CSE 417T
Introduction to Machine Learning

Lecture 21

Instructor: Chien-Ju (CJ) Ho

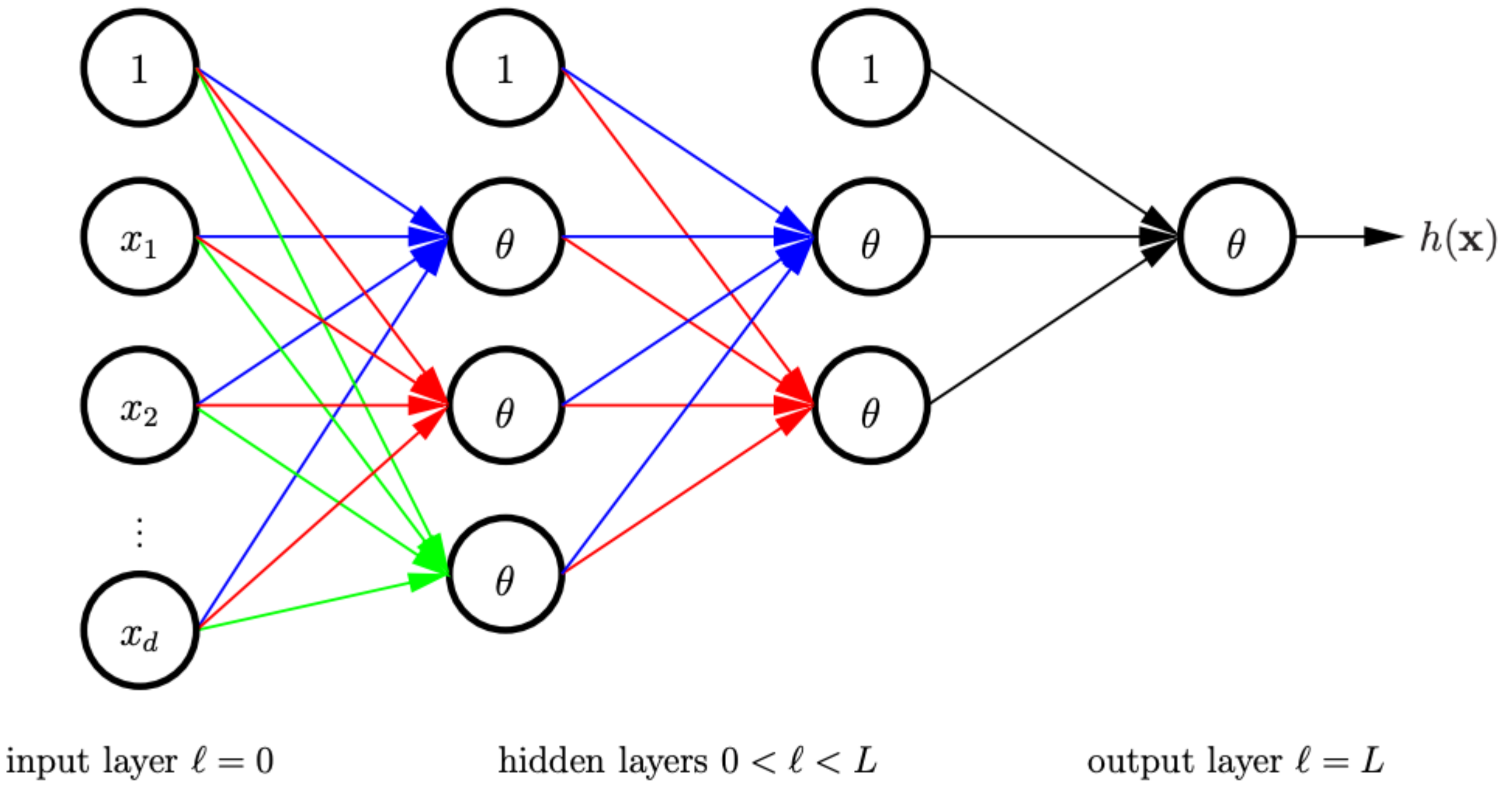
Logistics

- Homework 5 is due Dec 2 (Friday)

- Exam 2 will be on Dec 8 (Thursday)
  - Will focus on the topics in the second half of the semester
    - Note though knowledge is cumulative, so we still assume you know the concepts earlier
  - Format / logistics will be similar with what we have in Exam 1
    - Timed exam (75 min) during lecture time in the classroom
    - Closed-book exam with 2 letter-size cheat sheets allowed (4 pages in total)
      - No format limitations (it can be typed, written, or a combination)
  - Dec 6 (Tuesday) will be a review lecture

Recap

Neural Networks



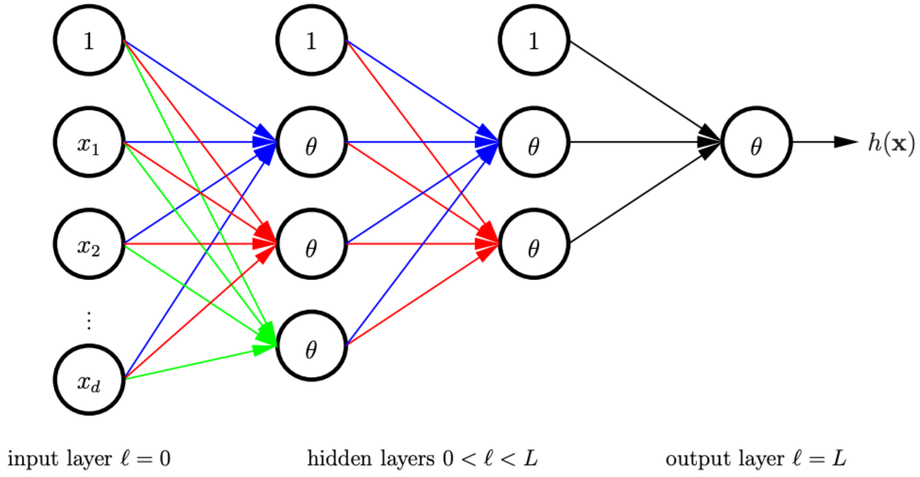
θ: activation function

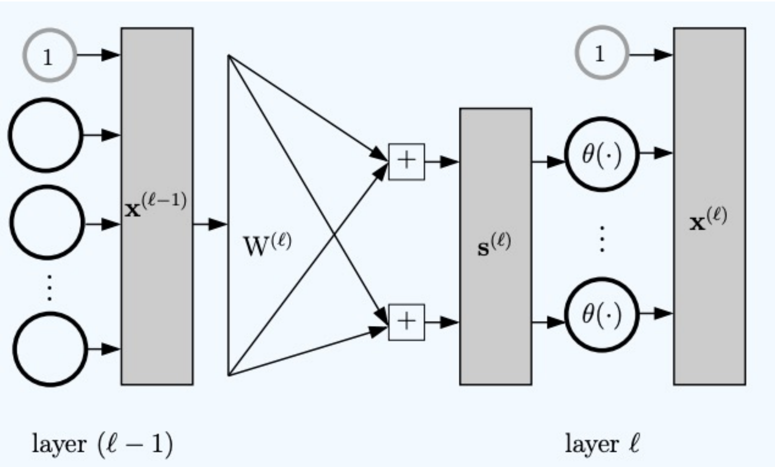
(Specify the "activation" of the neuron)

We mostly focus on feed-forward network structure

Notations of Neural Networks (NN)

- Notations:
  - $\ell = 0 \text{ to } L$: 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
    - $x_j^{(\ell)} = \theta\left(s_j^{(\ell)}\right)$





Evaluate $h(\vec{x})$ - Forward Propagation

- A NN 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 $h(\mathbf{x})$:

$\mathbf{x}^{(0)} \leftarrow \mathbf{x}$ **[Initialization]**

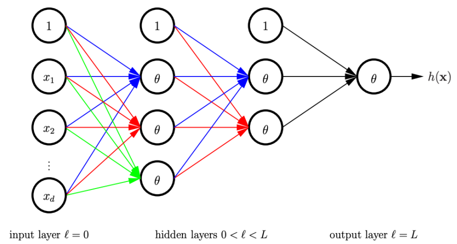
for $\ell = 1$ to L **do** **[Forward Propagation]**

  $\mathbf{s}^{(\ell)} \leftarrow (\mathrm{W}^{(\ell)})^{\mathrm{T}}\mathbf{x}^{(\ell-1)}$

  $\mathbf{x}^{(\ell)} \leftarrow \begin{bmatrix} 1 \\ \theta(\mathbf{s}^{(\ell)}) \end{bmatrix}$

end for

$h(\mathbf{x}) = \mathbf{x}^{(L)}$ **[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.

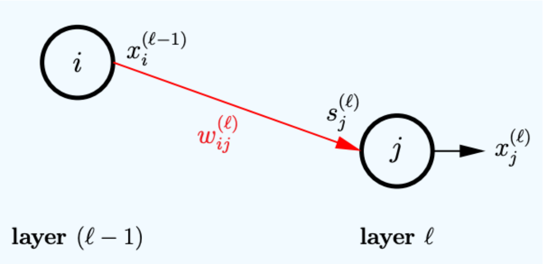
How to Learn NN From Data?

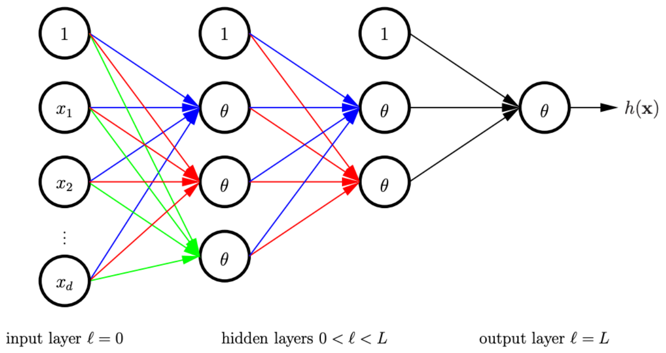
- Given D, how to learn the weights $W = \left\{w_{i,j}^{(\ell)}\right\}$?
- Intuition: Minimize $E_{in}(W) = \frac{1}{N}\sum_{n=1}^{N} e_n(W)$
- How?
  - Gradient descent: $W(t+1) \leftarrow W(t) - \eta \nabla_W E_{in}(W)$
  - Stochastic gradient descent $W(t+1) \leftarrow W(t) - \eta \nabla_W e_n(W)$
- Key step: we need to be able to evaluate the gradient…
  - Not trivial 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 $\delta_j^{(\ell)}$ (Using dynamic programming idea)
  - Boundary conditions
    - The output layer (assume regression)
      - $\delta_1^{(L)} = 2\left(s_1^{(L)} - y_n\right)$ (generalizable to other differentiable error)
  - Backward recursive formulation
    - $\delta_j^{(\ell)} = \sum_{k=1}^{d^{(\ell+1)}} \frac{\partial e_n(W)}{\partial s_k^{(\ell+1)}} \frac{\partial s_k^{(\ell+1)}}{\partial x_j^{(\ell)}} \frac{\partial x_j^{(\ell)}}{\partial s_j^{(\ell)}} = \sum_{k=1}^{d^{(\ell+1)}} \delta_k^{(\ell+1)} w_{j,k}^{(\ell+1)} \theta'\left(s_j^{(\ell)}\right)$
  - Backward propagation

Backpropagation 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

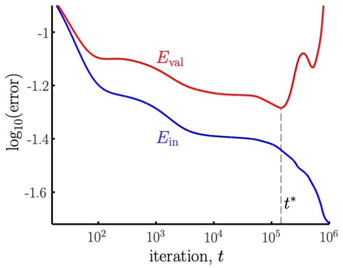
Discussion

- 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

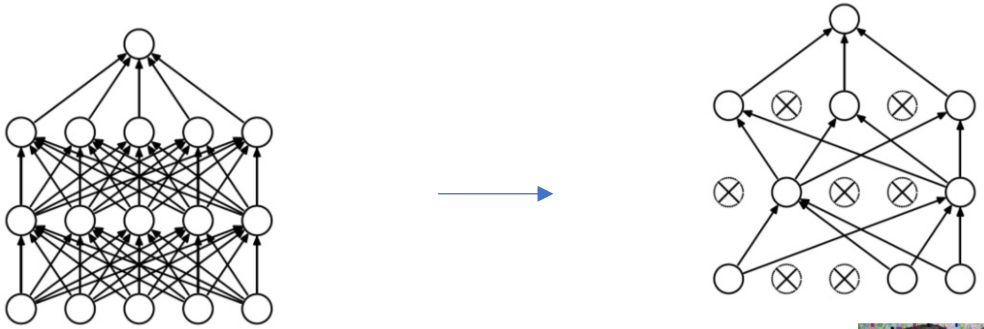
- Potential approaches:
  - Run it many times
  - Choose better initializations (the choice of initialization matters)

Regularizations in Neural Networks

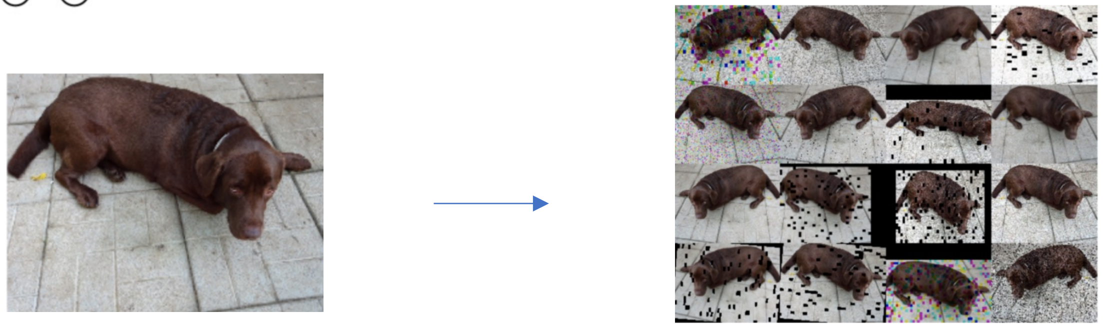
- Weight-based regularization
- Early stopping



- Dropout



- Adding noises
  (Data augmentation)



Today's Lecture

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

Let me know if you spot errors.

Deep Learning

ImageNet Challenge 2012

Task 1: Classification

Car

- Predict a class label

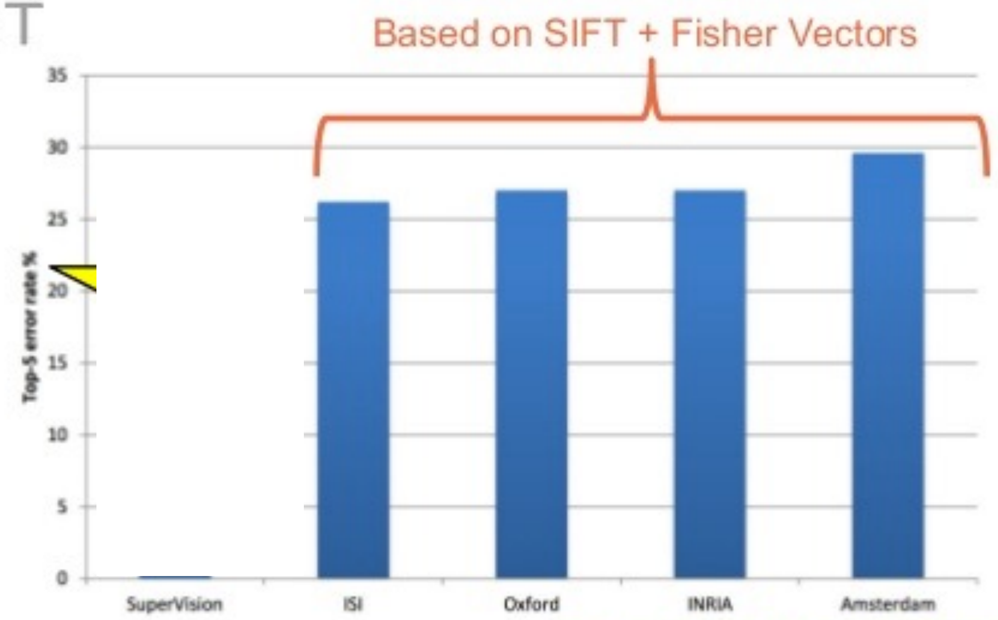
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 50% of training.

ImageNet Challenge

Image Classification 2012

IMAGENET

Slide credit:
Rob Fergus (NYU)



Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2014). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575.* [web]

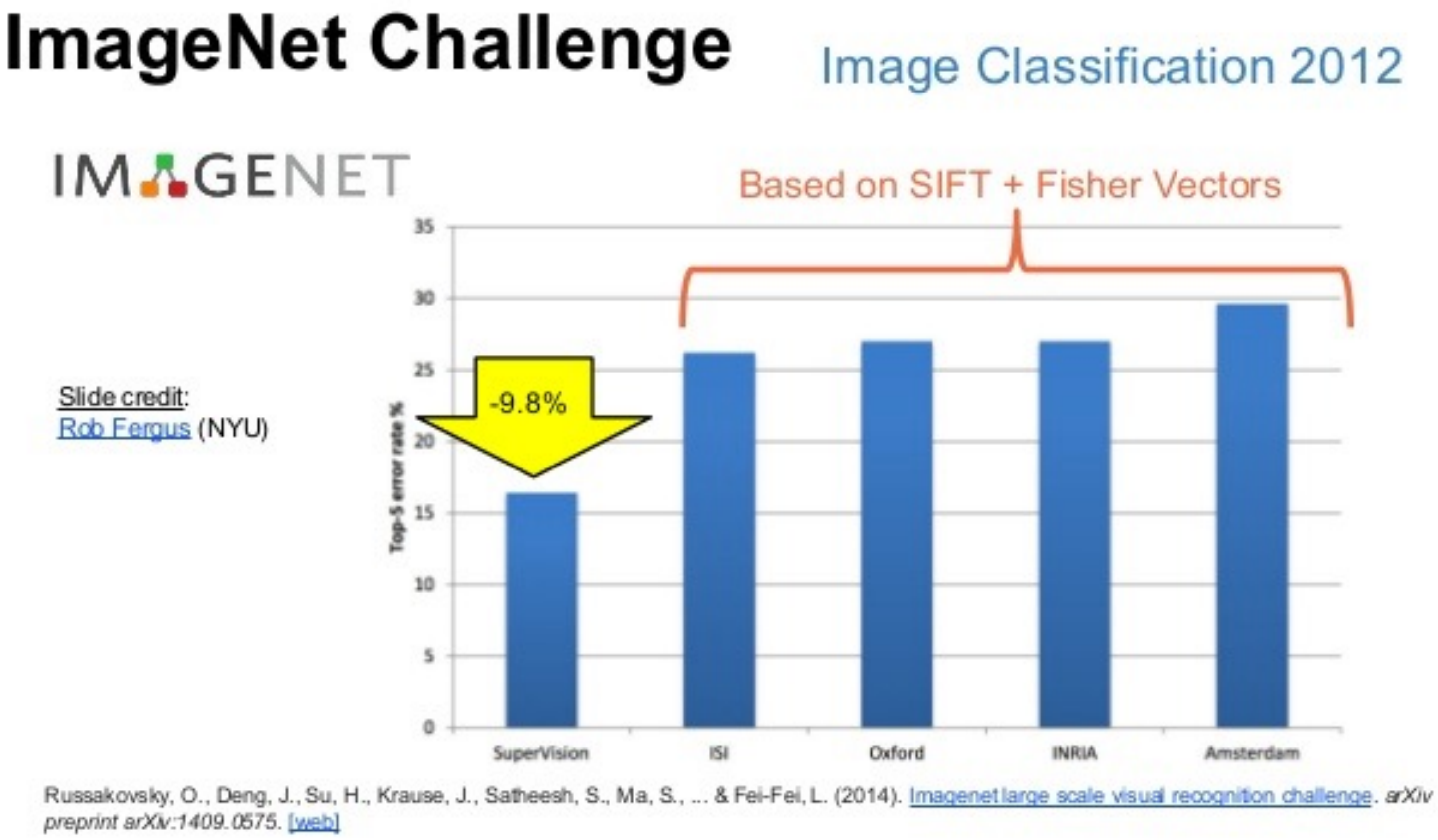
ImageNet Challenge 2012

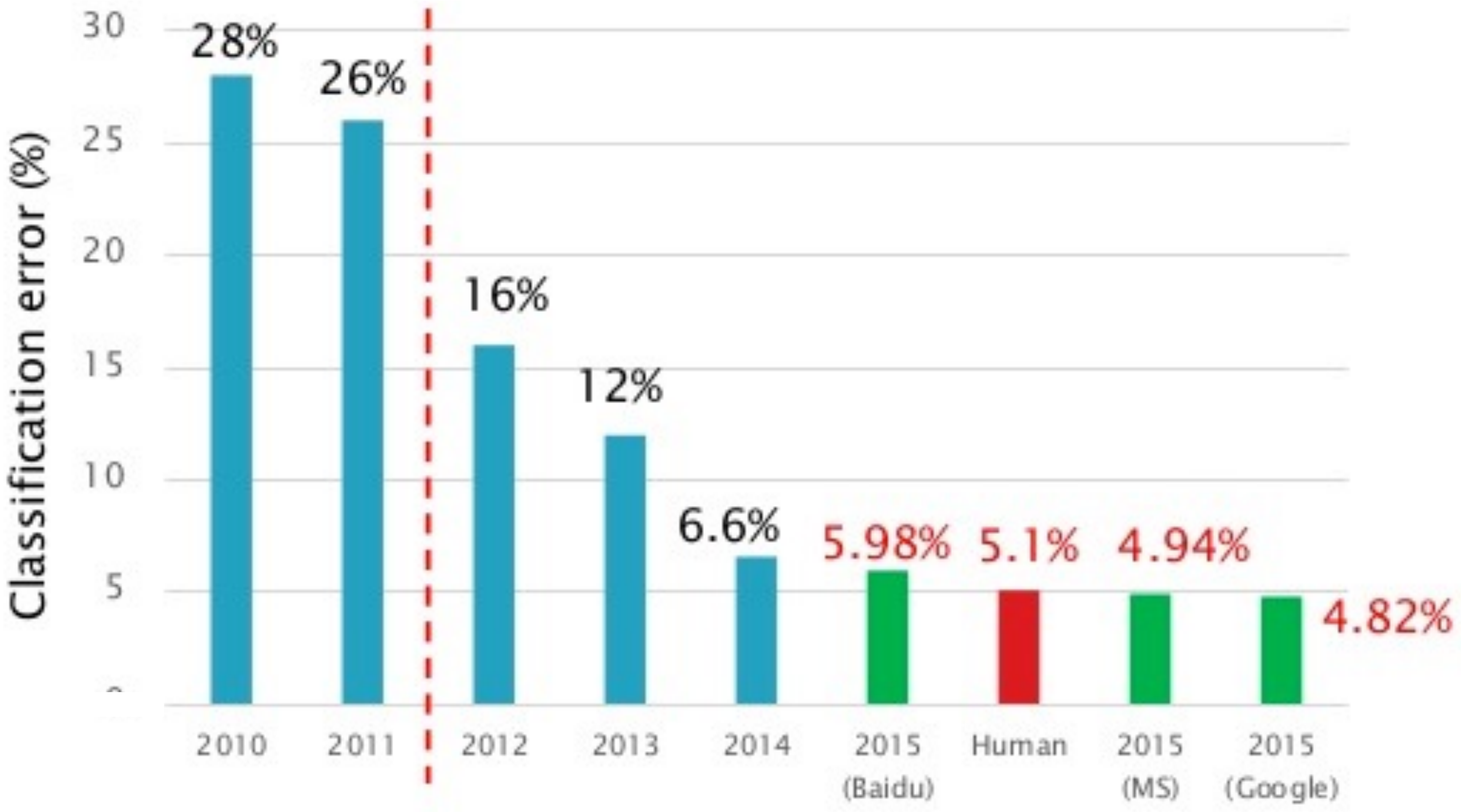
Task 1: Classification

Car

- Predict a class label

- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 50% of training.





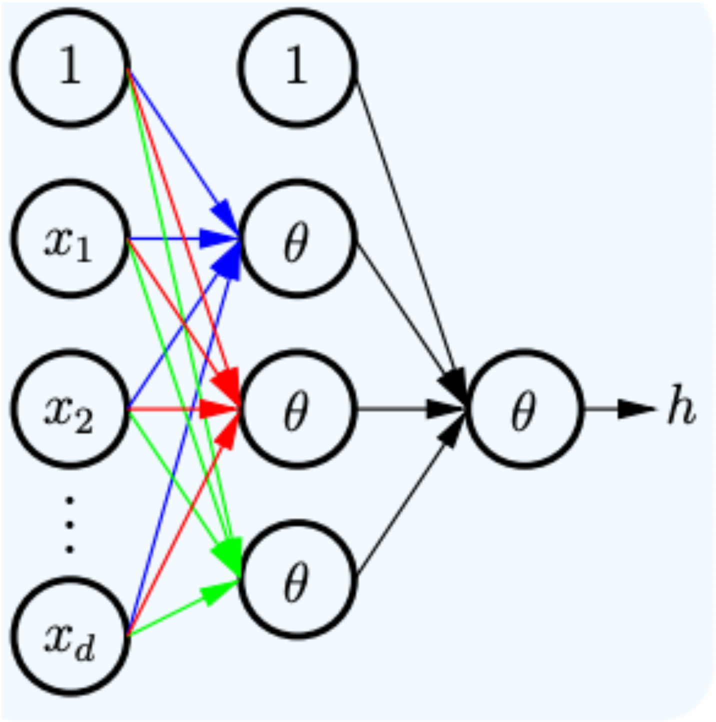
He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", arXiv, 2015.
Ioffe et al., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv, 2015.

What is “Deep” Learning

Neural networks with many layers

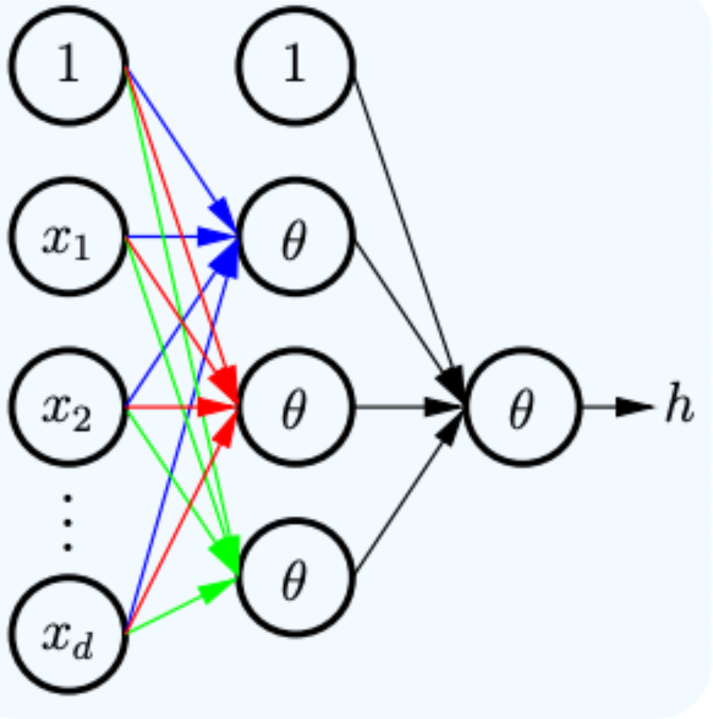
Single Hidden-Layer Neural Network

- How do we write a hypothesis in a single-hidden layer NN mathematically?



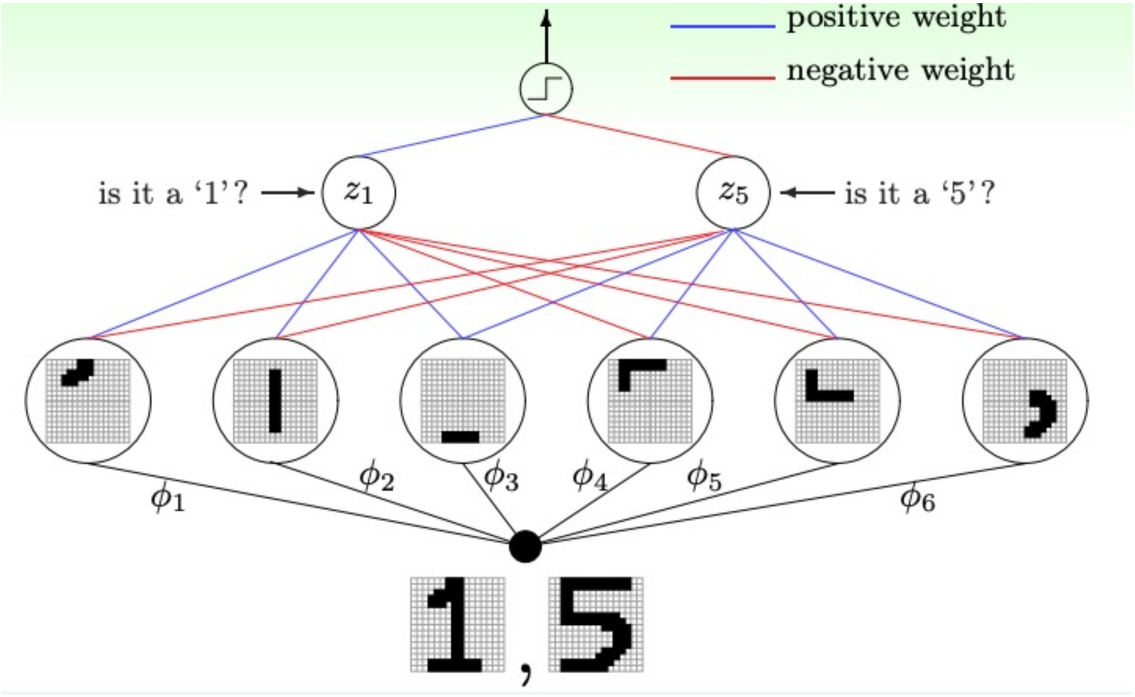
Single Hidden-Layer Neural Network

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



Deep Neural Network

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



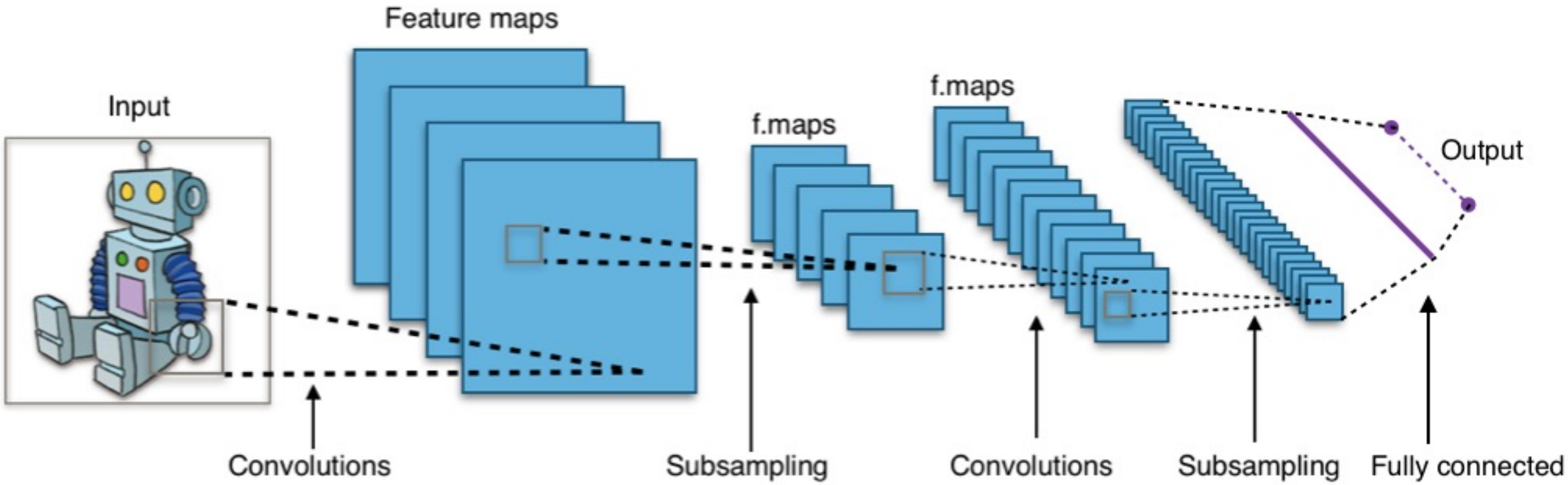
Each layer captures features of the previous layers.

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 (CNN)

- Captures the localized properties of features hierarchically



Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 2 & 2 & 1 & 0 \\ \hline
0 & 2 & 4 & 4 & 2 & 0 \\ \hline
0 & 1 & 3 & 3 & 1 & 0 \\ \hline
0 & 1 & 2 & 3 & 1 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 \\ \hline
\end{array}
\quad * \quad
\begin{array}{|c|c|c|}
\hline
0 & 1 & 0 \\ \hline
1 & -4 & 1 \\ \hline
0 & 1 & 0 \\ \hline
\end{array}
$$

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

*

0	1	0
1	-4	1
0	1	0

=

0			

$$
\begin{aligned}
&(0*0)+(0*1)+(0*0)+\\
&(0*1)+(1*-4)+(2*1)+\\
&(0*0)+(2*1)+(4*0)\\
&=0
\end{aligned}
$$

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

$$\begin{bmatrix} 0 & \boxed{0 & 0 & 0} & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 2 & 2 & 1 & 0 \\ \hline
0 & 2 & 4 & 4 & 2 & 0 \\ \hline
0 & 1 & 3 & 3 & 1 & 0 \\ \hline
0 & 1 & 2 & 3 & 1 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 \\ \hline
\end{array}
\quad * \quad
\begin{array}{|c|c|c|}
\hline
0 & 1 & 0 \\ \hline
1 & -4 & 1 \\ \hline
0 & 1 & 0 \\ \hline
\end{array}
\quad = \quad
\begin{array}{|c|c|c|c|}
\hline
0 & -1 & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
 & & & \\ \hline
\end{array}
$$

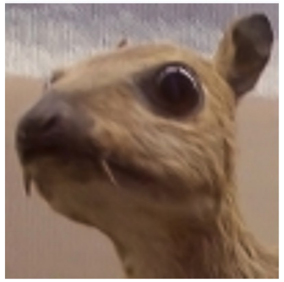
$$
\begin{aligned}
&(0 * 0) + (0 * 1) + (0 * 0) + \\
&(1 * 1) + (2 * -4) + (2 * 1) + \\
&(2 * 0) + (4 * 1) + (4 * 0) \\
&= -1
\end{aligned}
$$

Convolutional Filters

- A convolutional filter is like a matrix version of a dot product.

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}
*
\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}
=
\begin{bmatrix} 0 & -1 & -1 & 0 \\ -2 & -5 & -5 & -2 \\ 2 & -2 & -1 & 3 \\ -1 & 0 & -5 & 0 \end{bmatrix}
$$

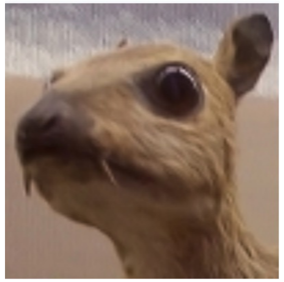
Convolutional Filters

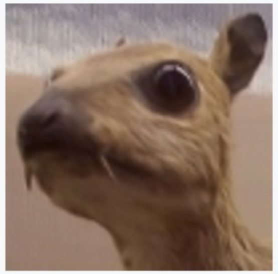


Operation	Kernel ω	Image result g(x,y)
	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

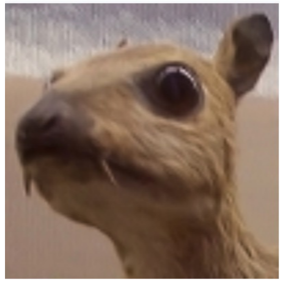
Convolutional Filters

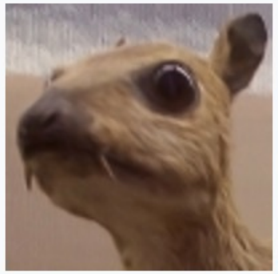
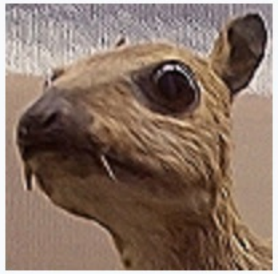


Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

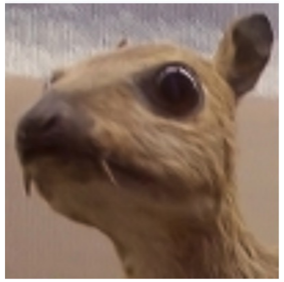
Convolutional Filters

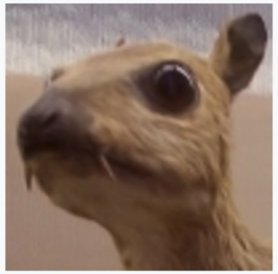
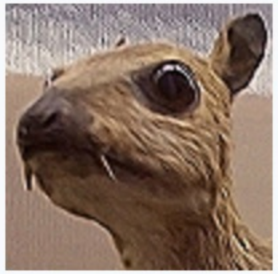


Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

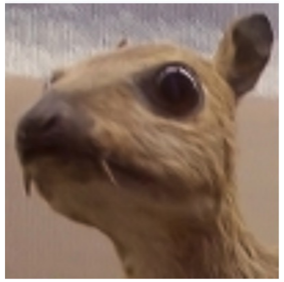
Convolutional Filters

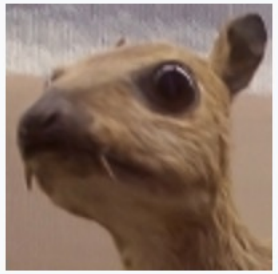
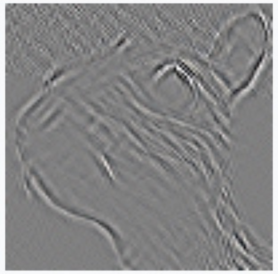
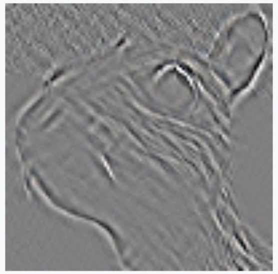


Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

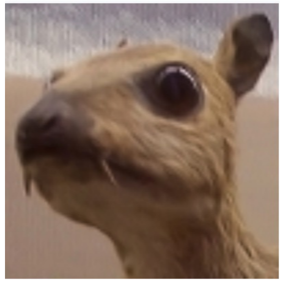
Convolutional Filters



Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

Convolutional Filters



Operation	Kernel ω	Image result g(x,y)
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

Connection to Neural Networks

- Convolutions can be represented by a network structure
  - Nodes in the previous layer are only connected to “adjacent” nodes in the next layer.
  - Many of the weights have the same value.

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}
*
\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}
=
\begin{bmatrix} \boxed{0} & -1 & -1 & 0 \\ -2 & -5 & -5 & -2 \\ 2 & -2 & -1 & 3 \\ -1 & 0 & -5 & 0 \end{bmatrix}
$$

Pooling Layers

- Commonly used in convolutional neural networks.

  A subsampling / down-sampling process:

  - Combines multiple adjacent nodes into a single node

0	-1	-1	0
-2	-5	-5	-2
2	-2	2	3
-1	1	-5	0

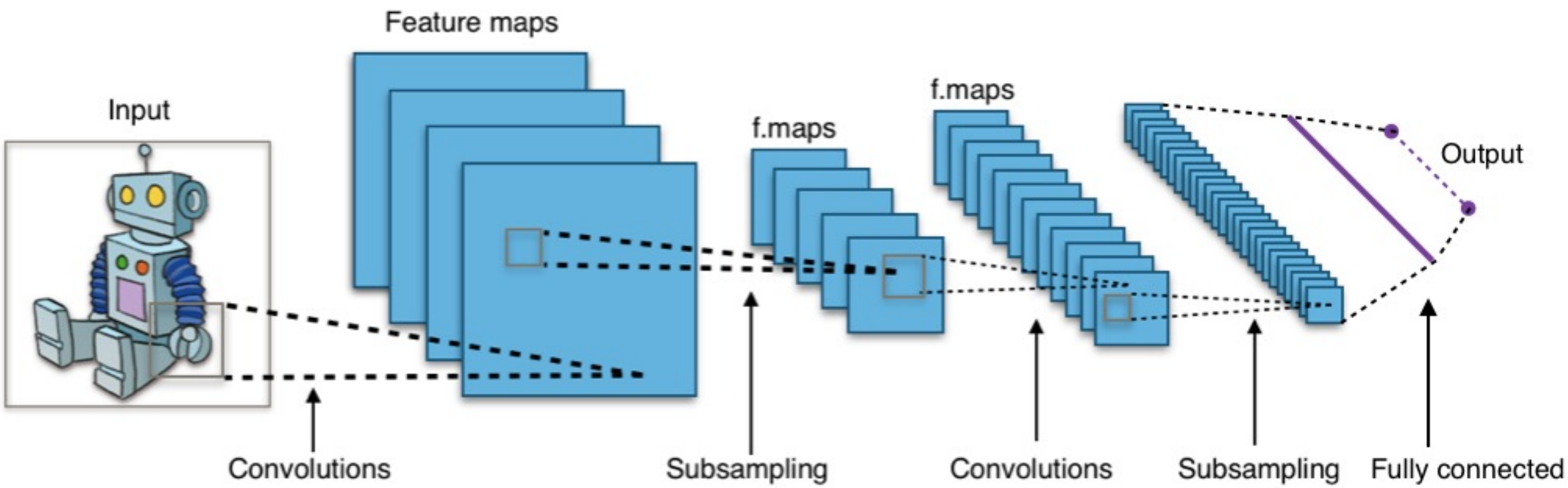
average pooling

-2	-2
0	0

- Reduce the dimensionality of input. More robust to noise.

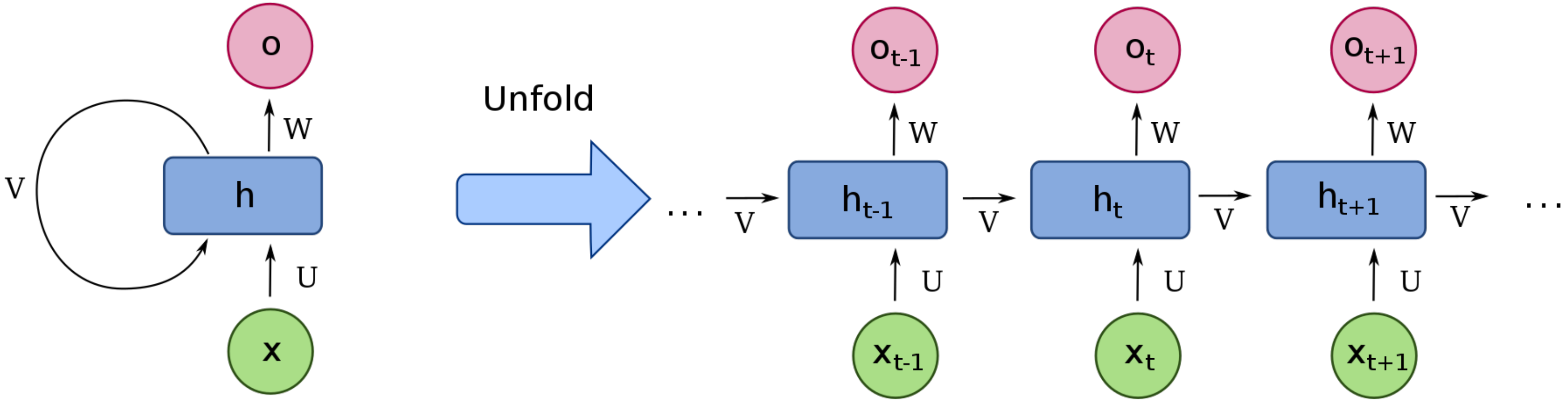
Convolutional Neural Networks (CNN)

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



Another Example Network Structure [Safe to Skip for the Exam]

- Recurrent Neural Network (RNN)
  - Aim to deal with time-series data, such as natural language processing
  - Using hidden layers to store temporal information
  - Allow previous outputs to be used as inputs and keep hidden states



[Images from Wikipedia]

Some Techniques in Improving Deep Learning

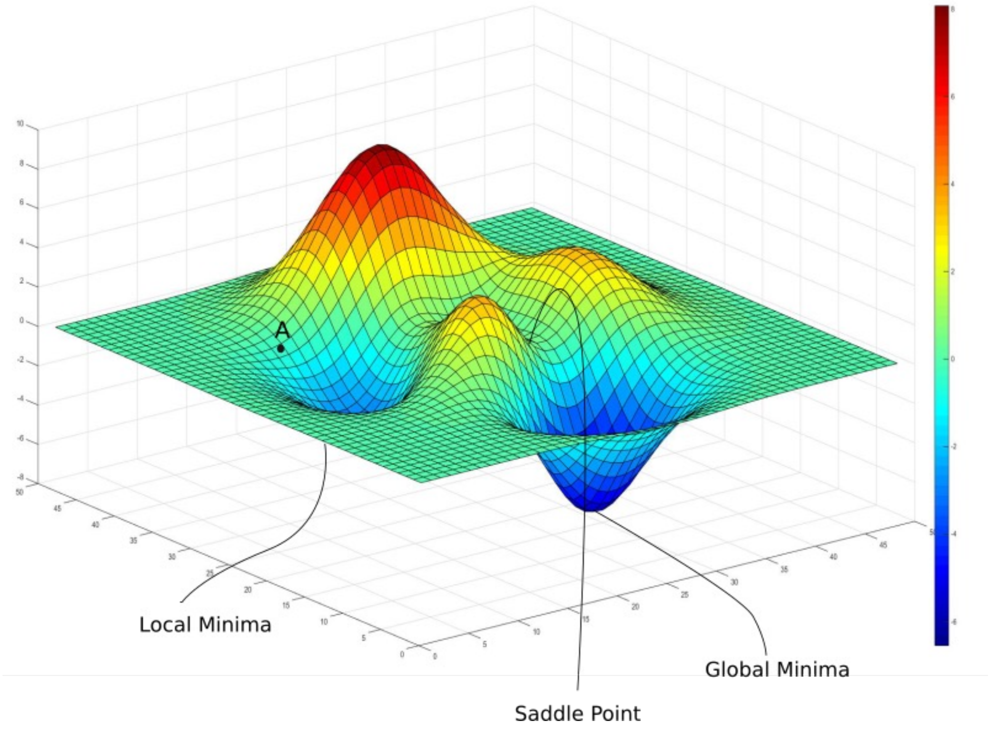
- Regularization to mitigate overfitting
  - Weight-based, early stopping, dropout, etc
- Incorporating domain knowledges
  - Network architectures (e.g., Convolutional Neural Nets)
- Improving computation with huge amount of data
  - Hardware architecture to improve parallel computation
- Improving gradient-based optimization
  - See more in LFD 7.5 (Steepest descent, conjugate gradient, higher-order optimization)
  - Choosing better initialization points

Initialization

Why initialization matters in deep learning

- Error is nonconvex in NN
- Vanishing/exploding gradient problem

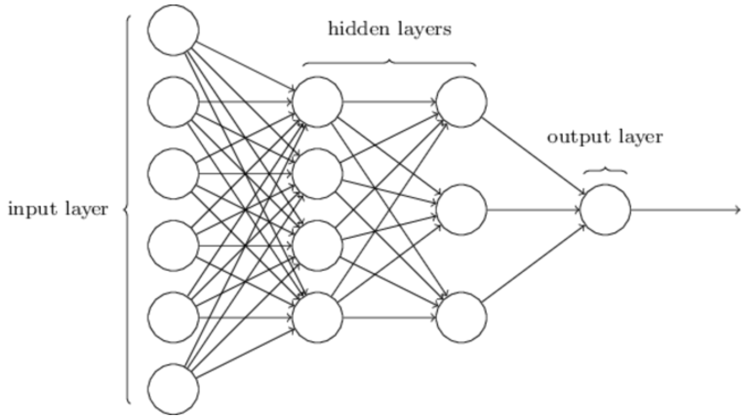
Error is Nonconvex in Neural Networks



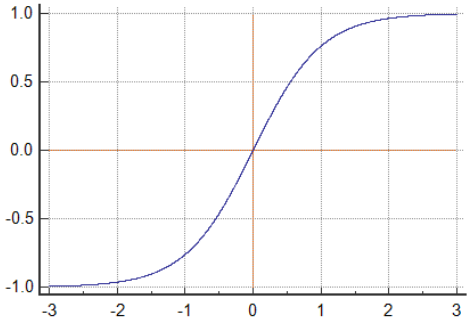
- We mostly adopt gradient-descent-style algorithms for optimization
- No guarantee to converge to global optimal
- Could run it many times
- Initialization matters

Vanishing Gradient Problem

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



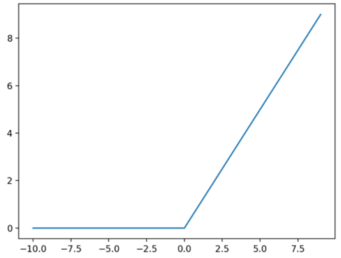
- If we use activation function $\theta(s) = \tanh(s)$
  - $\theta'(s) = 1 - \theta(s)^2 < 1$



- In deep learning with a lot of layers,
  - the gradient might 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 different activation functions
    - One common choice is Rectified Linear Unit (ReLU) and its variant
      - $\theta(s) = \max(0, s)$
  - Choose better initialization
    - Many approaches



Weight Initialization

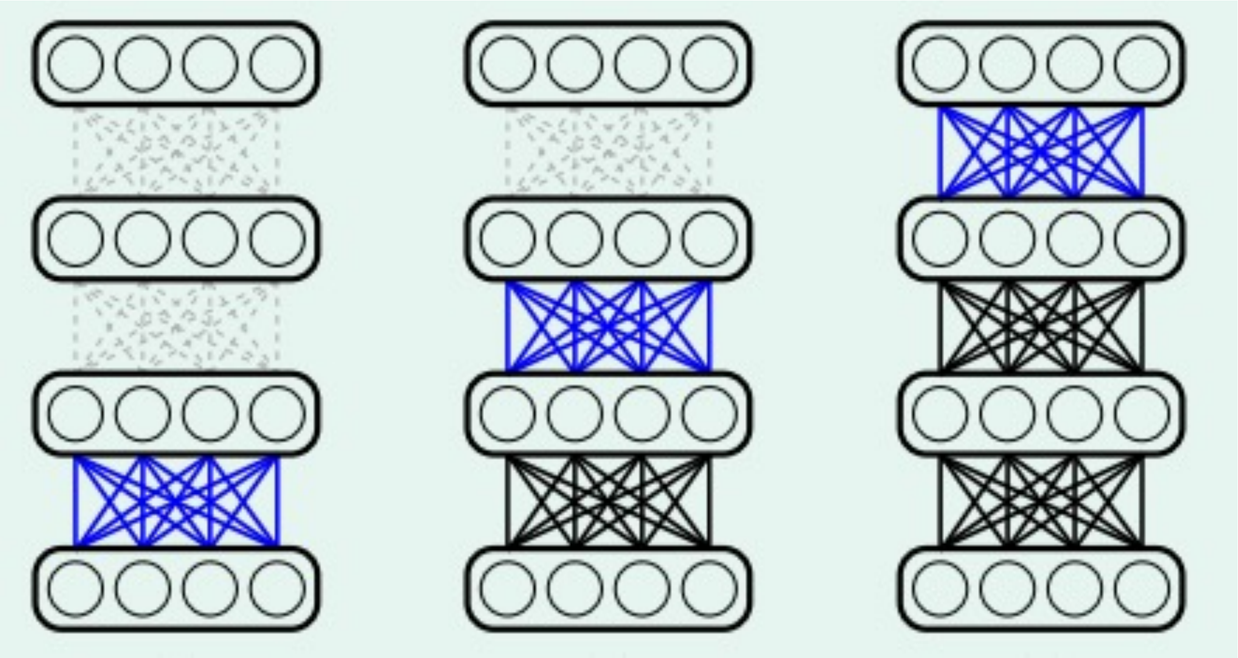
- Initializing weights to all 0 is a bad idea
  - Q6b of HW1
  - Hint: Look at the backpropagation formulation
- Randomly Initializing weights to regions so that vanishing/exploding gradients are less likely to happen
  - Activation-function dependent
    - e.g., Xavier initialization for tanh

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

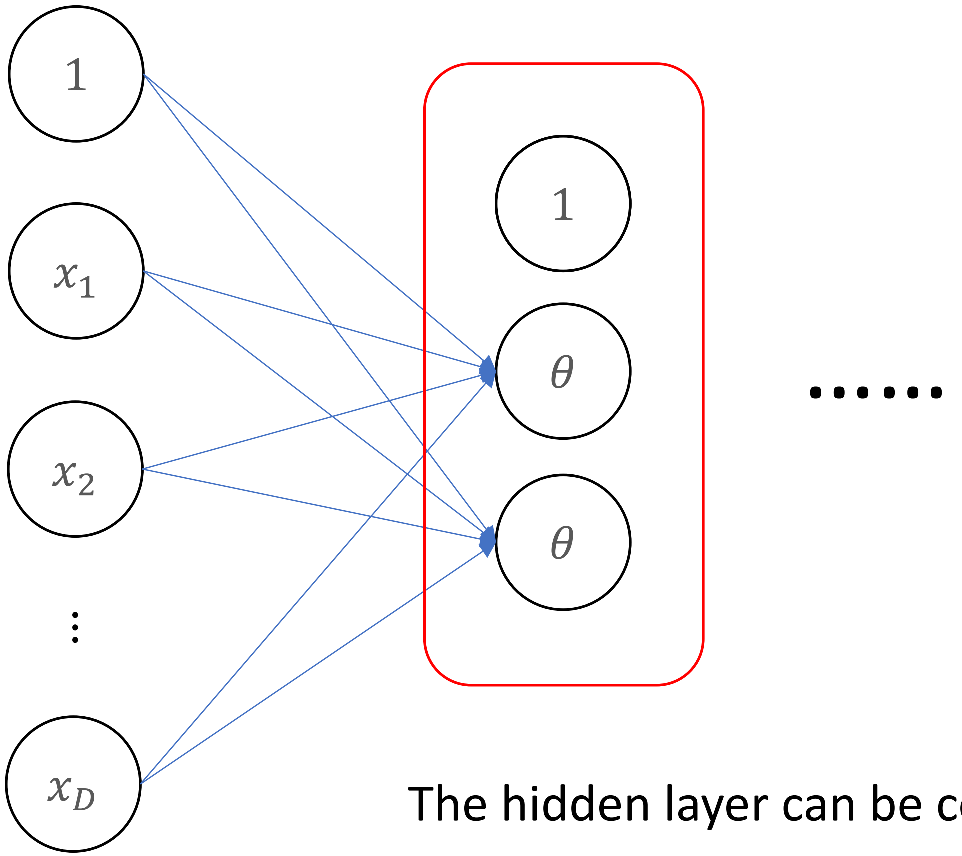
- Learning the initialization that might be closer to the optimal
  - E.g., using autoencoder

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.



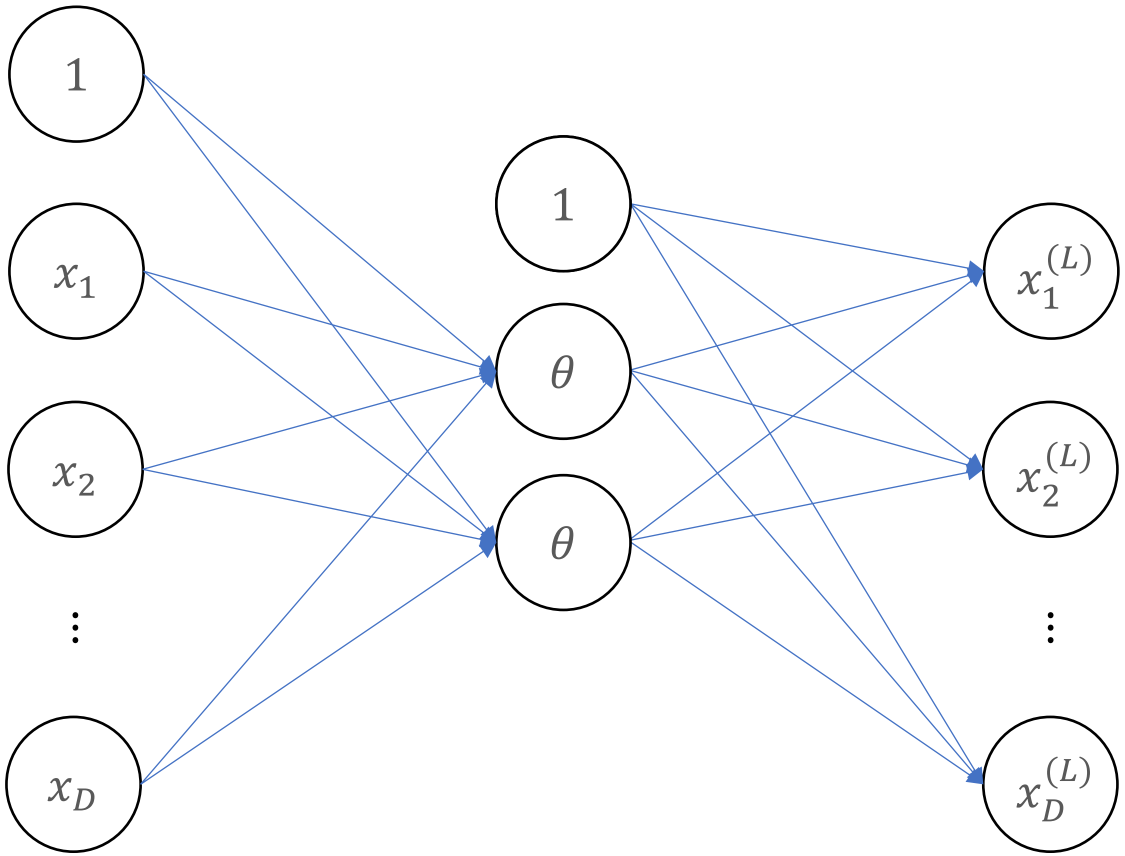
Autoencoders



The hidden layer can be considered as a feature transform.

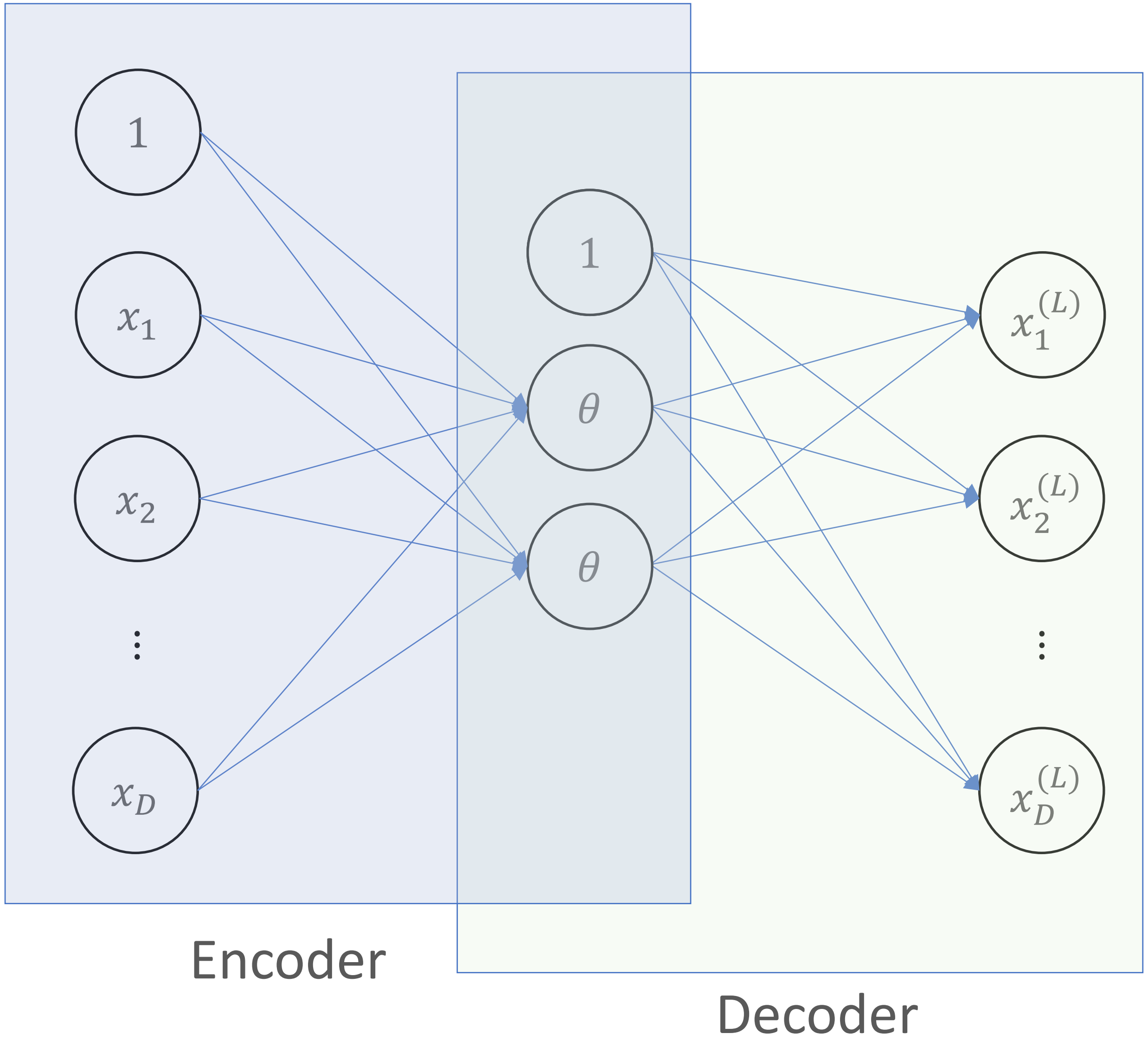
Can we ensure this transformation contain as much information about the original input as possible?

Autoencoders

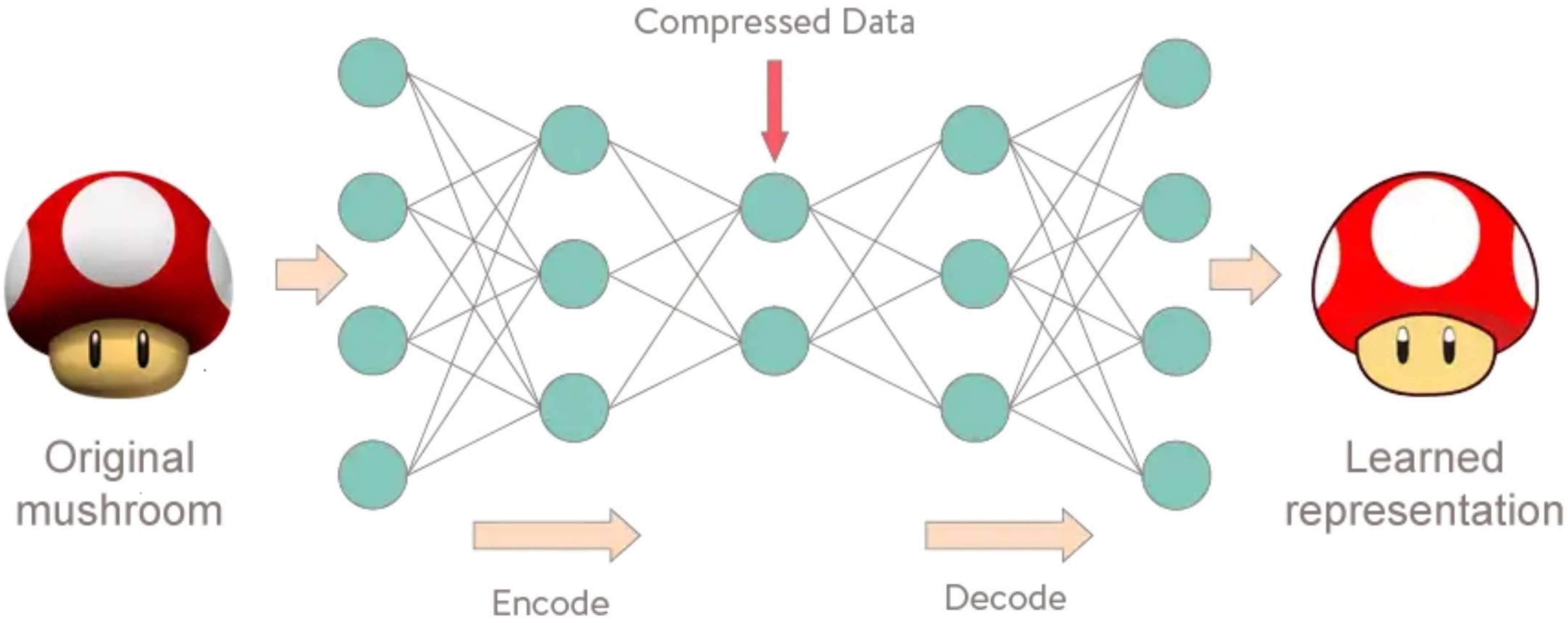


Minimize error of $\left\|\vec{x} - \overrightarrow{x^L}\right\|$

Autoencoders



Autoencoders



Unsupervised learning!

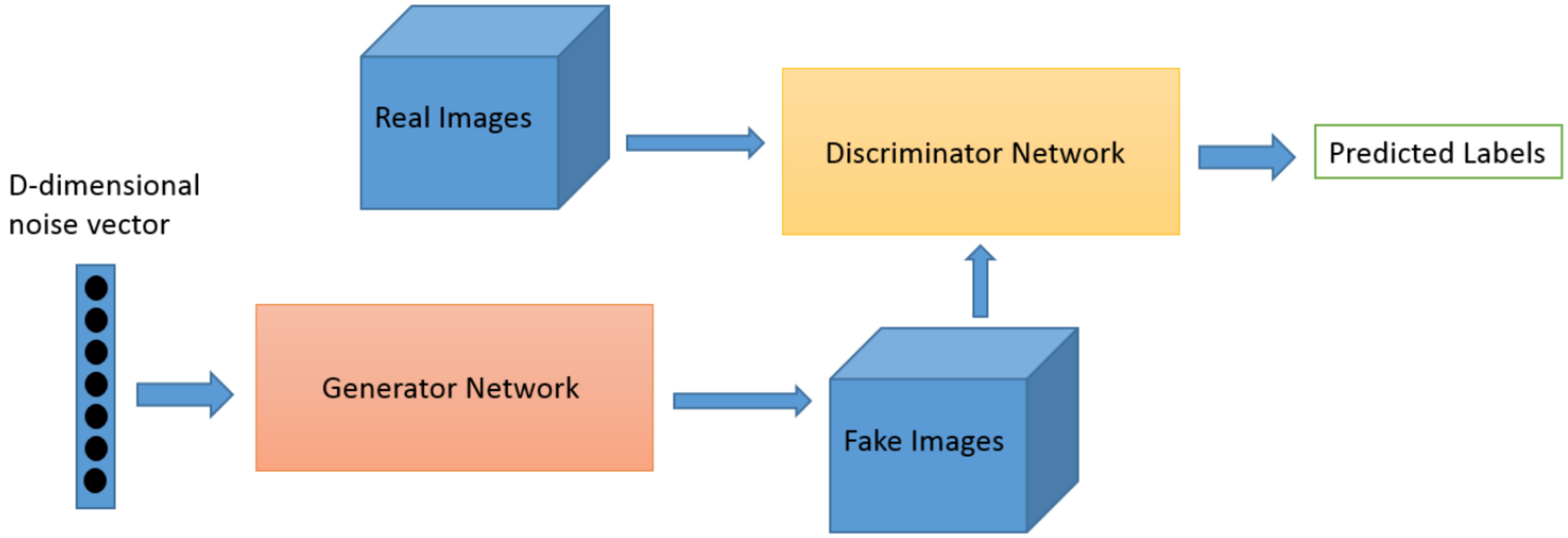
https://www.meetup.com/Le-Mans-School-of-AI/photos/29159837/477498782/

Cool Stuffs for Deep Learning

[Safe to Skip for the exam]

Generative Adversarial Nets (GAN)

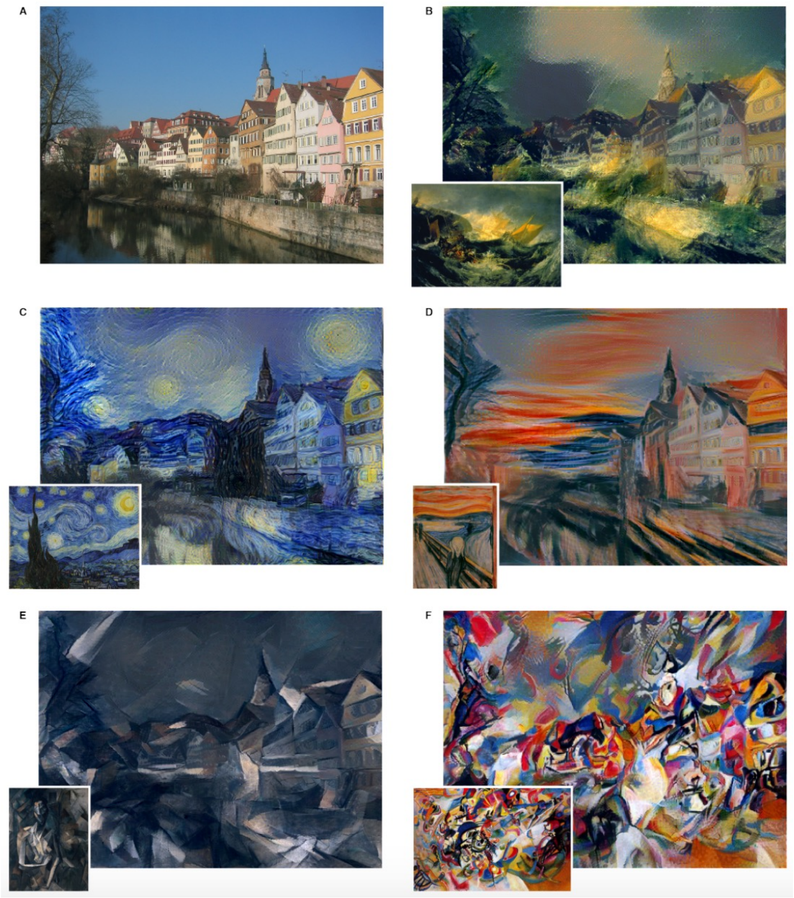
- A Competition: Generator vs Discriminator
  - Discriminator wants to correctly classify the images (true images or not)
  - Generator wants to generate images that discriminator can't classify



[Safe to Skip for the Exam]

https://thisPersonDoesNotExist.com/

Style Transfer

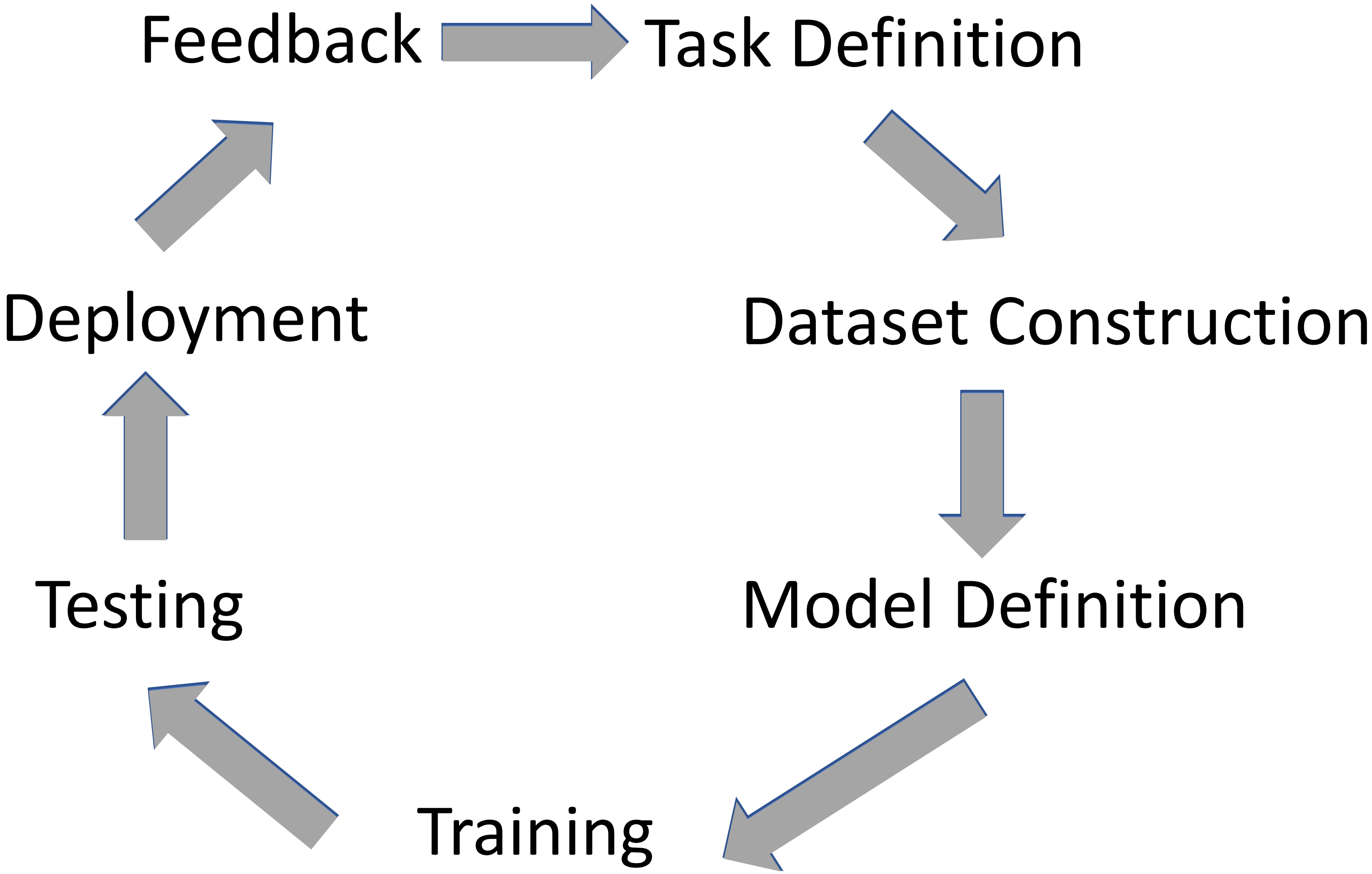


- Informal intuitions:
  - Recall that we can treat hidden layers as feature transforms
  - Deep learning is learning representation of data
  - How to achieve style transfer:
    - Learn a content representation for an image using hidden layers
    - Learn a style representation for an image using hidden layers
    - Compute an image that jointly minimizes the distance from the content image's content representation and the style image's style representation
    - https://arxiv.org/pdf/1508.06576.pdf

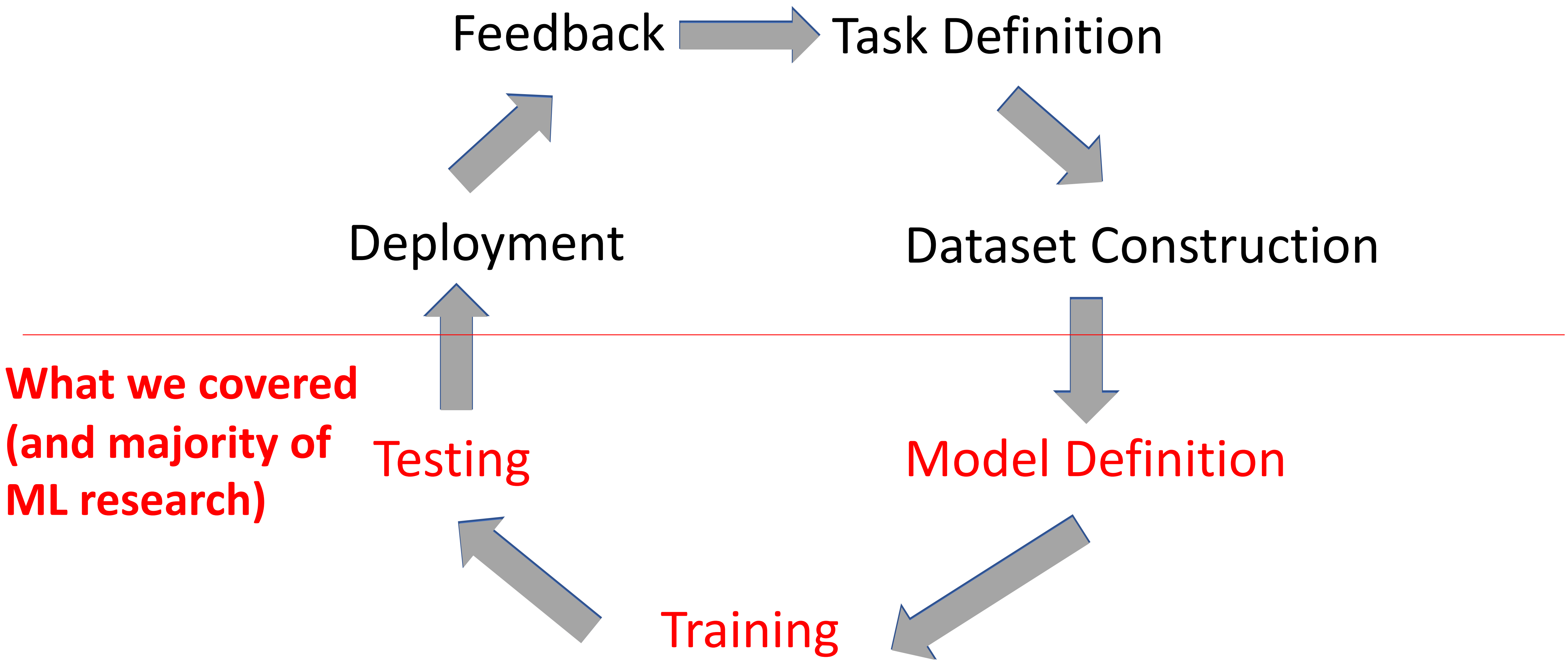
[Safe to Skip for the Exam]

Machine Learning Life Cycle

Machine Learning Lifecycle

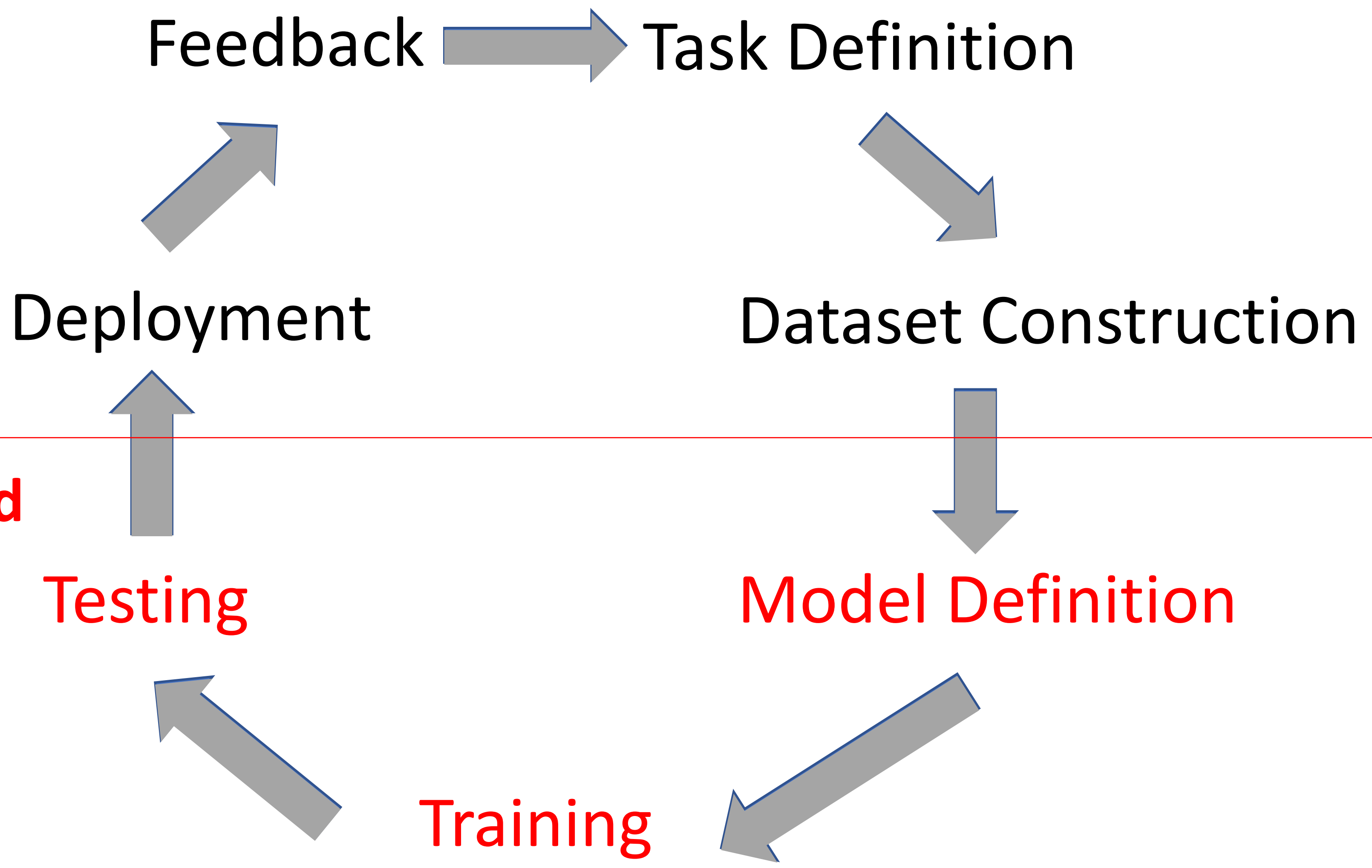


Machine Learning Lifecycle



Machine Learning Lifecycle

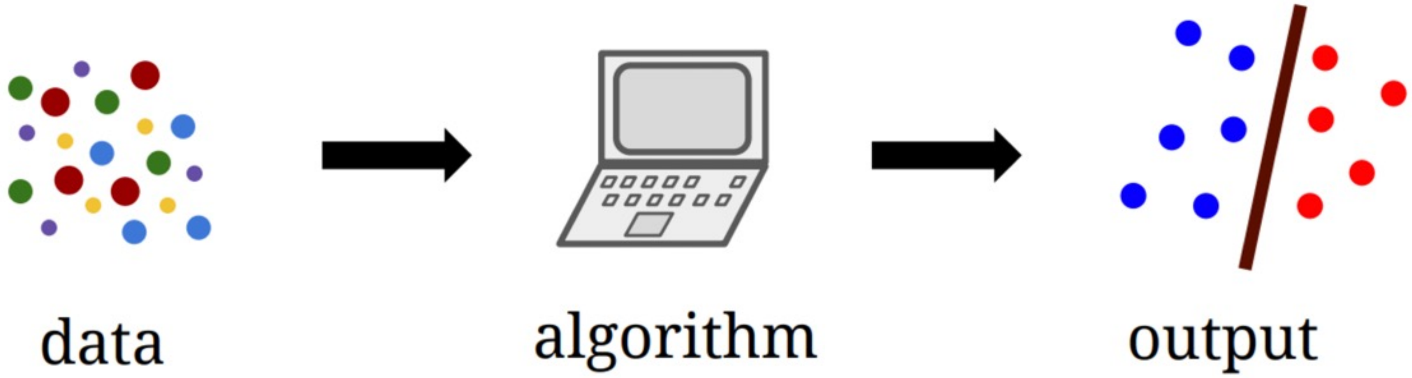
To have "positive" impacts, we need to be careful in every stage



What we covered (and majority of ML research)

Supervised Learning

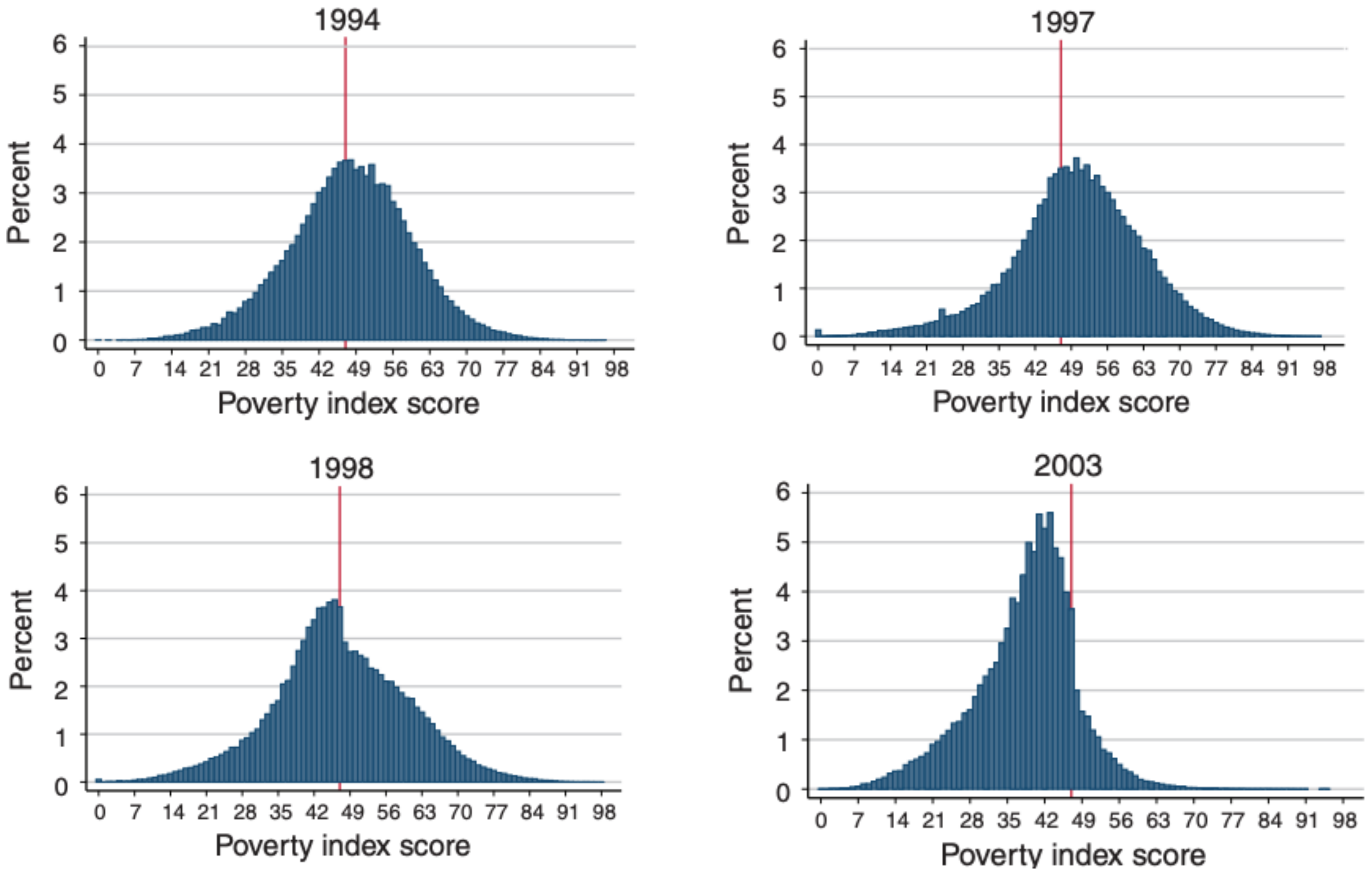
- Standard setup of (supervised) machine learning



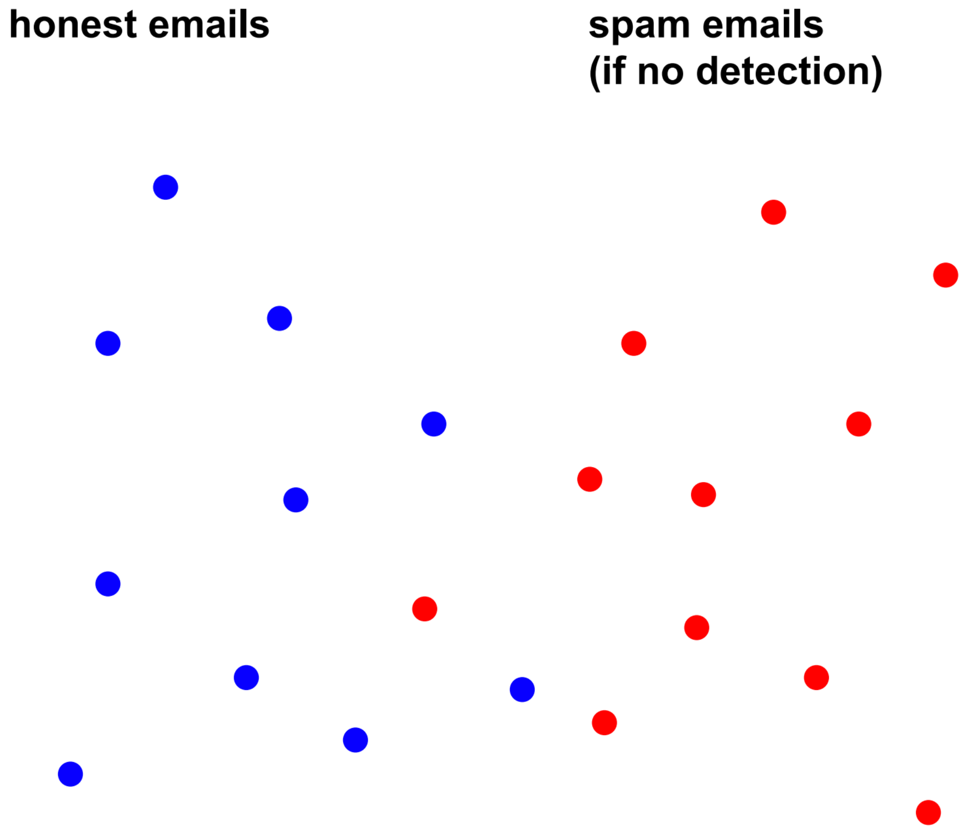
  - Finding patterns from the given training datasets
  - Use the pattern to make predictions on new testing data

- Fundamental assumption:
  - Training and testing data points are i.i.d. drawn from the same distribution

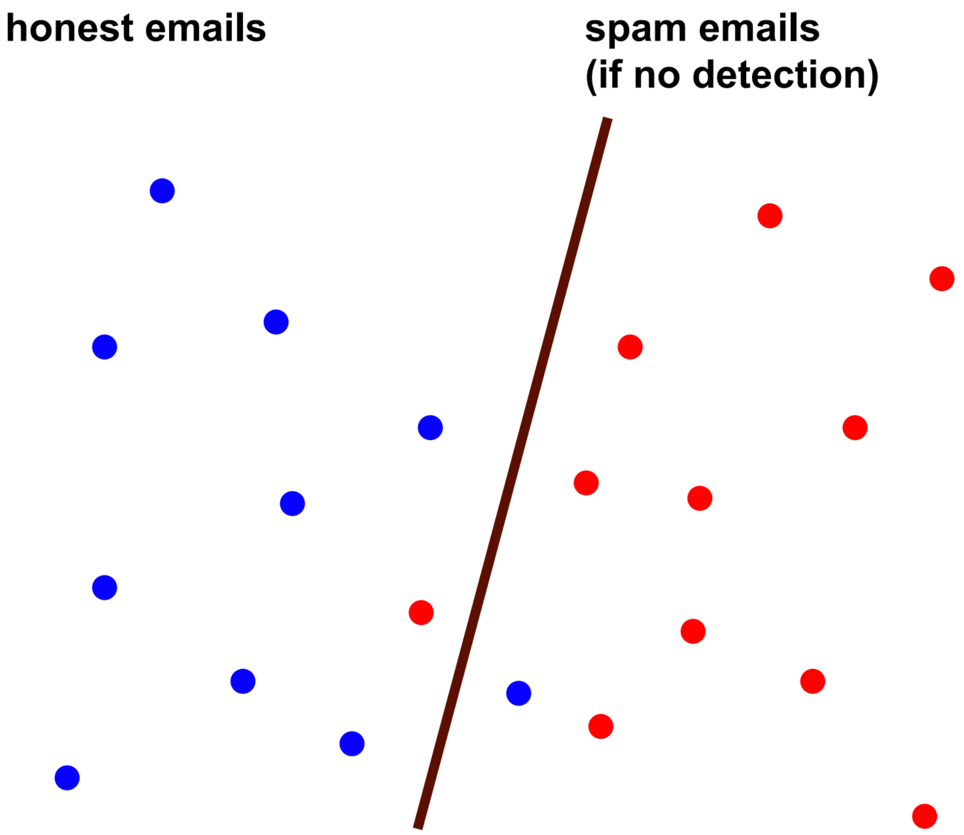
Social Program Eligibility [Camacho and Conover, 2012]



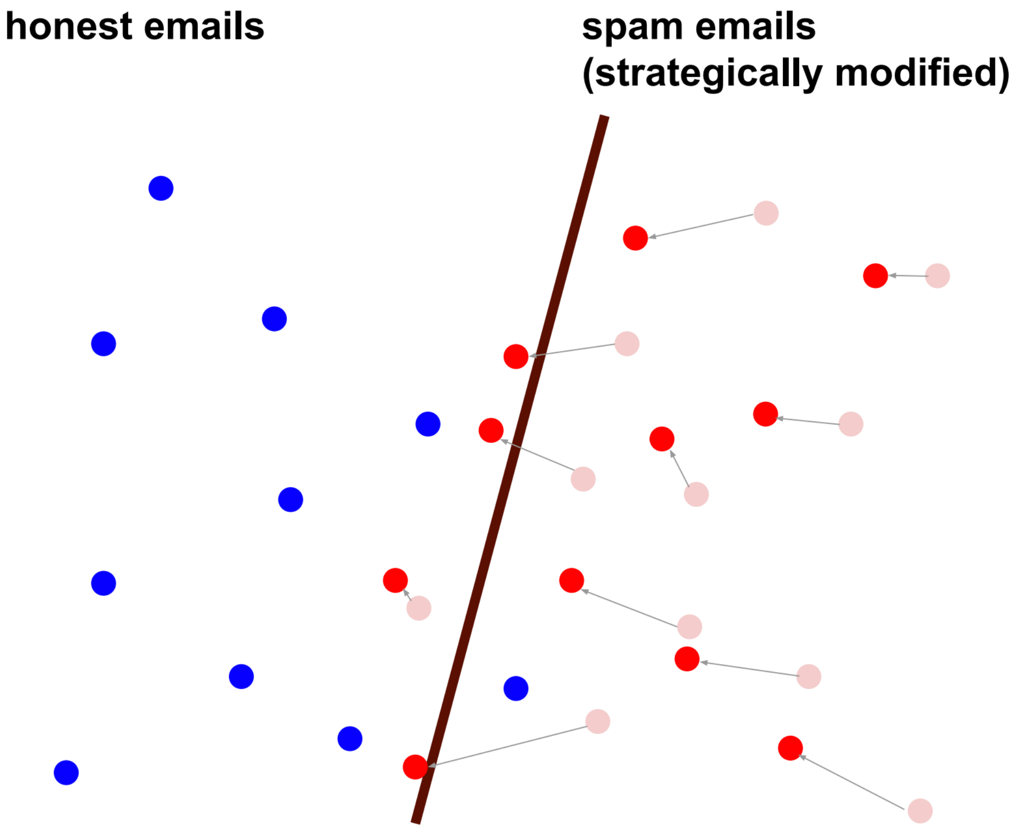
A More ML Example: Spam Filter



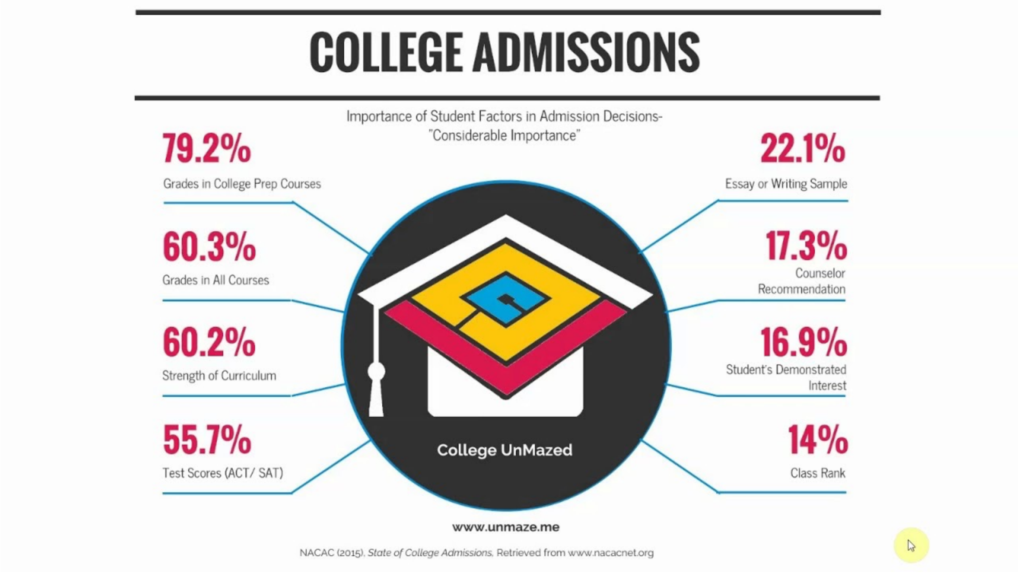
A More ML Example: Spam Filter

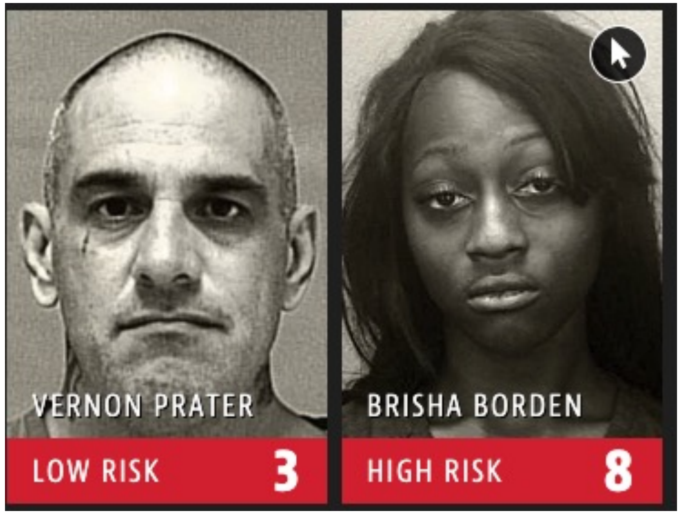


A More ML Example: Spam Filter

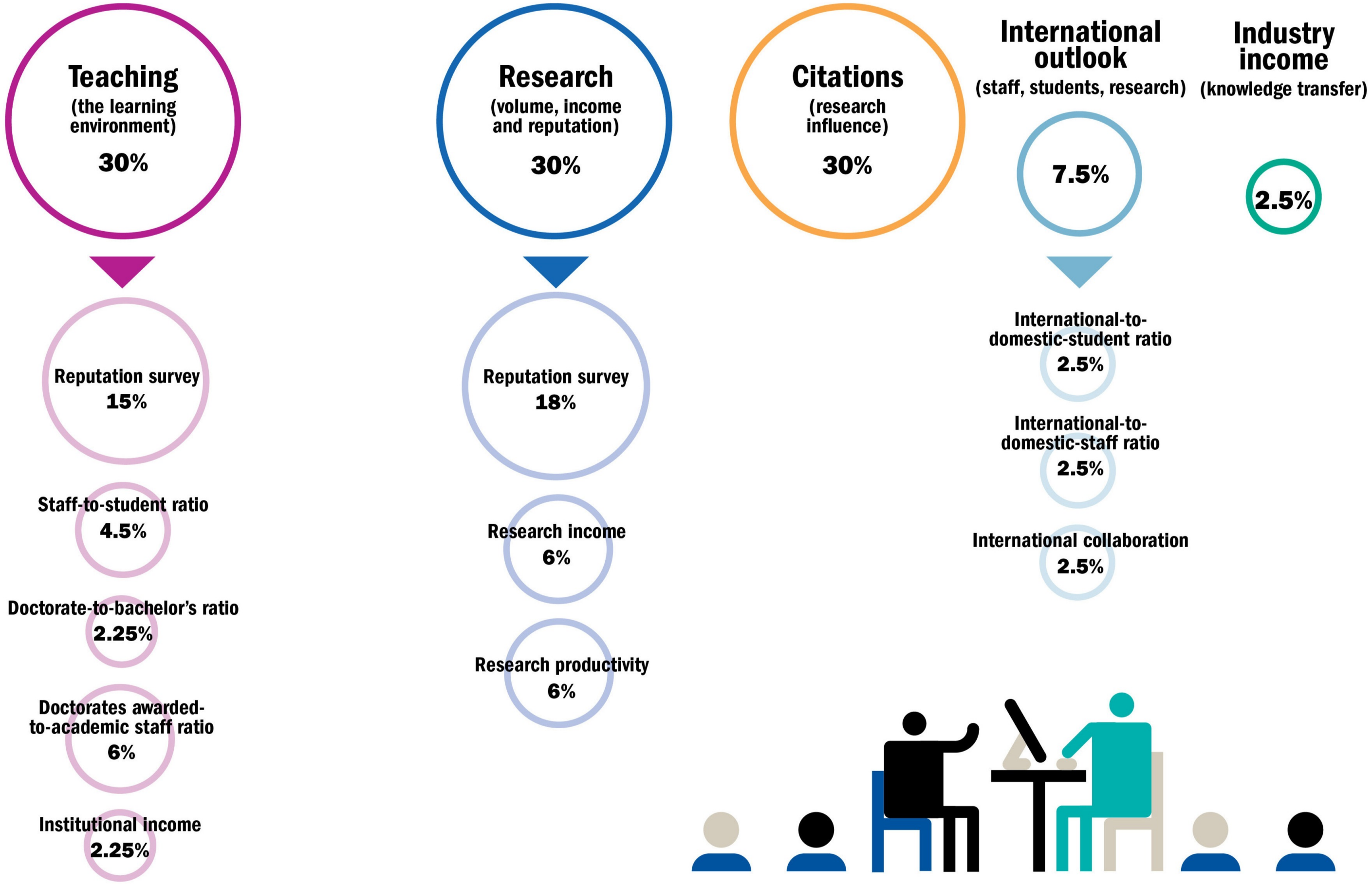


Goodhart’s law:
“If a measure becomes the public’s goal,
it is no longer a good measure.”

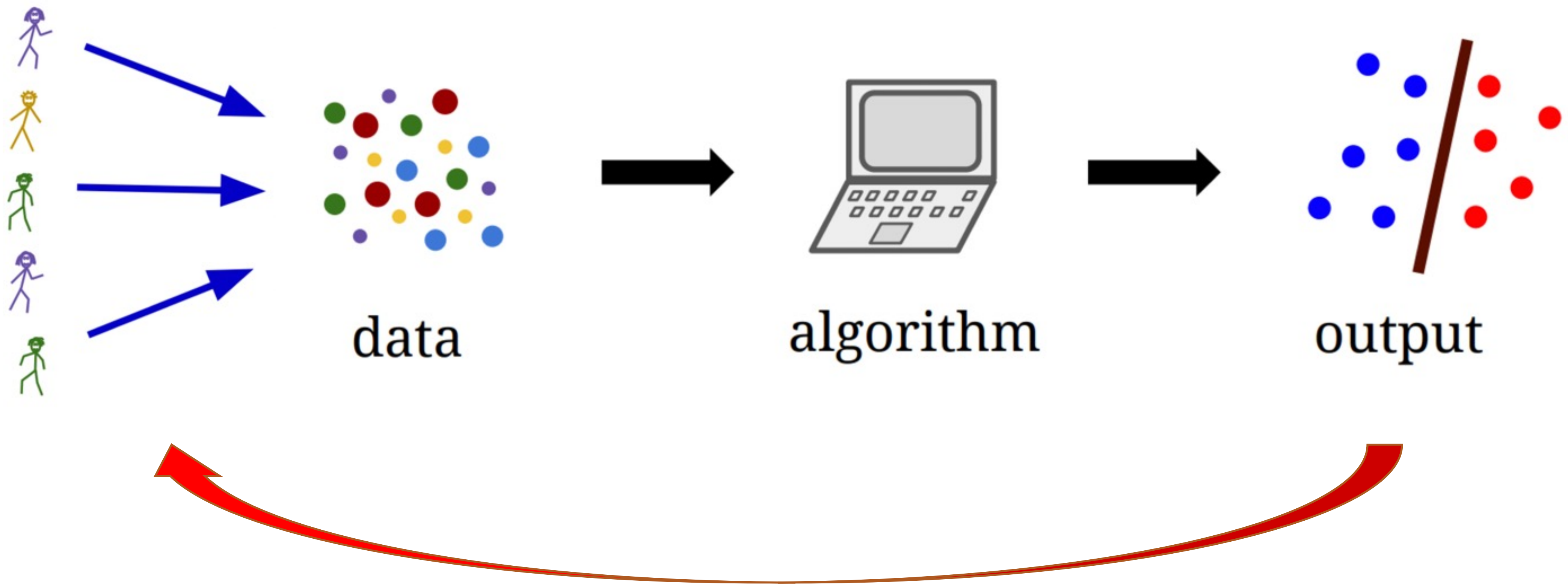




Methodology



Strategic Classification



Machine Learning Lifecycle