CS826 Deep Learning Theory and Practice

University of Strathclyde Glasgow

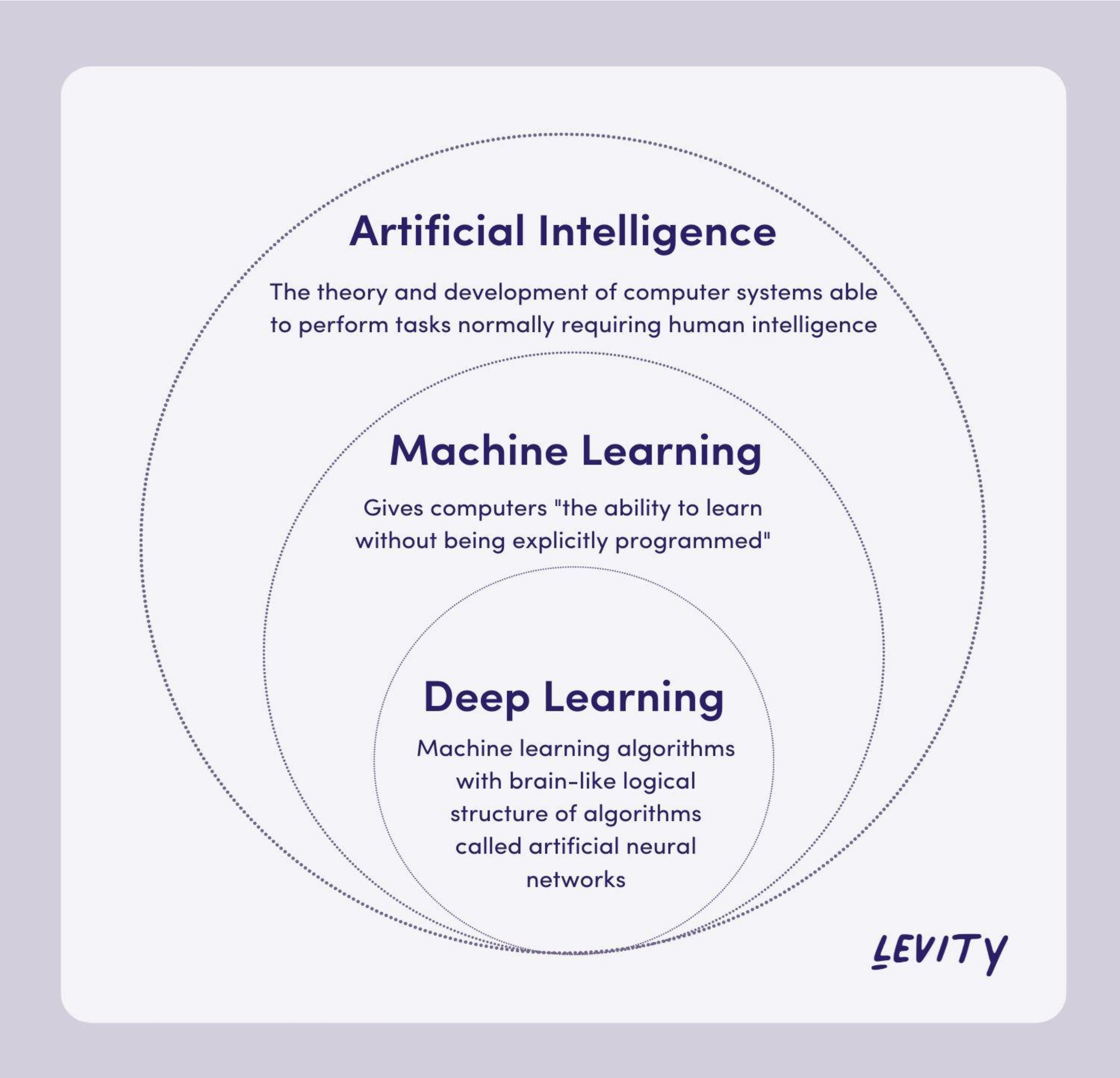
Getting Started with Neural Networks

Edited by Nur Naim, 20/9/22

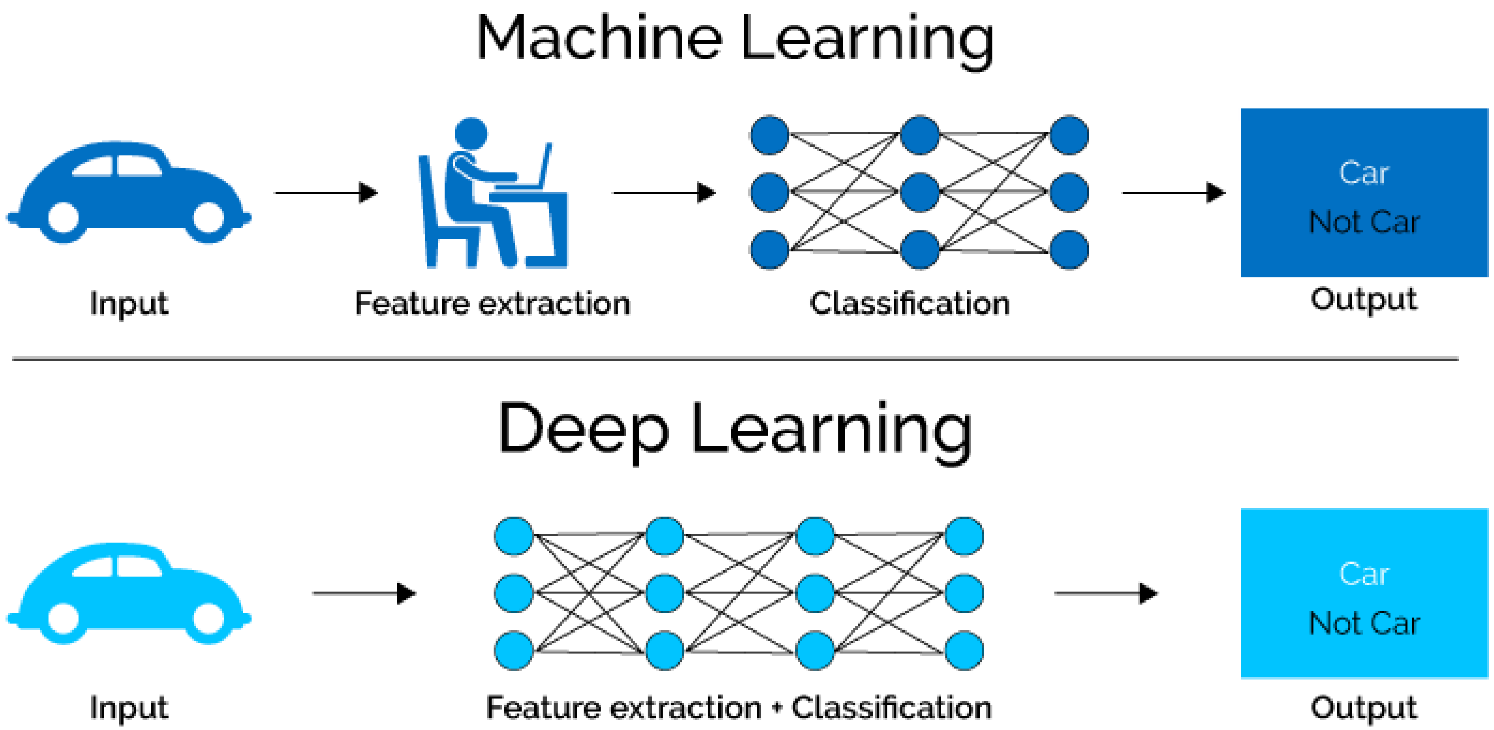
LEARNING OBJECTIVES

- Understand the fundamental of deep learning, the general concept
- Able to discuss the current trending about deep learning and provide solutions and opinions regarding deep learning
- Understand the mathematical steps
- Understand the overall architecture
- Use Google Colab or other python IDEs or code editors.

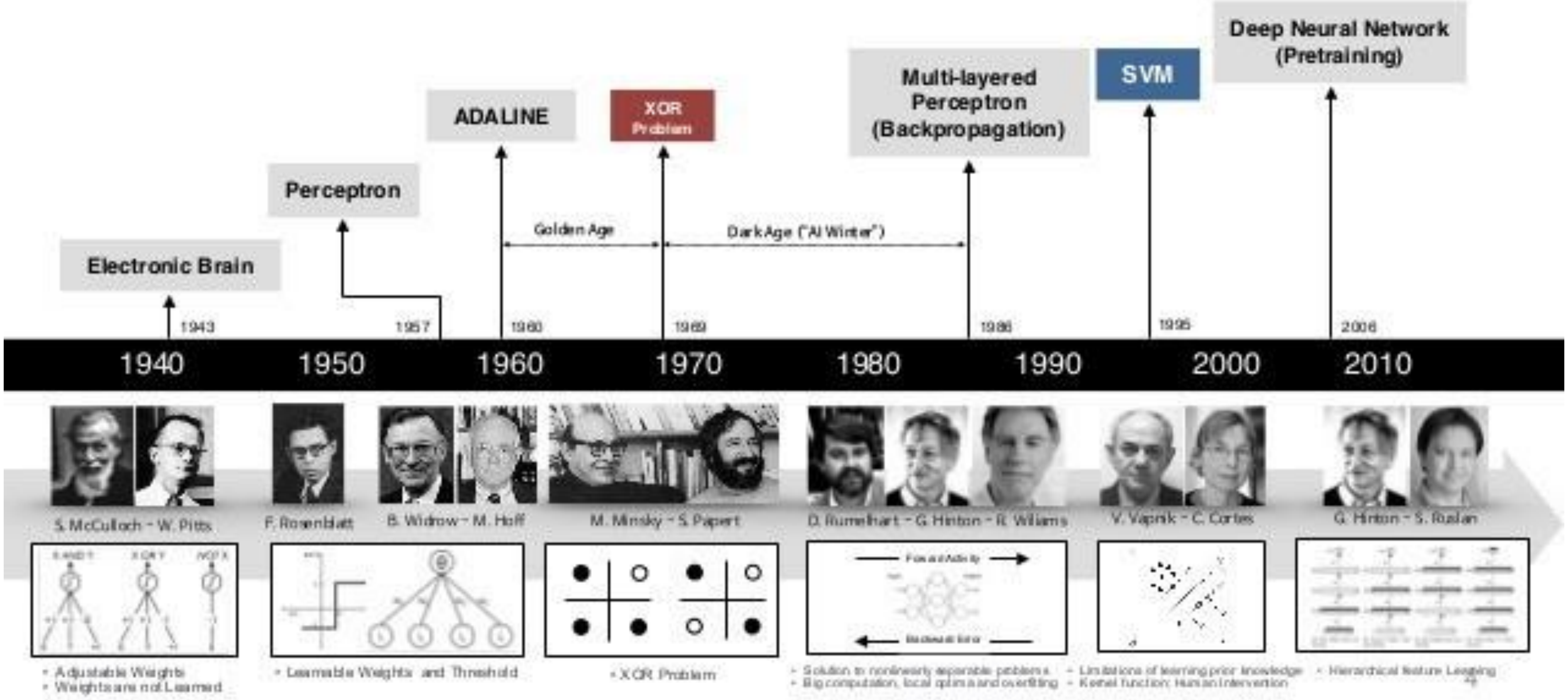
INTRODUCTION



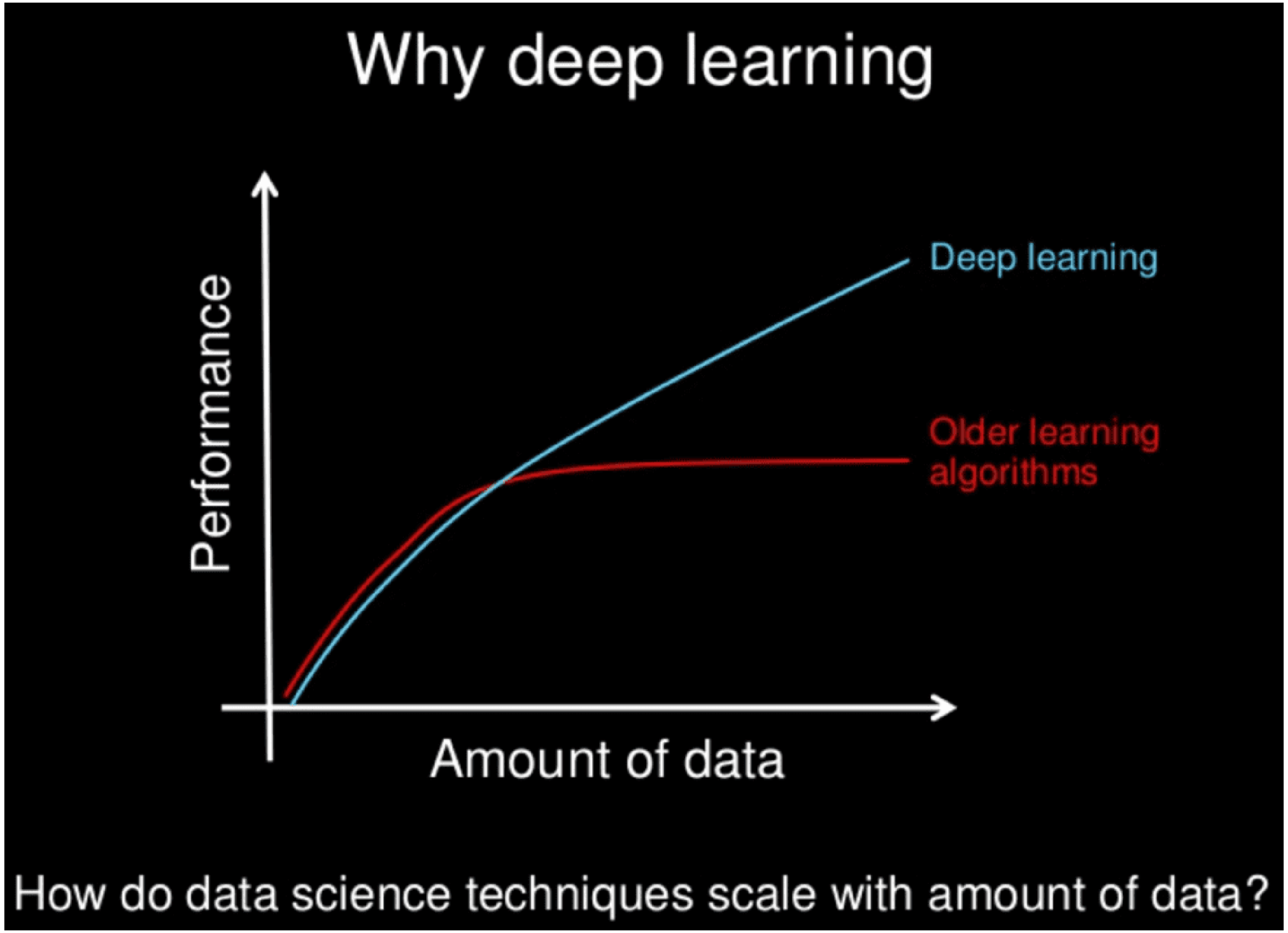
WHAT IS DEEP LEARNING?



Brief History of Neural Network



WHY DEEP LEARNING?



WHY NOW?

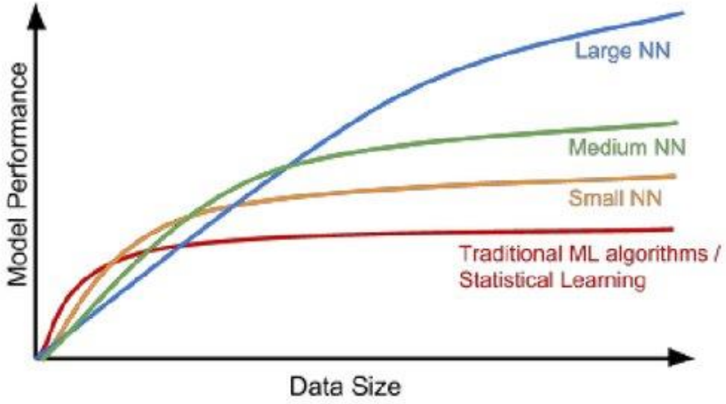
Why is Deep Learning Important now?

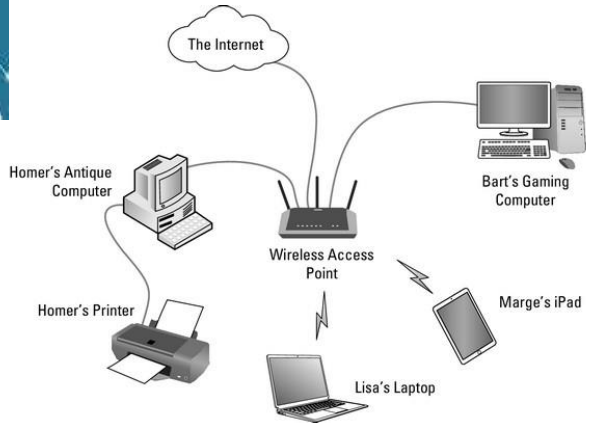
Deep learning requires large amounts of data

Deep learning requires substantial computing power

- High-performance GPUs have a parallel architecture that is efficient for deep learning

Well-trained Deep Neural Network can handle tasks that were previously considered impossible





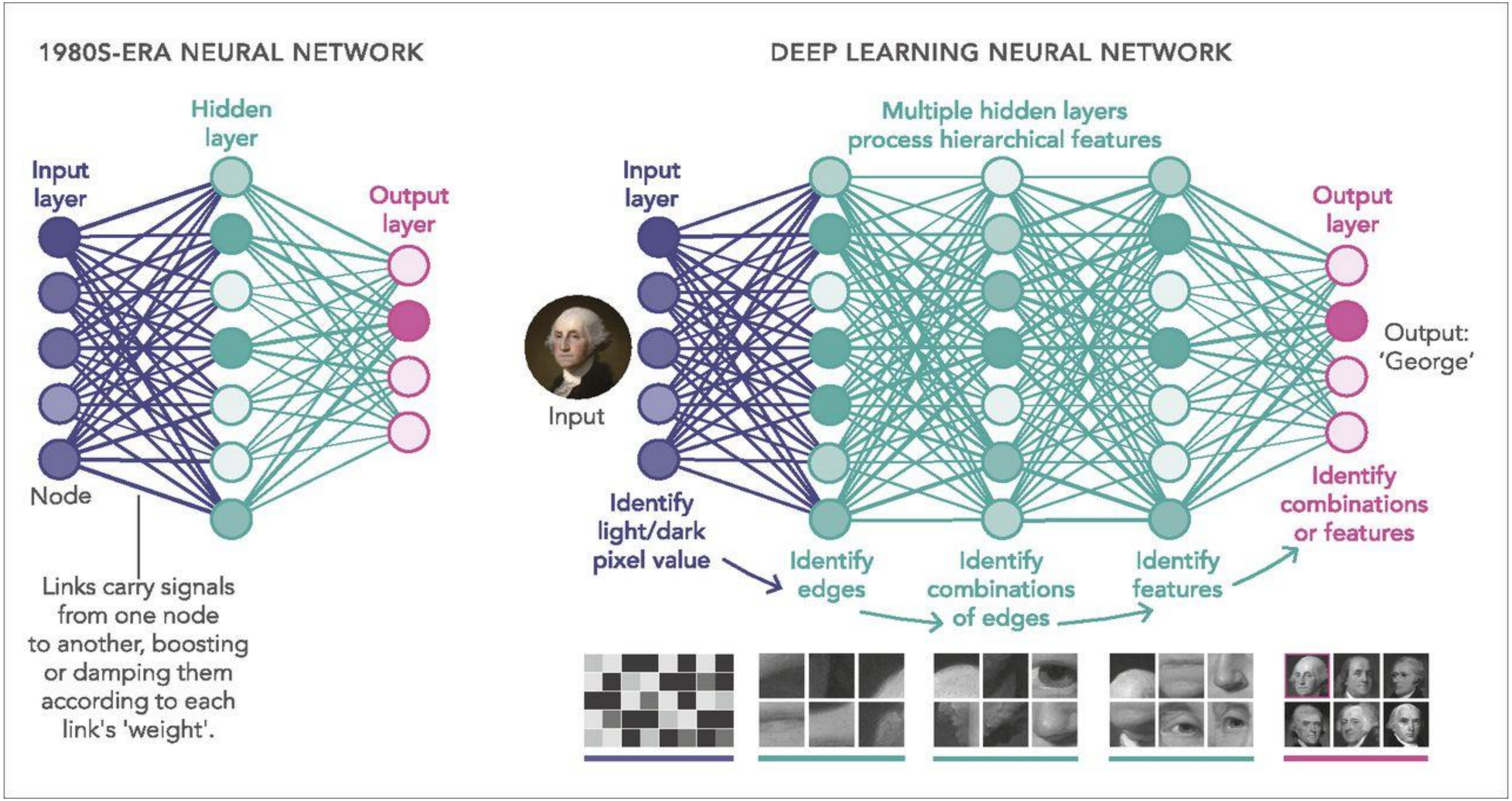
Since 1980s: Form of models hasn't changed much,
but lots of new tricks…

- More hidden units
- Better (online) optimization
- New nonlinear functions (ReLUs)
- Faster computers (CPUs and GPUs)

WHAT ARE THE CHALLENGES IN DEEP LEARNING?

- Not enough training data
- Poor Quality of data
- Irrelevant Features
- Nonrepresentative training data
- Overfitting and Underfitting

ARCHITECTURES

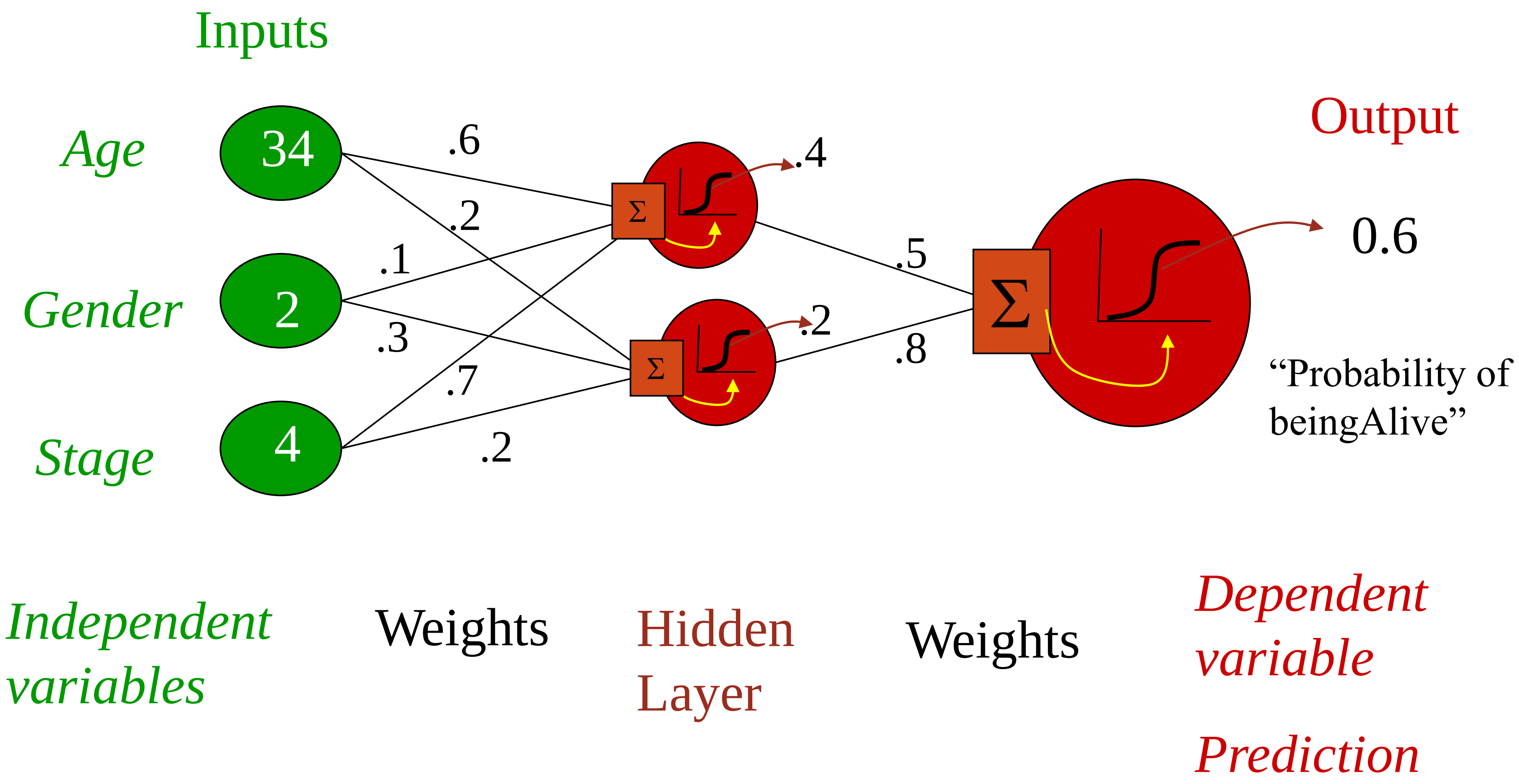

1980S-ERA NEURAL NETWORK
Hidden layer
Input layer
Output layer
Node
Links carry signals from one node to another, boosting or damping them according to each link's 'weight'.
DEEP LEARNING NEURAL NETWORK
Multiple hidden layers process hierarchical features
Input layer
Output layer
Input
Output: 'George'
Identify light/dark pixel value
Identify edges
Identify combinations of edges
Identify features
Identify combinations or features

DESIGN YOUR NN

Even for a basic Neural Network, there are many design decisions to make:

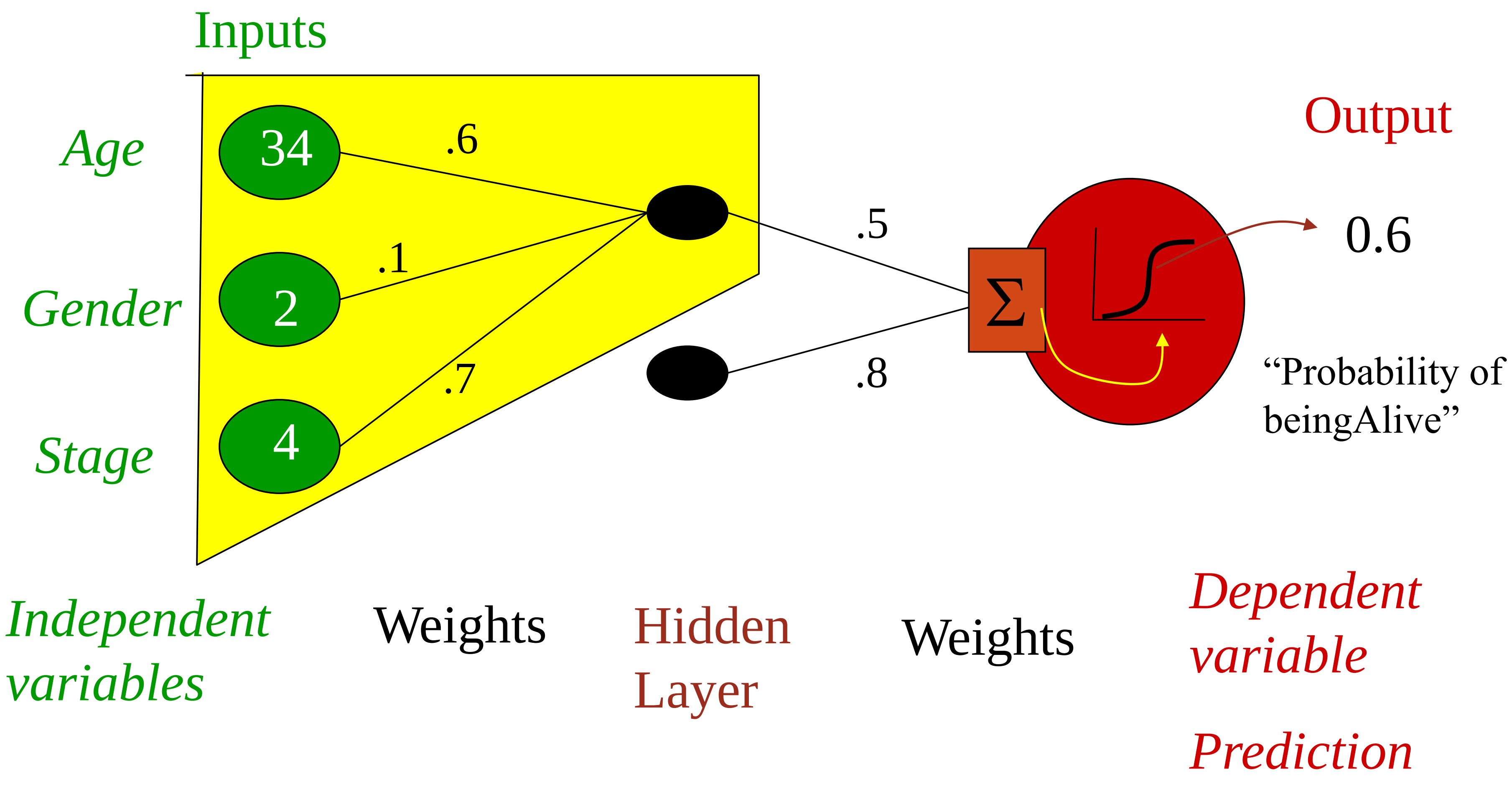
1. # of hidden layers (depth)
2. # of units per hidden layer (width)
3. Type of activation function (nonlinearity)
4. Form of objective function

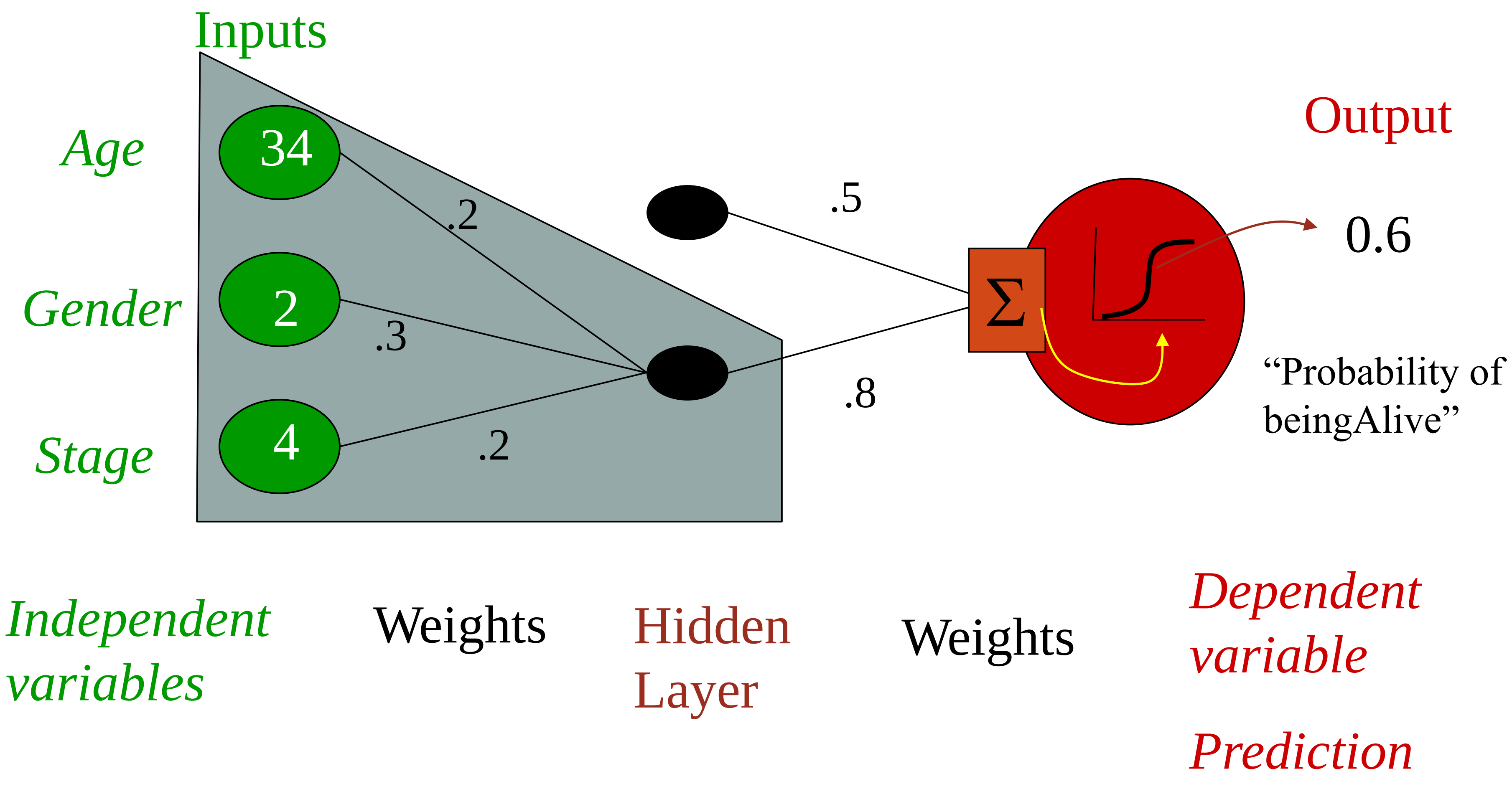
NEURAL NETWORK MODEL

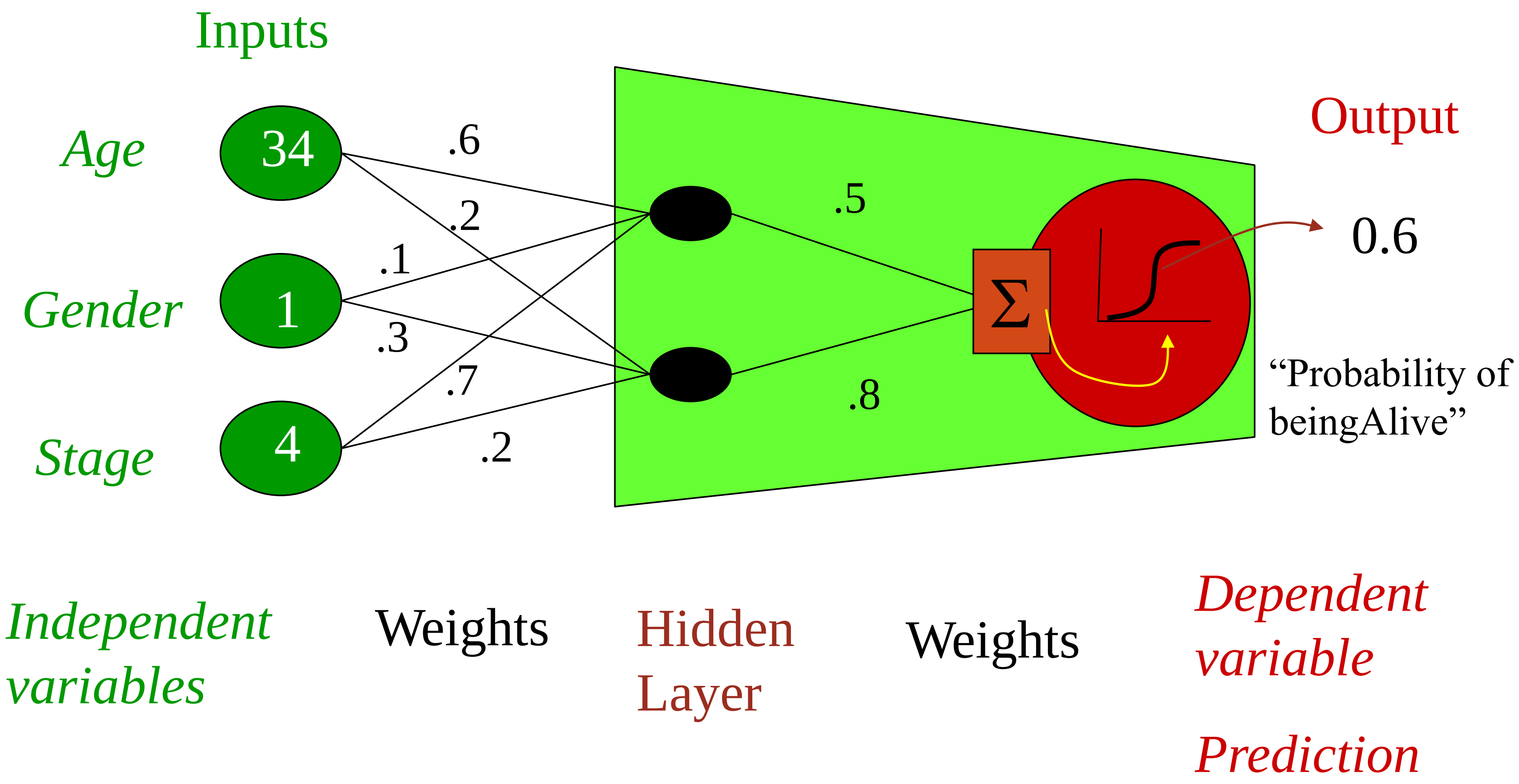


Source: © Eric Xing @ CMU, 2006-2011

“COMBINED LOGISTIC MODELS”

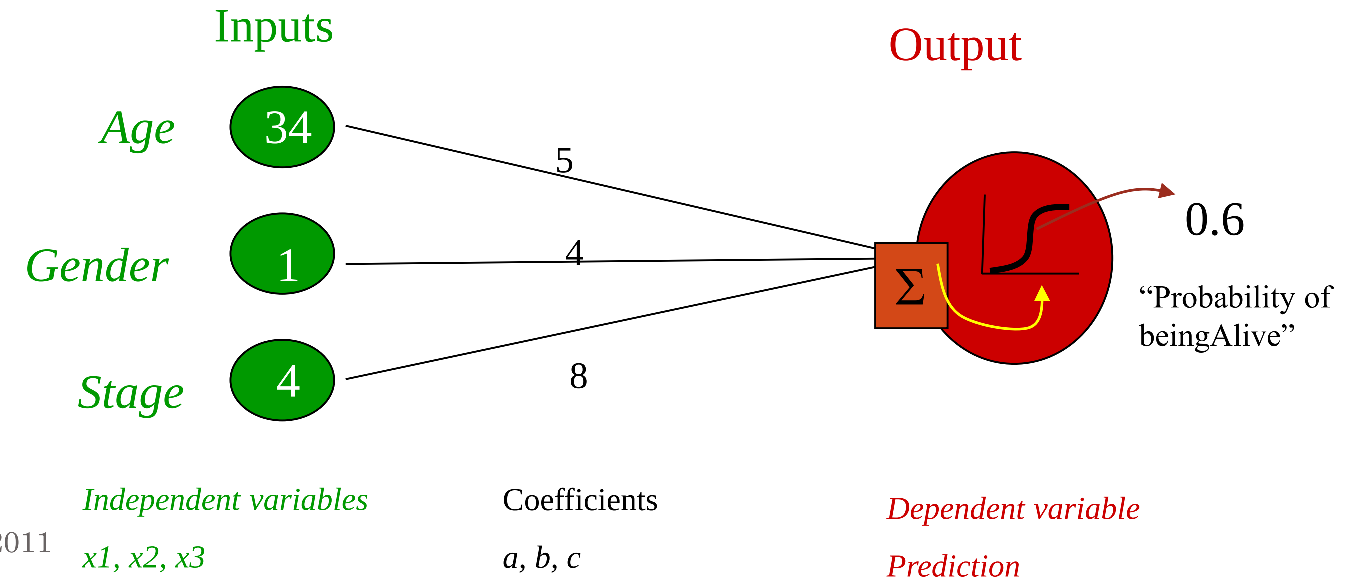



Inputs
Age
Gender
Stage
34
2
4
.2
.3
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction


Inputs
Age
34
Gender
1
Stage
4
.6
.2
.1
.3
.7
.2
.5
.8
Σ
Output
0.6
"Probability of beingAlive"
Independent variables
Weights
Hidden Layer
Weights
Dependent variable
Prediction

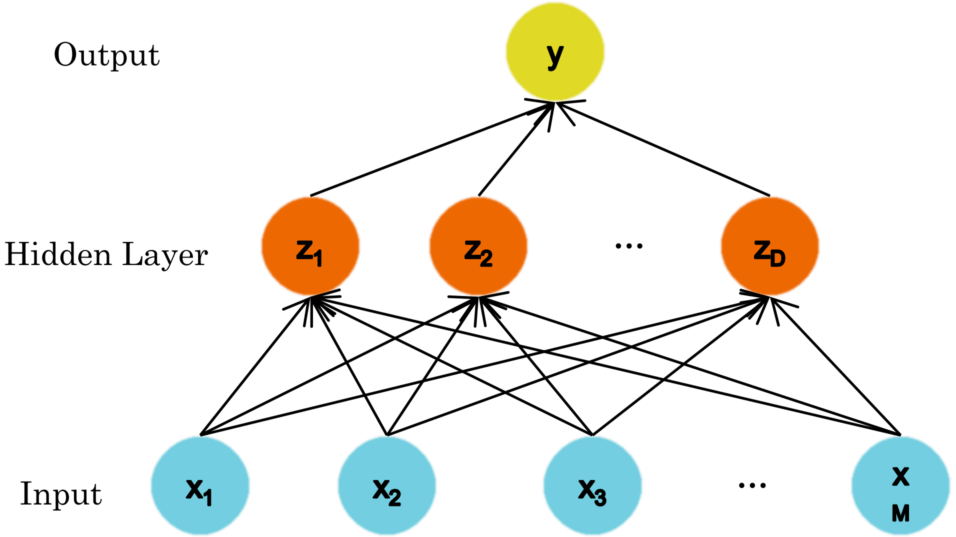
JARGON PSEUDO-CORRESPONDENCE

- Independent variable = input variable
- Dependent variable = output variable
- Coefficients = "weights"
- Estimates = "targets"
- Logistic Regression Model (the sigmoid unit)



Decision Functions

NEURAL NETWORK



(E) Output (sigmoid)
$y = \frac{1}{1+\exp(-b)}$

(D) Output (linear)
$b = \sum_{j=0}^{D} \beta_j z_j$

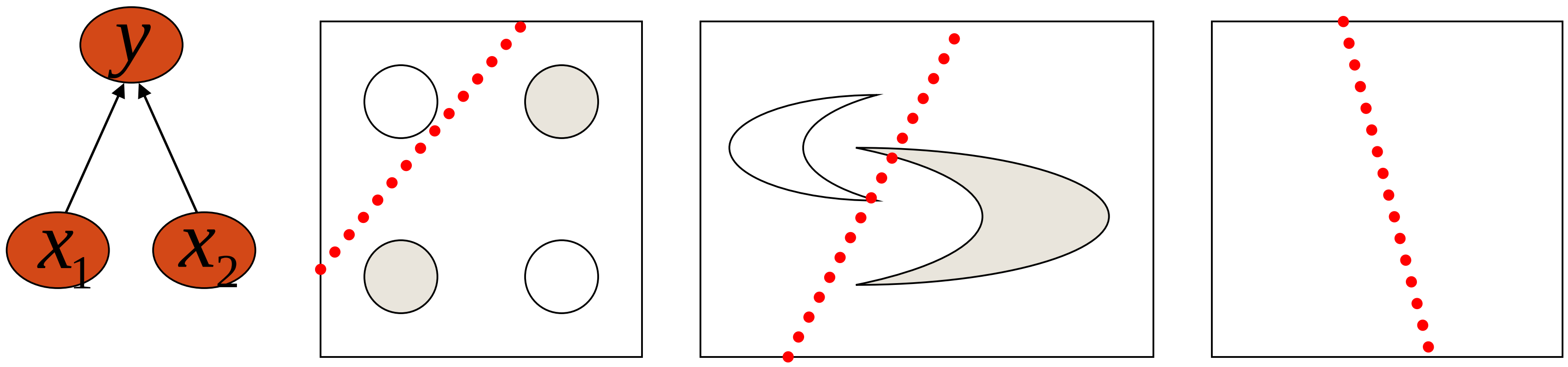
(C) Hidden (sigmoid)
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

(A) Input
Given $x_i, \ \forall i$

DECISION BOUNDARY

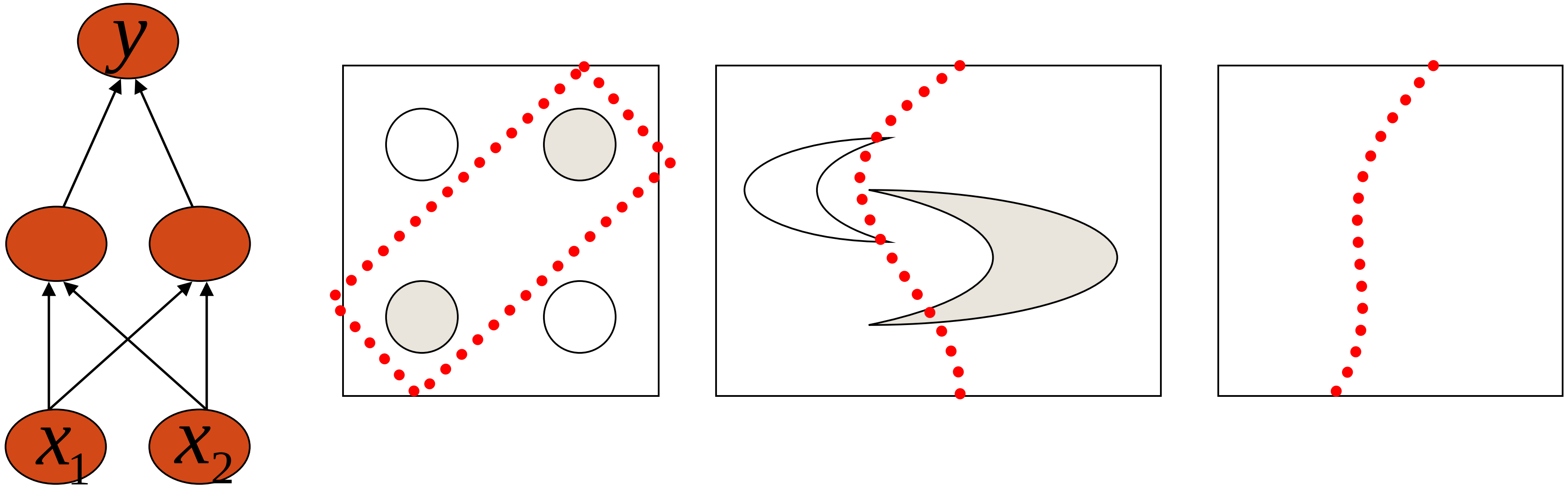
- 0 hidden layers: linear classifier
  - Hyperplanes



Example from to Eric Postma via Jason Eisner

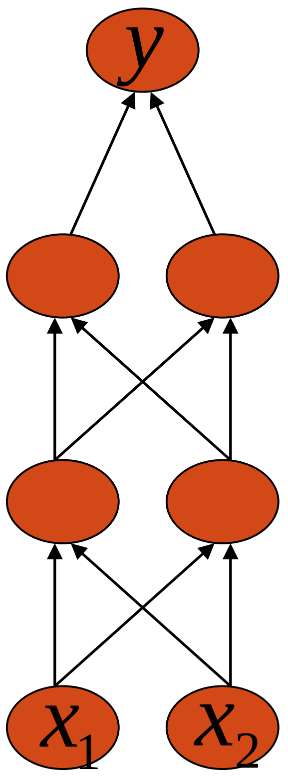
DECISION BOUNDARY

- 1 hidden layer
  - Boundary of convex region (open or closed)

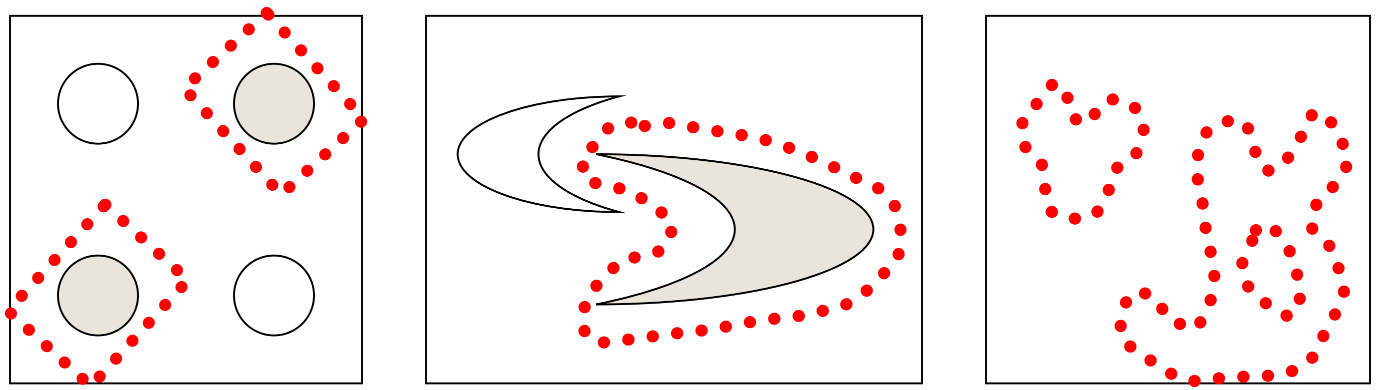


Example from to Eric Postma via Jason Eisner

DECISION BOUNDARY



- 2 hidden layers
  - Combinations of convex regions



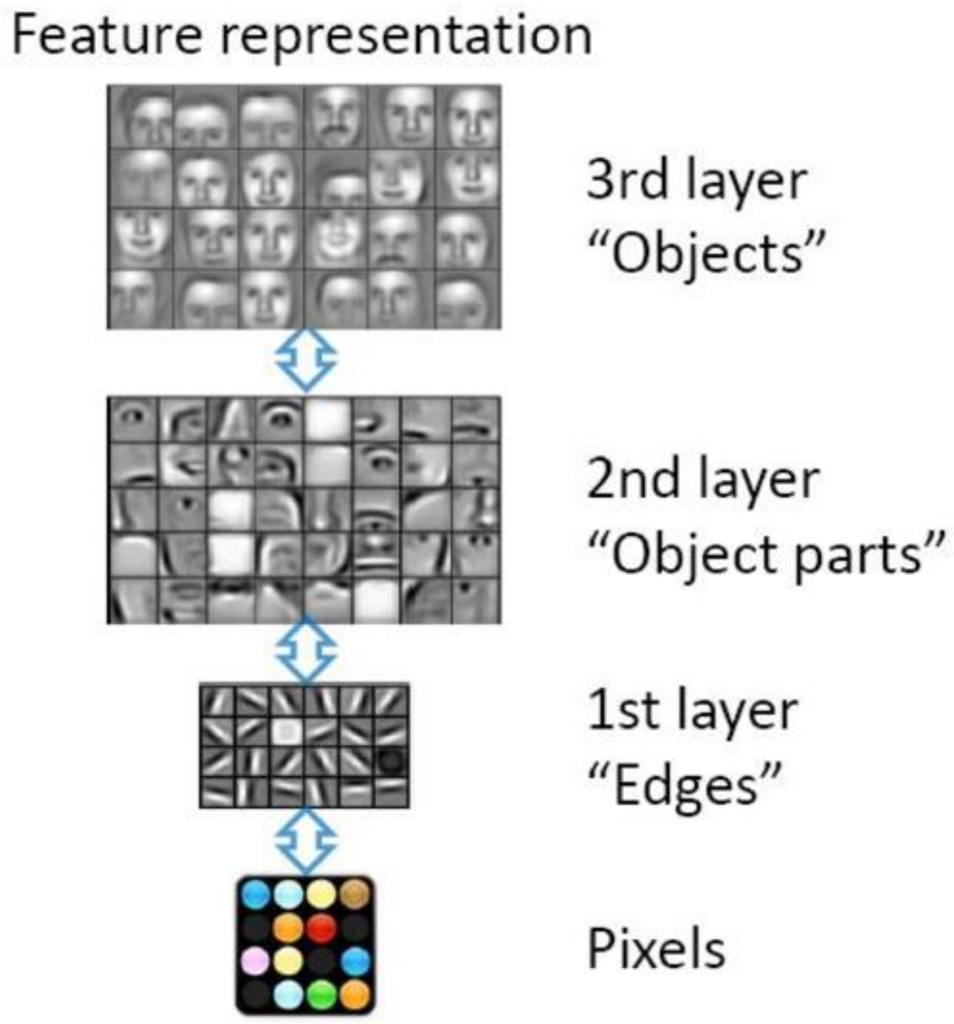
Example from to Eric Postma via Jason Eisner

DIFFERENT LEVELS OF ABSTRACTION

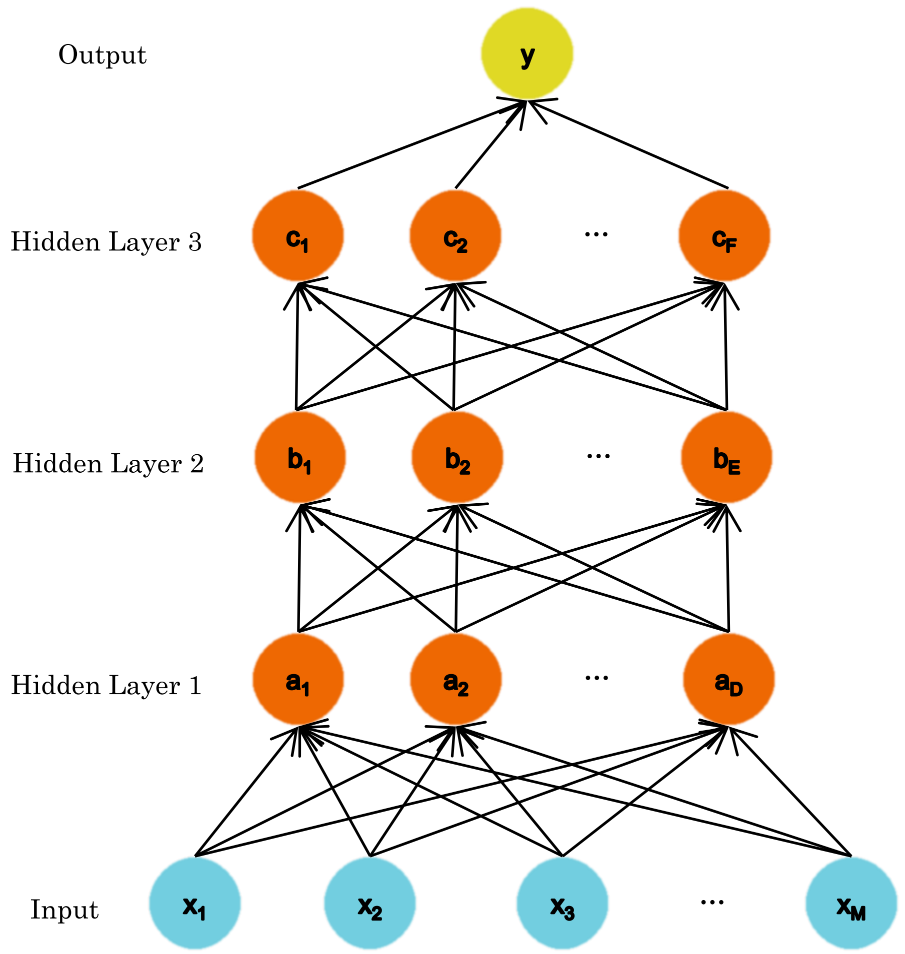
- We don't know the "right" levels of abstraction
- So let the model figure it out!

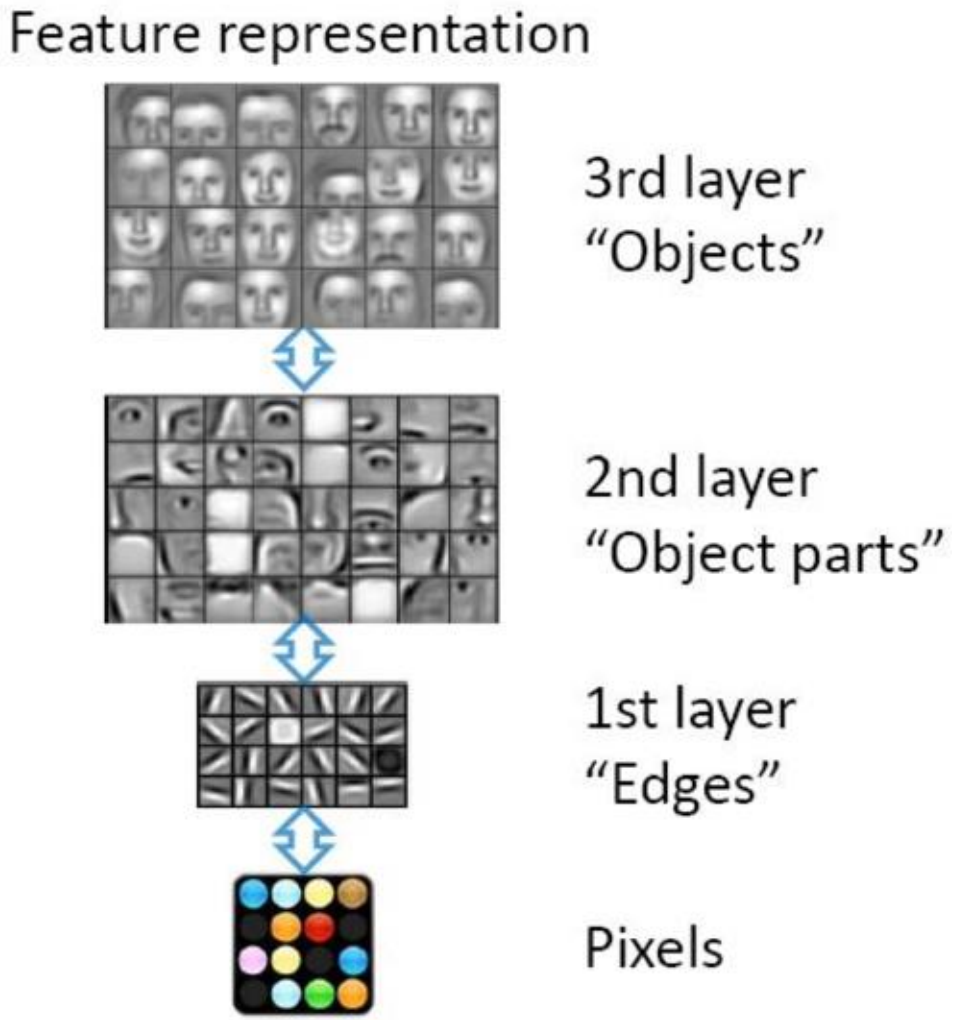
Face Recognition:

- Deep Network can build up increasingly higher levels of abstraction
- Lines, parts, regions



Example from Honglak Lee (NIPS 2010)

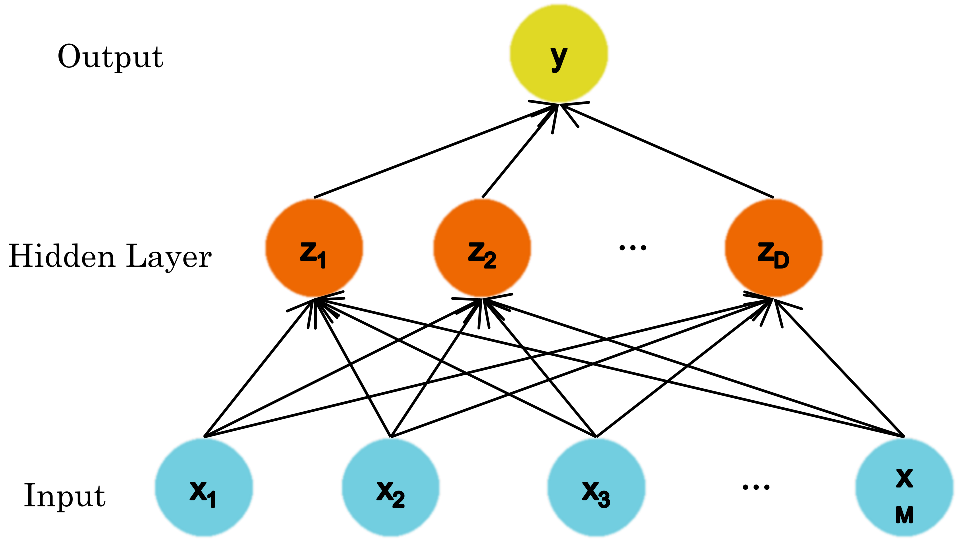




Example from Honglak Lee (NIPS 2010)

ACTIVATION FUNCTIONS

Neural Network with
sigmoid activation functions



(F) **Loss**
$J = \frac{1}{2}(y - y^*)^2$

(E) **Output (sigmoid)**
$y = \frac{1}{1+\exp(-b)}$

(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

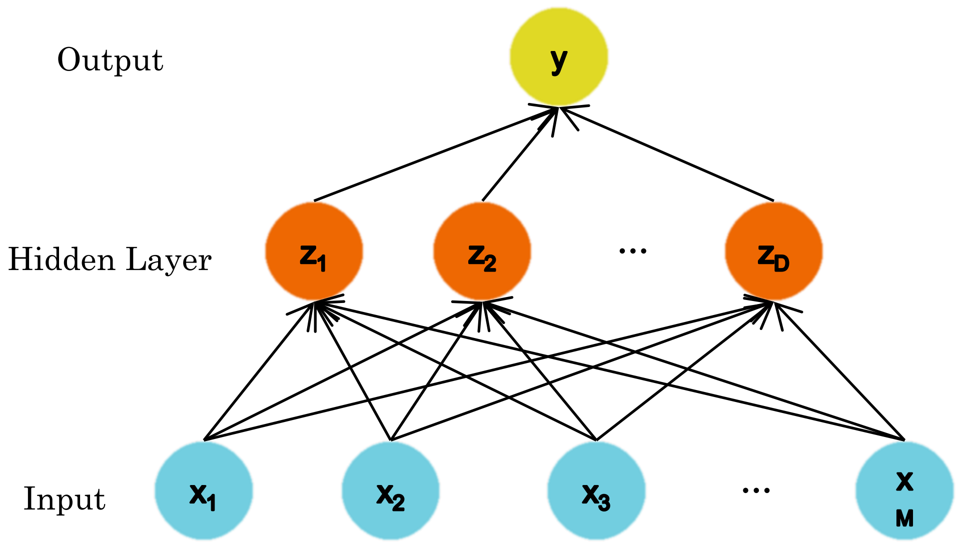
(C) **Hidden (sigmoid)**
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

(A) **Input**
Given $x_i, \ \forall i$

ACTIVATION FUNCTIONS

Neural Network with arbitrary nonlinear activation functions



(F) **Loss**
$J = \frac{1}{2}(y - y^*)^2$

(E) **Output (nonlinear)**
$y = \sigma(b)$

(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) **Hidden (nonlinear)**
$z_j = \sigma(a_j), \;\; \forall j$

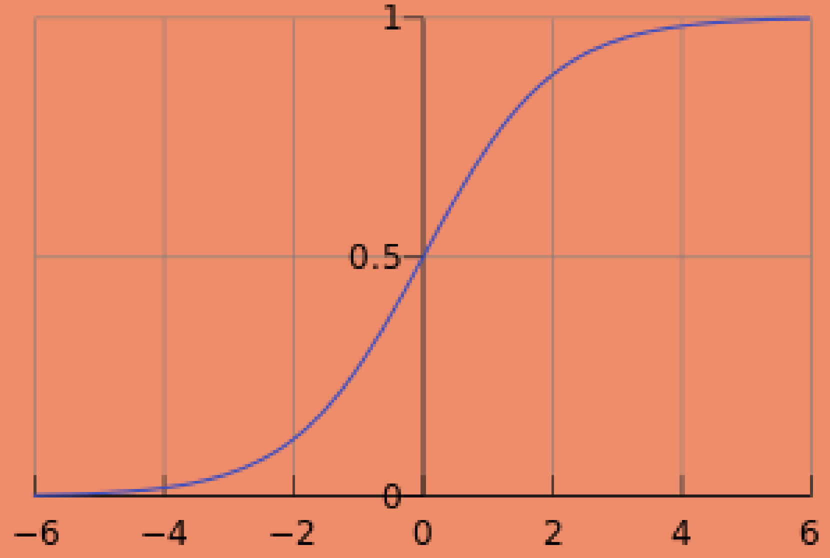
(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \;\; \forall j$

(A) **Input**
Given $x_i, \;\; \forall i$

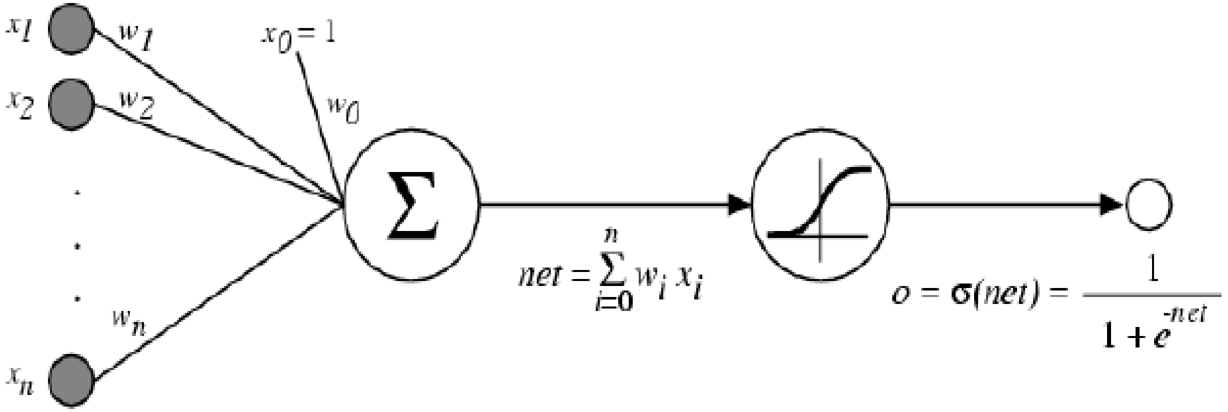
ACTIVATION FUNCTIONS

Sigmoid / Logistic Function

$$\text{logistic}(u) \circ \frac{1}{1+e^{-u}}$$

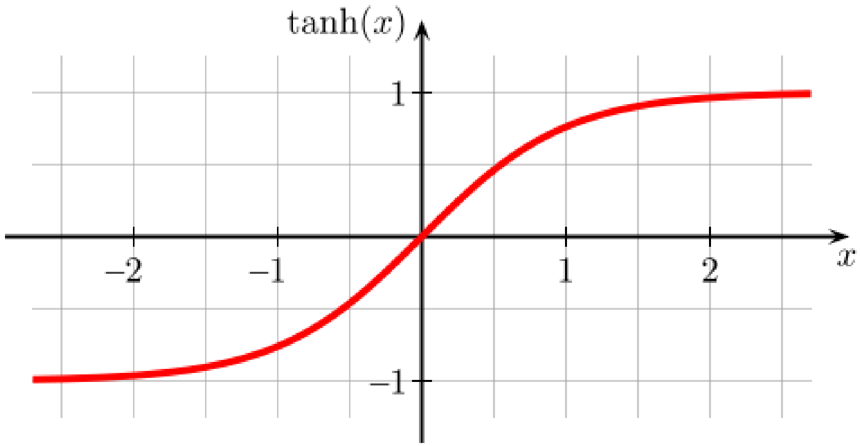


So far, we've assumed that the activation function (nonlinearity) is always the sigmoid function…



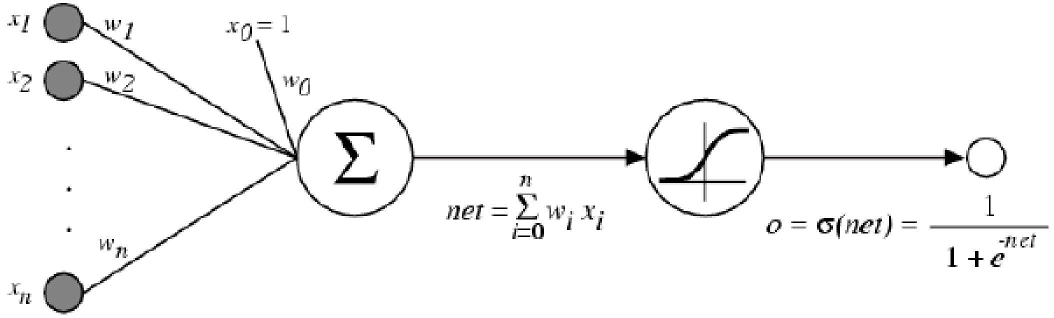
ACTIVATION FUNCTIONS

- A new change: modifying the nonlinearity
  - The logistic is not widely used in modern ANNs



Alternate 1: tanh

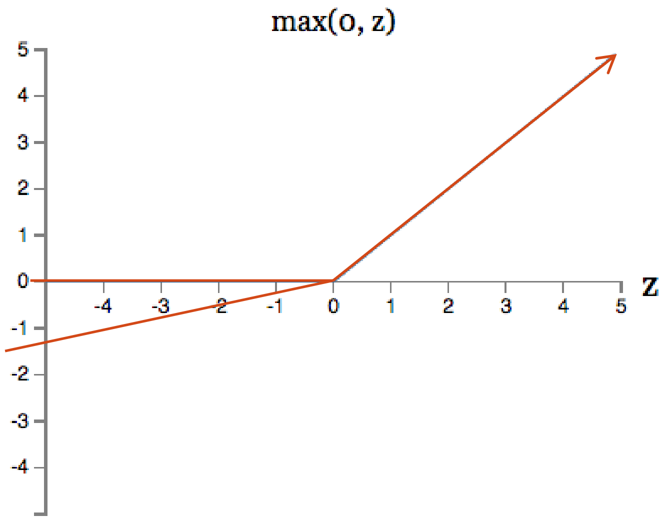
Like logistic function but shifted to range [-1, +1]



Slide from William Cohen

ACTIVATION FUNCTIONS

- A new change: modifying the nonlinearity
  - ReLU often used in vision tasks

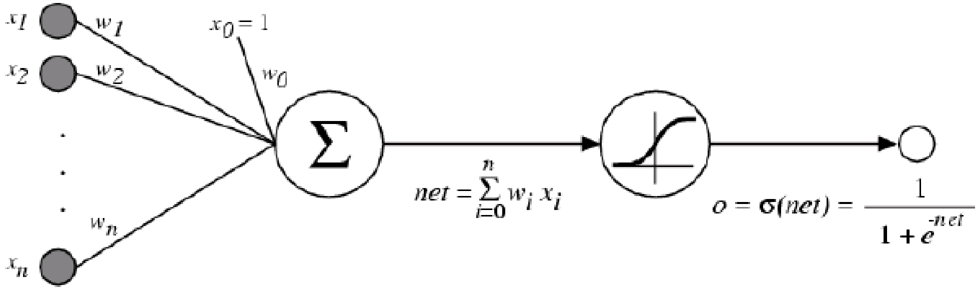


$$\max(0, w \cdot x + b).$$

Alternate 2: rectified linear unit

Linear with a cutoff at zero

(Implementation: clip the gradient when you pass zero)



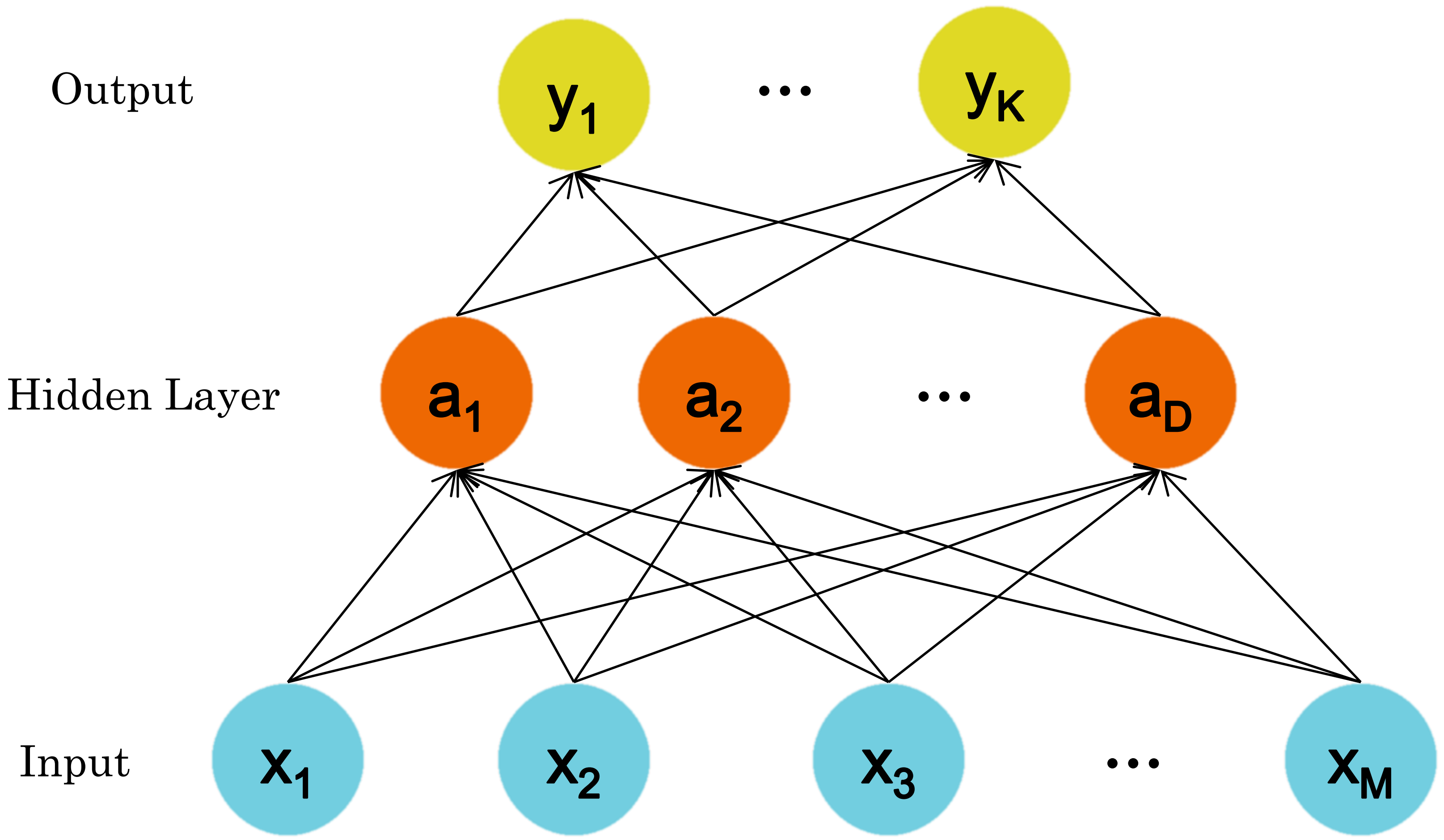
Slide from William Cohen

OBJECTIVE FUNCTIONS FOR NNS

- Regression:
  - Use the same objective as linear regression
  - Quadratic loss (i.e. mean squared)
- Classification:
  - Use the error same objective as logistic regression
  - Cross-entropy (i.e. negative log likelihood)
  - This requires probabilities, so we add an additional "softmax" layer at the end of our network

	Forward	Backward
Quadratic	$J = \frac{1}{2}(y - y^*)^2$	$\frac{dJ}{dy} = y - y^*$
Cross Entropy	$J = y^* \log(y) + (1 - y^*) \log(1 - y)$	$\frac{dJ}{dy} = y^* \frac{1}{y} + (1 - y^*) \frac{1}{y - 1}$

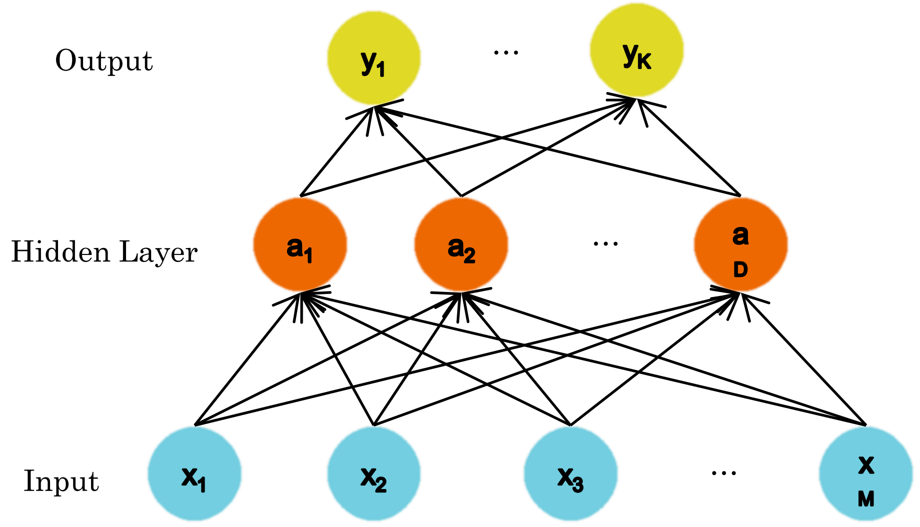
MULTI-CLASS OUTPUT



MULTI-CLASS OUTPUT

Softmax:

$$y_k = \frac{\exp(b_k)}{\sum_{l=1}^{K} \exp(b_l)}$$



(F) Loss
$J = \sum_{k=1}^{K} y_k^* \log(y_k)$

(E) Output (softmax)
$y_k = \frac{\exp(b_k)}{\sum_{l=1}^{K} \exp(b_l)}$

(D) Output (linear)
$b_k = \sum_{j=0}^{D} \beta_{kj} z_j \ \forall k$

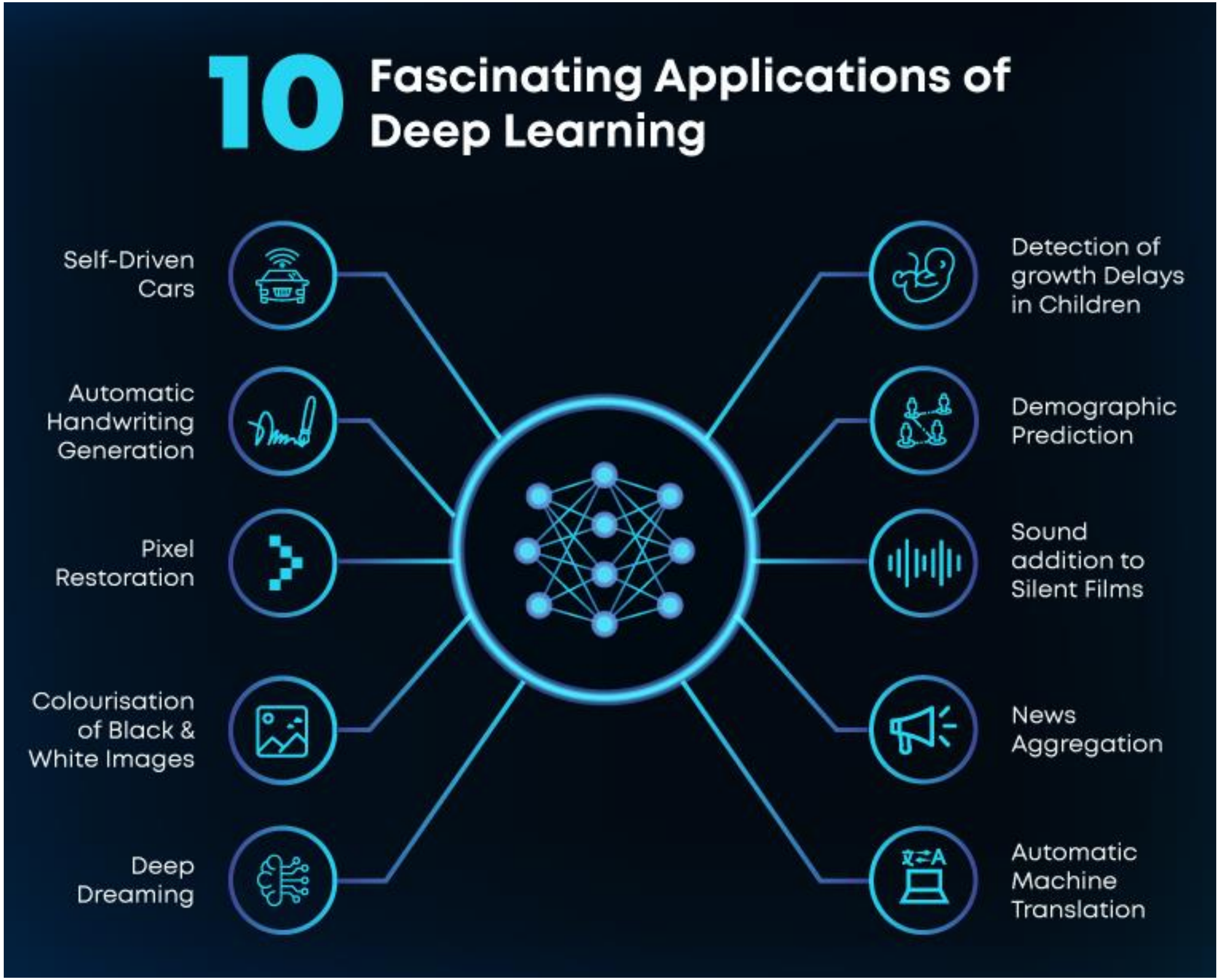
(C) Hidden (nonlinear)
$z_j = \sigma(a_j), \ \forall j$

(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

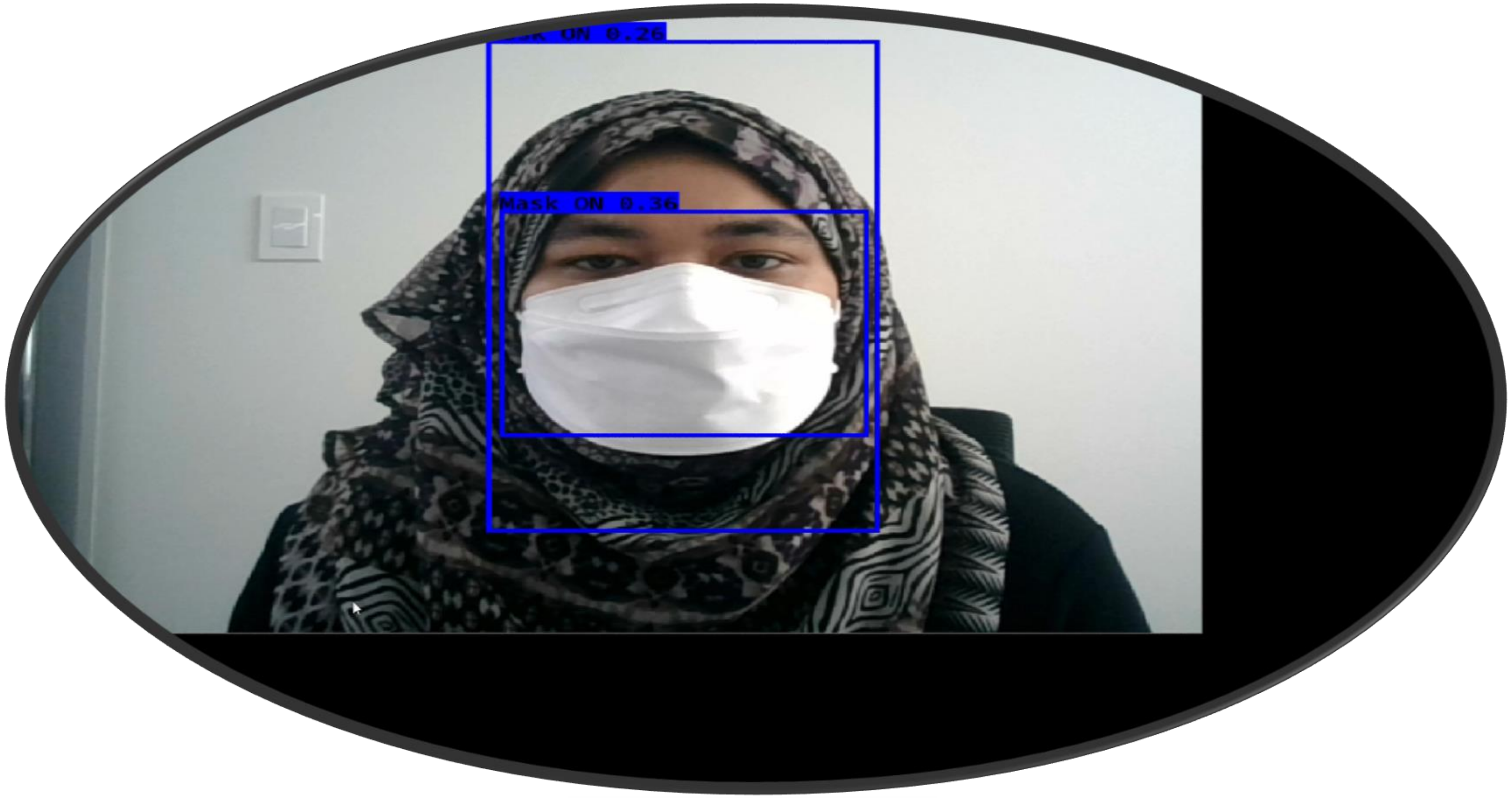
(A) Input
Given $x_i, \ \forall i$

APPLICATIONS

Examples of Deep Learning Applications


10 Fascinating Applications of Deep Learning
Self-Driven Cars
Automatic Handwriting Generation
Pixel Restoration
Colourisation of Black & White Images
Deep Dreaming
Detection of growth Delays in Children
Demographic Prediction
Sound addition to Silent Films
News Aggregation
Automatic Machine Translation

APPLICATIONS




FPS: 1

REFERENCES

- https://www.slideshare.net/databricks/introduction-to-neural-networks-122033415
- https://www.cs.wmich.edu/~elise/courses/cs6800/Neural-Networks.ppt
- https://www.cs.cmu.edu/~mgormley/courses/10601b-f16/lectureSlides/lecture15-neural-nets.pptx
- Google (images) – deep learning, why deep learning now, applications.