

# Lecture 14: Deep Neural Networks

## CS486/686 Intro to Artificial Intelligence

2023-6-27

Pascal Poupart  
David R. Cheriton School of Computer Science



# Outline

- Deep Neural Networks
  - Gradient Vanishing
    - Rectified linear units
  - Overfitting
    - Dropout
- Breakthroughs
  - Acoustic modeling in speech recognition
  - Image recognition

# Deep Neural Networks

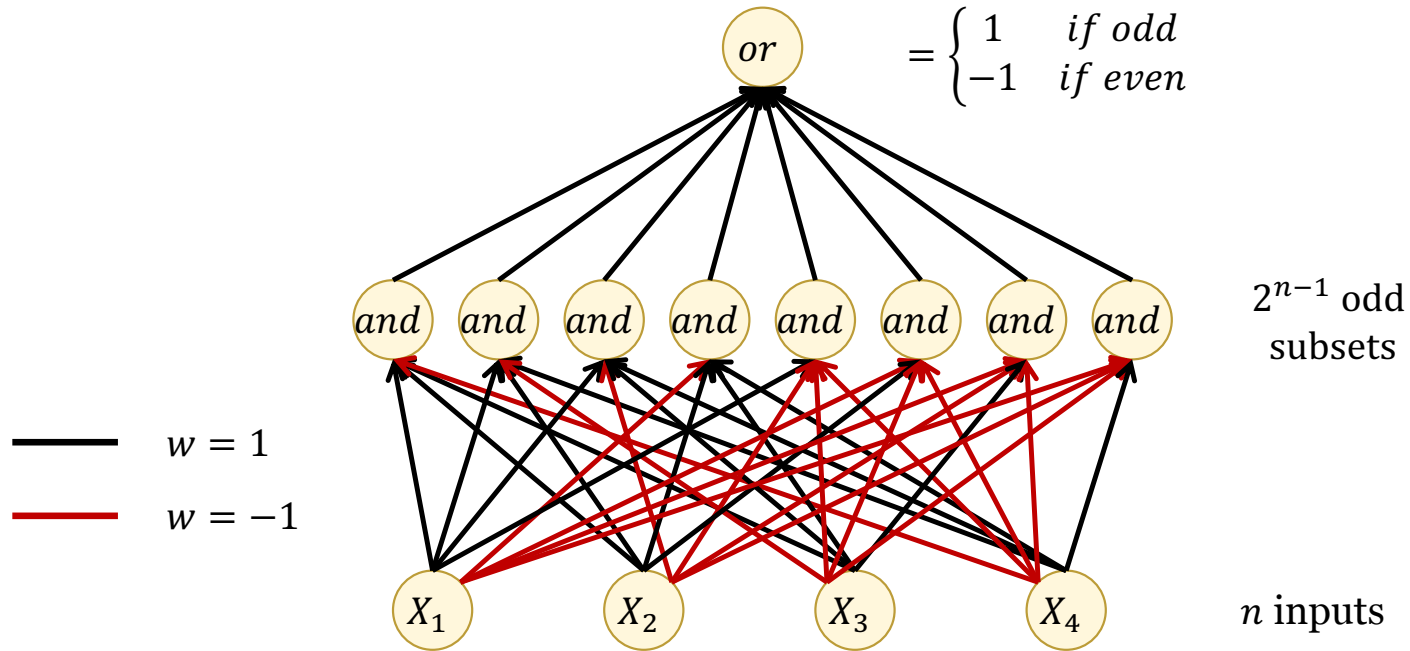
- Definition: neural network with **many hidden layers**
- Advantage: **high expressivity**
- Challenges:
  - **How should we train a deep neural network?**
  - **How can we avoid overfitting?**

# Expressiveness

- Neural networks with one hidden layer of sigmoid/hyperbolic units can approximate arbitrarily closely neural networks with several layers of sigmoid/hyperbolic units
- However as we increase the number of layers, the number of units needed may decrease exponentially (with the number of layers)

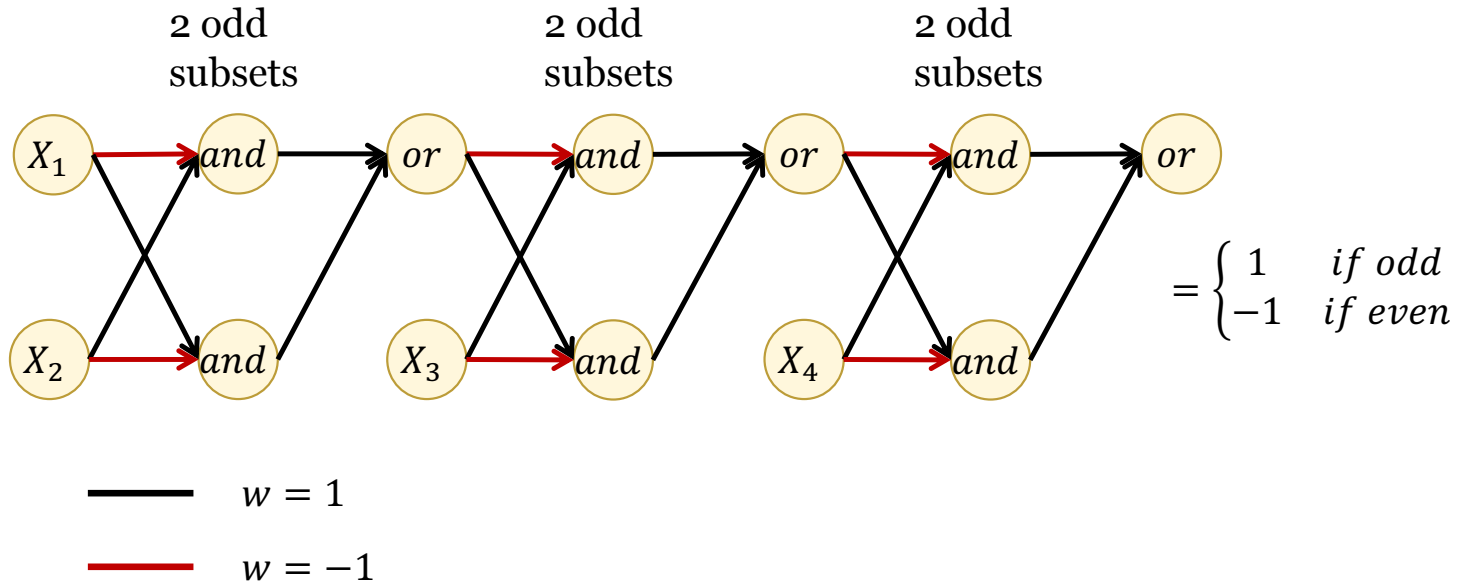
# Example - Parity Function

- Single layer of hidden nodes



# Example - Parity Function

- $2n - 2$  layers of hidden nodes

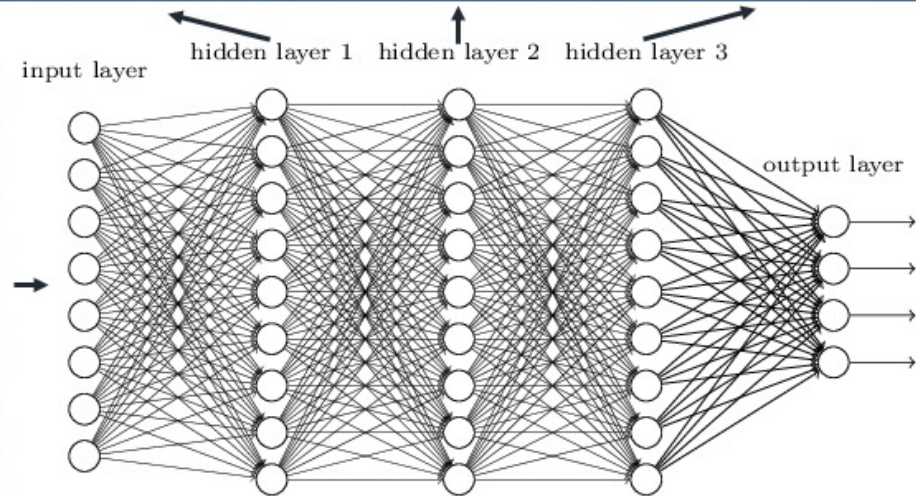


# The power of depth (practice)

Deep neural networks learn hierarchical feature representations



- Challenge: how to train deep NNs?



# Speech

- 2006 (Hinton, al.): first effective algorithm for deep NN
  - layerwise training of Stacked Restricted Boltzmann Machines (SRBM)s



# Speech

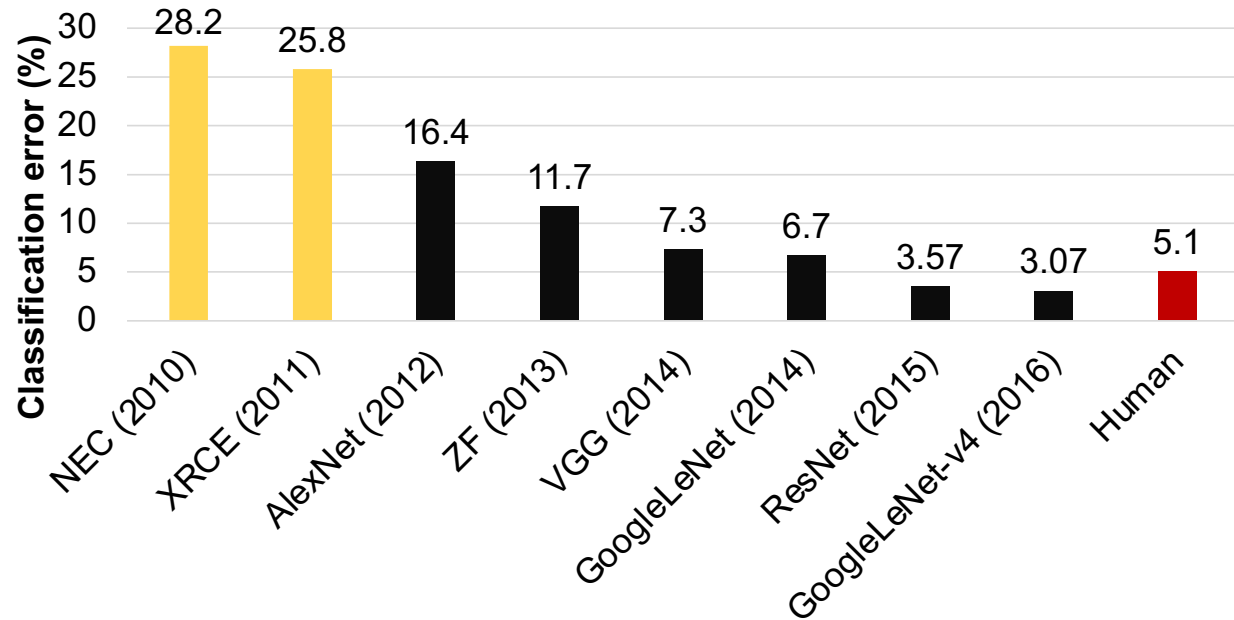
- 2006 (Hinton, al.): first effective algorithm for deep NN
  - layerwise training of Stacked Restricted Boltzmann Machines (SRBM)s
- 2009: **Breakthrough in acoustic modeling**
  - replace Gaussian Mixture Models by SRBMs
  - Improved speech recognition at Google, Microsoft, IBM

# Speech

- 2006 (Hinton, al.): first effective algorithm for deep NN
  - layerwise training of Stacked Restricted Boltzmann Machines (SRBM)s
- 2009: **Breakthrough in acoustic modeling**
  - replace Gaussian Mixture Models by SRBMs
  - Improved speech recognition at Google, Microsoft, IBM
- 2013-2019: recurrent neural nets (LSTM)
  - Google error rate: **23% (2013) → 8% (2015)**
  - Microsoft error rate: **5.9% (Oct 17, 2016) same as human performance**

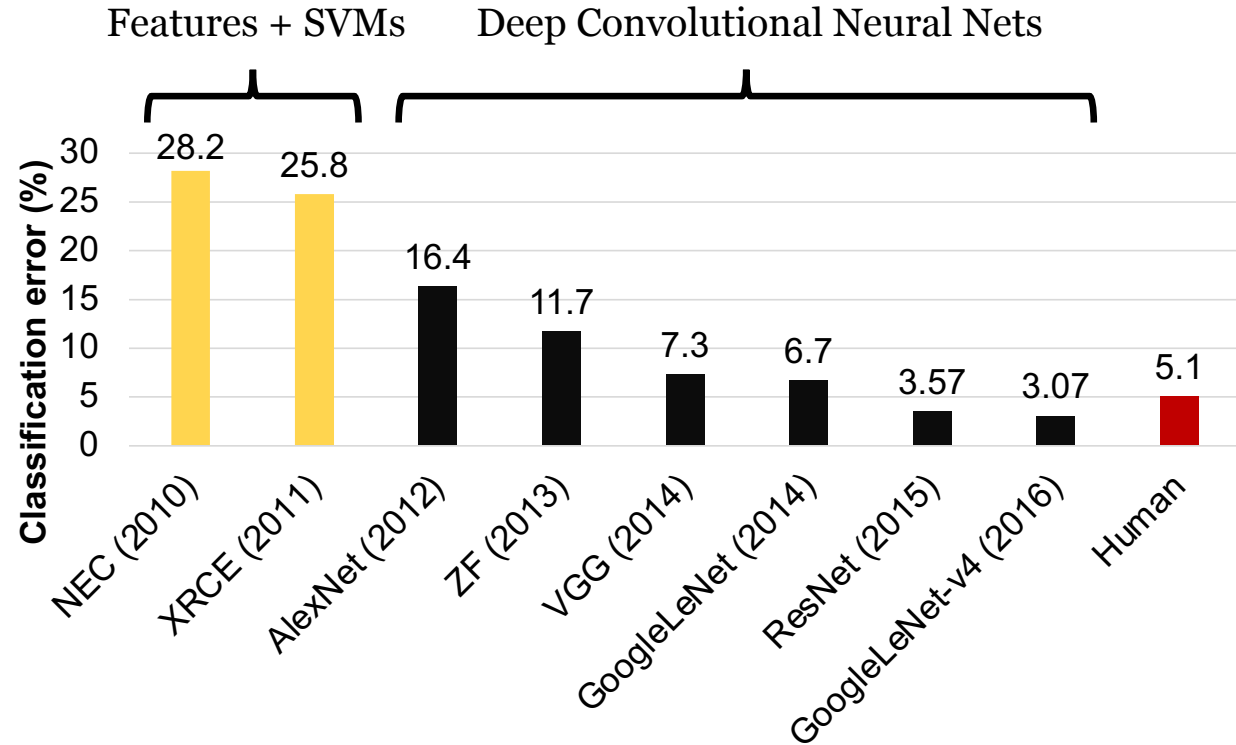
# Image Classification

- ImageNet Large Scale Visual Recognition Challenge



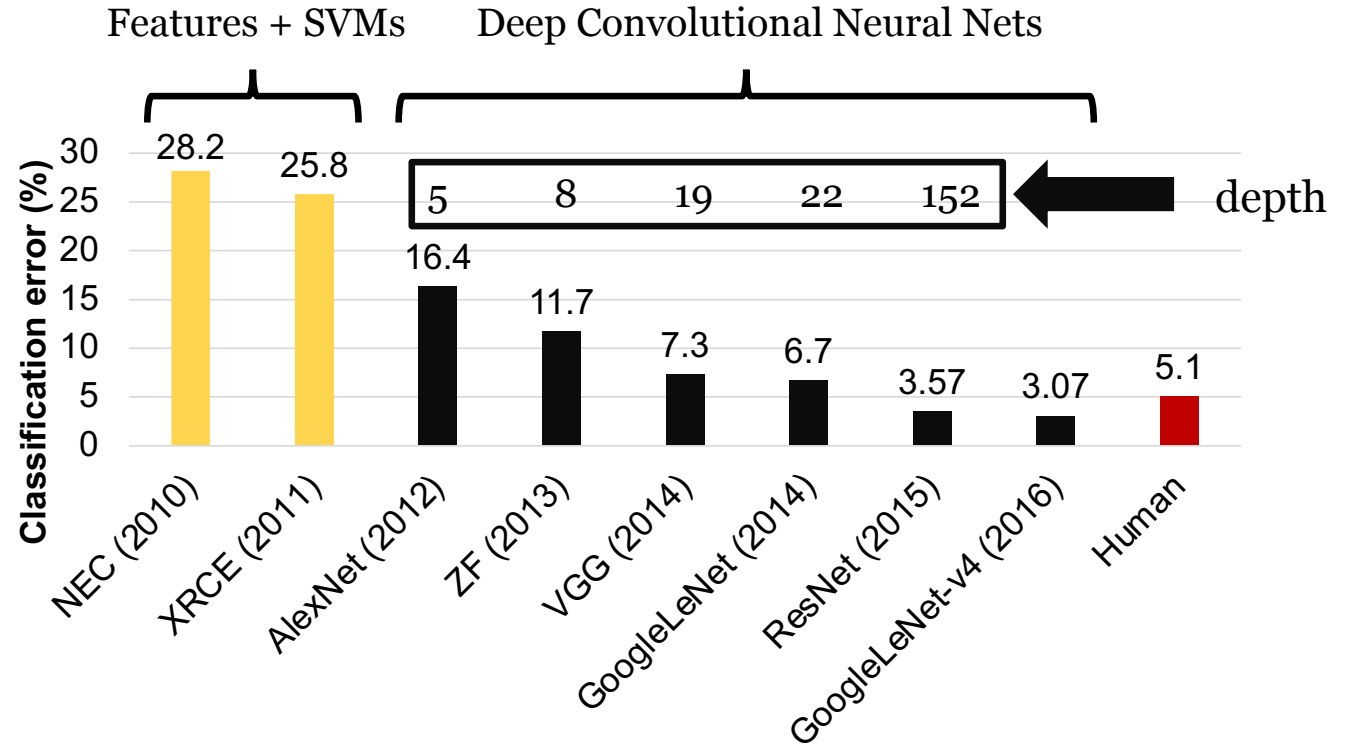
# Image Classification

- ImageNet Large Scale Visual Recognition Challenge



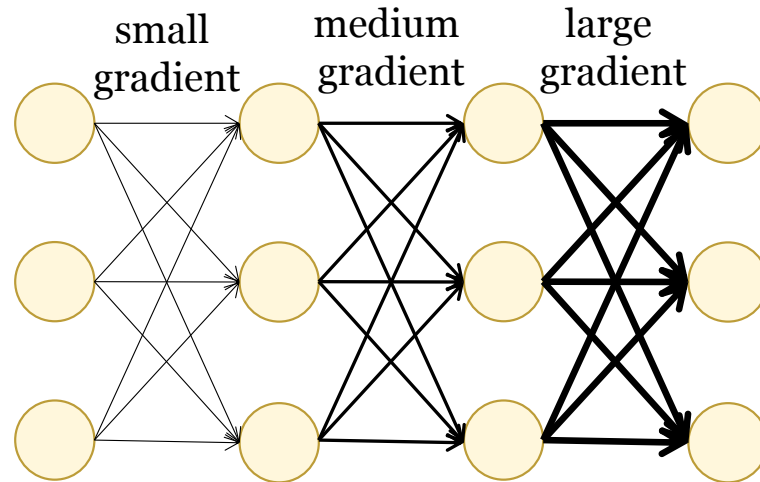
# Image Classification

- ImageNet Large Scale Visual Recognition Challenge



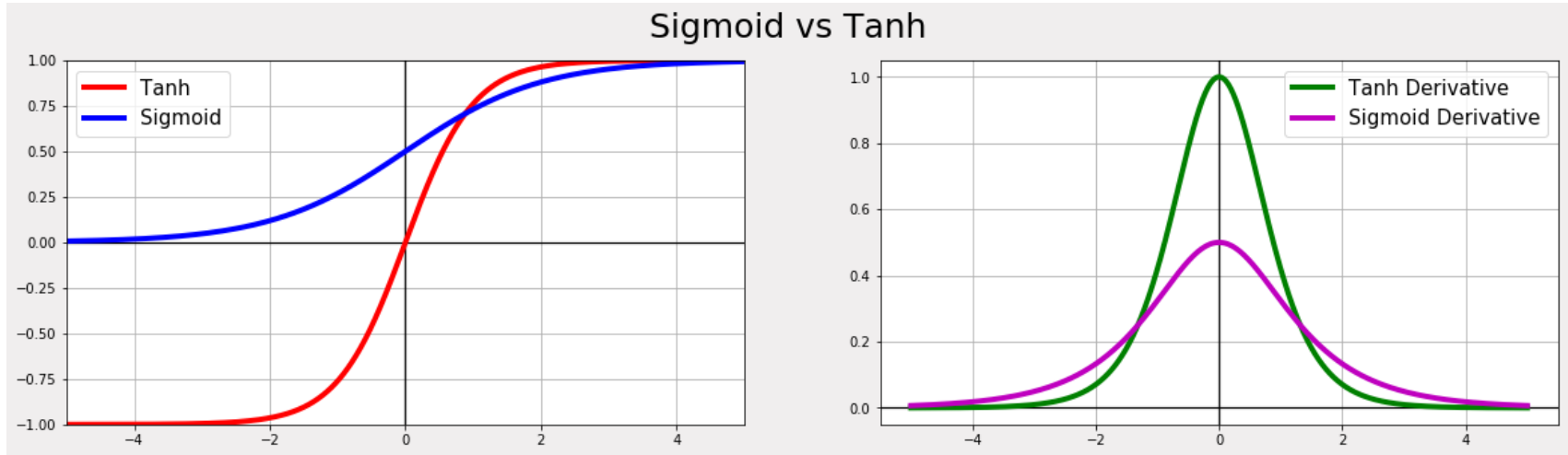
# Vanishing Gradients

- Deep neural networks of sigmoid and hyperbolic units often suffer from **vanishing gradients**



# Sigmoid and hyperbolic units

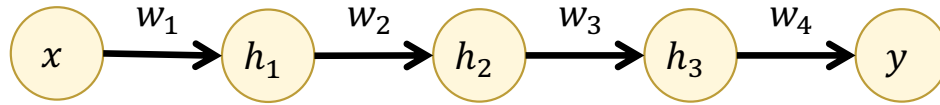
- Derivatives are always less than 1



From Aidan Wilson ([https://a-i-dan.github.io/math\\_nn](https://a-i-dan.github.io/math_nn))

# Simple Example

- $y = \sigma \left( w_4 \sigma \left( w_3 \sigma \left( w_2 \sigma \left( w_1 x \right) \right) \right) \right)$



- Common weight initialization in  $(-1,1)$
- Sigmoid function and its derivative always less than 1
- This leads to vanishing gradients:

$$\frac{\partial y}{\partial w_4} = \sigma'(a_4)\sigma(a_3)$$

$$\frac{\partial y}{\partial w_3} = \sigma'(a_4)w_4\sigma'(a_3)\sigma(a_2)$$

$$\frac{\partial y}{\partial w_2} = \sigma'(a_4)w_4\sigma'(a_3)w_3\sigma'(a_2)\sigma(a_1)$$

$$\frac{\partial y}{\partial w_1} = \sigma'(a_4)w_4\sigma'(a_3)w_3\sigma'(a_2)w_2\sigma'(a_1)x$$

As products of factors less than 1 gets longer, gradient vanishes



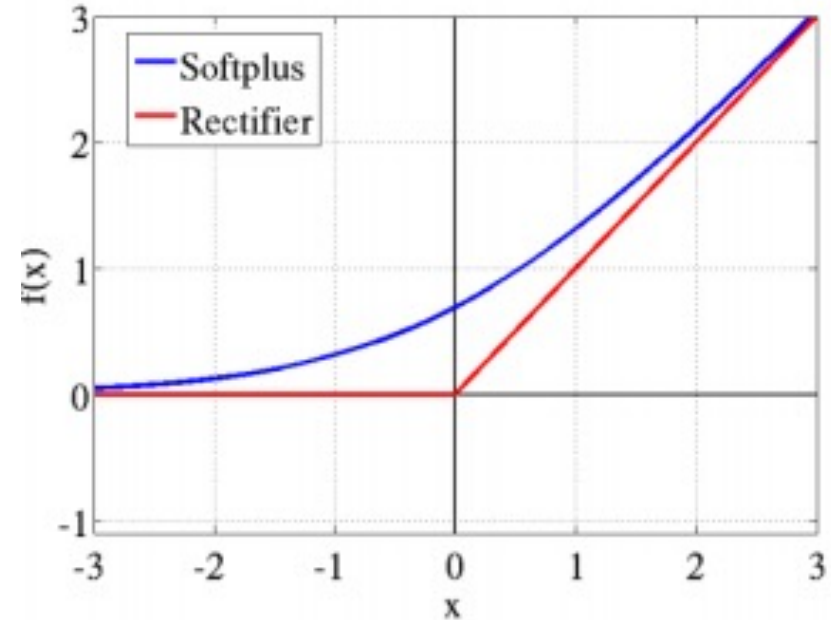
# Avoiding Vanishing Gradients

- Several popular solutions:
  - Pre-training
  - **Rectified linear units**
  - Skip connections
  - Batch normalization

# Rectified Linear Units

- Rectifier (ReLU):  $h(a) = \max(0, a)$ 
  - Gradient is 0 or 1
  - Sparse computation
- Soft version (“Softplus”) :  
 $h(a) = \log(1 + e^a)$
- Warning: softplus does not prevent gradient vanishing (gradient  $< 1$ )

From Abhinav Ralhan (<https://medium.com/@abhinavr8/activation-functions-neural-networks-66220238e1ff>)



# Overfitting

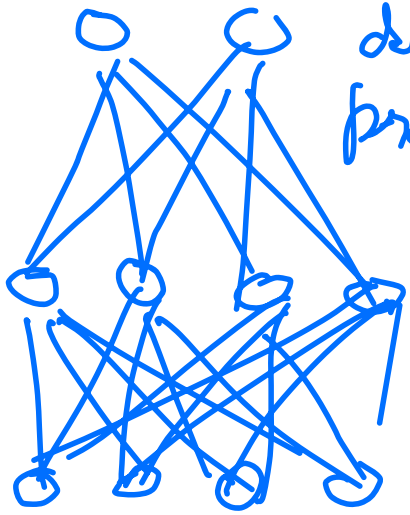
- High expressivity increases the risk of overfitting
  - # of parameters is often larger than the amount of data
  
- Some solutions:
  - Regularization
  - **Dropout**
  - Data augmentation

# Dropout

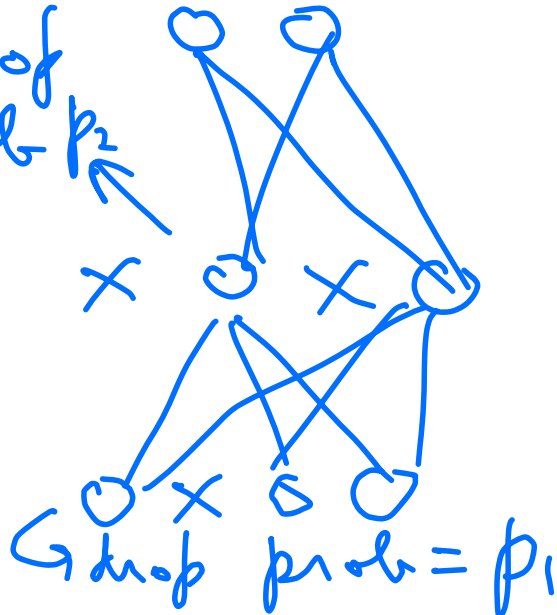
- Idea: randomly “drop” some units from the network when training
- Training: at each iteration of gradient descent
  - Each input unit is dropped with probability  $p_1$  (e.g., 0.2)
  - Each hidden unit is dropped with probability  $p_2$  (e.g., 0.5)
- Prediction (testing):
  - Multiply each input unit by  $1 - p_1$
  - Multiply each hidden unit by  $1 - p_2$

# Dropout Illustration

original net



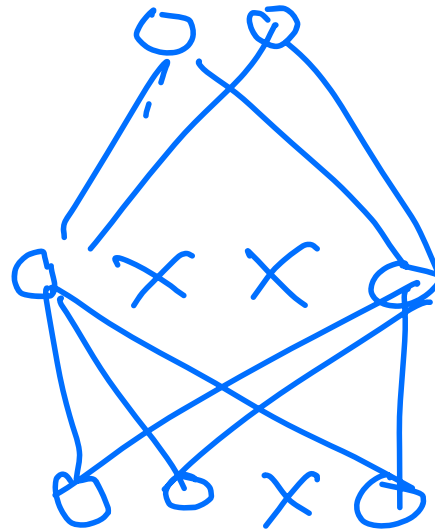
drop prob  $p_2$



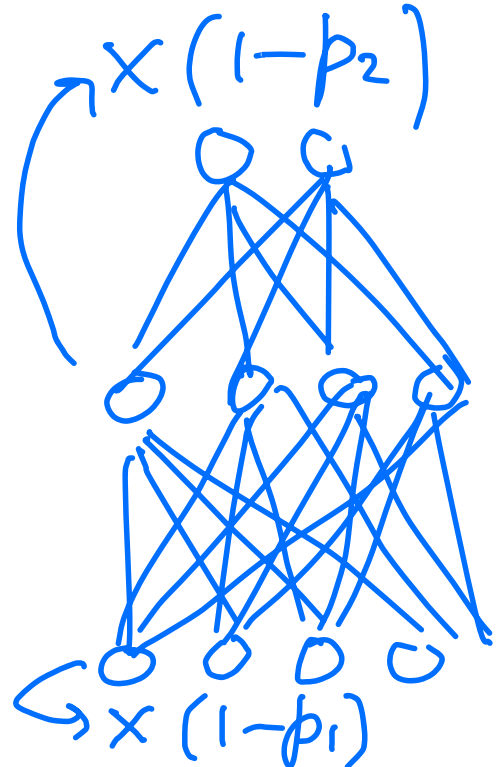
Training

batch #1

batch #2



prediction



# Dropout Algorithm

Training: let  $\odot$  denote elementwise multiplication

- Repeat
  - For each training example  $(\mathbf{x}_n, y_n)$  do
    - Sample  $\mathbf{z}_n^{(l)}$  from  $Bernoulli(1 - p_l)^{k_l}$  for  $1 \leq l \leq L$
    - Neural network with dropout applied:

$$f_n(\mathbf{x}_n, \mathbf{z}_n; \mathbf{W}) = h_l \left( \mathbf{W}^{(L)} \left[ \dots h_2 \left( \mathbf{W}^{(2)} \left[ h_1 \left( \mathbf{W}^{(1)} \left[ \bar{\mathbf{x}}_n \odot \mathbf{z}_n^{(1)} \right] \odot \mathbf{z}_n^{(2)} \right) \dots \odot \mathbf{z}_n^{(L)} \right] \right) \right]$$

- Loss:  $Err(y_n, f_n(\mathbf{x}_n, \mathbf{z}_n; \mathbf{W}))$
  - Update:  $w_{kj} \leftarrow w_{kj} - \eta \frac{\partial Err}{\partial w_{kj}}$
  - End for
- Until convergence

Prediction:  $f(\mathbf{x}_n; \mathbf{W}) = h_l \left( \mathbf{W}^{(L)} \left[ \dots h_2 \left( \mathbf{W}^{(2)} \left[ h_1 \left( \mathbf{W}^{(1)} \left[ \bar{\mathbf{x}}_n (1 - p_1) \right] (1 - p_2) \right) \dots (1 - p_L) \right] \right) \right]$

# Intuition

- Dropout can be viewed as an approximate form of ensemble learning
- In each training iteration, a different subnetwork is trained
- At test time, these subnetworks are “merged” by averaging their weights

# Early Applications of Deep Neural Networks

- **Speech Recognition**
- **Image recognition**
- Machine translation
- Control



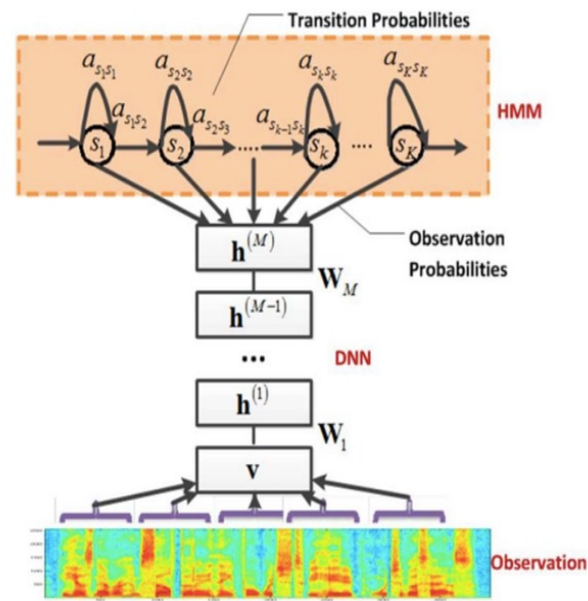
# Acoustic Modeling in Speech Recognition

## Architecture of a DNN-HMM hybrid system

TABLE III

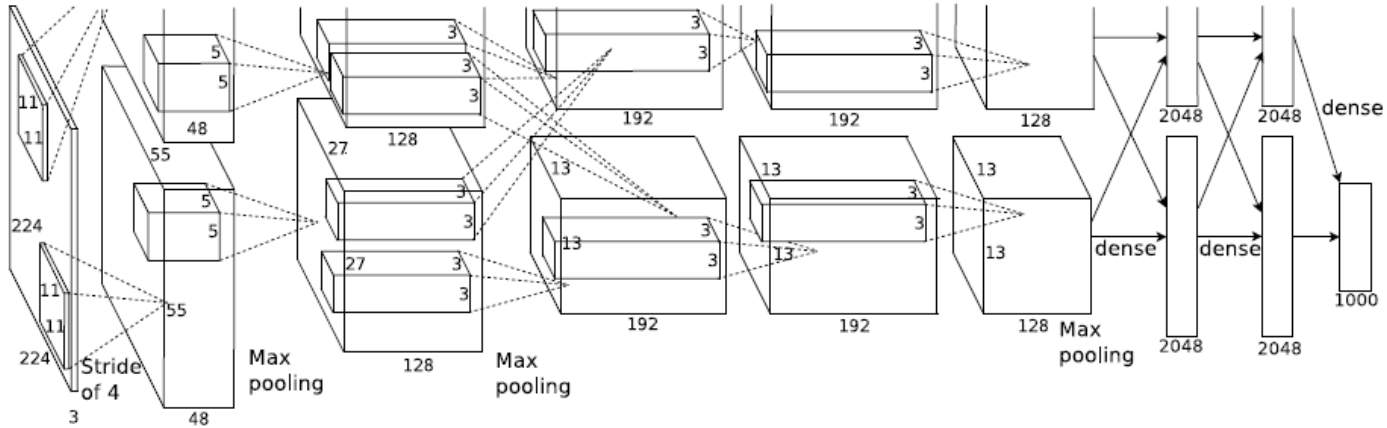
A comparison of the Percentage Word Error Rates using DNN-HMMs and GMM-HMMs on five different large vocabulary tasks.

task	hours of training data	DNN-HMM	GMM-HMM with same data	GMM-HMM with more data
Switchboard (test set 1)	309	18.5	27.4	18.6 (2000 hrs)
Switchboard (test set 2)	309	16.1	23.6	17.1 (2000 hrs)
English Broadcast News	50	17.5	18.8	
Bing Voice Search (Sentence error rates)	24	30.4	36.2	
Google Voice Input	5,870	12.3		16.0 (>>5,870hrs)
Youtube	1,400	47.6	52.3	



# Image Recognition

- Convolutional Neural Network
  - With rectified linear units and dropout
  - Data augmentation for transformation invariance



# ImageNet Breakthrough

- Results: ILSVRC-2012
- From Krizhevsky, Sutskever, Hinton

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
<i>SIFT + FVs [7]</i>	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	<b>16.4%</b>
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	<b>15.3%</b>

Table 2: Comparison of error rates on ILSVRC-2012 validation and test sets. In *italics* are best results achieved by others. Models with an asterisk\* were “pre-trained” to classify the entire ImageNet 2011 Fall release. See Section 6 for details.

# ImageNet Breakthrough

- From Krizhevsky, Sutskever, Hinton

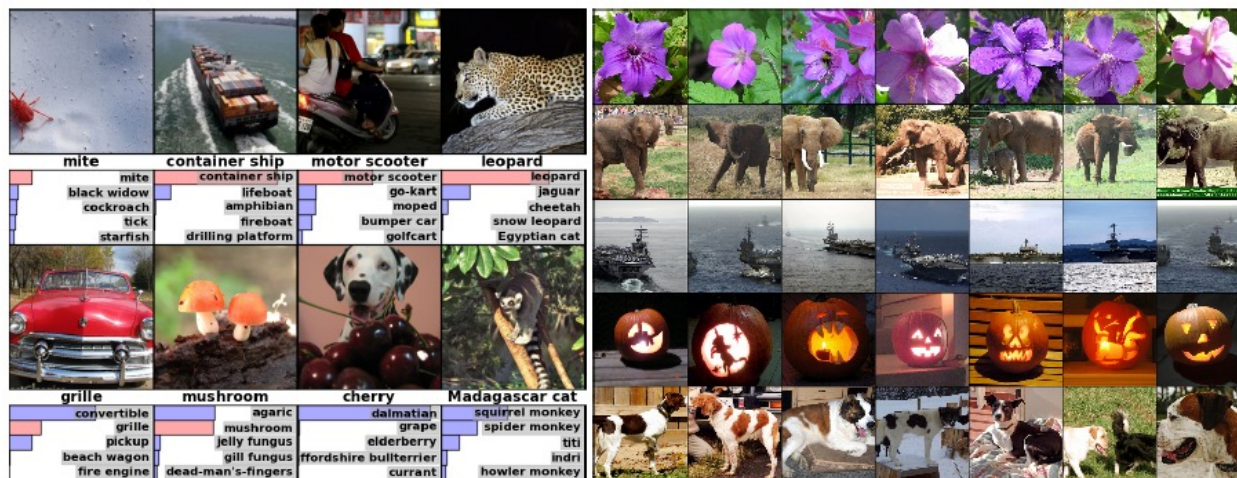


Figure 4: (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.