Lecture Notes on
Pattern Recognition

Module 09(A): Deep Learning Overview

Gang Li
School of Information Technology
Deakin University, VIC 3125, Australia

0

Road map

- Deep Learning Overview
  - Deep Learning
    - Why
    - End-to-End Learning
  - FFN
    - Matrix Operation
    - Learning FFN
  - BP
    - Forward Pass
    - Backward Pass

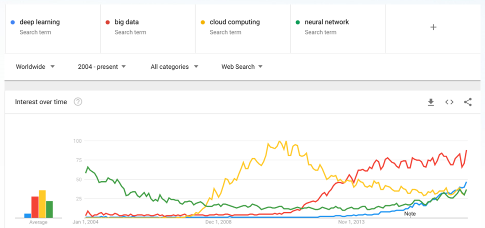
1

Deep Learning

- Deep Learning
- Why Deep Learning?
  - Modularization
  - End-to-End Learning

2

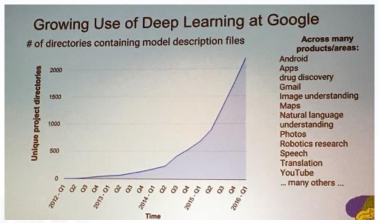
Google Trends



3

Deep learning trends at Google

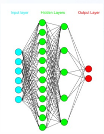
- SIGMOD 2016, Jeff Dean



4

What is Deep Learning?

- Fast answer:
  - Fast answer: multilayer perceptron (aka deep neural networks) of the 1980s rebranded in 2006.
  - But early nets go stuck at 1-2 hidden layers.

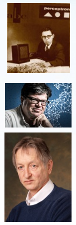


- Slow answer:
  - distributed representation, multiple steps of computation, modelling the compositionality of the world, a better prior, advances in compute, data & optimization, neural architectures, etc.

5

From ANN to Deep Learning

- 1958: Perceptron (linear model)
- 1969: Perceptron has limitation
- 1980s: Multi-layer perceptron
  - Do not have significant difference from DNN today
- 1986: Backpropagation
  - Usually more than 3 hidden layers is not helpful
- 1989: 1 hidden layer is "good enough"
  - why deep?



TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 6

6

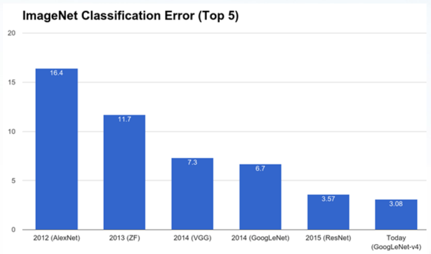
From ANN to Deep Learning

- 2006: RBM initialization
- 2009: GPU
- 2012: win ILSVRC image competition
- 2015.2:
  - Image recognition surpassing human-level performance
- 2016.3: Alpha GO beats Lee Sedol
- 2016.10:
  - Many system as good as humans

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 7

7

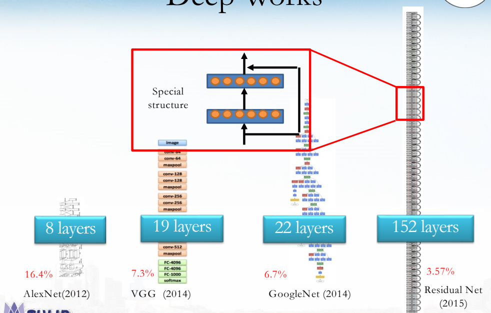
Deep Works



TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 8

8

Deep Works



TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 9

9

The Best of Machine Learning

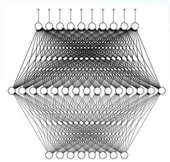
- Strong/flexible priors:
  - Good features: Feature engineering
  - Data structure: HMM, CRF, MRF, Bayesian nets
  - Model structure, VC-dimension, regularization, sparsity:
    - SVM, compressed sensing
  - Manifold assumption, class/region separation:
    - Metric + semi-supervised learning
  - Factors of variation: PCA, ICA, FA
- Uncertainty quantification:
  - Bayesian, ensemble: RF, GBM
- Sharing statistical strength:
  - model reuse: transfer learning, domain adaption, multitask learning, lifelong learning

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 10

10

Universality Theorem

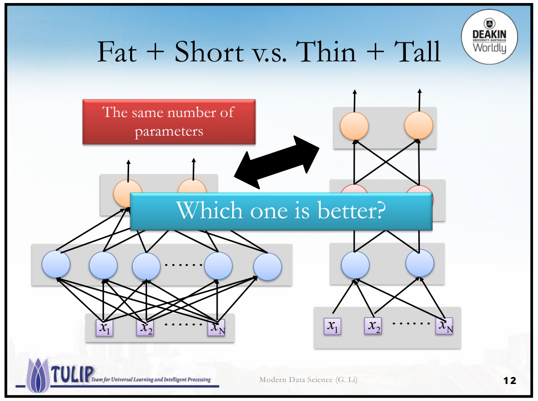
- Shallow network can represent any function.
- Given **enough** hidden neurons, Any continuous function $f: R^N \rightarrow R^M$ can be realized by a network with one hidden layer
- However, using deep structure is more effective.



Reference for the reason: http://neuralnetworksanddeeplearning.com/chap4.html

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 11

11



12

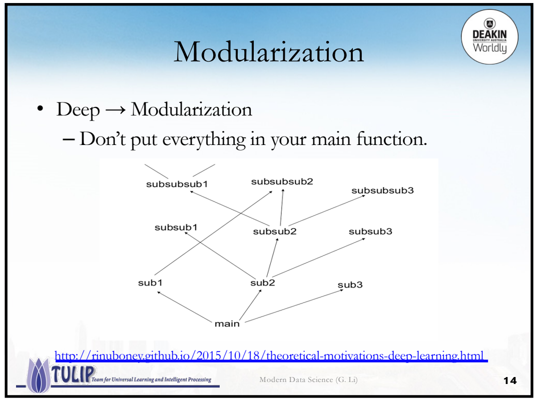
Fat + Short v.s. Thin + Tall

Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4	Why?	
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

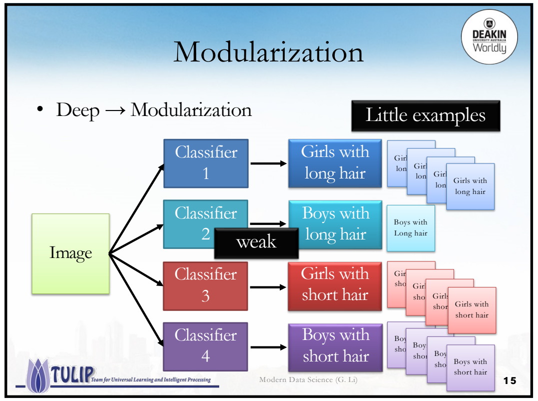
- Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 13

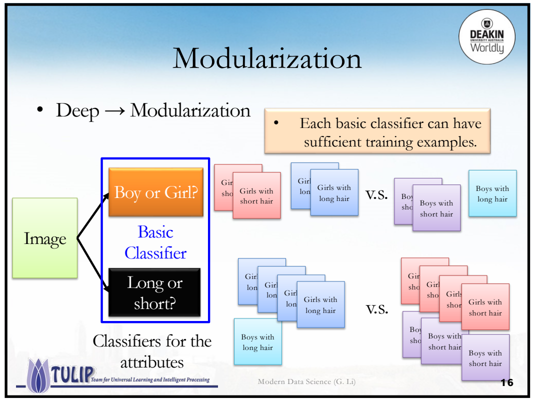
13



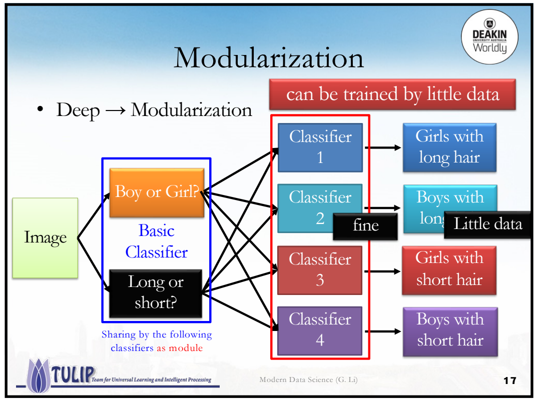
14



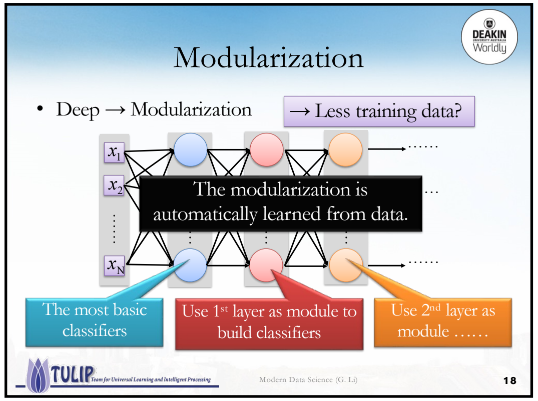
15



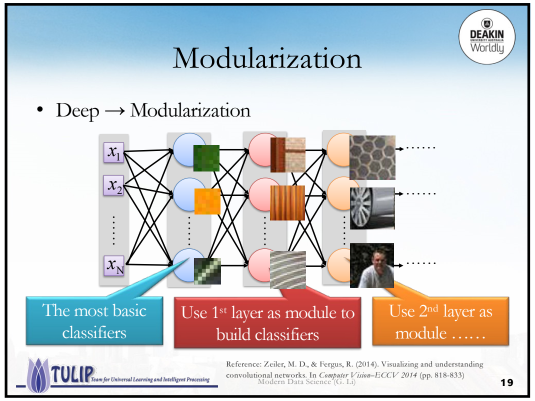
16



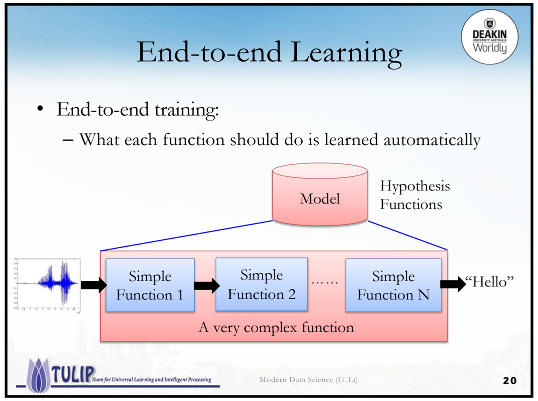
17



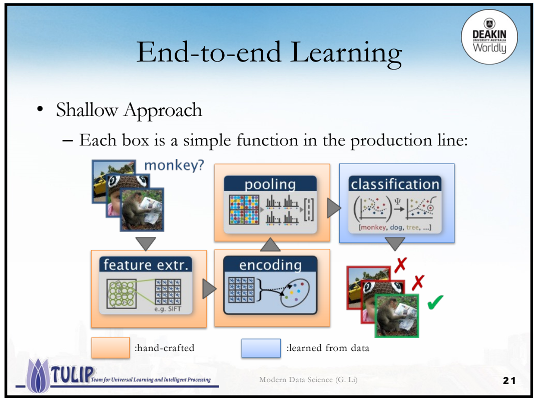
18



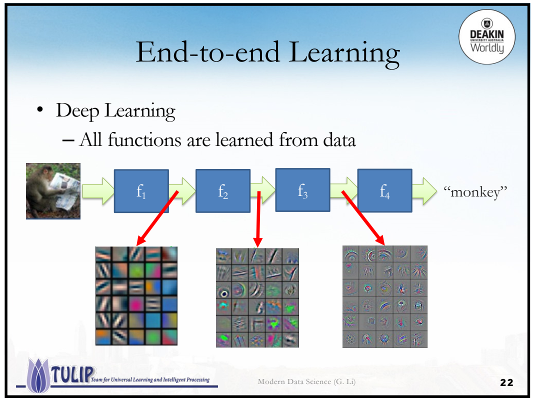
19



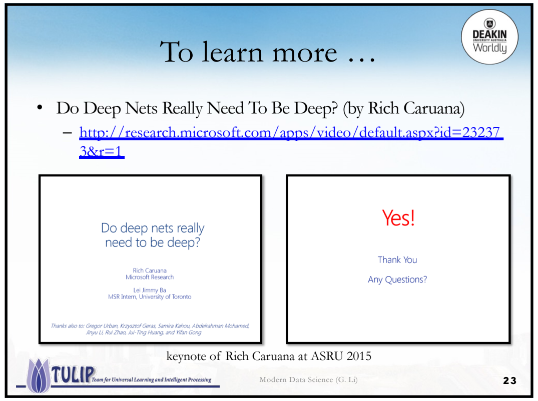
20



21



22



23

Tensorflow or MXNet?

Tensorflow vs. MXNET

	Languages	MultiGPU	Distributed	Mobile	Runtime Engine
Tensorflow	Python	Yes	No	Yes	?
MXNET	Python, R, Julia, Go	Yes	Yes	Yes	Yes
(Googlenet) E5-1650/980	Tensor flow	Torch7	Caffe	MXNET	
Time	940ms	172ms	170ms	180ms	
Memory	all (OOM after 24)	2.1GB	2.2GB	1.6GB	

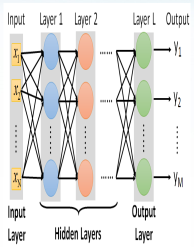
Marianas Labs Carnegie Mellon University

K + TensorFlow

K mxnet

24

FFN

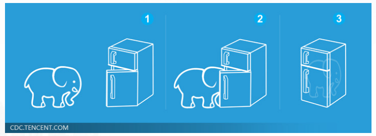


- FFN
- Matrix Operation
- Learning

25

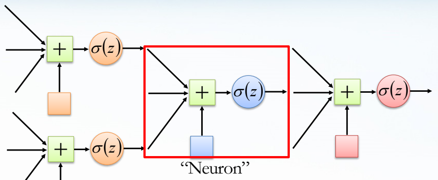
Deep Learning is so simple ……

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

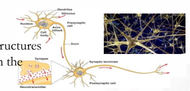


26

Neural Network

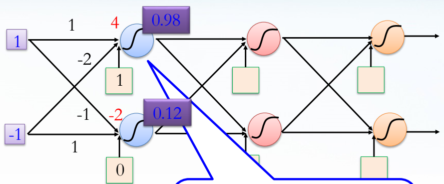


Neural Network:
- Different connection leads to different network structures
- Network parameter θ: all the weights and biases in the "neurons"



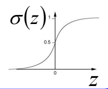
27

Fully Connected Feedforward Network



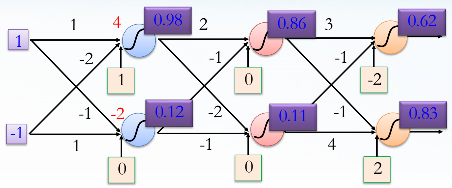
Sigmoid Function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

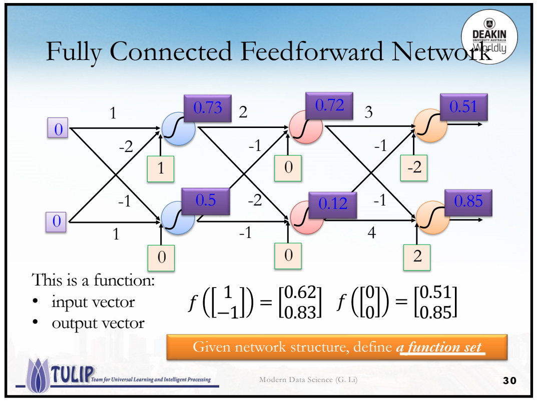


28

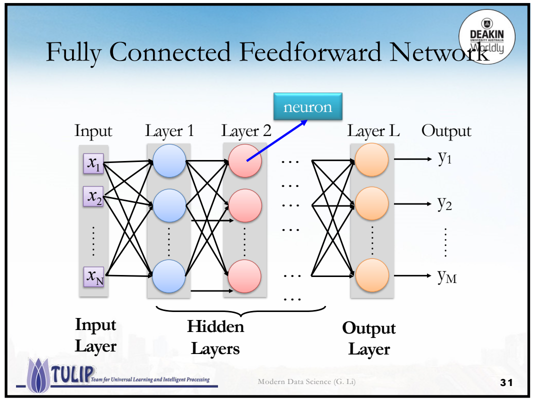
Fully Connected Feedforward Network



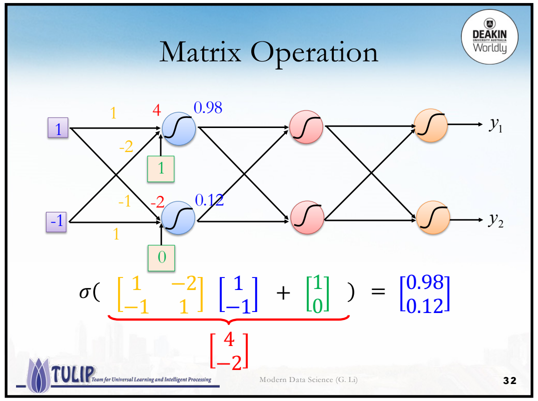
29



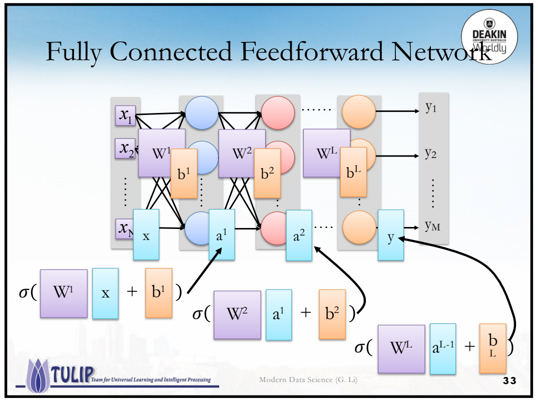
30



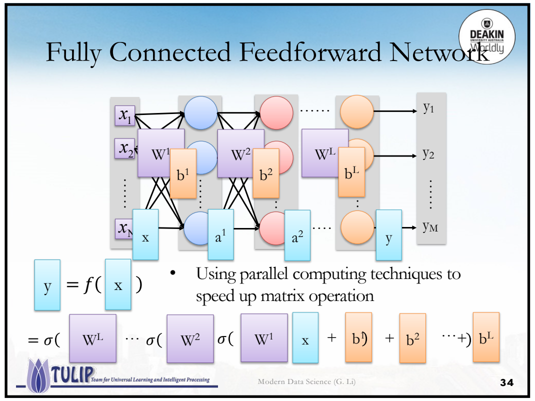
31



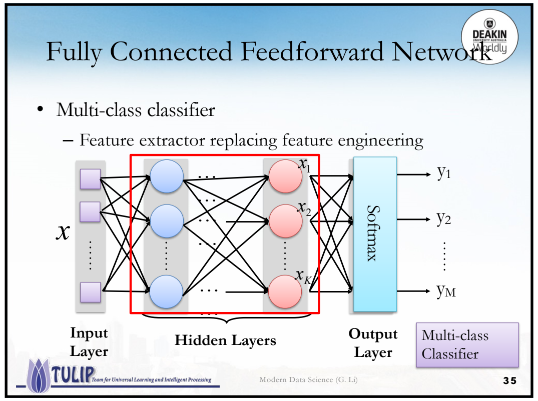
32



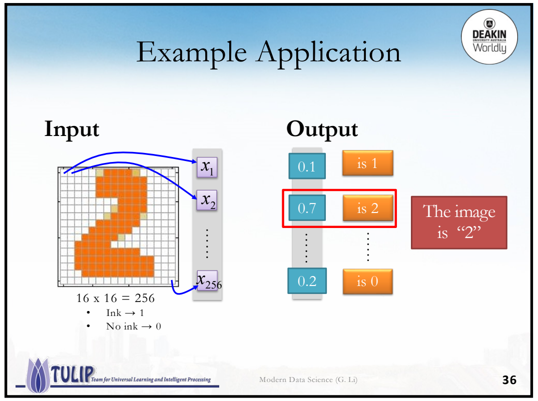
33



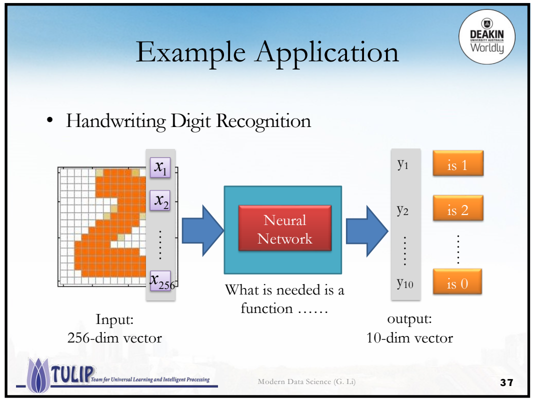
34



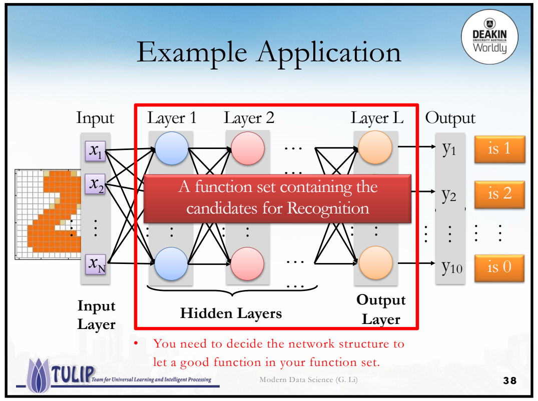
35



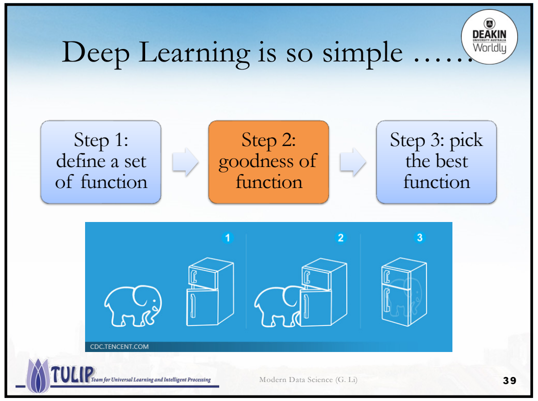
36



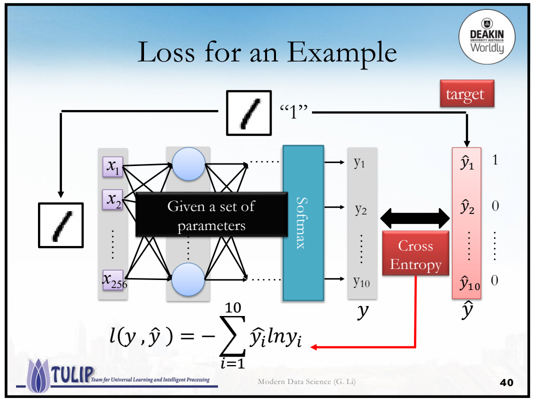
37



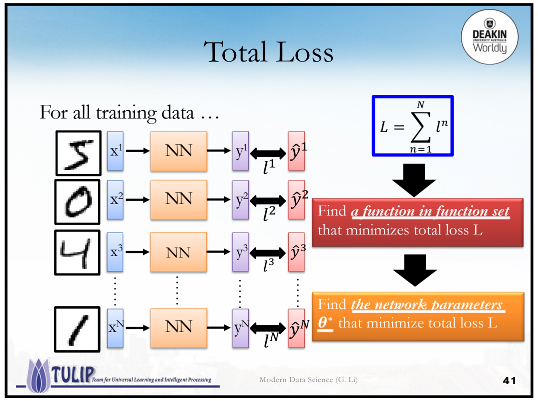
38



39



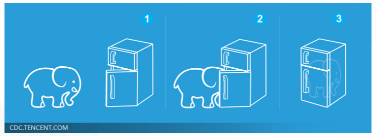
40



41

Deep Learning is so simple ……

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

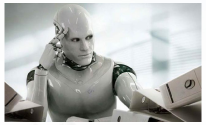


TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 42

42

Gradient Descent

- Gradient Descent is the "learning" of machines in deep learning ……
  - Even alpha go using this approach.



TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 43

43

Gradient Descent

θ

w_1: 0.2 $\xrightarrow[-\mu\,\partial L/\partial w_1]{\text{Compute } \partial L/\partial w_1}$ 0.15

w_2: -0.1 $\xrightarrow[-\mu\,\partial L/\partial w_2]{\text{Compute } \partial L/\partial w_2}$ 0.05

⋮

b_1: 0.3 $\xrightarrow[-\mu\,\partial L/\partial b_1]{\text{Compute } \partial L/\partial b_1}$ 0.2

⋮

$$\nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial b_1} \\ \vdots \end{bmatrix}$$

gradient

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 44

44

Gradient Descent

θ

w_1: 0.2 $\xrightarrow[-\mu\,\partial L/\partial w_1]{\text{Compute } \partial L/\partial w_1}$ 0.15 $\xrightarrow[-\mu\,\partial L/\partial w_1]{\text{Compute } \partial L/\partial w_1}$ 0.09 ……

w_2: -0.1 $\xrightarrow[-\mu\,\partial L/\partial w_2]{\text{Compute } \partial L/\partial w_2}$ 0.05 $\xrightarrow[-\mu\,\partial L/\partial w_2]{\text{Compute } \partial L/\partial w_2}$ 0.15 ……

⋮

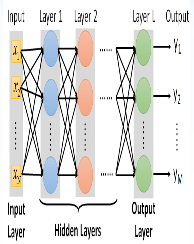
b_1: 0.3 $\xrightarrow[-\mu\,\partial L/\partial b_1]{\text{Compute } \partial L/\partial b_1}$ 0.2 $\xrightarrow[-\mu\,\partial L/\partial b_1]{\text{Compute } \partial L/\partial b_1}$ 0.10 ……

⋮

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 45

45

Back Propagation



- Forward Pass
- Backward Pass

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 46

46

Gradient Descent

Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

Starting Parameters $\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow \cdots\cdots$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \vdots \\ \partial L(\theta)/\partial b_1 \\ \partial L(\theta)/\partial b_2 \\ \vdots \end{bmatrix}$$

Compute $\nabla L(\theta^0)$ $\quad \theta^1 = \theta^0 - \eta \nabla L(\theta^0)$

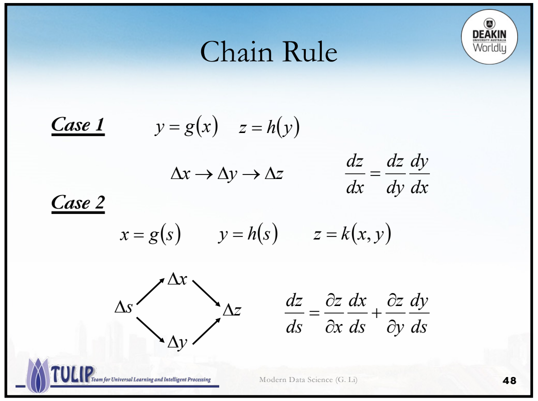
Compute $\nabla L(\theta^1)$ $\quad \theta^2 = \theta^1 - \eta \nabla L(\theta^1)$

Millions of parameters ……

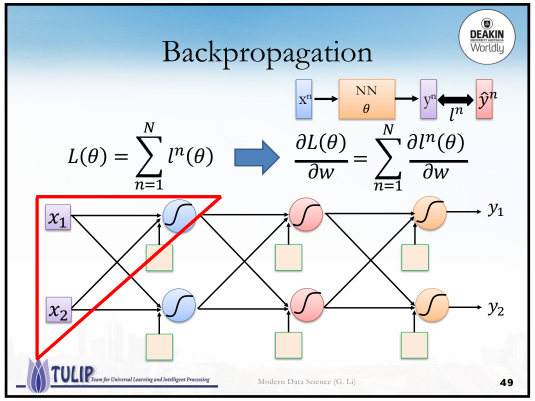
To compute the gradients efficiently, we use ***backpropagation***.

TULIP Team for Universal Learning and Intelligent Processing — Modern Data Science (G. Li) — 47

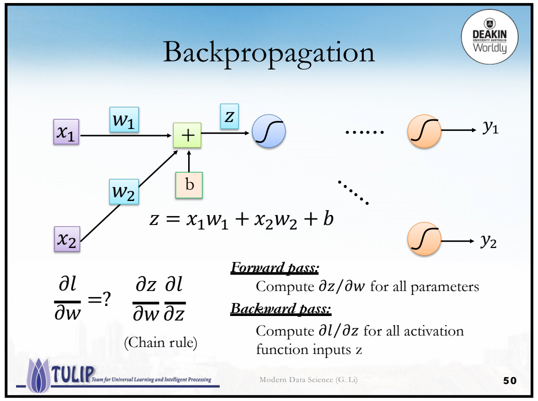
47



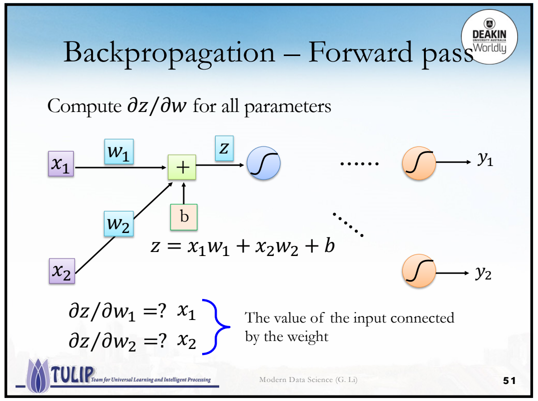
48



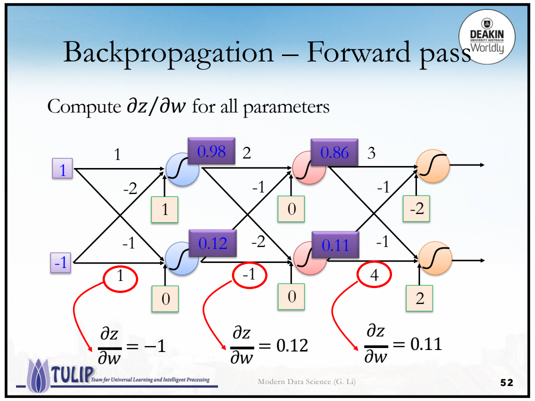
49



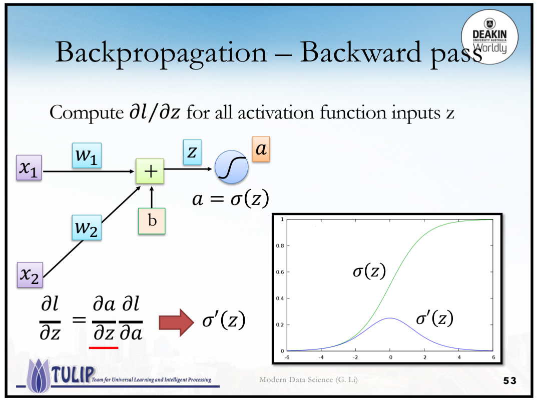
50



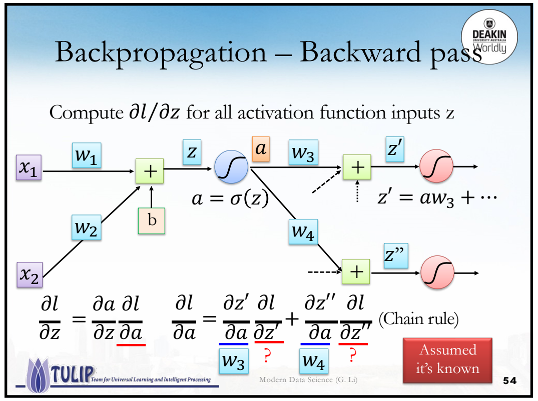
51



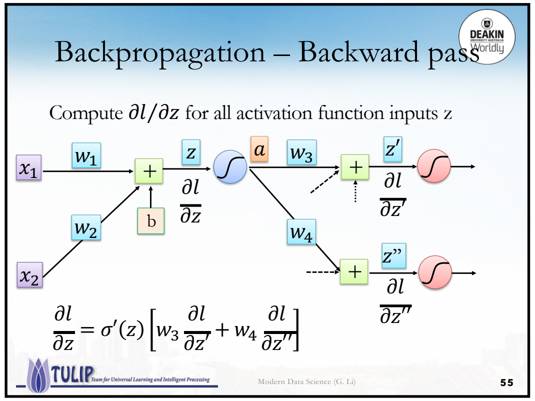
52



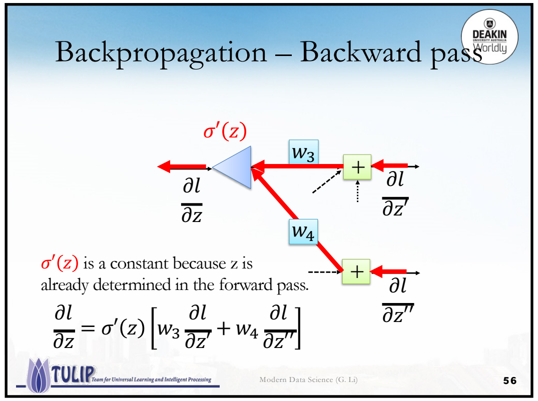
53



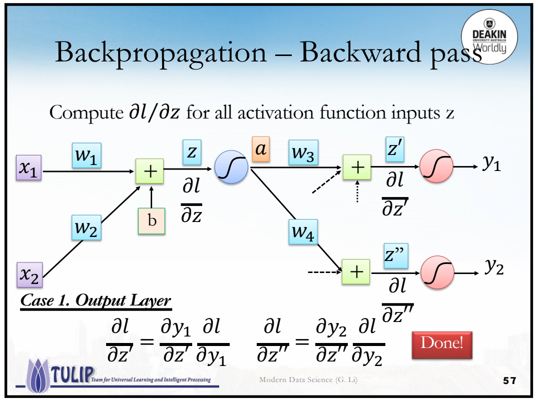
54



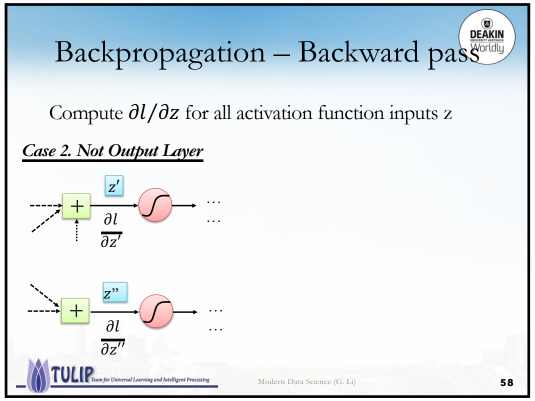
55



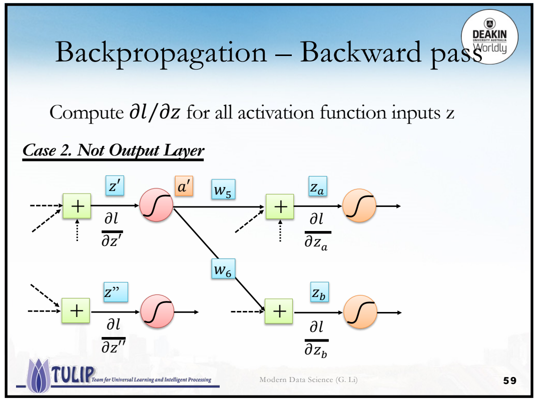
56



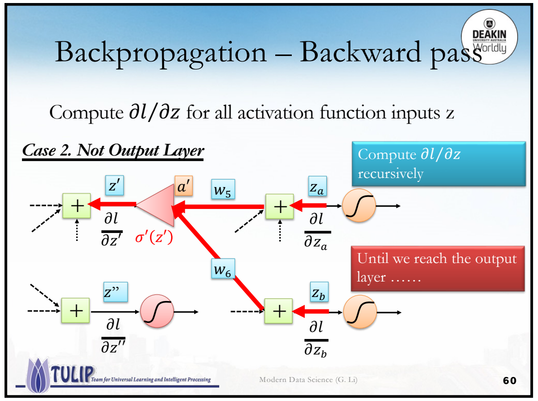
57



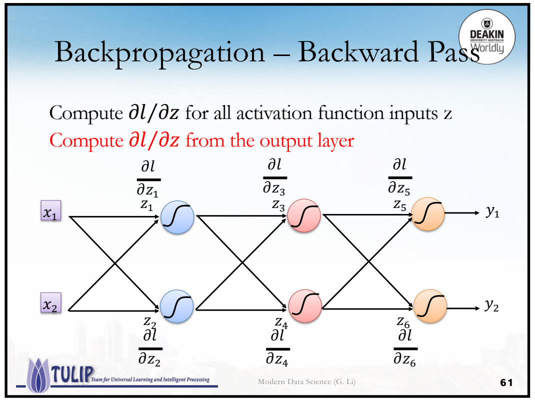
58



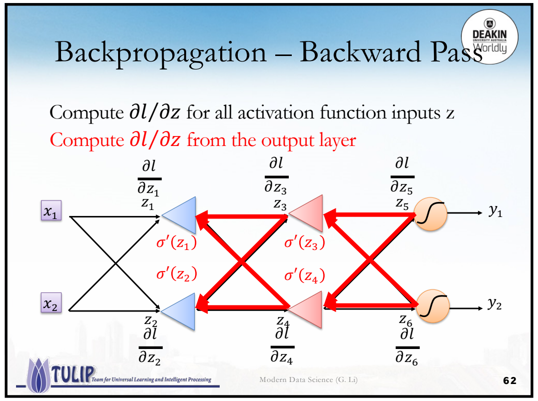
59



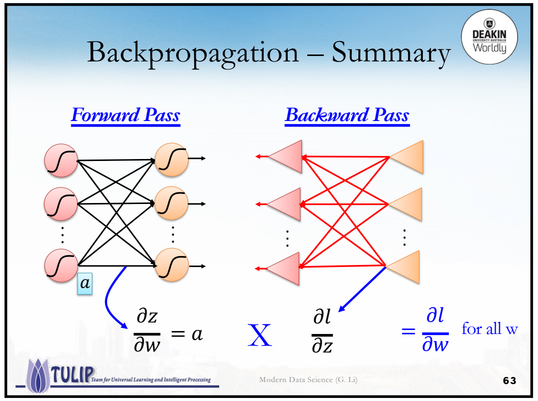
60



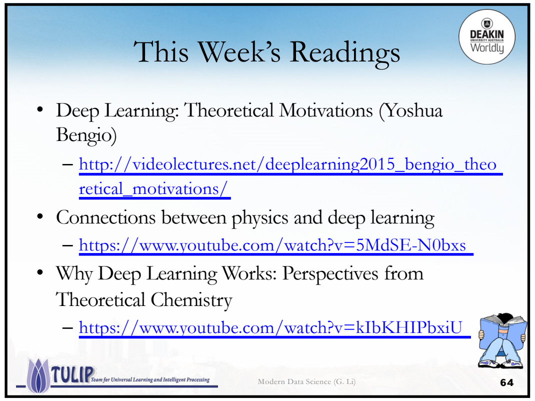
61



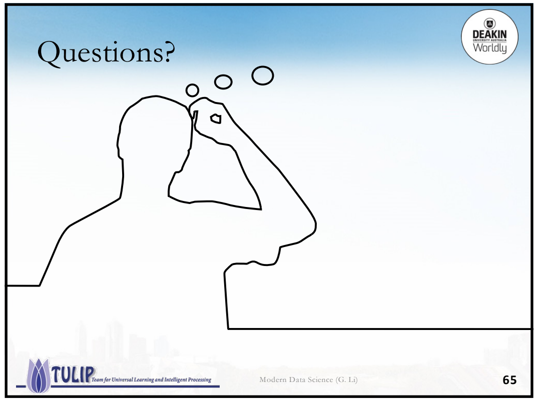
62



63



64



65