CSE417T – Lecture 23

- Please **mute** yourself and **turn off videos** to save bandwidth.

- If you have questions during the lecture
  - Use chatrooms to post your questions
    - I'll review chatrooms in batches
  - You can also un-mute yourself and ask the questions directly
- The slides are posted on the course website

- RECORD THE LECTURE!
  - Please remind me if I forget to do so.

Logistics: Homework and Exam 2

- Homework 4 was due Monday.
  - Latest submission by today if you use 3 late days
- Homework 5 will be due on April 19 (Sunday), **11:30AM**
  - At most two late days can be used in this homework
  - I'll post the submission link on Friday to avoid conflicts with HW4 submissions
- Test exam is on Canvas till **Friday 1pm**
  - Strongly encouraged to do it make sure the setup works in your environment
  - No additional extension will be given for exam 2 due to unfamiliarity of the tool
- Exam 2 will be online on Canvas on April 23 (Thursday)
  - See Slides on April 7 for more details

Logistics: Exam 2

- Exam 2 will be on April 23 using **Canvas Quiz + Lockdown Browser.**

- Exam duration: **80 minutes**
  - 5 more minutes than Exam 1 as the buffer for online exam

- Start time
  - **11:30am CDT** (lecture start time).
  - If you cannot make it, please let me know by **tomorrow, April 17**
  - Only students who get approved can take the exam at a different time.
  - Unless there is a strong reason, everyone should take the exam on April 23.

Logistics: Exam 2

- Format
  - A mix of long questions and multiple choice questions
  - Likely dominated by multiple choices due to the constraint of math typing
  - I will try to minimize the need to write math in the long questions.
- Open book
  - You can reference any materials in hard copies. Searching information online is not allowed. Talking with other people is not allowed.
    - LockDown Browser will prohibit you from accessing other information in your computer
- Randomized questions
  - The questions will be randomly drawn from a “question bank”, so everyone might be getting different questions. I’ll take that into account in final grades.

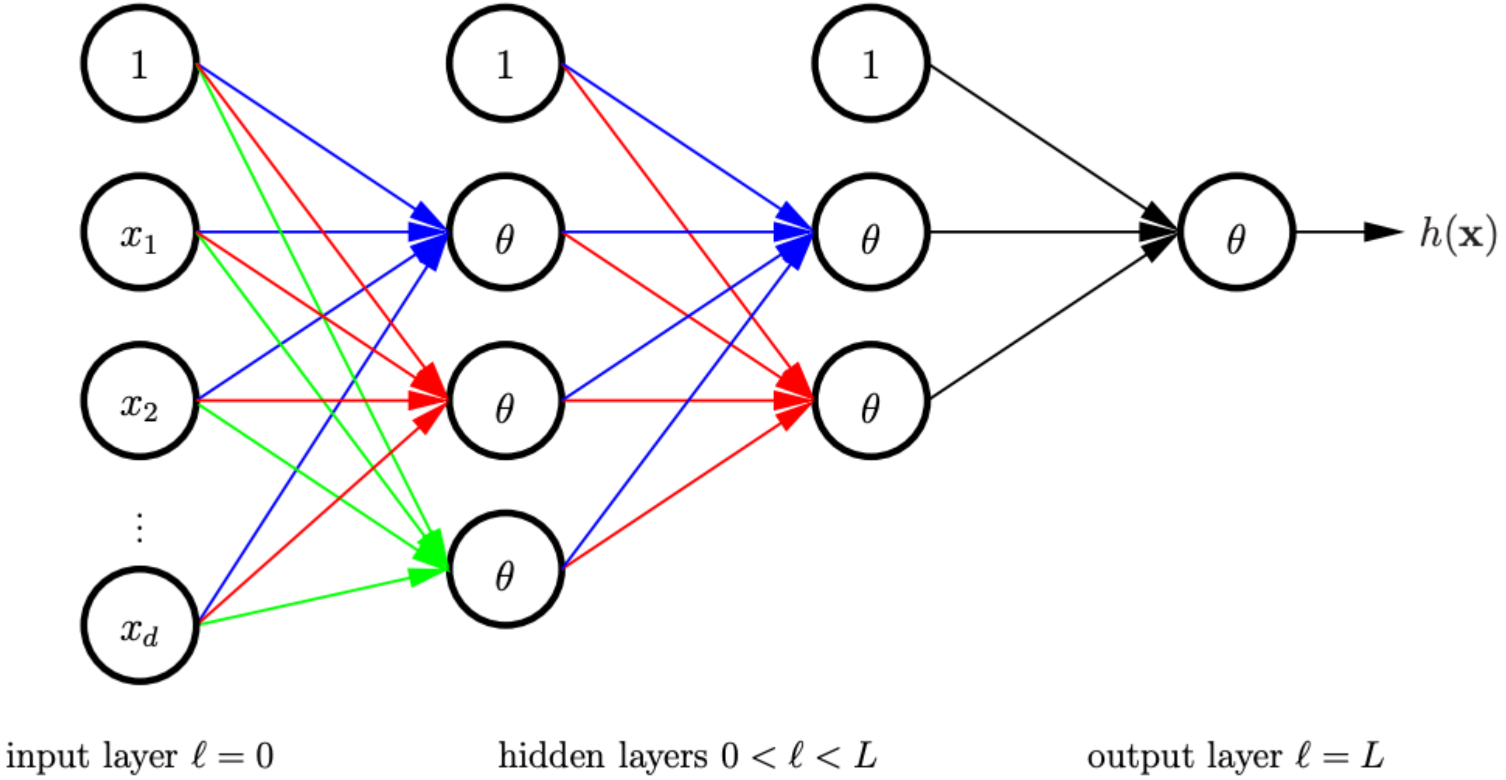
Logistics: Exam 2

- Topic
  - The focus will be on the 2nd half of the semester
    - Knowledge is cumulative, and concepts in Exam 1 might also be included
  - Everything in lectures and in readings on the course website is included (except for the parts marked as safe to skip).

- Remaining lectures
  - ~~Apr 14 (Tue): Learning in Neural Networks~~
  - Apr 16 (Thu): Discussion on Neural Networks
  - Apr 21 (Tue): Brief review and office hour
  - Apr 23 (Thu): Exam 2

Recap

Neural Networks



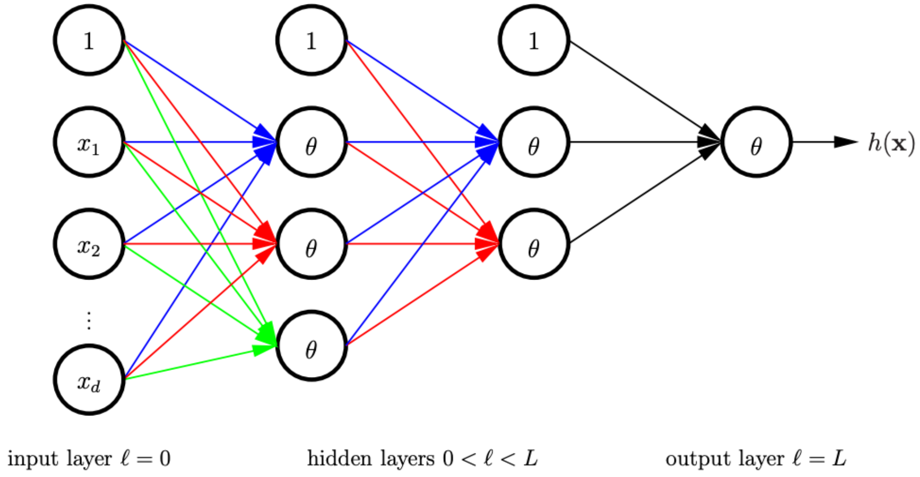
θ: activation function

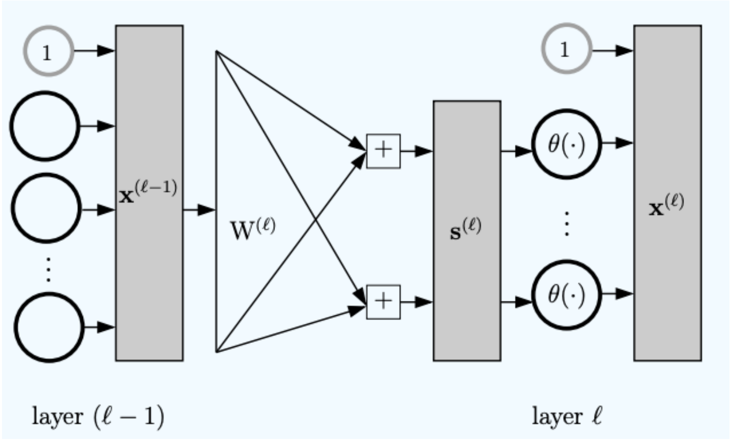
(Specify the "activation" of the neuron)

We mostly focus on feed-forward network structure

Notations of Neural Networks (NN)

- Notations:
  - $\ell = 0$ to L: layer
  - $d^{(\ell)}$: dimension of layer ℓ
  - $\vec{x}^{(\ell)}$: the nodes in layer ℓ
  - $w_{i,j}^{(\ell)}$: weights; characterize hypothesis in NN
  - $s_j^{(\ell)} = \sum_{i=0}^{d^{(\ell-1)}} w_{i,j}^{(\ell)} x_i^{(\ell-1)}$: linear signals
  - θ: activation function
    - $x_j^{(\ell)} = \theta\left(s_j^{(\ell)}\right)$





Forward Propagation (evaluate $h(\vec{x})$)

- A Neural network hypothesis h is characterized by $\left\{w_{i,j}^{(\ell)}\right\}$
- How to evaluate $h(\vec{x})$?

$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{\mathrm{W}^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{\mathrm{W}^{(2)}} \mathbf{s}^{(2)} \xrightarrow{\theta} \mathbf{x}^{(2)} \cdots \xrightarrow{\mathrm{W}^{(L)}} \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x}).$$

Forward propagation to compute $h(\mathbf{x})$:

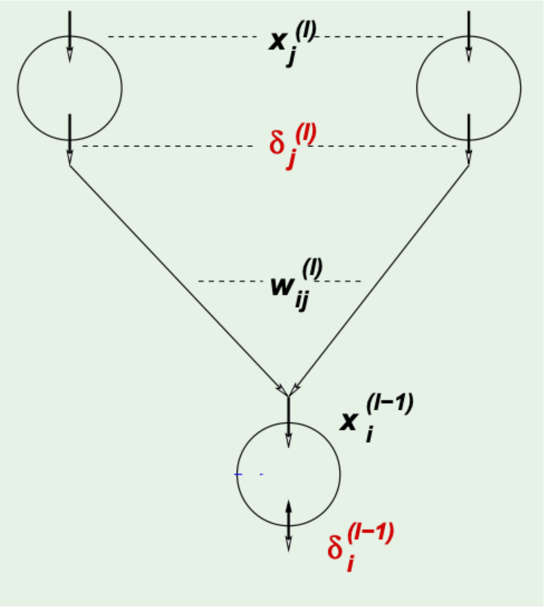
1: $\mathbf{x}^{(0)} \leftarrow \mathbf{x}$ **[Initialization]**
2: **for** $\ell = 1$ to L **do** **[Forward Propagation]**
3: $\quad \mathbf{s}^{(\ell)} \leftarrow (\mathrm{W}^{(\ell)})^{\mathrm{T}} \mathbf{x}^{(\ell-1)}$
4: $\quad \mathbf{x}^{(\ell)} \leftarrow \begin{bmatrix} 1 \\ \theta(\mathbf{s}^{(\ell)}) \end{bmatrix}$
5: **end for**
6: $h(\mathbf{x}) = \mathbf{x}^{(L)}$ **[Output]**

Given weights $w_{i,j}^{(\ell)}$ and $\vec{x}^{(0)} = \vec{x}$, we can calculate all $\vec{x}^{(\ell)}$ and $\vec{s}^{(\ell)}$ through forward propagation.

How to Learn NN From Data?

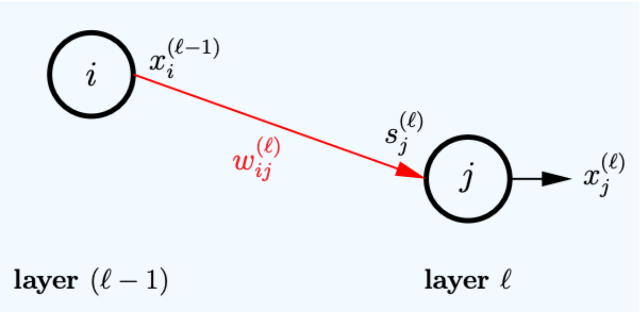
- Given D, how to learn the weights $W = \left\{w_{i,j}^{(\ell)}\right\}$?
- Intuition: Minimize $E_{in}(W) = \frac{1}{N}\sum_{n=1}^{N} e_n(W)$
- How?
  - Gradient descent: $W(t+1) \leftarrow W(t) - \eta \nabla_W E_{in}(W)$
  - Stochastic gradient descent $W(t+1) \leftarrow W(t) - \eta \nabla_W e_n(W)$
- Key step: we need to be able to evaluate the gradient…
  - Not trivial to do given the network structure
  - Backpropagation is an algorithmic procedure to calculate the gradient

Compute the Gradient $\nabla_W e_n(W)$



- Applying chain rule

$$\frac{\partial e_n(W)}{\partial w_{i,j}^{(\ell)}} = \frac{\partial e_n(W)}{\partial s_j^{(\ell)}} \frac{\partial s_j^{(\ell)}}{\partial w_{i,j}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}$$



- Calculating $\delta_j^{(\ell)}$
  - Backward recursive formulation
    - $\delta_j^{(\ell)} = \sum_{k=1}^{d^{(\ell+1)}} \frac{\partial e_n(W)}{\partial s_k^{(\ell+1)}} \frac{\partial s_k^{(\ell+1)}}{\partial x_j^{(\ell)}} \frac{\partial x_j^{(\ell)}}{\partial s_j^{(\ell)}} = \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)} \theta'\left(s_j^{(\ell)}\right)$
  - Boundary conditions
    - The output layer (assume regression)
      - $\delta_1^{(L)} = 2\left(s_1^{(L)} - y_n\right)$ (generalizable to other differentiable error)
  - Backward propagation

Backprogation Algorithm

- Recall that $\frac{\partial e_n(W)}{\partial w_{i,j}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}$
- Backpropagation Algorithm
  - Initialize $w_{i,j}^{(\ell)}$ randomly
  - For $t = 1$ to T
    - Randomly pick a point from D (for stochastic gradient descent)
    - Forward propagation: Calculate all $x_i^{(\ell)}$ and $s_i^{(\ell)}$
    - Backward propagation: Calculate all $\delta_j^{(\ell)}$
    - Update the weights $w_{i,j}^{(\ell)} \leftarrow w_{i,j}^{(\ell)} - \eta \delta_j^{(\ell)} x_i^{(\ell-1)}$
  - Return the weights

Discussion

- Backpropagation is gradient descent with efficient gradient computation
- Note that the E_{in} is not convex in weights
- Gradient descent doesn't guarantee to converge to global optimal

- Common approaches:
  - Run it many times
  - Each with a different initialization (the choice of initialization matters)

Lecture Notes Today

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.

Let me know if you spot errors.

Regularization in Neural Networks

Weight-Based Regularization

- Weight decay

$$E_{aug}(W) = E_{in}(W) + \frac{\lambda}{N}\sum_{i,j,\ell}\left(w_{i,j}^{(\ell)}\right)^2$$

  - Backpropagation applies (changing the error measure)

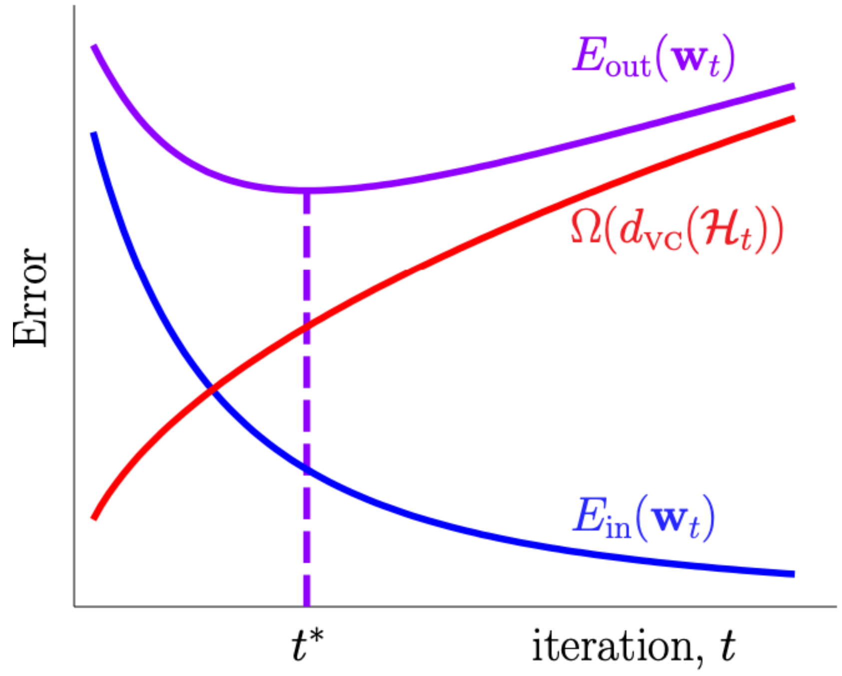
- Weight elimination

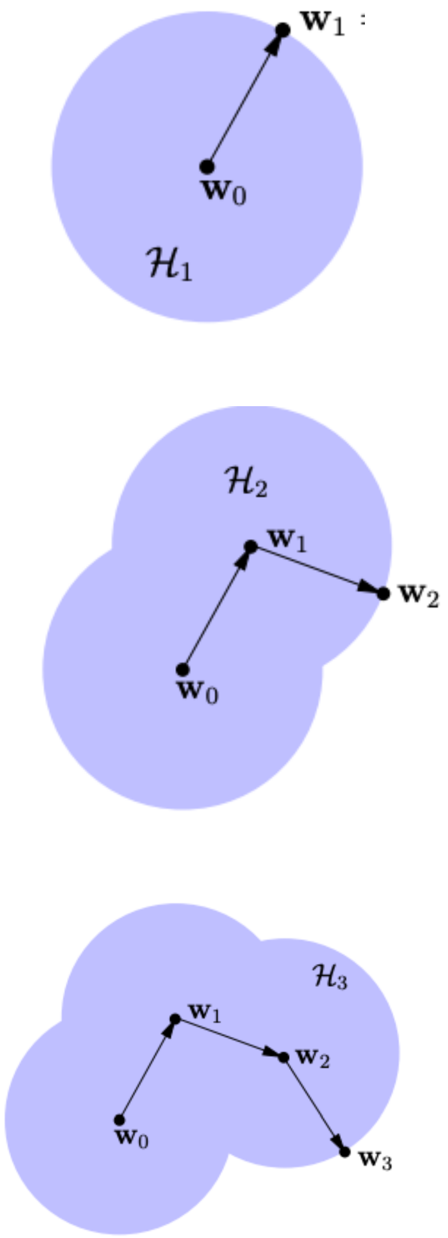
$$E_{aug}(W) = E_{in}(W) + \frac{\lambda}{N}\sum_{i,j,\ell}\frac{\left(w_{i,j}^{(\ell)}\right)^2}{1+\left(w_{i,j}^{(\ell)}\right)^2}$$

  - When $w_{i,j}^{(\ell)}$ is small, approximates weight decay
  - When $w_{i,j}^{(\ell)}$ is large, approximates adding a constant (no impacts to gradient)
  - "Decaying" more on smaller weights (i.e., eliminating small weights)

Early Stopping

- Consider gradient descent (GD)
  - H_1: the set of hypothesis GD can reach at $t = 1$
  - H_T: the set of hypothesis GD can reach at $t = T$
  - $H_1 \subseteq H_2 \subseteq H_3 \subseteq \cdots$

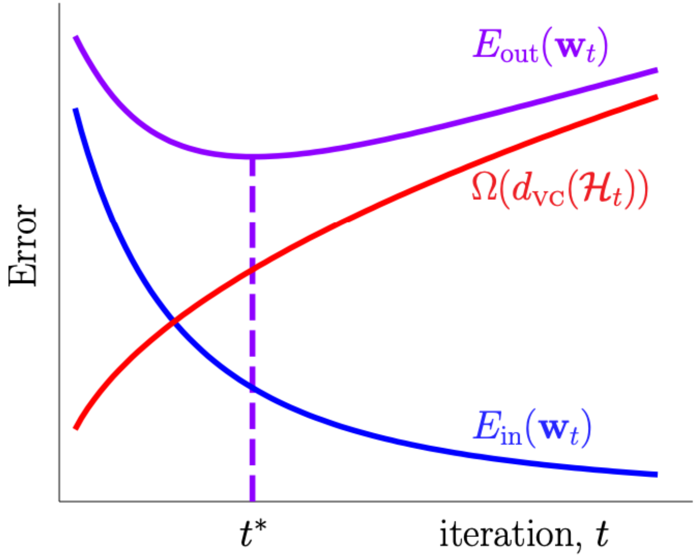


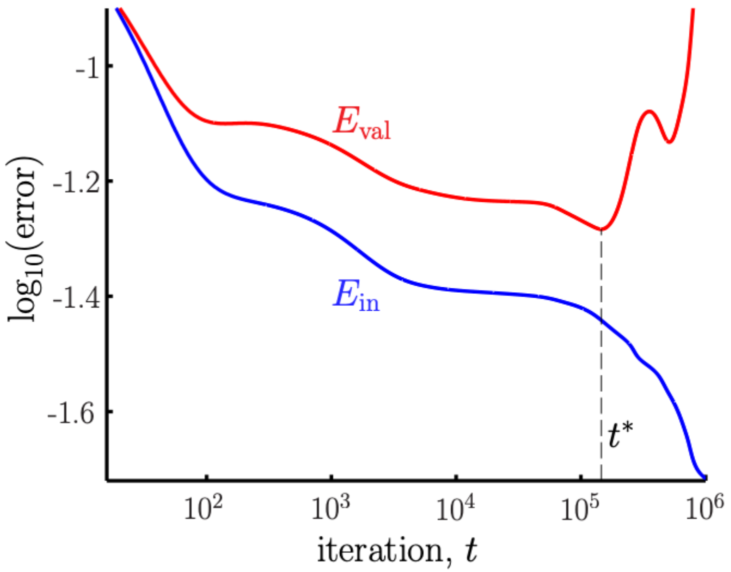


Early Stopping

- Stopping gradient descent early is a regularization method
  - “Constrained the hypothesis set”

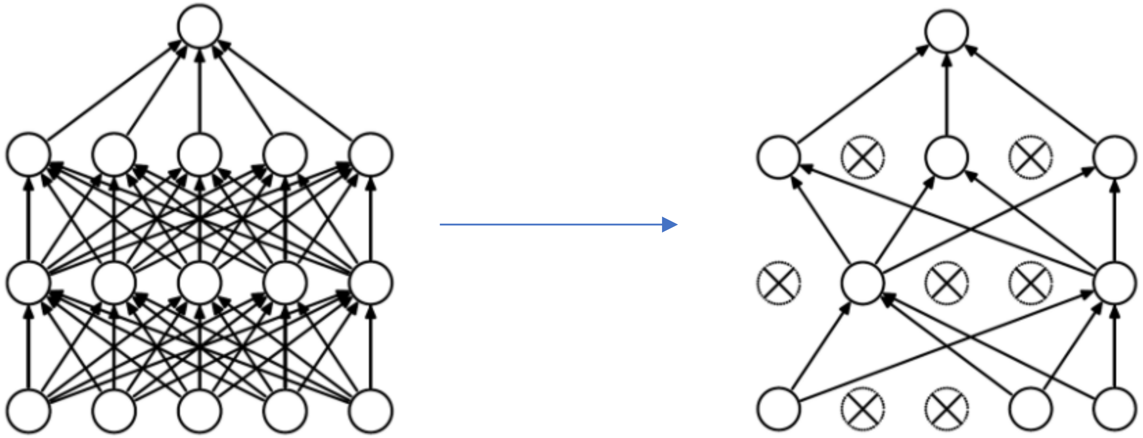
- How to find the optimal stopping point t^*?
  - Using validation is a common approach





Dropout

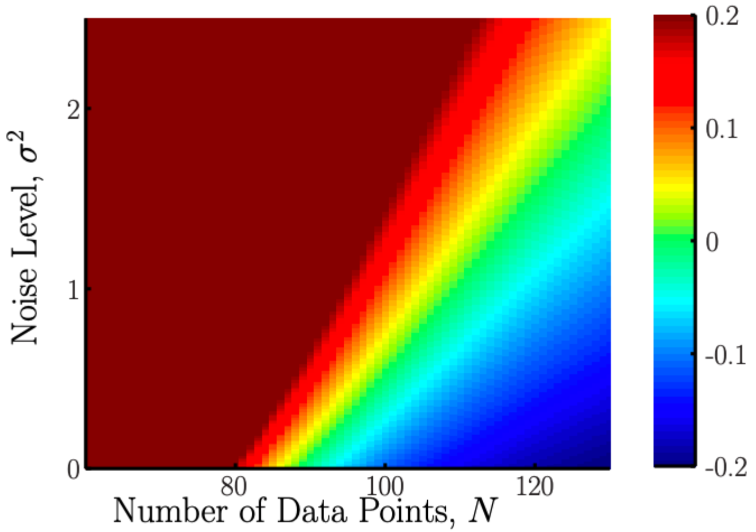
- Neural networks is very expressive (low bias, potentially high variance)
- Dropout
  - Randomly drop p portion of the weights during training

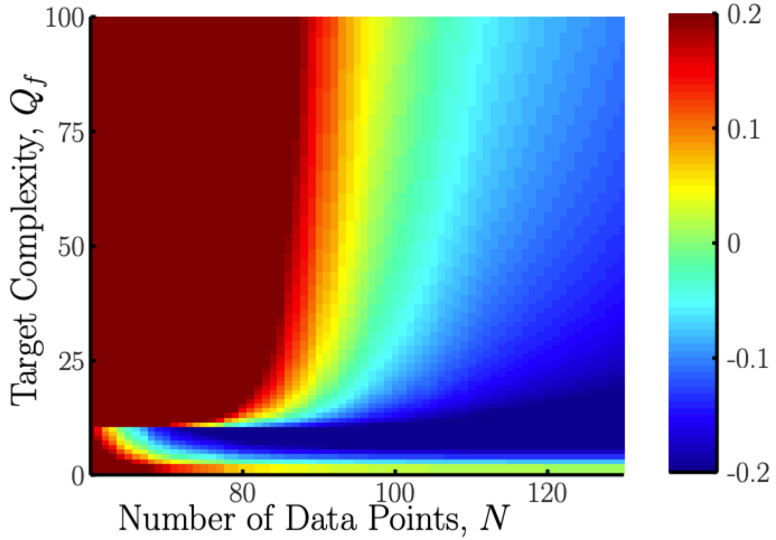


  - Learn many models with dropout
  - Average them during prediction (reduce weights by a ratio of p)

A Nontraditional Method to Avoid Overfitting

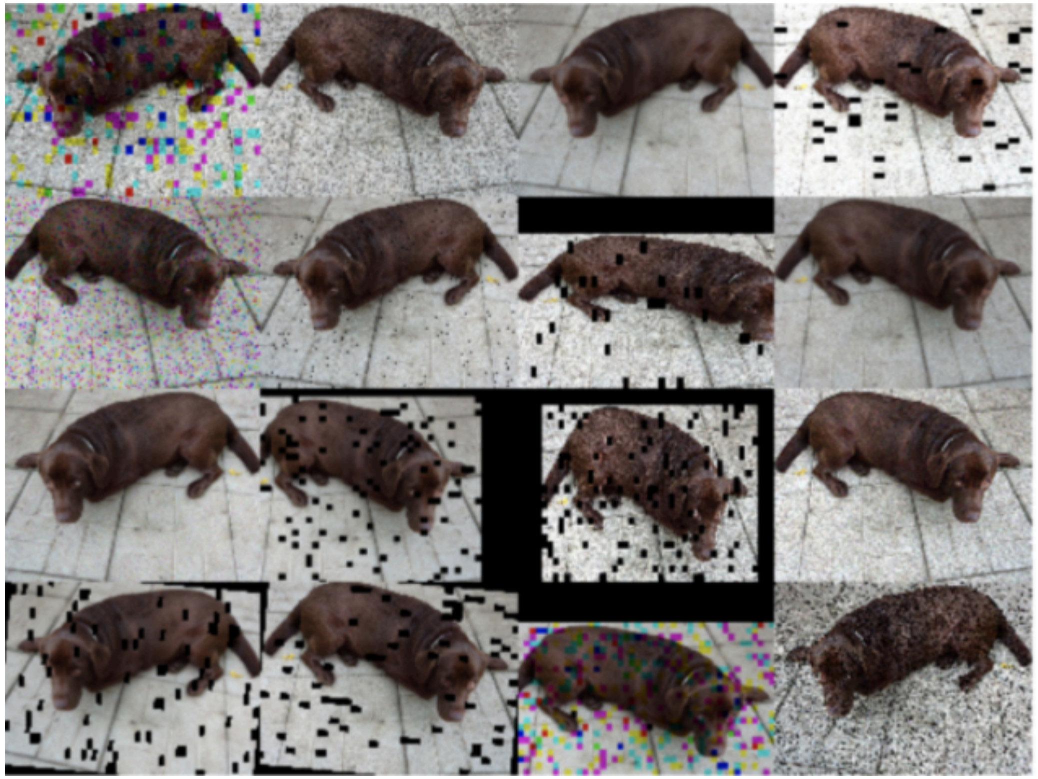
- What's the cause of overfitting?





- Fitting the noise instead of the target
- Regularization: Constrain H so it's not that powerful to fit noise
- How about adding noises to data?

Adding Noises as Regularization

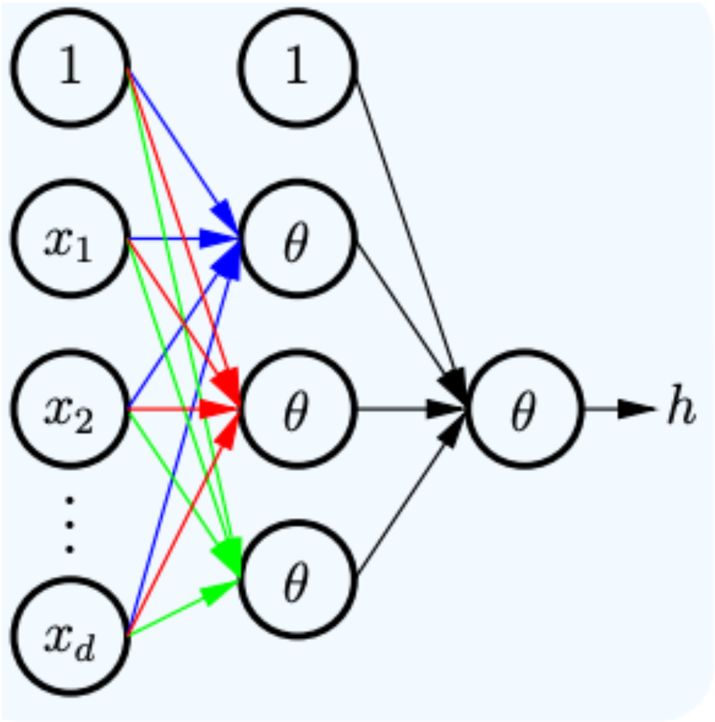


Short Break and Q&A

Deep Learning

Single Hidden-Layer Neural Network

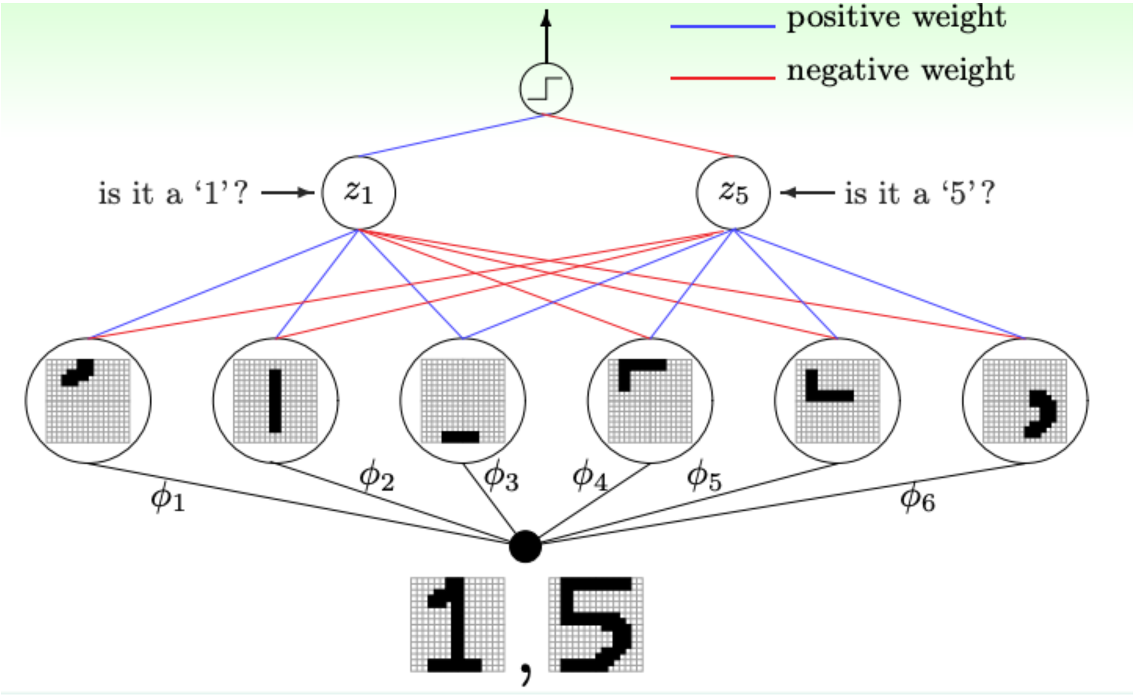
- How do we write a hypothesis in single-hidden layer mathematically?
  - $h(\vec{x}) = \theta\left(w_{0,1}^{(2)} + \sum_{j=1}^{d^{(1)}} w_{j,1}^{(2)} x_{j,1}^{(1)}\right)$
    $= \theta\left(w_{0,1}^{(2)} + \sum_{j=1}^{d^{(1)}} w_{j,1}^{(2)} \theta\left(\sum_{i=0}^{d^{(0)}} w_{i,j}^{(1)} x_i\right)\right)$
- How do we write a Kernel SVM hypothesis
  - $g(\vec{x}) = \theta\left(b^* + \sum_{\alpha_n^*>0} \alpha_n^* y_n K(\vec{x}_n, \vec{x})\right)$
- How do we write a linear model with nonlinear transform
  - $h(\vec{x}) = \theta(w_0 + \sum w_i \phi_i(\vec{x}))$
- Interpretation:
  - The hidden layer is like feature transform
  - Shallow learning vs. deep learning



[Safe to Skip Starting this Slide]

Deep Neural Network

- “Shallow” neural network is powerful (universal approximation theorem holds with a single hidden layer). Why “deep” neural networks?



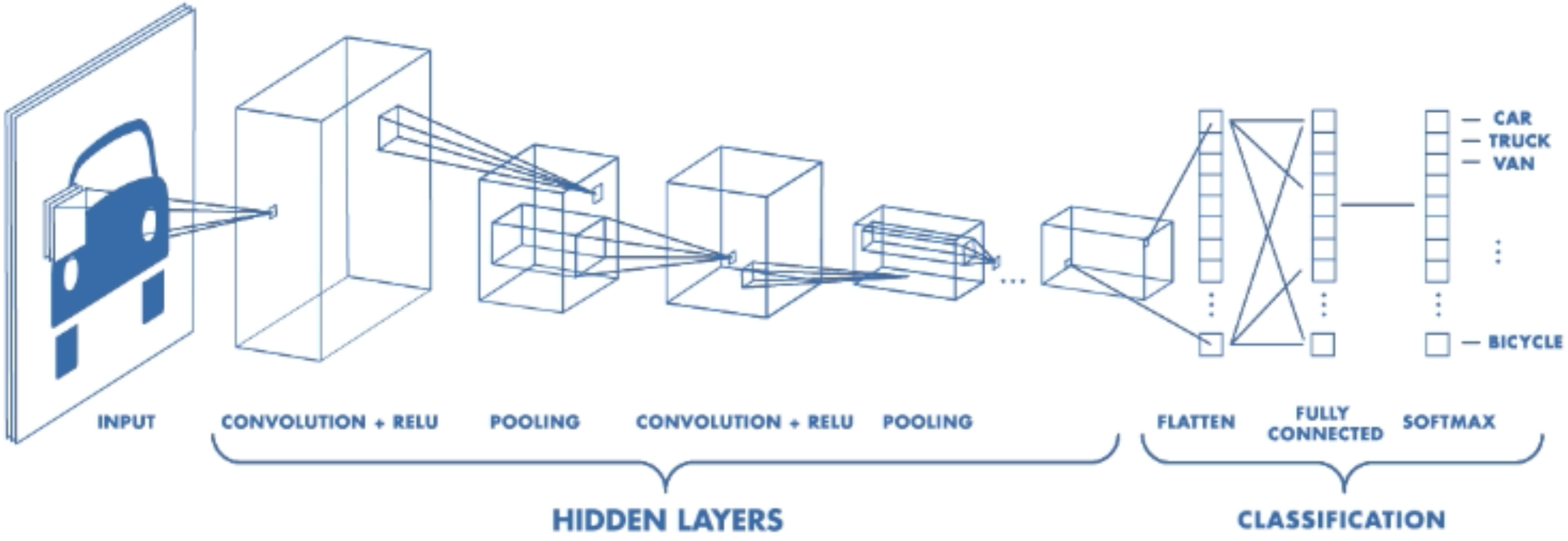
Each layer captures features of the previous layers.

We can use “raw data” (e.g., pixels of an image) as input. The hidden layer are extracting the features.

Design different network architectures to incorporate domain knowledge.

Convolutional Neural Networks

- Captures the localized properties of features
  - Particularly suitable for computer vision (images)
  - Go (AlphaGo) is another famous application of CNN

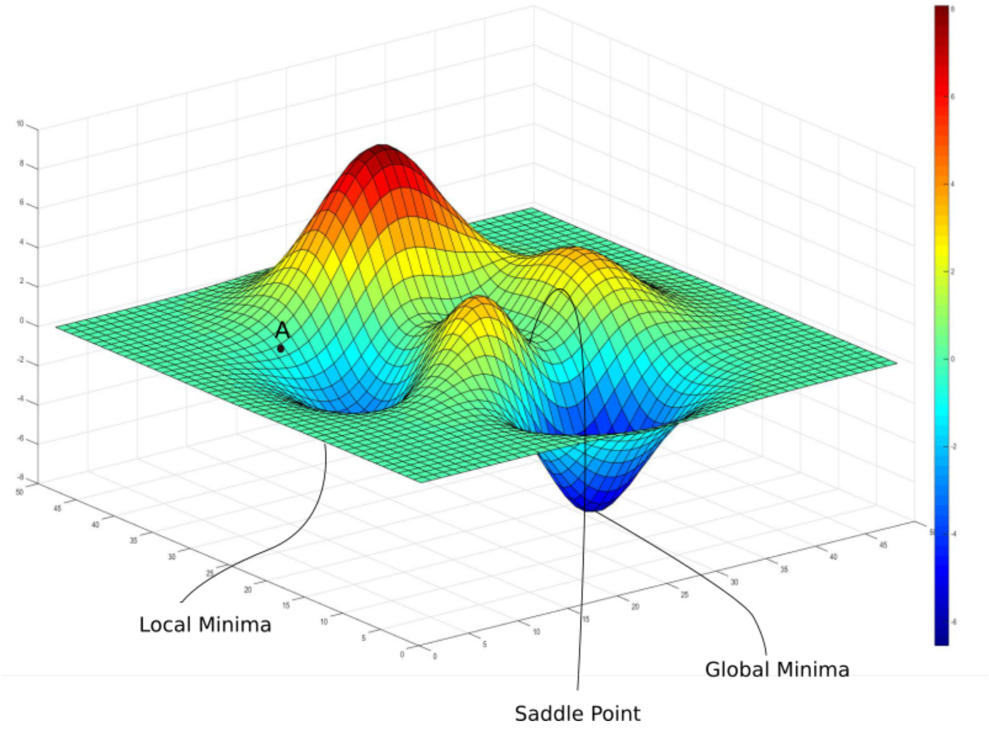


Some Techniques in Improving Deep Learning

- Regularization to mitigate overfitting
  - Weight-based, early stopping, dropout, etc
- Incorporating domain knowledges
  - Network architectures (e.g., Convolutional Neural Nets)
- Improving computation with huge amount of data
  - Hardware architecture to improve parallel computation
- Improving gradient-based optimization
  - Choosing better initialization points

Initialization

Minimizing Error is Nonconvex in Deep Learning

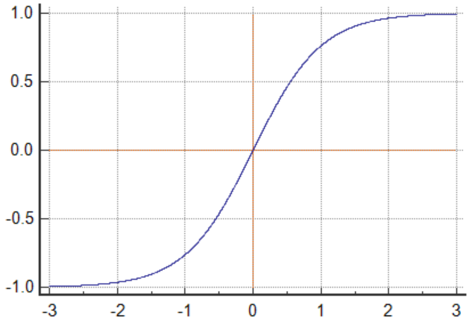


- We mostly adopt gradient-descent-style algorithms for optimization.
- No guarantee to converge to global optimal.
- Need to run it many times.
- Initialization matters!

Vanishing Gradient Problem

- Backpropagation
  - $\frac{\partial e_n(W)}{\partial w_{i,j}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}$
  - $\delta_j^{(\ell)} = \theta'\left(s_j^{(\ell)}\right) \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)}$

- If we use activation function $\theta(s) = \tanh(s)$
  - $\theta'(s) = 1 - \theta(s)^2 < 1$



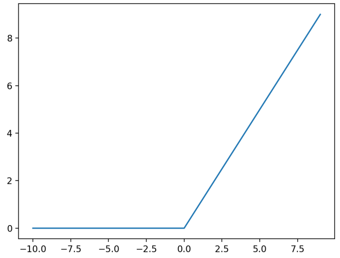
- In deep learning with a lot of layers,
  - the gradient will vanish
  - hard to update the early layers

Vanishing Gradient Problem

- $\delta_j^{(\ell)} = \theta'\left(s_j^{(\ell)}\right) \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)}$

- There is also a corresponding "exploding gradient problem"

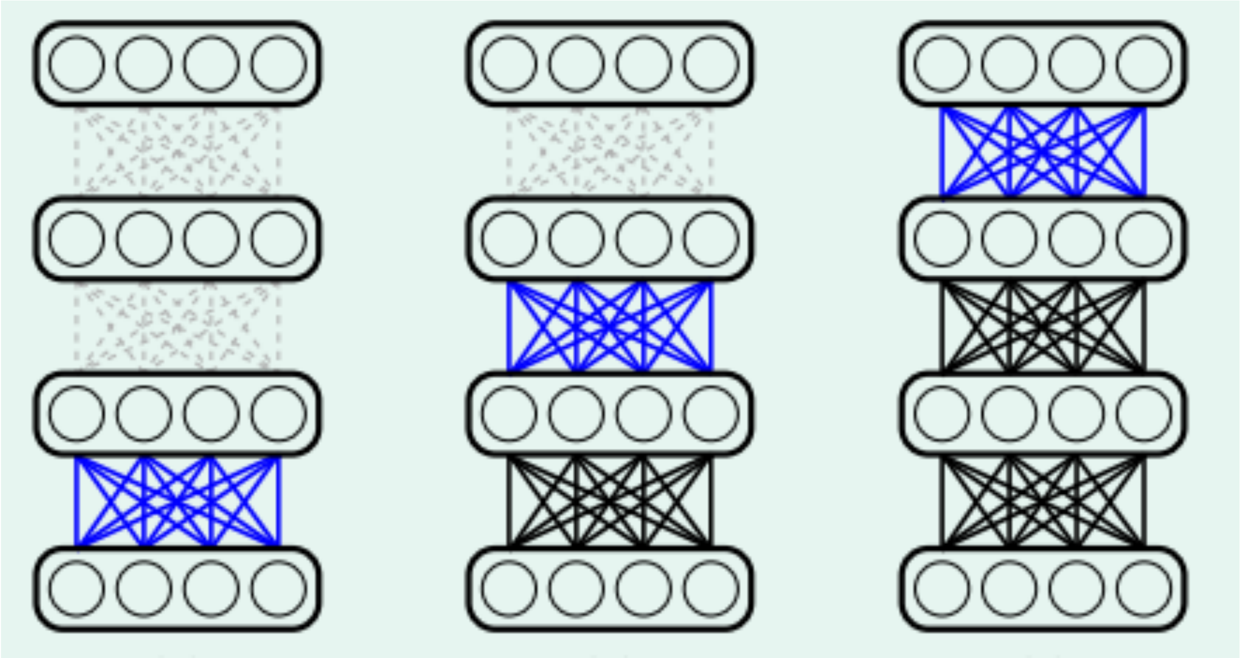
- What can we do
  - Choose more suitable activation functions
    - One common choice is Rectified Linear Unit (ReLU) and its variant
      - $\theta(s) = \max(0, s)$



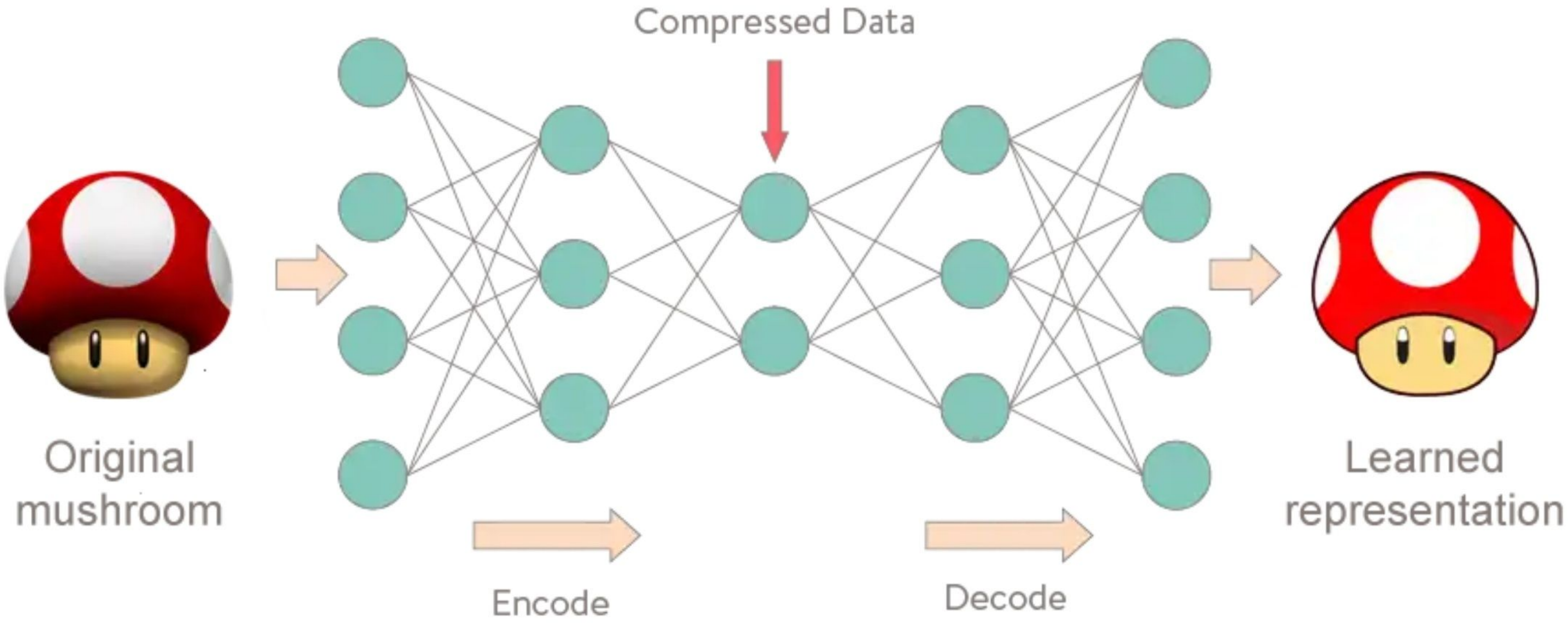
  - Choose better initialization

Initialization

- Hard to initialize the entire network well.
- Intuition: Initialize the weights layer by layer such that each layer preserves the properties of the previous layer.



Autoencoder



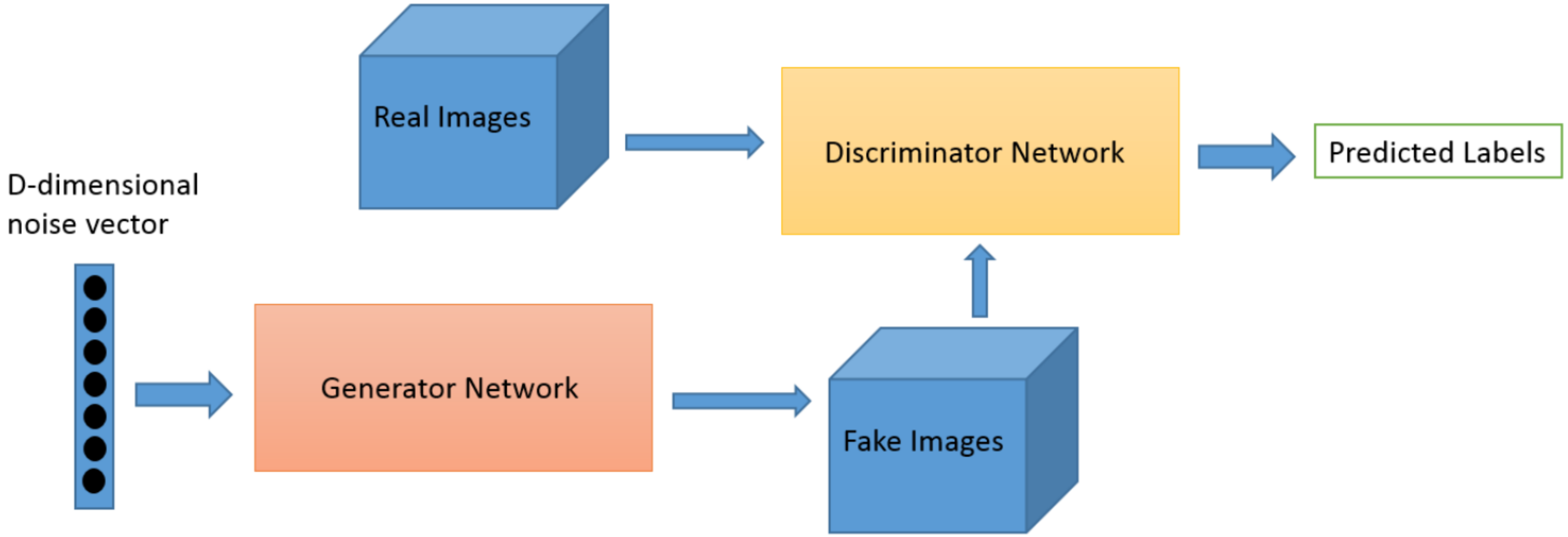
Unsupervised learning!

https://www.meetup.com/Le-Mans-School-of-AI/photos/29159837/477498782/

Generative Adversarial Nets (GAN)

A Competition: Generator vs Discriminator

- Discriminator wants to correctly classify the images (true images or not)
- Generator wants to generate images that discriminator can’t classify



Credit: O’Reilly

https://thisPersonDoesNotExist.com/