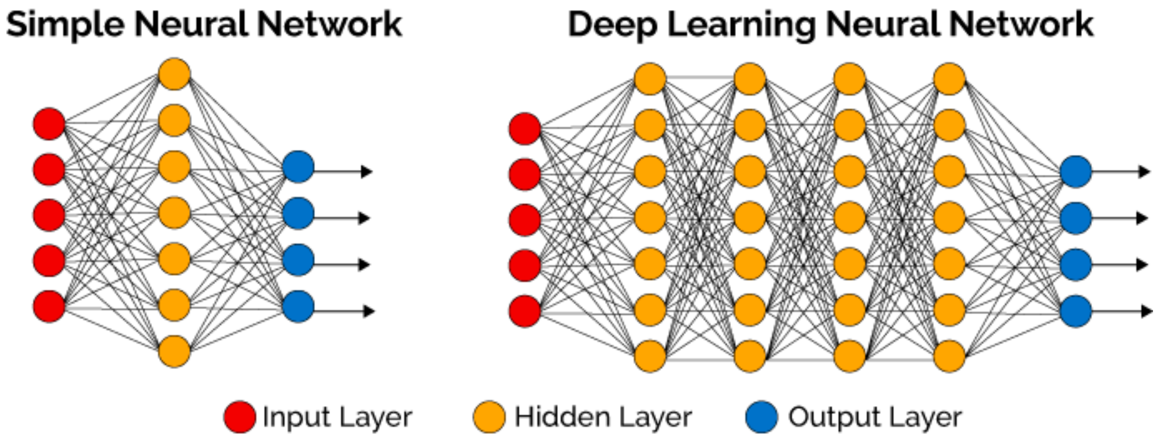
Deep Neural Networks

INSTRUCTOR: HONGJIE CHEN

JUNE 21ST 2022

Pros and Challenges

- Deep Neural Networks (DNN) are neural networks with many layers



- Pros: Highly expressive, accurate, mainstream method
- Challenges:
  - How to train a DNN?
  - How to avoid overfitting?

VIRGINIA TECH

Expressiveness

- A shallow flat NN can approximate abitrarily closely to deep narrow NN.
- When deep, fewer neurons are required to reach the same expressiveness.

Parity Function Example

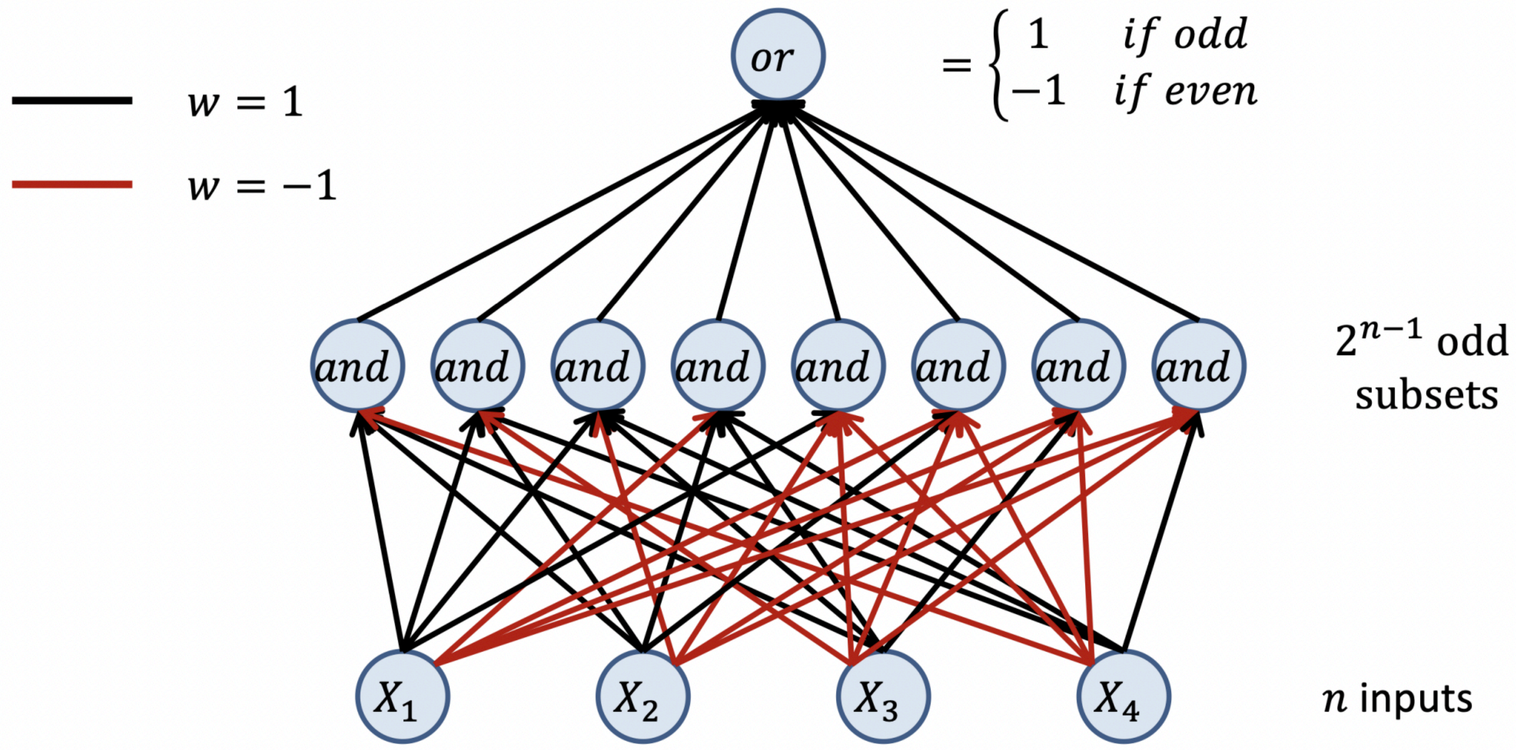
- $Y, X_1, X_2, X_3, X_4 \in \{-1,1\}$

$$Y = \begin{cases} 1, & X_1 + X_2 + X_3 + X_4 \mod 2 = 1 \\ -1, & (X_1 + X_2 + X_3 + X_4) \mod 2 = 0 \end{cases}$$

- 8 situations when predicted as 1

A Shallow Flat NN for Parity Function

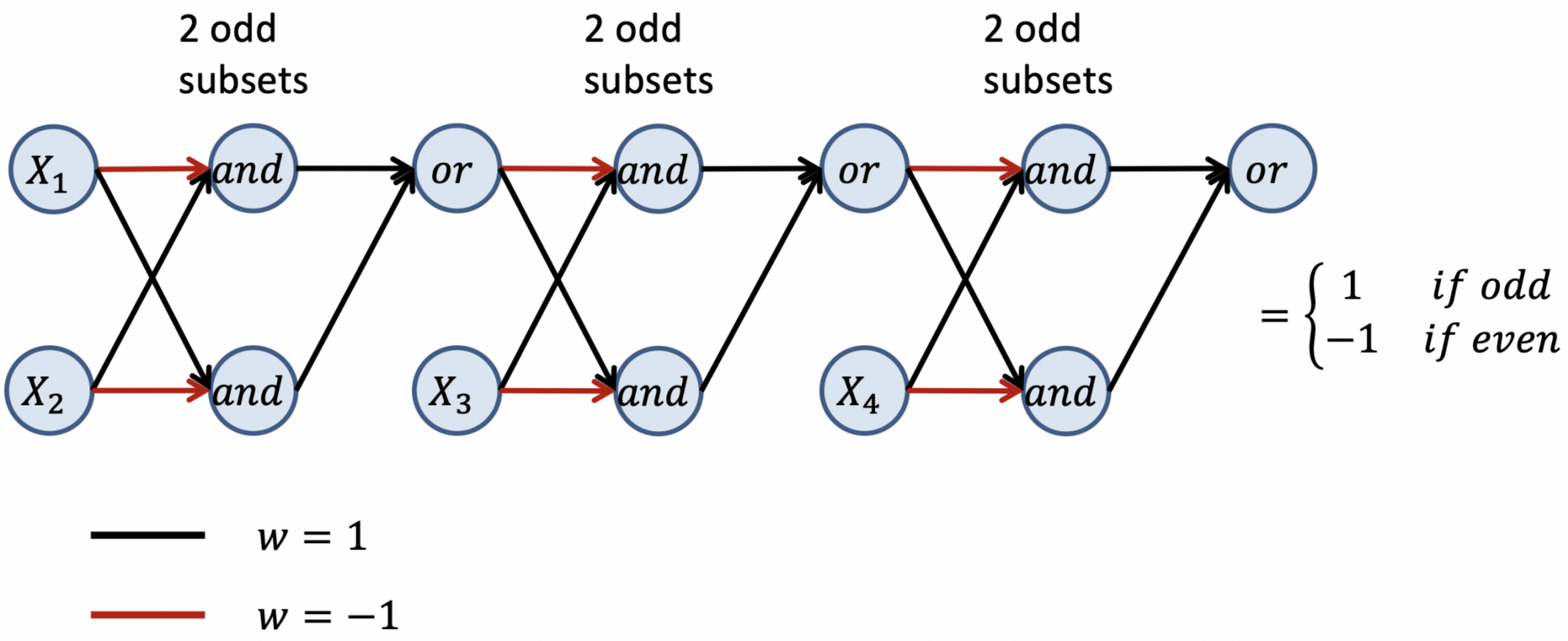
- 2^{n-1}, $n = 4$ hidden units



VIRGINIA TECH

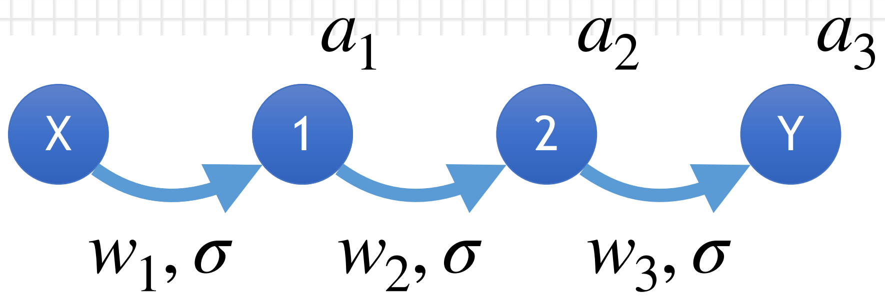
A Narrow Deep NN for Parity Function

- $2n - 2,\ n = 4$ hidden units



A Simple DNN Example

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



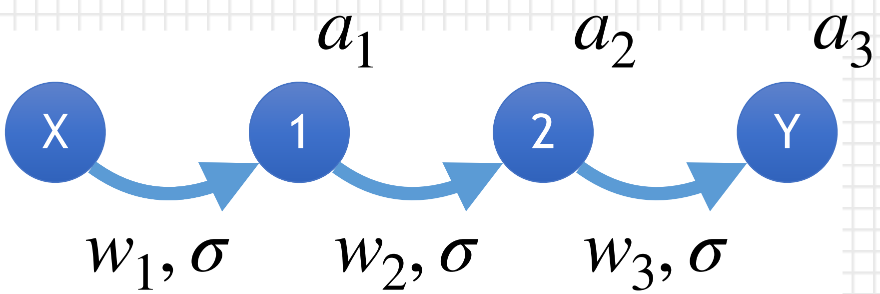
$$\frac{\partial y}{\partial w_3} = \sigma'(a_3)\sigma(a_2)$$

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

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

More and more terms

Vanishing Gradients



- $\frac{\partial y}{\partial w_1} = \sigma'(a_3)w_3\sigma'(a_2)w_2\sigma'(a_1)x$
- Weights are in $[0,1]$ or $[-1,1]$
- Activation functions and their derivatives are in $[-1,1]$
  - For example, sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, $\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right)$
- More terms mean $\frac{\partial y}{\partial w_1}$ is close to zero
  - Vanishing gradients close to the starting layers

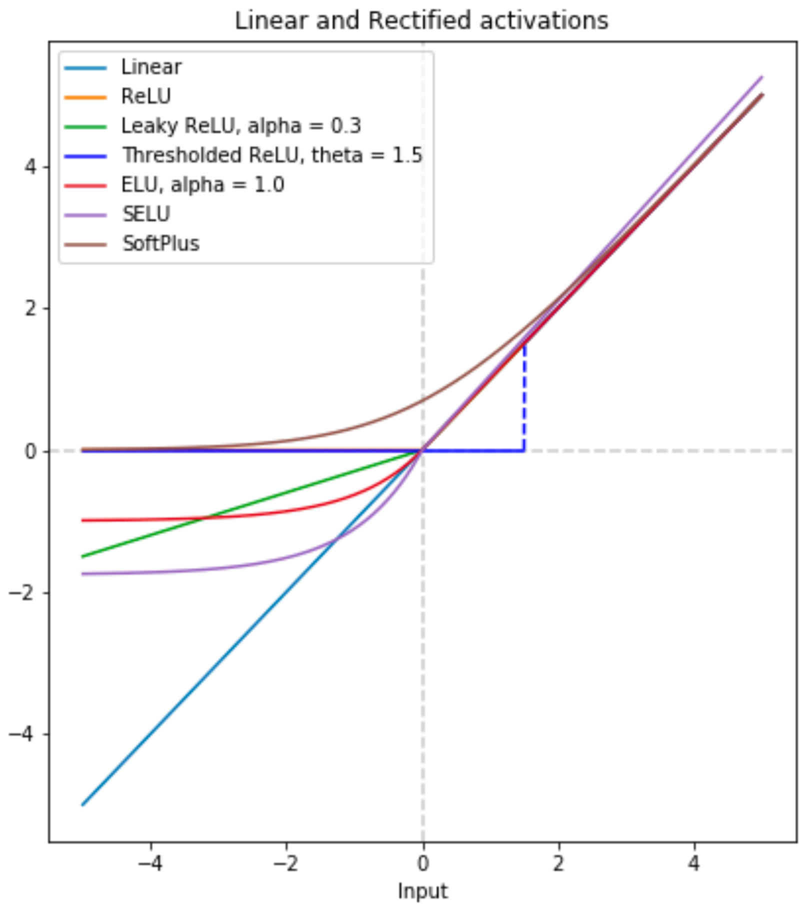
VIRGINIA TECH

Addressing Vanishing Gradients

- Popular solutions:
  - Smarter units, maxout units (Rectified Linear Units)
  - Skip connections
  - Batach normalization

Rectified Linear Units (ReLU)

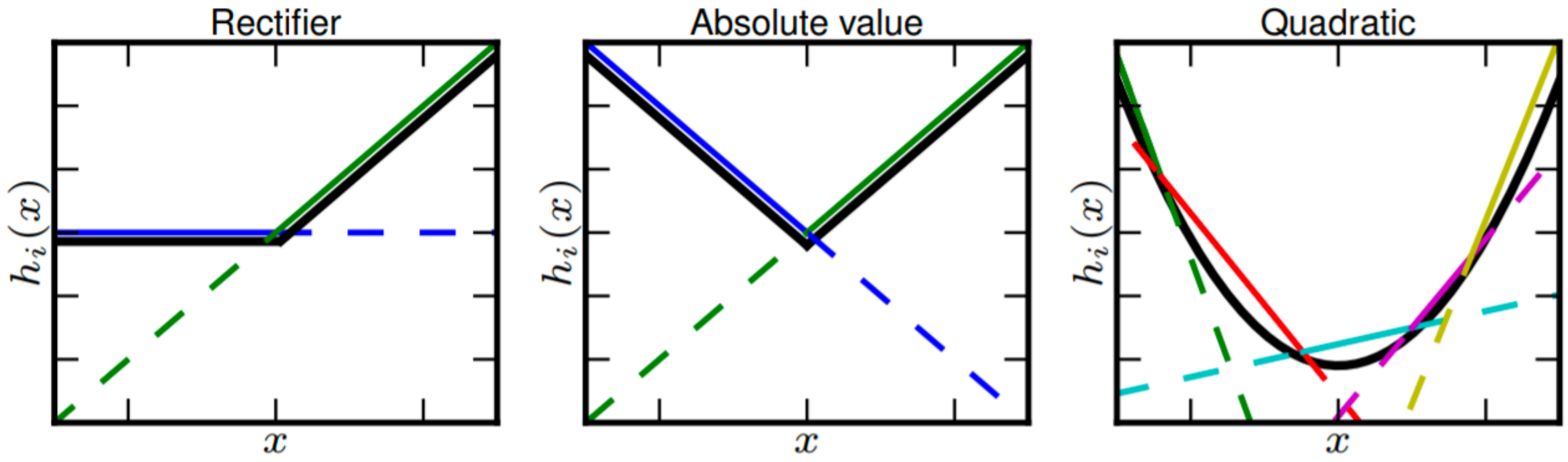
- ReLU
  - $h(a) = \max(0, a)$
    - Gradient $h'(a)$ is either 0 or 1
    - Computationally efficient
- LeakyReLU
  - $h(a) = \begin{cases} ka, \ a < 0 \\ a, \ a >= 0 \end{cases}$, k is a small constant
    - Fix dying ReLU, when there are many negative values
    - Gradient is either k or 1
- Counterexample: Softplus
  - $h(a) = \log(1 + e^a)$
    - Gradient is still smaller than 1
    - Making it differentiable at $x = 0$ does not help



VIRGINIA TECH

Maxout Units

- A generalization of ReLU units
- $f_{\max}(h_1, h_2, \ldots, h_n) = \max(h_1, h_2, \ldots, h_n)$
- h_i denotes the hidden state value from i(th) input hidden node



VIRGINIA TECH

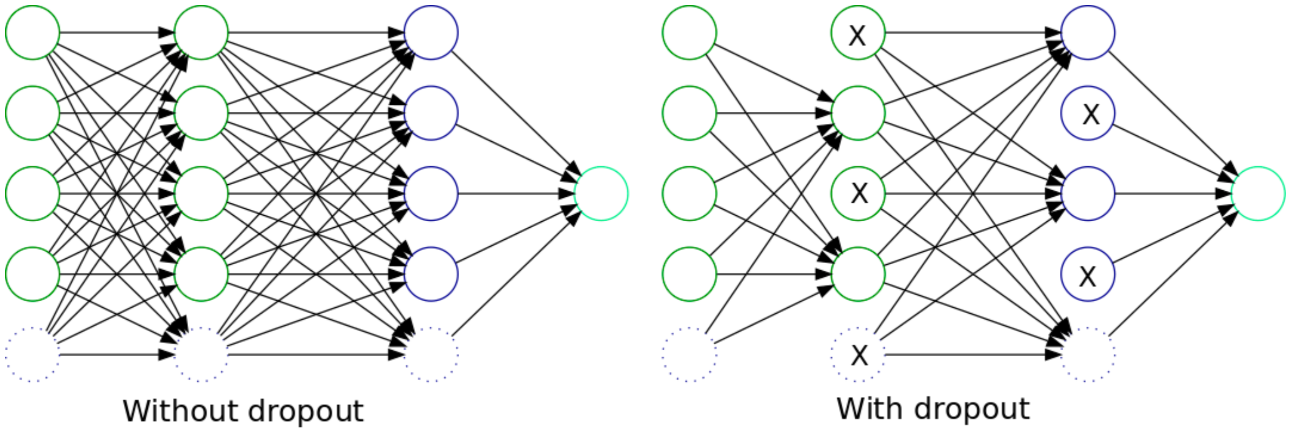
Addressing Overfitting

- High expressivity increases the risk of overfitting
- Memorizing everything causes a bad generalization

- Popular solutions
  - Dropout (turn off some neurons)
  - Regularization
  - Data augmentation

Dropout in Training Stage

- Randomly turn some units down
- For each iteration
  - Each input unit is dropped with a probability p_1 (e.g., 0.2)
  - Each hidden unit is dropped with a probability p_2 (e.g., 0.5)



Dropout in Testing Stage

- When testing, the dropout is not used
  - To utilize all input information
  - Too excited
- Compensation
  - Multiply each input unit by $1 - p_1$
  - Multiply each hidden unit by $1 - p_2$

Dropout seen as Ensemble

- Dropout can be viewed as a type of ensemble learning
- In each training iteration, a different subnetwork is trained
- In testing, these subnetworks are aggregated.
- Recall Boostrapping Aggregation (Bagging) in decision trees