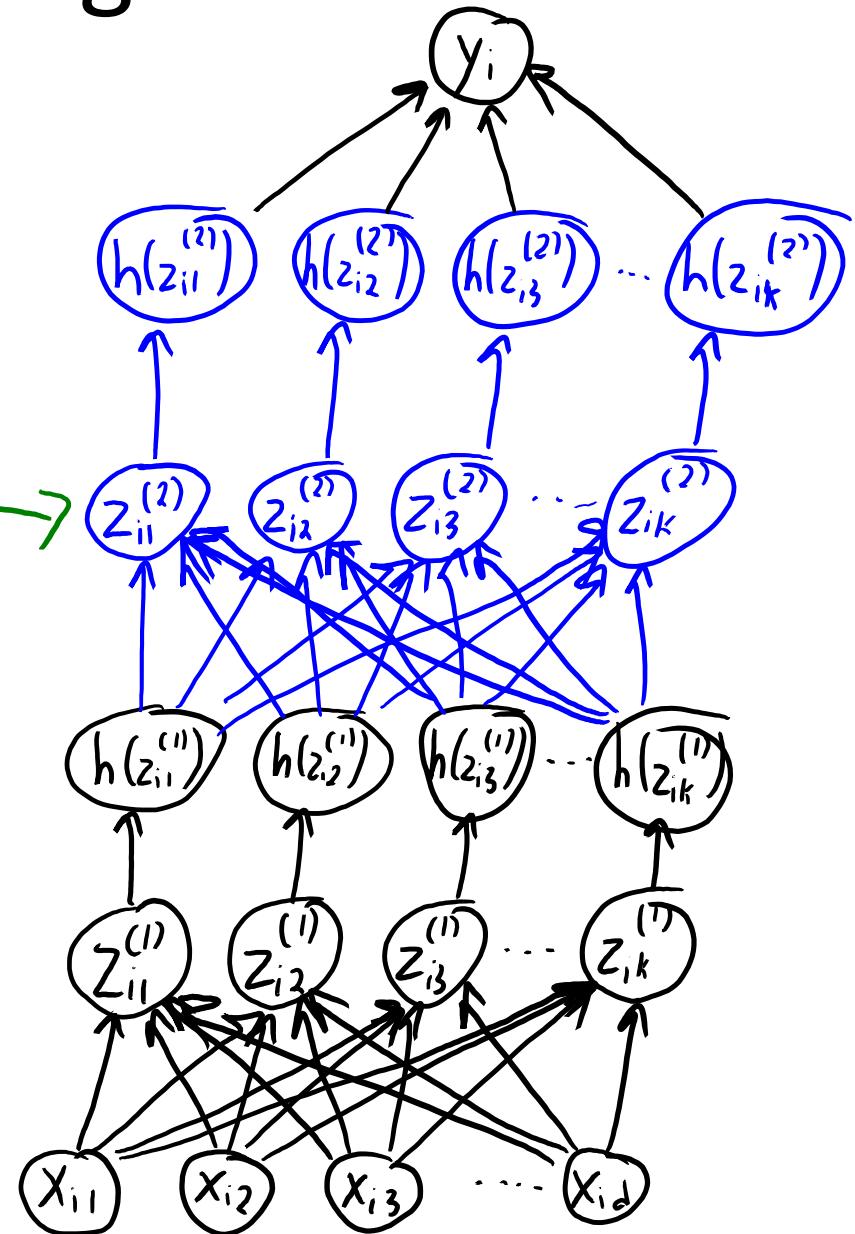
Neural network:



Deep learning:

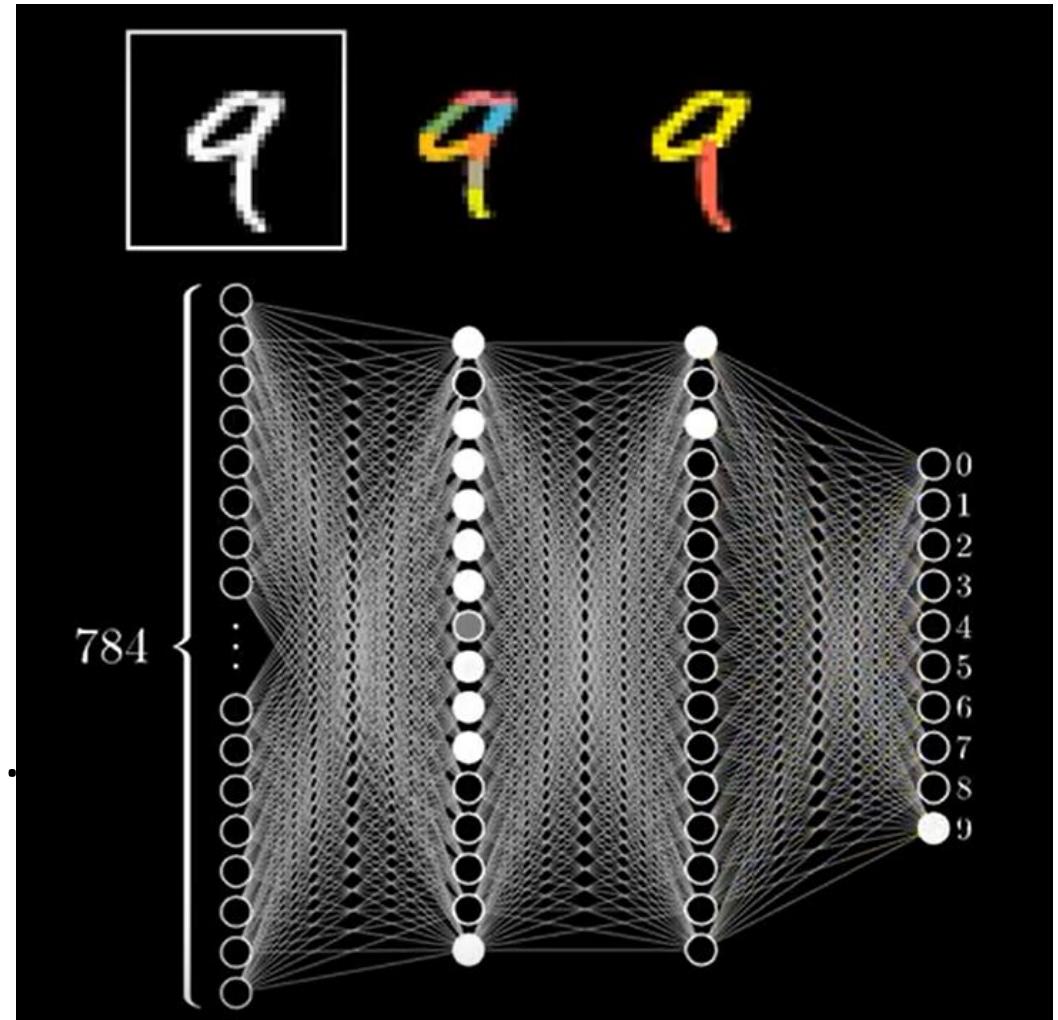
Second "layer" of latent features

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



“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>



Deep Learning

Linear model:

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

Neural network with 1 hidden layer:

$$\hat{y}_i = v^\top h(Wx_i)$$

Neural network with 2 hidden layers:

$$\hat{y}_i = v^\top h(W^{(2)} h(W^{(1)} x_i))$$

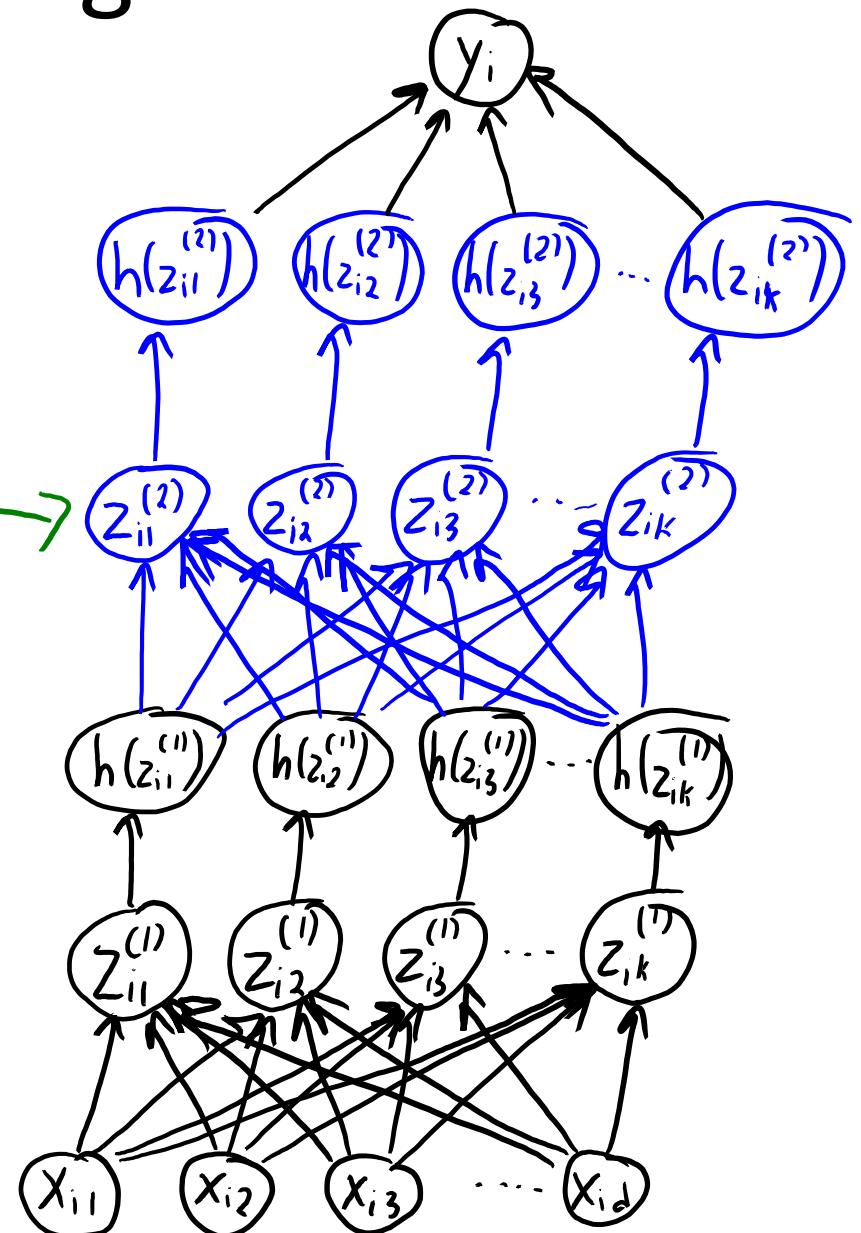
Neural network with 3 hidden layers

$$\hat{y}_i = v^\top h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i)))$$

Deep learning:

Second "layer" of latent features

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



Deep Learning

- For 4 layers, we could write the prediction as:

$$\hat{y}_i = v^T h(w^{(4)} h(w^{(3)} h(w^{(2)} h(w^{(1)} x_i))))$$

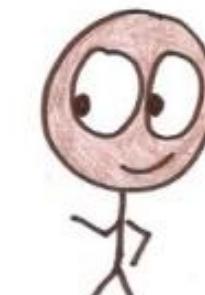
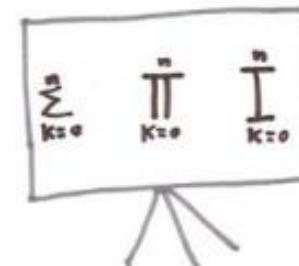
Symbol:

$$\prod_{k=0}^n f_k(t)$$

- For 'm' layers, we could use:

Meaning: $f_n \circ f_{n-1} \circ f_{n-2} \circ \dots \circ f_1 \circ f_0(t)$

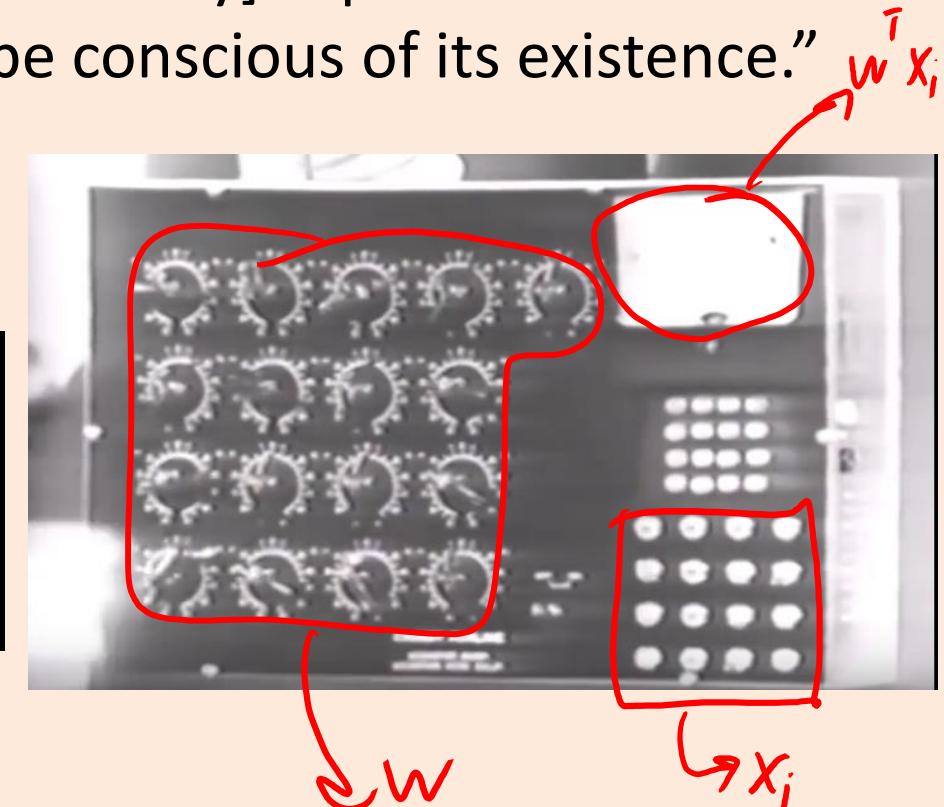
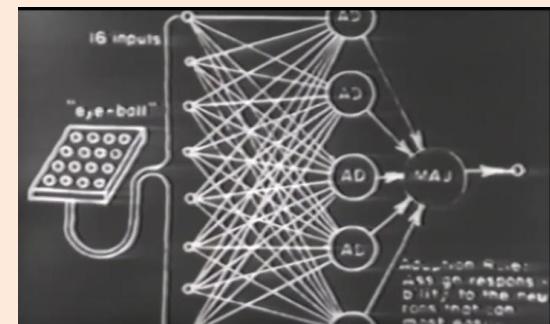
$$\hat{y}_i = v^T \left(\prod_{l=1}^m h(w^{(l)} x_i) \right)$$



- I knew something was missing!

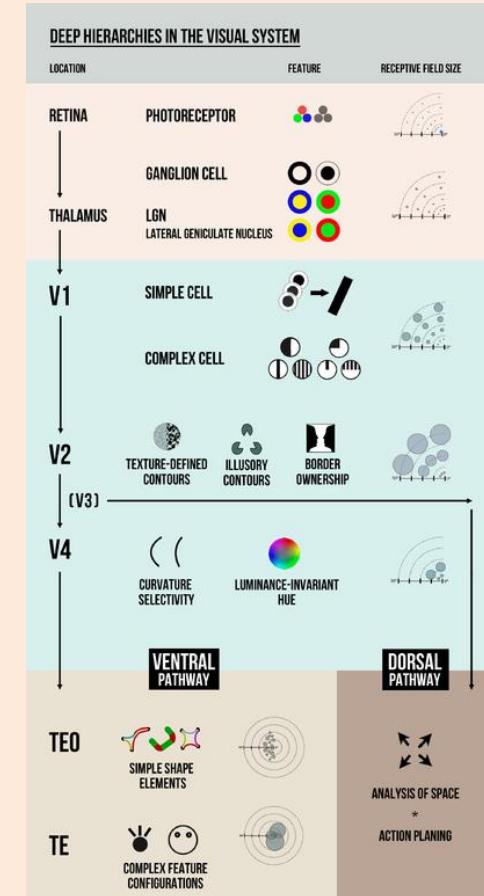
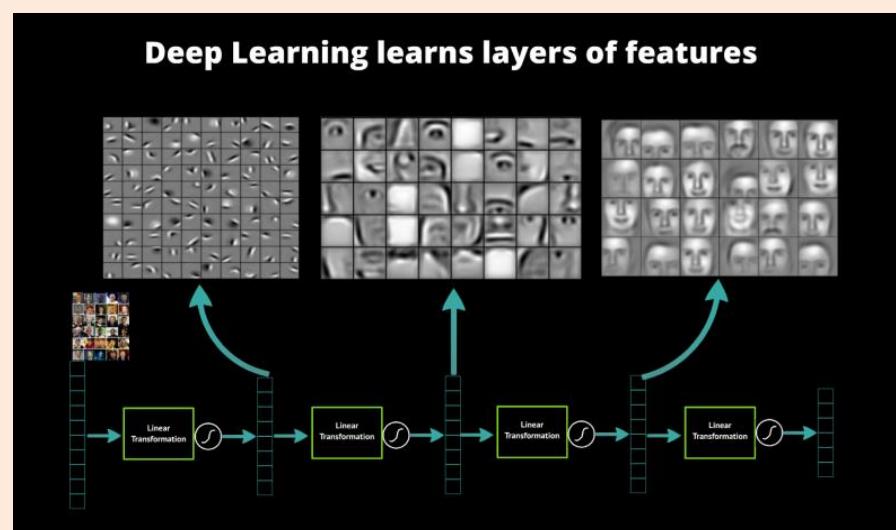
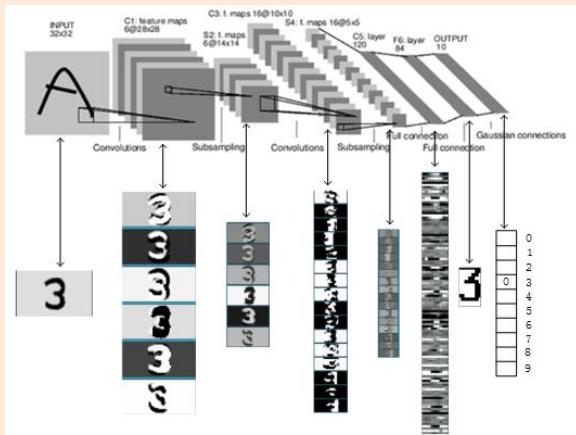
ML and Deep Learning History

- 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.**



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/>

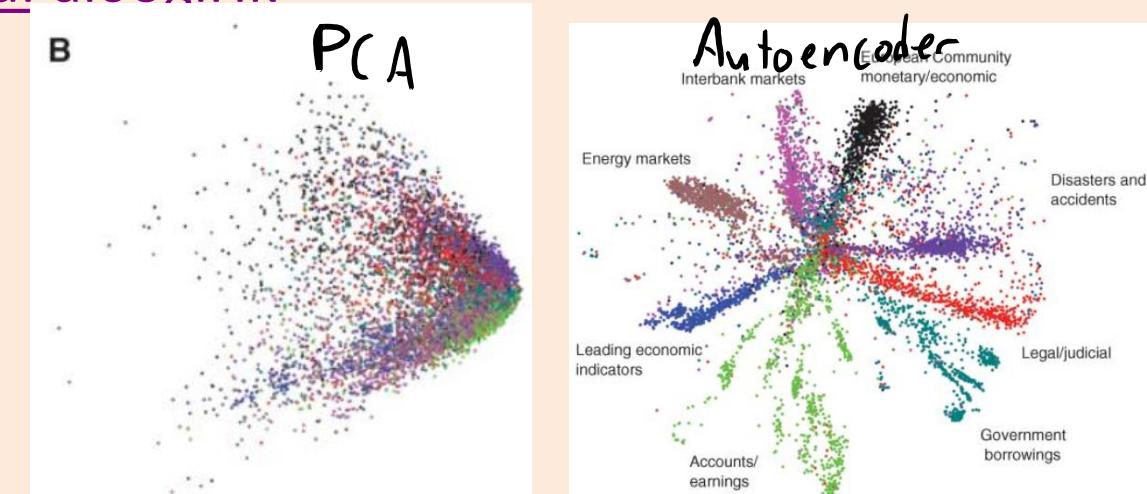
<http://blog.csdn.net/strint/article/details/44163869>

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.

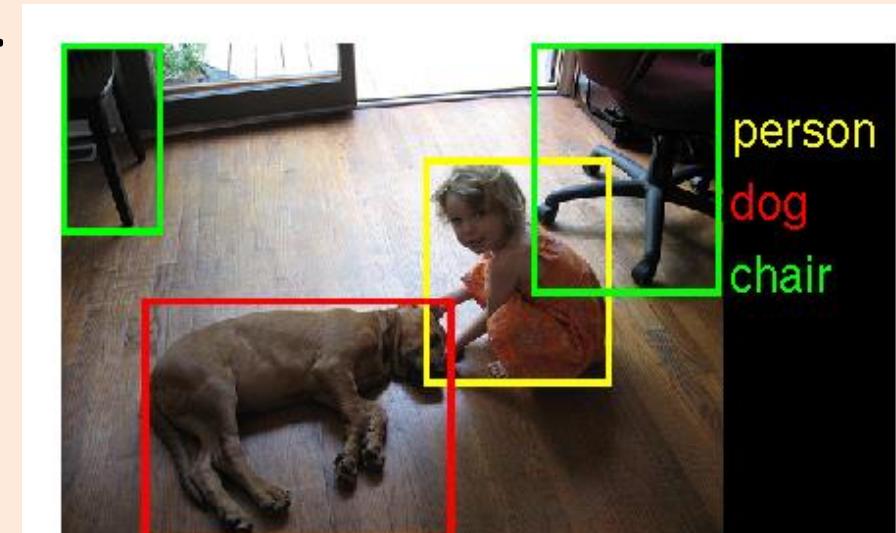
ML and Deep Learning History

- 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.

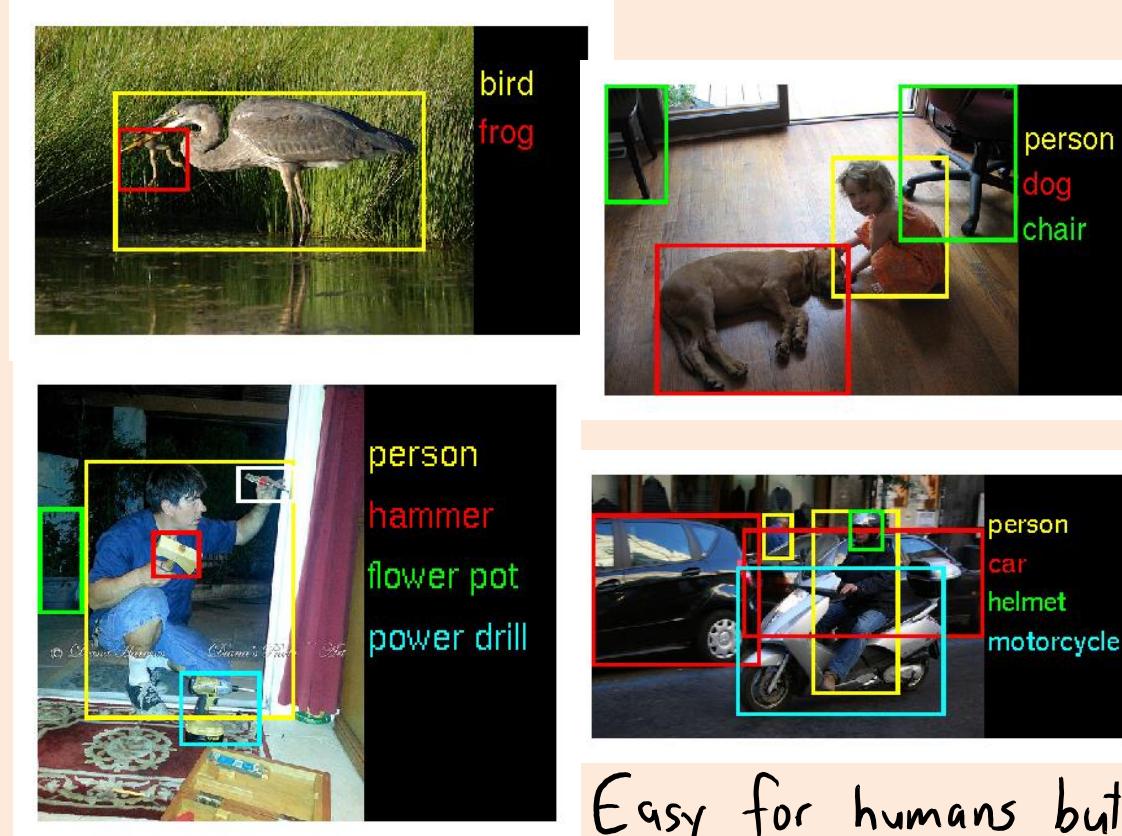


2010s: DEEP LEARNING!!!

- 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.

ImageNet Challenge

- Millions of labeled images, 1000 object classes.



Easy for humans but
hard for computers.

ImageNet Challenge

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

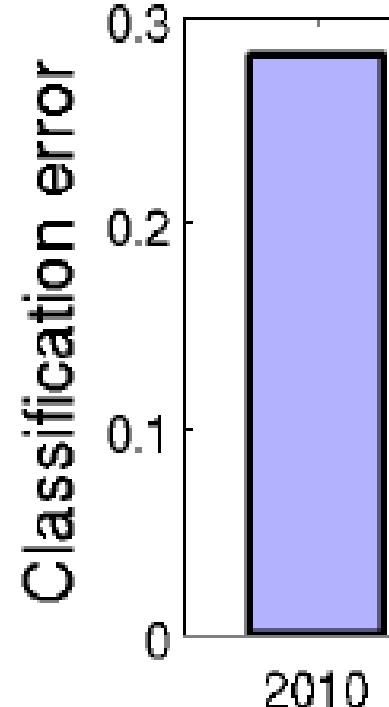


(a) Siberian husky



(b) Eskimo dog

Image classification



<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.

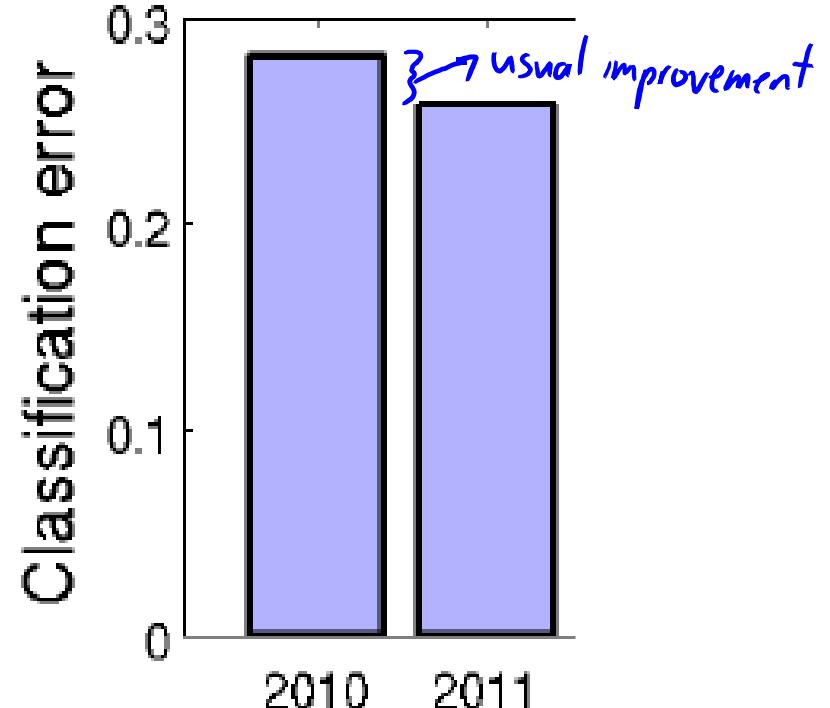


(a) Siberian husky



(b) Eskimo dog

Image classification



ImageNet Challenge

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

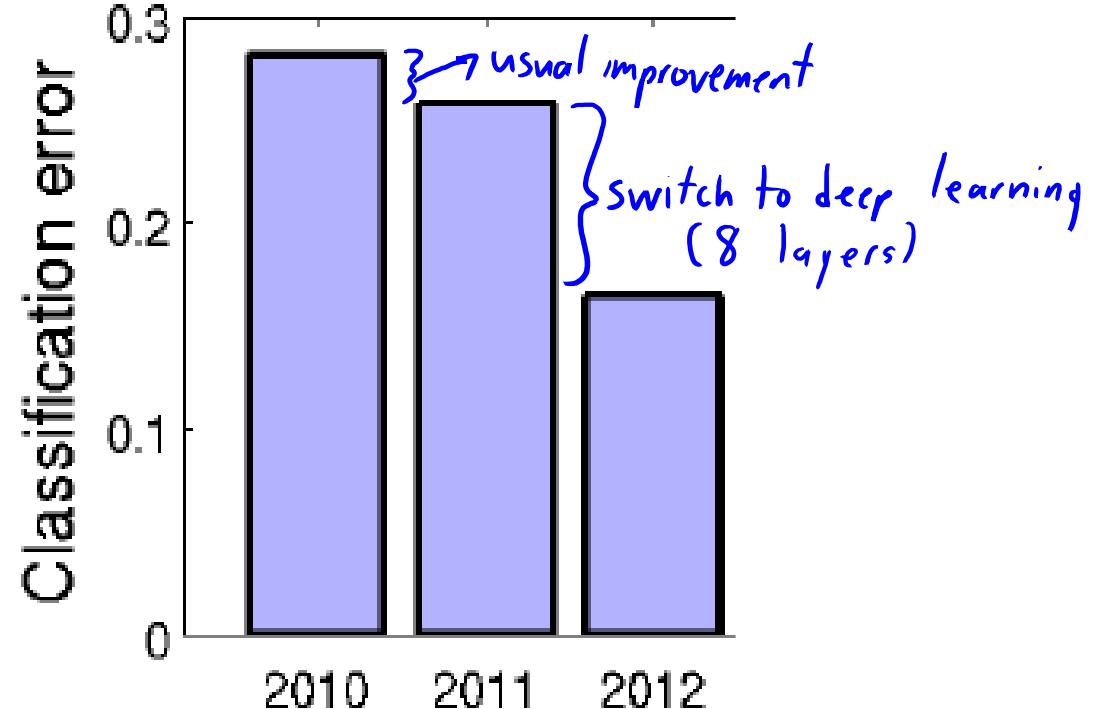


(a) Siberian husky



(b) Eskimo dog

Image classification

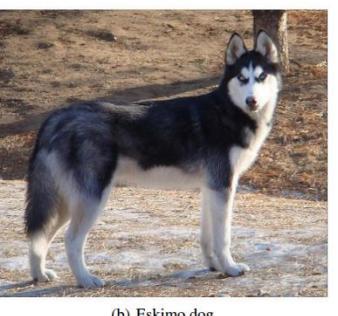


ImageNet Challenge

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

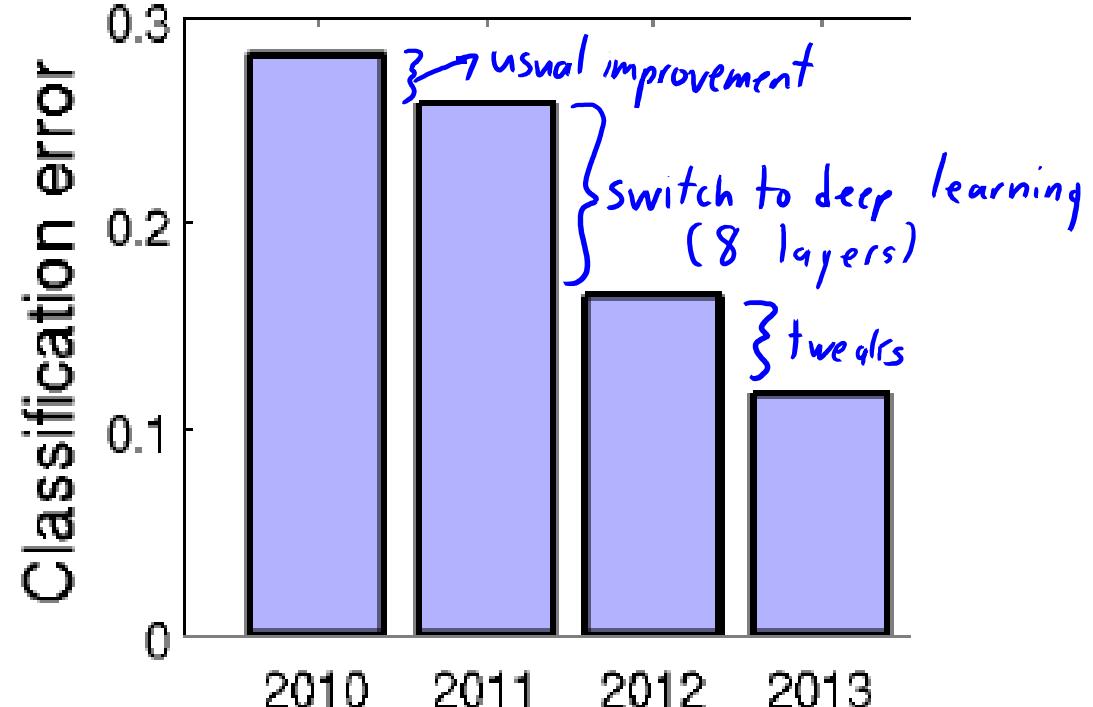


(a) Siberian husky



(b) Eskimo dog

Image classification



<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.

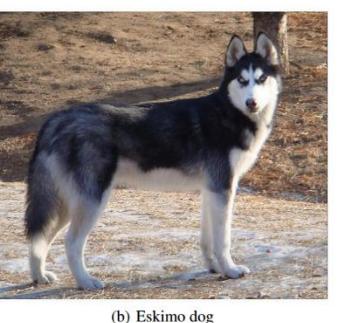
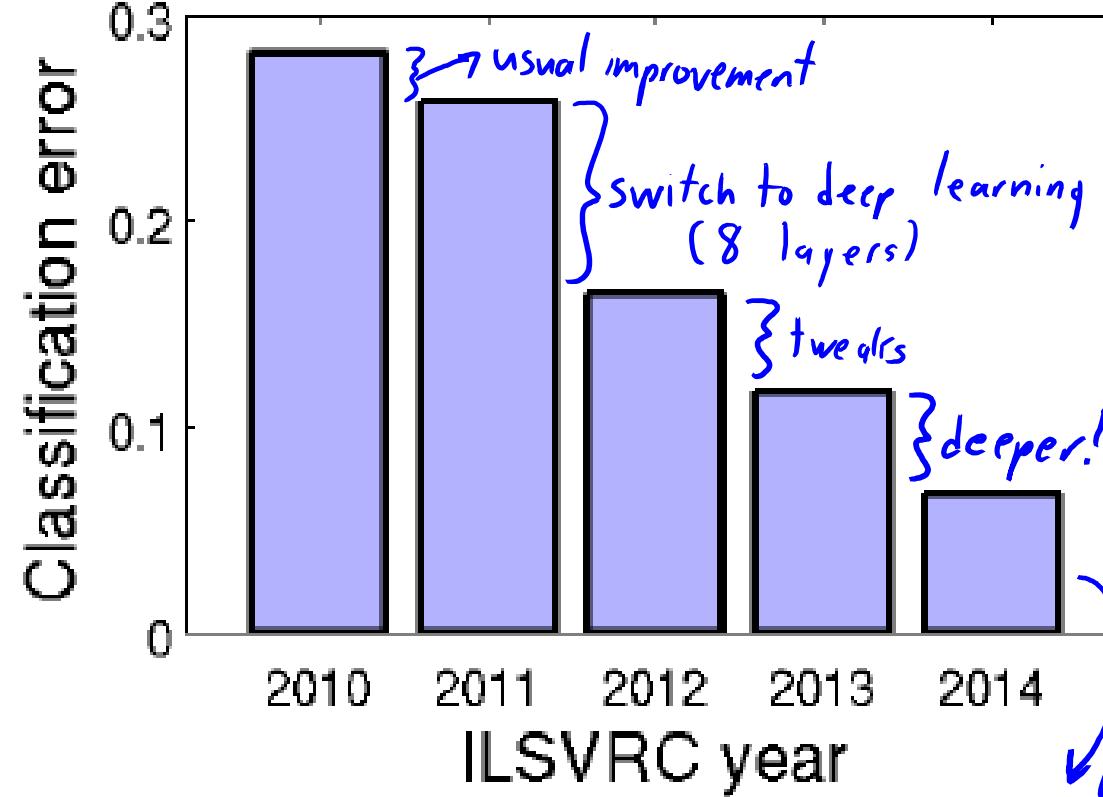
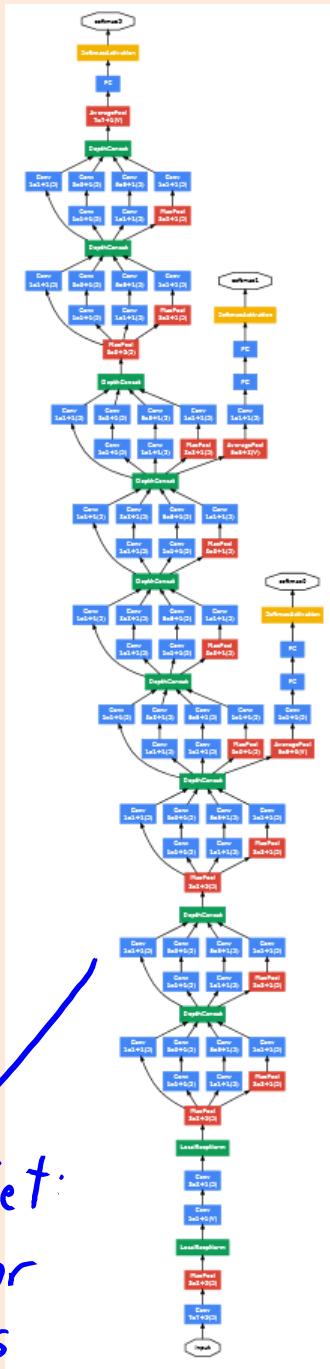


Image classification

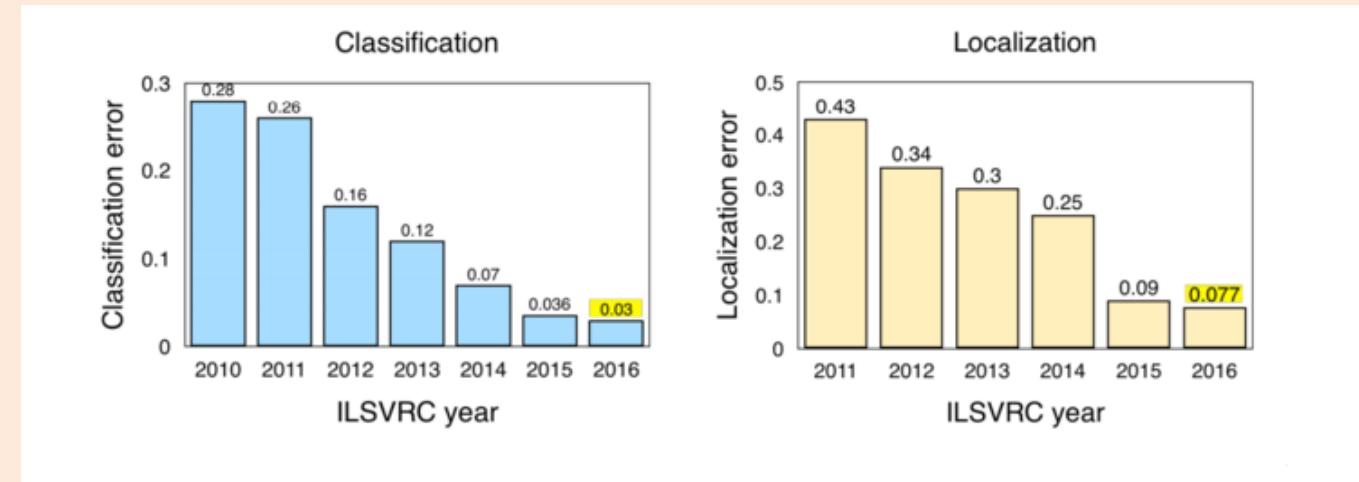


GoogLe Net:
6.7% error
22 layers



ImageNet Challenge

- Object detection task:
 - Single label per image.
 - Humans: ~5% error.
- 2015: Won by Microsoft Asia
 - 3.6% error.
 - 152 layers, “resnet” architecture.
 - 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.



Summary

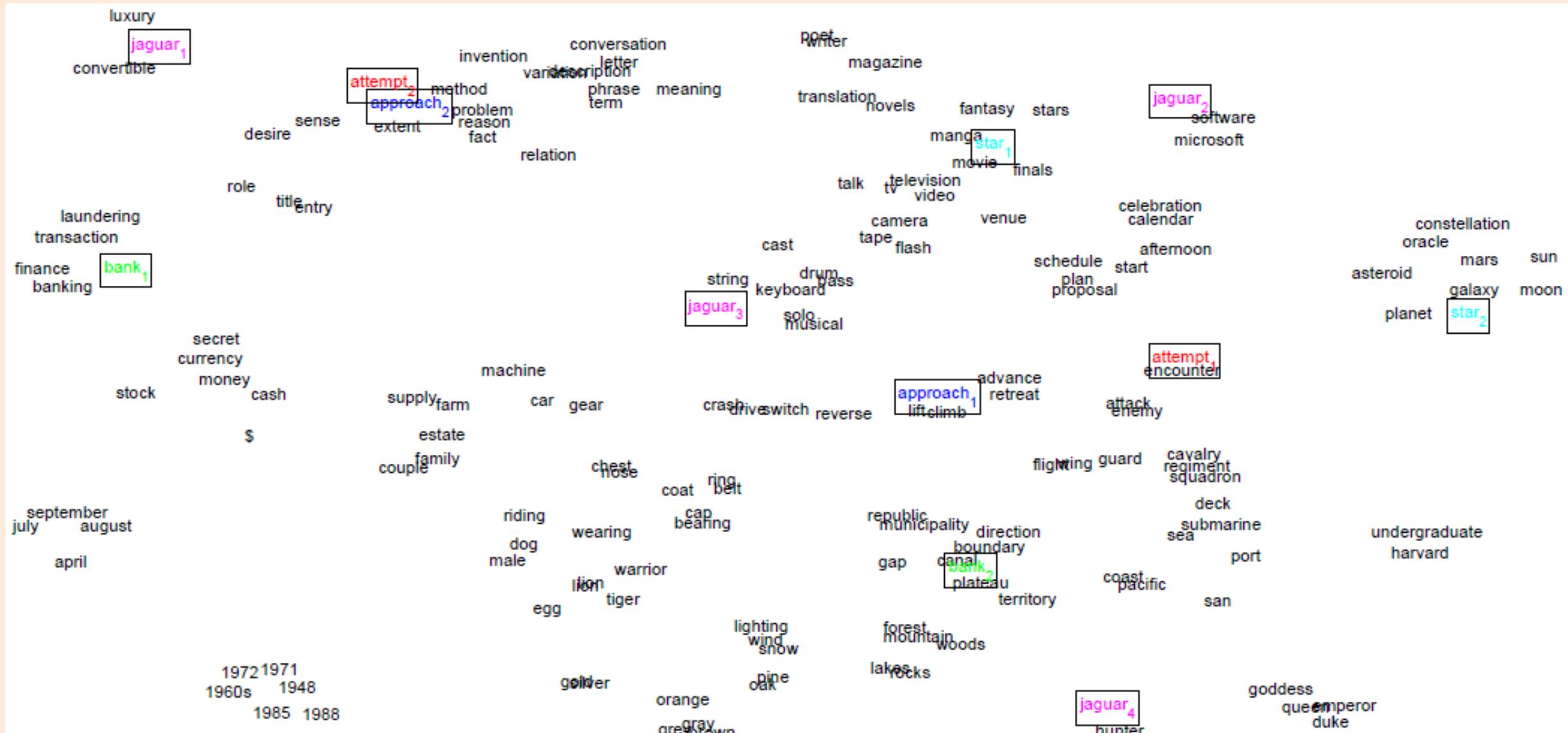
- Neural networks learn features z_i for supervised learning.
- Sigmoid function avoids degeneracy by introducing non-linearity.
- Biological motivation for (deep) neural networks.
- Deep learning considers neural networks with many hidden layers.
- Unprecedented performance on difficult pattern recognition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

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.

$$X_{j\text{distr}} = \begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \left. \right\} z_{j1} \\ \left. \right\} z_{j2} \\ \left. \right\} z_{j3}$$

Multiple Word Prototypes

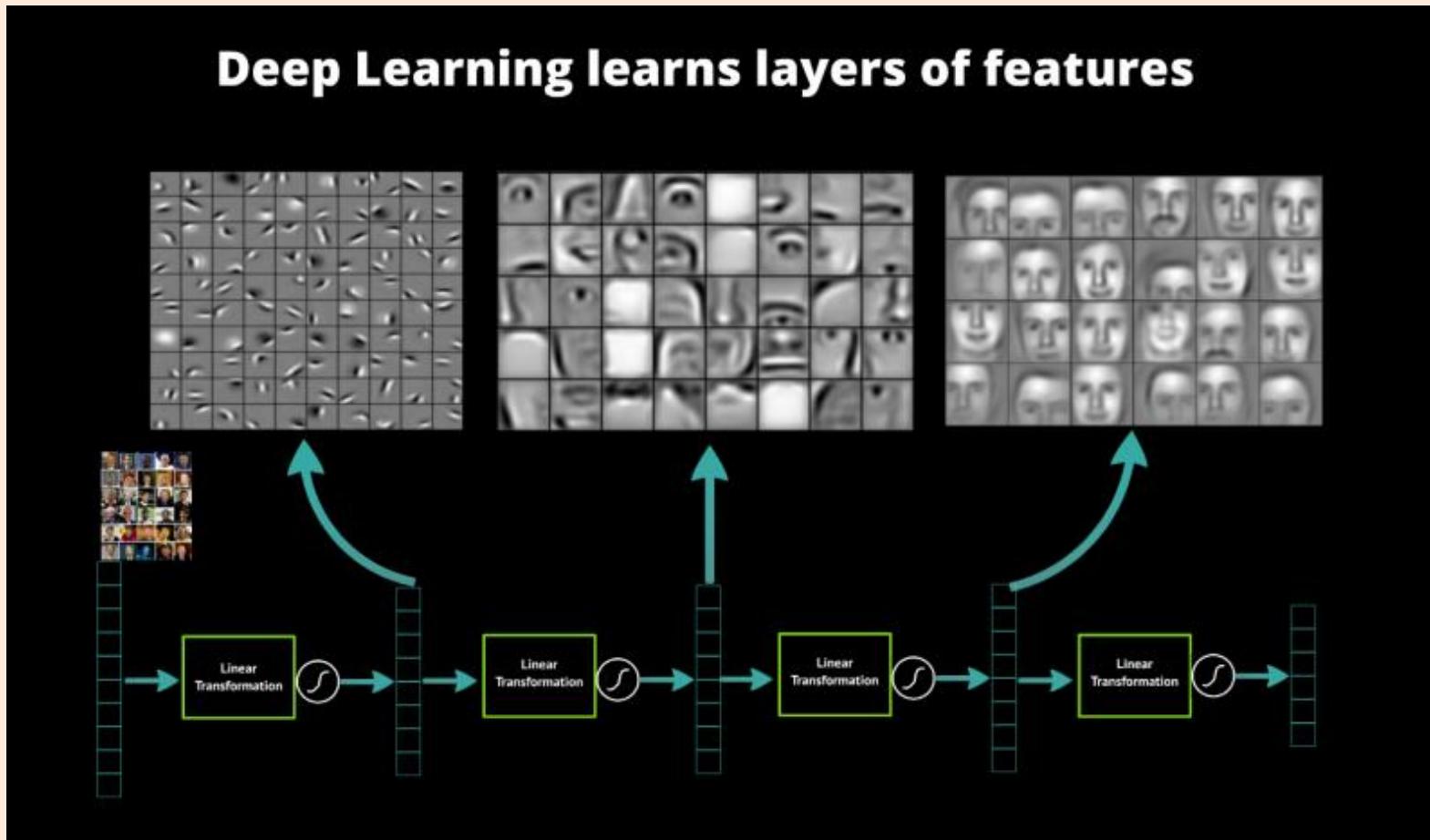


Why $z_i = Wx_i$?

- 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**.

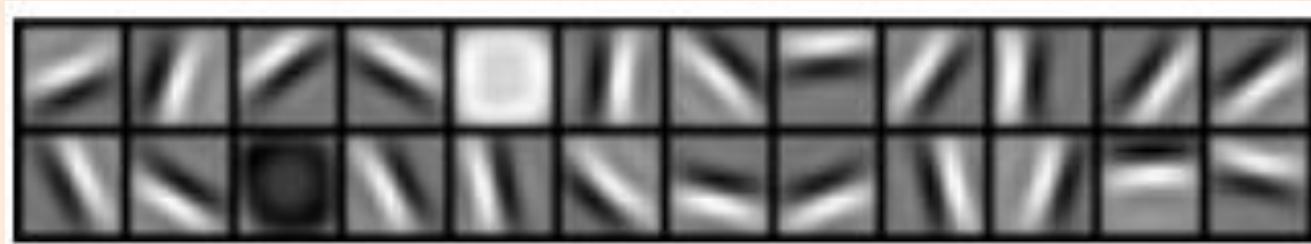
Cool Picture Motivation for Deep Learning

- Faces might be composed of different “parts”:



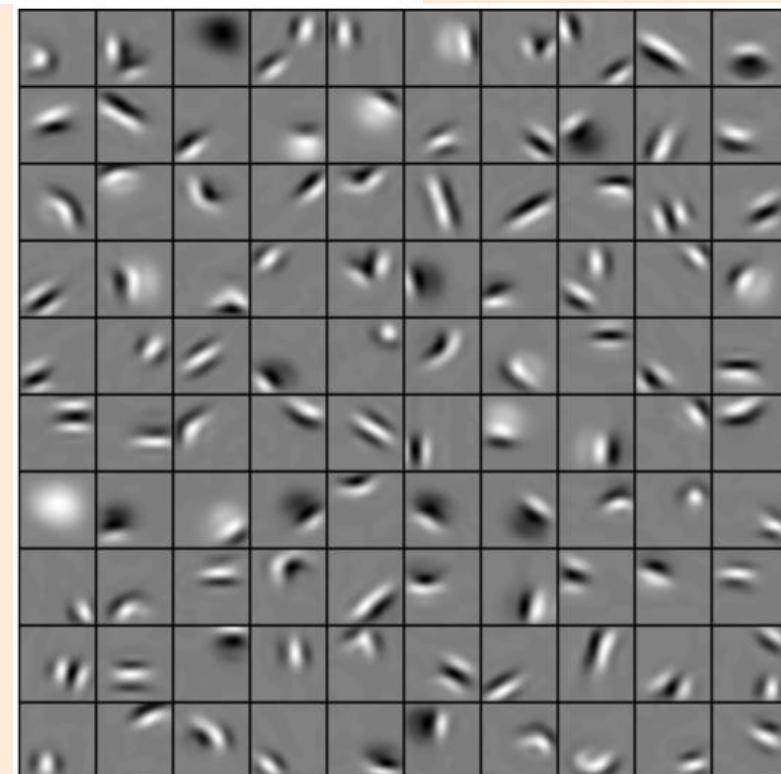
Cool Picture Motivation for Deep Learning

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



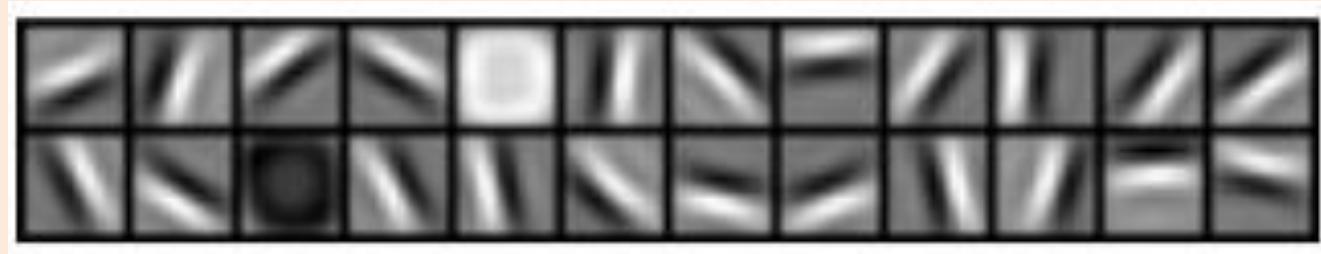
} "Gabor filters"

- 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

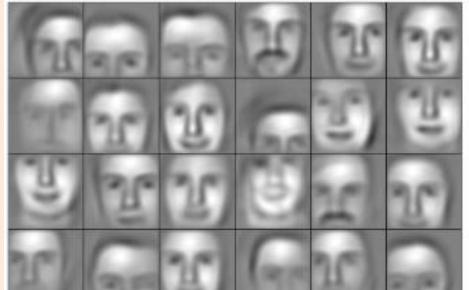
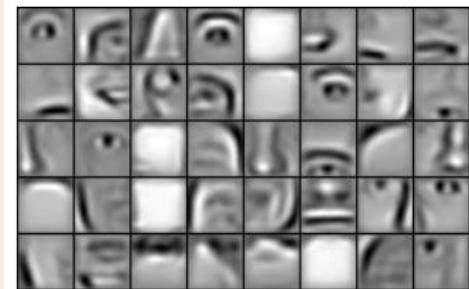
- First layer of z_i trained on 10 by 10 image patches:



} "Gabor filters"

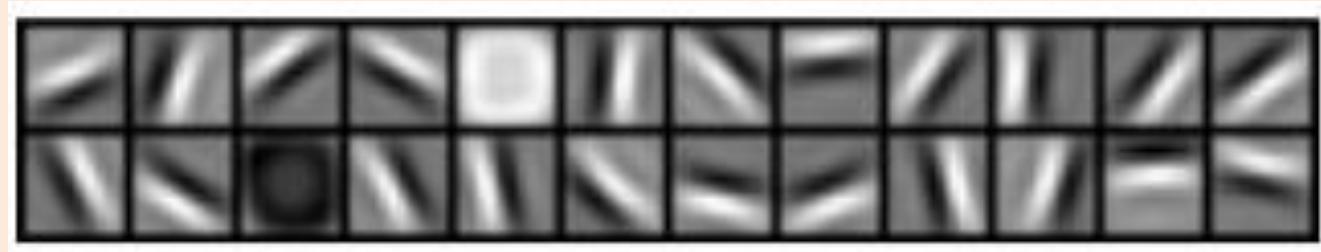
- Visualization of second and third layers trained on specific objects:

faces



Cool Picture Motivation for Deep Learning

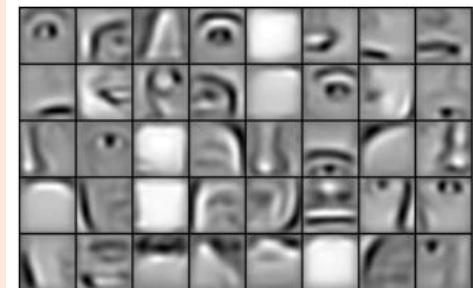
- First layer of z_i trained on 10 by 10 image patches:



} "Gabor filters"

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

faces

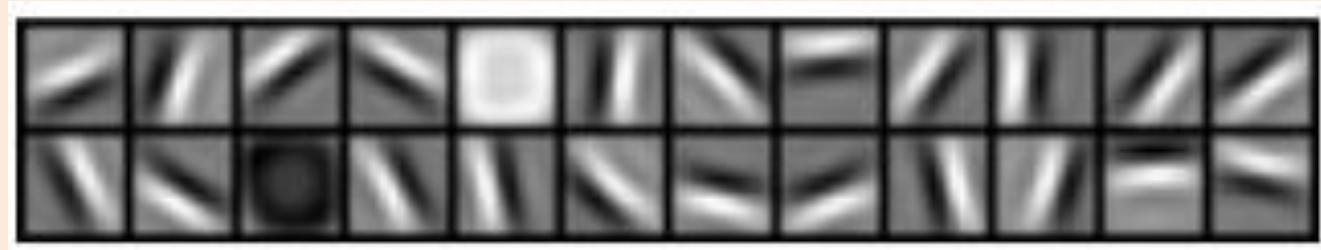


cars



Cool Picture Motivation for Deep Learning

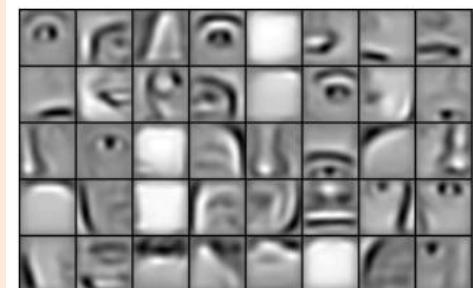
- First layer of z_i trained on 10 by 10 image patches:



} "Gabor filters"

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

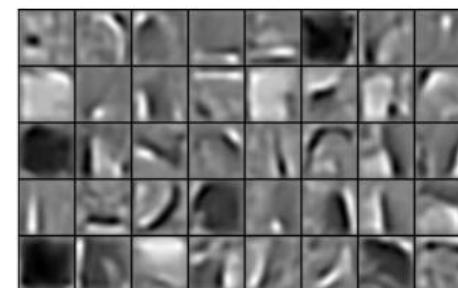
faces



cars

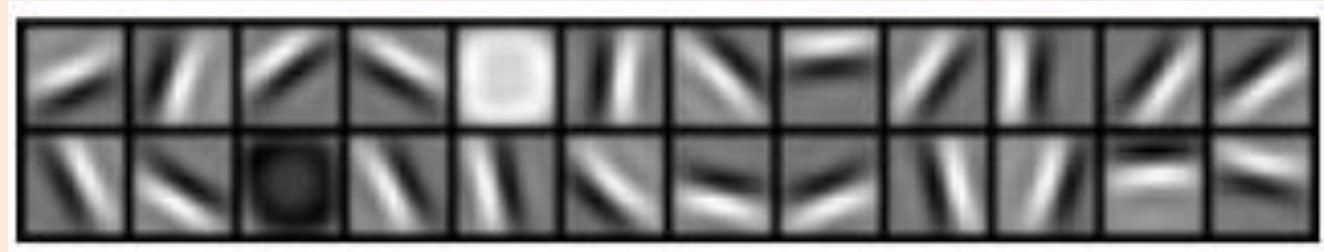


elephants



Cool Picture Motivation for Deep Learning

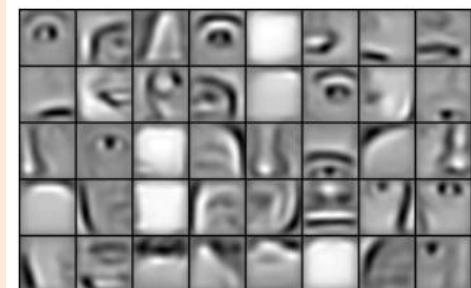
- First layer of z_i trained on 10 by 10 image patches:



} "Gabor filters"

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

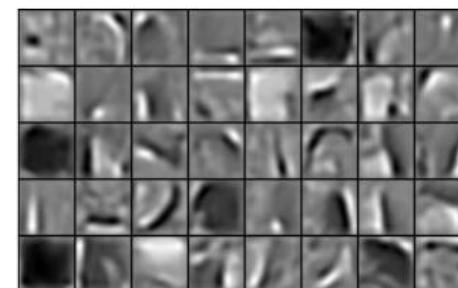
faces



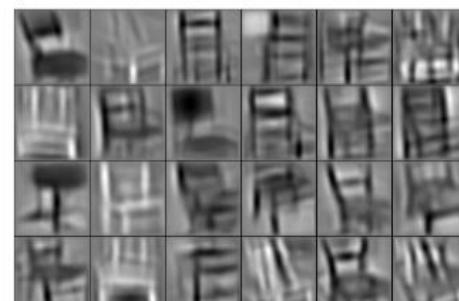
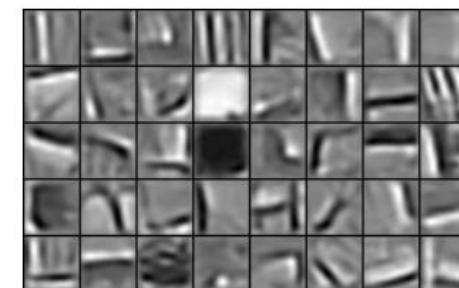
cars



elephants

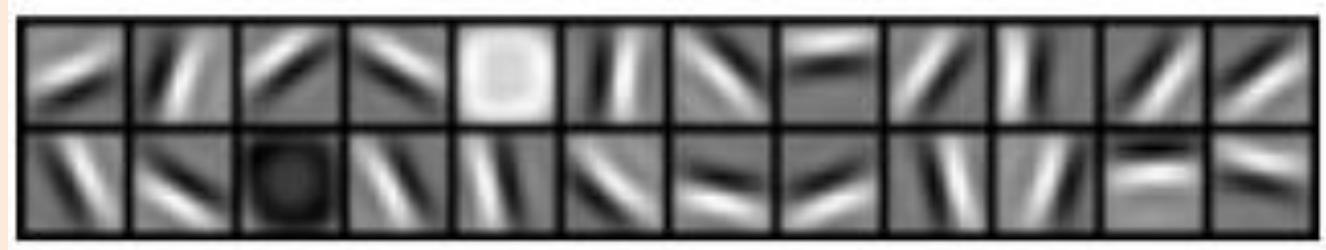


chairs



Cool Picture Motivation for Deep Learning

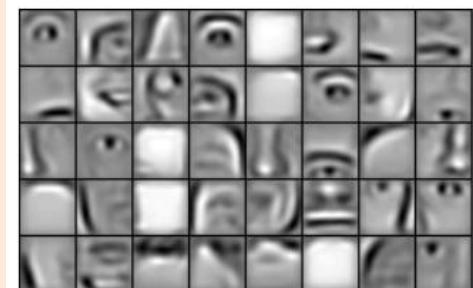
- First layer of z_i trained on 10 by 10 image patches:



} "Gabor filters"

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

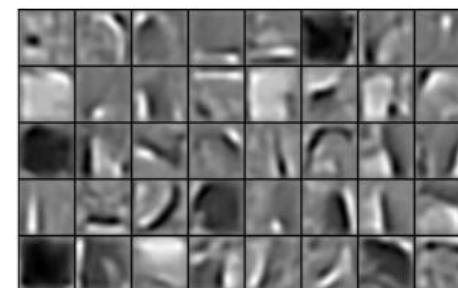
faces



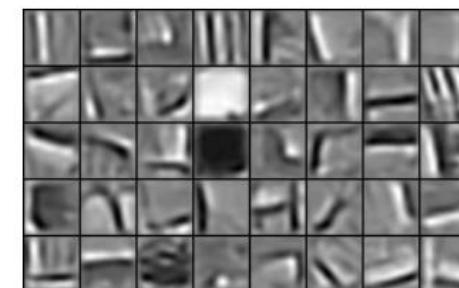
cars



elephants



chairs



faces, cars, airplanes, motorbikes

