

# Deep Learning

## lecture 1

Yi Wu, IIIS Tsinghua University

Spring 2025

Feb-17

# About the Course

- (Hopefully) English Teaching
- Monday 13:30 to 16:05 六教6A117, Week 1-16  
**Special#1: No formal lecture at 11<sup>th</sup> week.**  
**Special#2: Poster session will be on 16<sup>th</sup> week Thursday**
- Offline class required
  - For special cases, contact TA
- Office Hour
  - 六教6A117, Mon 12:30 to 13:30 (TAs will be always there)
  - Email me if you want a 1-on-1 conversation with me
- **Use 钉钉 for announcements and questions**
- Course Website: <https://openpsi-project.github.io/DeepLearning2025/>

# Staff



吴翼

Office: 清华科技园C19

jxwuyi@mails.tsinghua.edu.cn

Research Directions:

**Deep Reinforcement Learning, Large Reasoning Model;  
Multi-Agent Learning; Robot Learning; Distributed Learning System**

Feng Gao (TA)  
高枫  
E-mail:  
feng.gao220@gmail.com



Jinglun Zhao (TA)  
赵京伦  
E-mail:  
u3514271@gmail.com



Yutao Ouyang(TA)  
欧阳宇涛  
E-mail:  
oyyt24@mails.tsinghua.edu.cn



Jingzhi Cui(TA)  
崔景植  
E-mail:  
cjz24@mails.tsinghua.edu.cn

# Syllabus

- 8 writing homework (due in 1 week) + 4 coding project + final project (2~3 members/team) + final exam

**A lot of work!**

#	Date	Topic	Homework	Note
1	2/17	Overview	HW0	CP1 release
2	2/24	Supervised Learning (1)		CP2 release
3	3/3	Supervised Learning (2)	HW1	CP1 due;
4	3/10	Energy-Based Model	HW2	
5	3/17	Variational Autoencoder	HW3	CP2 due; CP3 release
6	3/24	Generative Adversarial Network	HW4	
7	3/31	Normalizing Flow & Diffusion Model	HW5	
8	4/7	Sequence Modeling (1)	HW6	CP 4 release; CP3 due;
9	4/14	Sequence Modeling (2)	HW7	Project announcement
10	4/21	LLM and Advanced Topics		
11	<b>4/28</b>	<b>NO COURSE!!</b>		<b>Q/A on Proposal;</b> CP4 due
12	5/10	Learning Structured Data	HW8	Project proposal due after holiday
13	5/12	Advanced Topics: Self-Supervised Learning, Meta-Learning, AutoML		
14	5/19	Security, Interpretability and Fairness		
15	5/26	Frontier and Applications (Guest Lecture)		
<sup>2/17</sup> 16	<b>6/5</b>	<b>Final Project Poster Session</b>		<b>Thursday! Project report due TBA</b>

# Grading

- Overall: Homework 20% + Coding 40% + Project 20% + Final 20%
  - Coding: 10% each
    - backpropagation; classification; generative models; language models;
  - Homework:
    - We will convert all the homework grades into 20 pts
    - **No late submission allowed**
  - Project: team project; report + poster presentation;
    - Each member's contribution must be clearly stated
  - **STRICT HOMEWORK POLICY!**
    - Discussion is encouraged
    - Ask your TAs, friends, or even AIs
    - But do your homework on your own

# About the Course

- Prerequisites
  - Linux and Python coding
  - Calculus & basic Machine Learning
  - Mentally ready for **heavy** engineering work 😊
- What will you learn?
  - Not about “how to import python packages”!
  - Not about “theorem proving”
- Formal Mission
  - How to train neural nets from data
  - How to use neural nets to solve problems

## Convergence Analysis of Two-layer Neural Networks with ReLU Activation

Yuanzhi Li  
Computer Science Department  
Princeton University  
yuanzhi@cs.princeton.edu

Yan  
Computer Sci  
Cornell  
yangyuan@cornell.edu

November 1, 2017

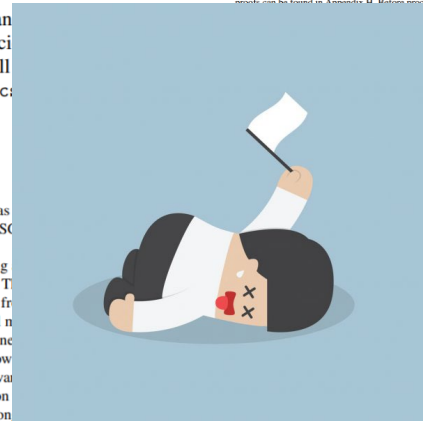
### Abstract

In recent years, stochastic gradient descent (SGD) based techniques has training neural networks. However, formal theoretical understanding of why SGD practice is largely missing.

In this paper, we make progress on understanding this mystery by providing on a rich subset of two-layer feedforward networks with ReLU activations. This special structure called “identity mapping”. We prove that, if input follows from standard  $O(1/\sqrt{d})$  initialization of the weights, SGD converges to the global minimum of steps. Unlike normal vanilla networks, the “identity mapping” makes our global minimum is unique. To complement our theory, we are also able to show networks with this mapping have better performance compared with normal vanilla networks.

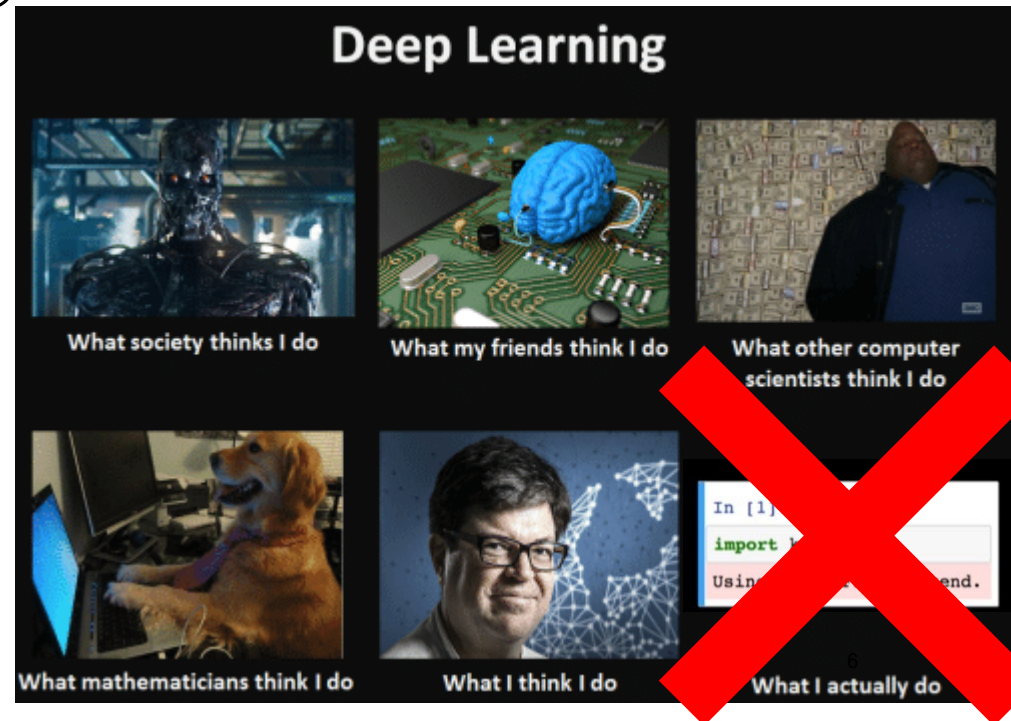
Our convergence theorem differs from traditional non-convex optimization converges to optimal in “two phases”: In phase I, the gradient points to the wrong direction and the function  $g$  gradually decreases. Then in phase II, SGD enters a nice one point convex region and converges. We also show that the identity mapping is necessary for convergence, as it moves the initial point to a better place for optimization. Experiment verifies our claims.

OpenStax @ IIIS  
As we saw in Theorem 3.3, if  $\|W\|_2, \|W^*\|_2$  is bounded by a constant  $\gamma = \frac{1}{100}$ , and the potential function  $g \leq 0.1$ ,  $L(W)$  is 0.03-one point convex, which will give us convergence guarantee according to Lemma 2.5. However,  $g$  could be larger than 0.1 initially, and as we run SGD,  $\|W_t\|_2$  might be larger than  $\frac{1}{100}$  as well. In this section, we address both problems by analyzing the dynamics of SGD, thus prove Theorem 3.2. The proofs can be found in Appendix B. Before proceeding to the interesting stuff, we need a simpler form of  $\nabla L(W)$



negative gradient of  $L(W)$  is approximately  
 $(W^* - W)^T - 2\text{Diag}(W^* - W) + gI + W$   
 For is  $\|Q(W) - [-\nabla L(W)]\|_2 \leq 61\gamma^2$ .  
 $\|L(W)\|_2 \leq 64\gamma$ .  
 We use subscript  $t$  under each variable to denote its value at the  $t$ -th iteration  $(W^* - W_t)$ . We first compute the update rule for  $g_t$ .  
 $g_t \geq 100, \eta \leq \frac{1}{200},$  then  $|g_{t+1}| \leq (1 - 0.95\eta d)|g_t| + 864\eta^2 \gamma^2 + \dots$   
 so, we also need to compute its updating rule:  
 $\|g_{t+1}\|_2 \leq (1 - \eta \frac{d+1}{2}) \|g_t\|_2 + \eta(6.61\gamma + 1.03|g_t| + \epsilon)\sqrt{d}$   
 show that  $g_t$  will shrink, conditioned on that  $\|W_t\|_2$  is bounded  
 $\|W_t\|_2 \leq \gamma \leq \frac{1}{100}, d \geq 100, \eta \leq \frac{1}{200}, \epsilon \leq \gamma^2,$  then  $|g_t|$  will keep  $\gamma$ , until  $|g_t| \leq 197\gamma^2$  for  $t_1 \leq \frac{1}{100}$ .  
 stays bounded by  $\gamma$  during the process described in Lemma C.5.  
 such that if  $\|W_t\|_2, \|W^*\|_2 \leq \gamma, d \geq 100, \eta \leq \frac{1}{200}, \epsilon \leq \gamma^2,$   
 always have  $\|W_T\|_2 \leq \gamma \leq \frac{1}{100}$  for any  $T > 0$ .  
 and  $\|W_T\|_2 \leq \gamma$ , which means we are in Phase II. The next  
 er.

Lemma C.7. There exists a constant  $\gamma_0 > 0$  such that if  $\|W_0\|_2, \|W^*\|_2 \leq \gamma_0, d \geq 100, \eta \leq \frac{1}{200}, \epsilon \leq \gamma^2,$  then after  $|g_{t_1}| \leq 197\gamma^2$ , Phase I ends and Phase II starts. That is, for every  $T > t_1, \|W_T\|_2 \leq \gamma$  and  $|g_T| \leq 0.1$ .  
 Proof for Theorem 3.2. We immediately get Theorem 3.2 by combining the above three lemmas. They show that  $g_t$  will decrease to a small value in Phase I (Lemma C.5),  $\|W_t\|_2$  will keep small during this process (Lemma C.6), and they all keep small afterwards (Lemma C.7). □



# About the Course

- Prerequisites
  - Linux and Python coding
  - Calculus & basic Machine Learning
  - Mentally ready for **heavy** engineering work 😊
- What will you learn?
  - Not about “how to import python packages”!
  - Not about “theorem proving”
- Formal Mission
  - How to **train neural nets** from **data**
  - How to use neural nets to **solve problems**

## Convergence Analysis of Two-layer Neural Networks with ReLU Activation

Yuanzhi Li  
Computer Science Department  
Princeton University  
yuanzhi@cs.princeton.edu

Yan  
Computer Sci  
Cornell  
yangyuan@cornell.edu

November 1, 2017

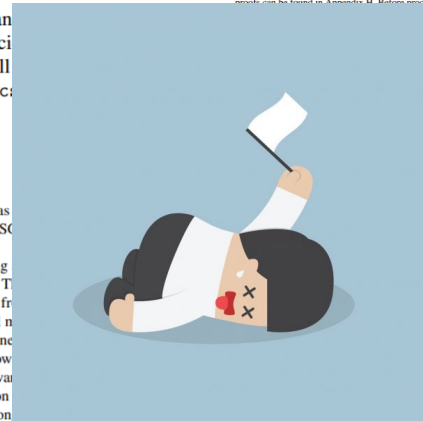
### Abstract

In recent years, stochastic gradient descent (SGD) based techniques has training neural networks. However, formal theoretical understanding of why SGD practice is largely missing.

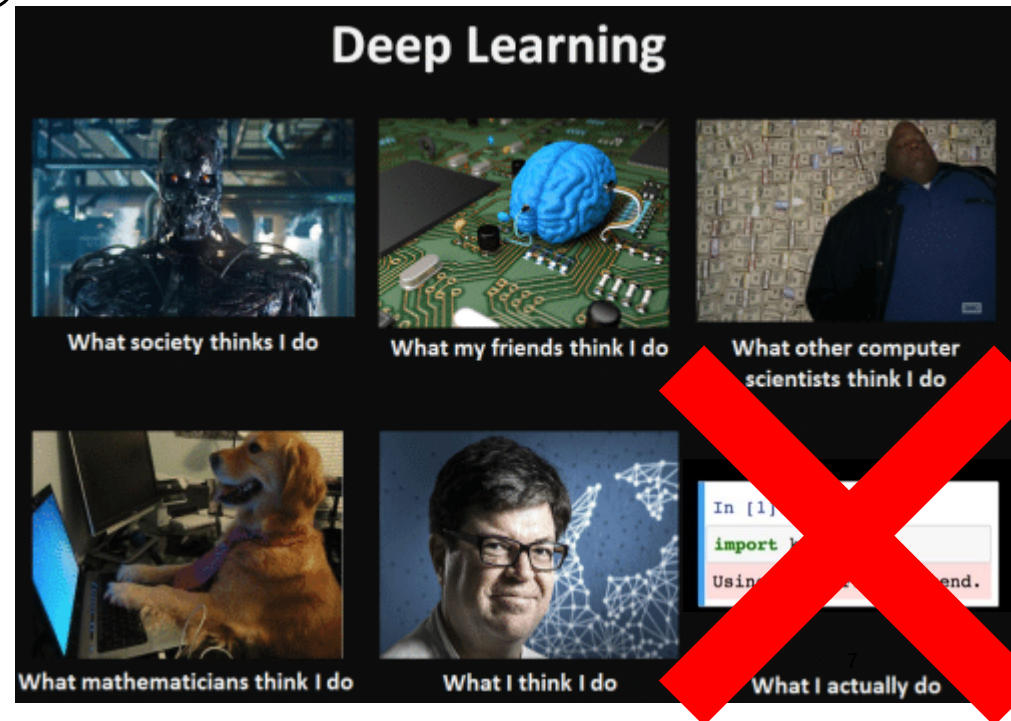
In this paper, we make progress on understanding this mystery by providing on a rich subset of two-layer feedforward networks with ReLU activations. This special structure called “identity mapping”. We prove that, if input follows from standard  $O(1/\sqrt{d})$  initialization of the weights, SGD converges to the global minimum of steps. Unlike normal vanilla networks, the “identity mapping” makes our global minimum is unique. To complement our theory, we are also able to show networks with this mapping have better performance compared with normal vanilla networks.

Our convergence theorem differs from traditional non-convex optimization converges to optimal in “two phases”: In phase I, the gradient points to the wrong direction and the function  $g$  gradually decreases. Then in phase II, SGD enters a nice one point convex region and converges. We also show that the identity mapping is necessary for convergence, as it moves the initial point to a better place for optimization. Experiment verifies our claims.

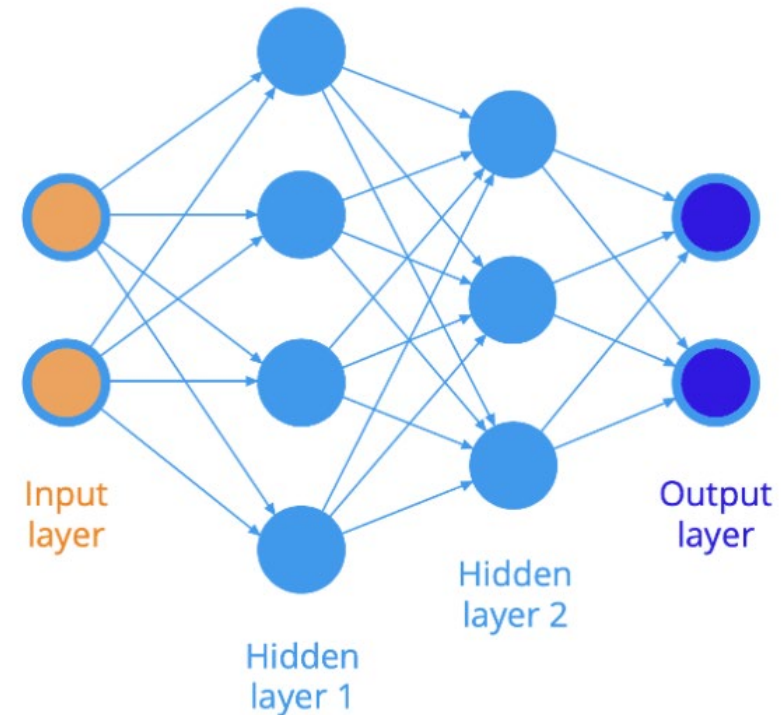
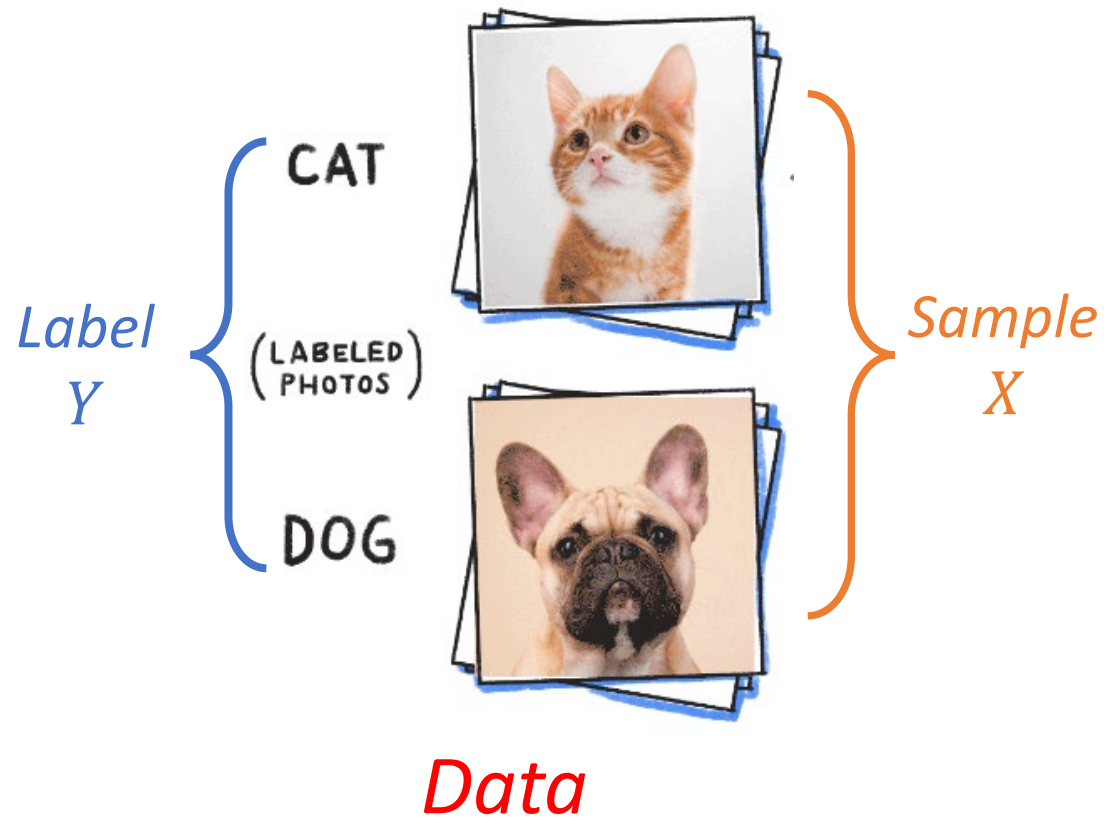
OpenSci @ IIIS  
As we saw in Theorem 3.3, if  $\|W\|_2$  is bounded by a constant  $\gamma = \frac{1}{100}$ , and the potential function  $g \leq 0.1$ ,  $L(W)$  is 0.03-one point convex, which will give us convergence guarantee according to Lemma 2.5. However,  $g$  could be larger than 0.1 initially, and as we run SGD,  $\|W_t\|_2$  might be larger than  $\frac{1}{100}$  as well. In this section, we address both problems by analyzing the dynamics of SGD, thus prove Theorem 3.2. The proofs can be found in Appendix B. Before proceeding to the interesting stuff, we need a simpler form of  $\nabla L(W)$



negative gradient of  $L(W)$  is approximately  
 $(W^* - W)^T - 2\text{Diag}(W^* - W) + gI - W$   
 For is  $\|Q(W) - [-\nabla L(W)]\|_2 \leq 61\gamma^2$ .  
 $\|L(W)\|_2 \leq 64\gamma$ .  
 We use subscript  $t$  under each variable to denote its value at the  $t$ -th iteration  $(W^* - W_t)_t$ . We first compute the update rule for  $g_t$ .  
 $g_t \geq 100, \eta \leq \frac{1}{200}$ , then  $|g_{t+1}| \leq (1 - 0.95\eta d)|g_t| + 864\eta\gamma^2 + \dots$   
 so, we also need to compute its updating rule:  
 $\|W_{t+1}\|_2 \leq (1 - \eta \frac{d+1}{2}) \|W_t\|_2 + \eta(6.61\gamma + 1.03|g_t| + \epsilon)\sqrt{d}$   
 show that  $g_t$  will shrink, conditioned on that  $\|W_t\|_2$  is bounded  
 $\|W_t\|_2 \leq \gamma \leq \frac{1}{100}, d \geq 100, \eta \leq \frac{1}{200}, \epsilon \leq \gamma^2$ , then  $|g_t|$  will keep  $\gamma$ , until  $|g_t| \leq 197\gamma^2$  for  $t_1 \leq \frac{1}{100}$ .  
 says bounded by  $\gamma$  during the process described in Lemma C.5.  
 such that if  $\|W_t\|_2, \|W^*\|_2 \leq \gamma_0, d \geq 100, \eta \leq \frac{1}{200}, \epsilon \leq \gamma^2$ ,  
 always have  $\|W_T\|_2 \leq \gamma \leq \frac{1}{100}$  for any  $T > 0$ .  
 and  $\|W_T\|_2 \leq \gamma$ , which means we are in Phase II. The next  
 er.  
 Lemma C.6. There exists a constant  $\gamma_0 > 0$  such that if  $\|W_t\|_2, \|W^*\|_2 \leq \gamma_0, d \geq 100, \eta \leq \frac{1}{200}, \epsilon \leq \gamma^2$ ,  
 then after  $|g_t| \leq 197\gamma^2$ , Phase I ends and Phase II starts. That is, for every  $T > t_1$ ,  $\|W_T\|_2 \leq \gamma$  and  $|g_T| \leq 0.1$ .  
 Proof for Theorem 3.2. We immediately get Theorem 3.2 by combining the above three lemmas. They show that  
 $g_t$  will decrease to a small value in Phase I (Lemma C.5),  $\|W_t\|_2$  will keep small during this process (Lemma  
 C.6), and they all keep small afterwards (Lemma C.7).  $\square$



# Some Terminologies

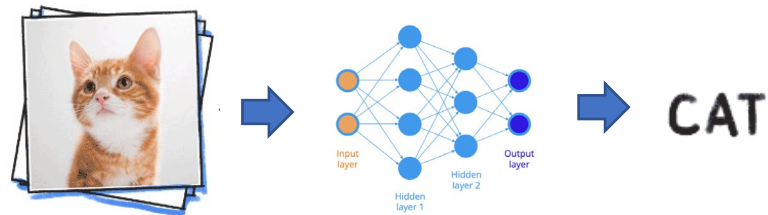


*Neural Network (model,  $f$ )*

**Goal: derive a neural net model ( $f$ ) from data**

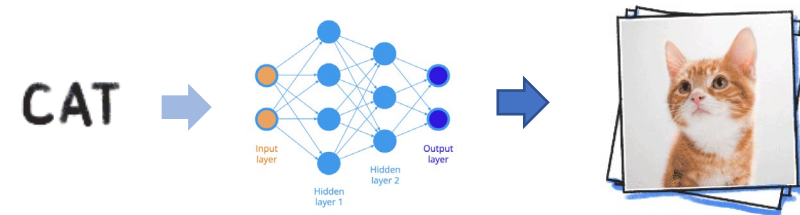


# The Problems to Solve (in a nutshell)



$X \longrightarrow Y$

*Classification  
(discriminative model)*



$Y \longrightarrow X$

*Generation  
(generative model)*

***(a bit more detailed) Goal: derive a discriminative/generative neural net model from data***

# About the Course



- What will be covered in the course?
  - How to design a neural network (*network architecture*)
  - **Algorithms** to learn a good model for  $X \rightarrow Y / Y \rightarrow X$
  - How to **apply** a learned model to solve different **real-world challenges**
  - Broad **understanding** of active research domains

A class about “*tools*” and “*ideas*” of neural nets for practical problems

***You don't need to understand every details in the class***

# Additional References

- Deep Learning
  - <https://mitpress.mit.edu/books/deep-learning>
- Dive into Deep Learning
  - <https://d2l.ai/>
- Use Google!
  - DeepMind and OpenAI resources
  - CMU & Stanford & Berkeley Courses
  - Some wonderful BLOGs
    - <https://towardsdatascience.com/>
    - <https://colah.github.io/>
    - <https://lilianweng.github.io/lil-log/>

# Lecture 1: Overview

- Part 1: A comprehensive history recap of deep learning
  - with many lessons
- Part 2: some programming basics
  - with HW0 released
  - CP1 will be released this week but no hurry

# Deep Learning is Everywhere



天气怎么样  
轻点以编辑 >

目前天气为晴，气温1°C。空气质量为良，指数61。

天气

北京市 晴朗 1°  
降雨概率：10% 1° / 0°

空气质量指数 61 — 良

晚上 8 点	☁️	1
晚上 9 点	☁️	1
晚上 10 点	☁️	0
晚上 11 点	☁️	0

更多信息，请访问 Weather.com

更多请见 QWeather.com

Google 翻译

英语 ↔ 中文 (简体)

The deep learning course is my favorite! ×

深度课程是我的最爱! ☆

Shēndù xuéxí kèchéng shì wǒ de zuì ài!

2 / 17



Copyright © IIIS, Tsinghua University

Yann LeCun<sup>1,2</sup>, Yoshua Bengio<sup>3</sup> & Geoffrey Hinton<sup>4,5</sup>

## Neural Networks

## Representation Learning

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

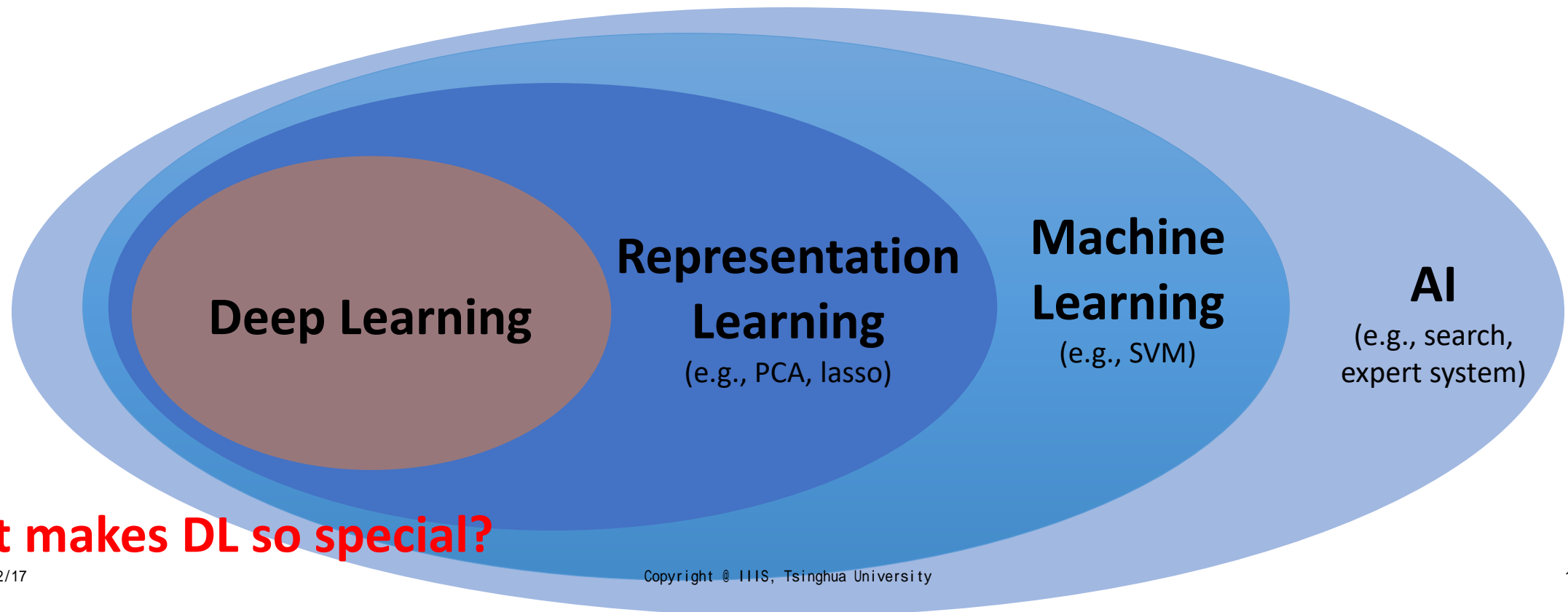
Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition<sup>1-4</sup> and speech recognition<sup>5-7</sup>, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules<sup>8</sup>, analysing particle accelerator data<sup>9,10</sup>, reconstructing brain circuits<sup>11</sup>, and predicting the effects of mutations in non-coding DNA on gene expression and disease<sup>12,13</sup>. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding<sup>14</sup>, particularly topic classification, sentiment analysis, question answering<sup>15</sup> and language translation<sup>16,17</sup>.

We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

# What is deep learning?

- Deep Learning is a class of ***machine learning*** methods that use ***neural networks*** to ***learn representations*** from raw data.



**What makes DL so special?**

# Deep learning

Lecture 1, Deep Learning, 2025 Spring

OpenPsi @ IIIS

Yann LeCun<sup>1,2</sup>, Yoshua Bengio<sup>3</sup> & Geoffrey Hinton<sup>4,5</sup>

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

## Feature Engineering

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition<sup>1-4</sup> and speech recognition<sup>5-7</sup>, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules<sup>8</sup>, analysing particle accelerator data<sup>9,10</sup>, reconstructing brain circuits<sup>11</sup>, and predicting the effects of mutations in non-coding DNA on gene expression and disease<sup>12,13</sup>. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding<sup>14</sup>, particularly topic classification, sentiment analysis, question answering<sup>15</sup> and language translation<sup>16,17</sup>.

We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.



# What is deep learning?

- Deep Learning is a class of ***machine learning*** methods that use ***neural networks*** to ***learn representations*** from raw data.

- What makes deep learning so special?

- Conventional machine learning (shallow learning)

First hand-design features  $\phi(x)$  for each  $x$

Then learn  $f$  w.r.t.  $\theta_f$  by  $y = f(\phi(x), \theta_f)$  w.  $\phi(x)$  fixed

- Deep Learning (end-to-end learning both  $\phi$  and  $f$  w.r.t  $\theta$ )

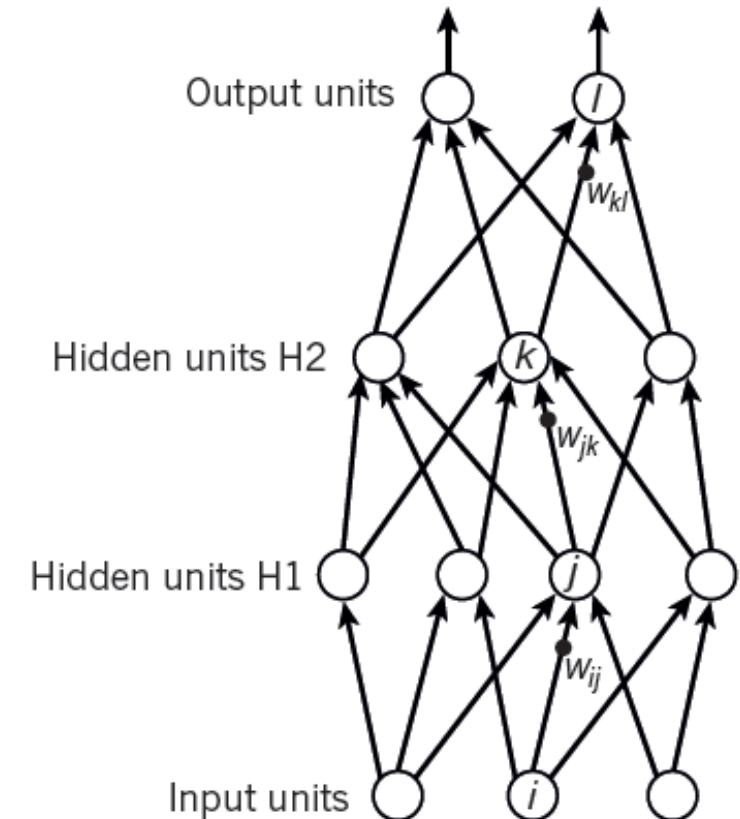
$y = f(NN(x; \theta_{NN}); \theta_f)$  w.  $NN(\cdot)$  a neural net with parameter  $\theta_{NN}$

**Assumption#1: a neural network can (approximately) represent any function**

**Assumption#2: a smart algorithm can (automatically) derive a good neural net**

# What is Neural Network?

- Artificial Neural Network (ANN, or NN) is *a computing system* based on a collection of *connected units or nodes* called artificial neurons, which *loosely* model the neurons in a biological brain.
  - Connections are also called edges
  - Neurons and edges typically has *weights* associated
  - A weight adjusts as learning proceeds
  - The output of each neuron is computed by some non-linear function of the sum of its inputs



A feedforward NN

- How is it invented?

# The Rise of Connectionism

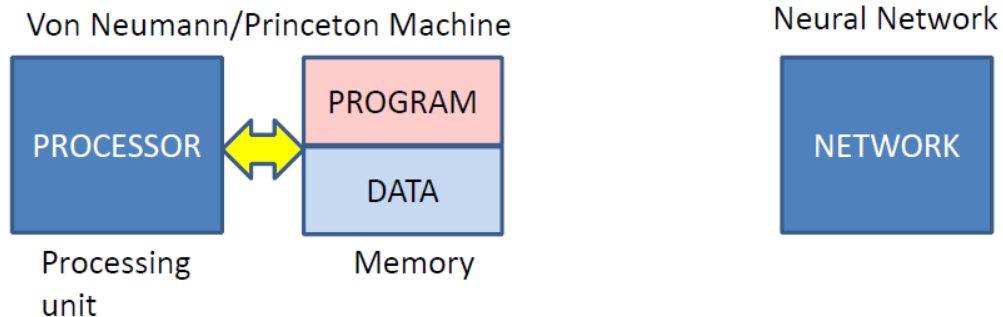
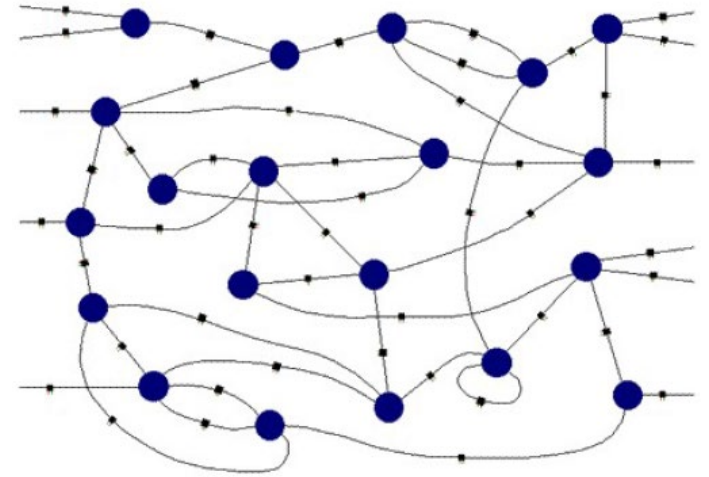
- The human brain is a connectionist machine
  - Bain, A. (1873). Mind and body. The theories of their relation. London: Henry King.
  - Ferrier, D. (1876). The Functions of the Brain. London: Smith, Elder and Co.
- The processing/capacity of the brain is a function of these connections



Many neurons connect in to each neuron  
Each neuron connects out to many neurons  
Neurons can be activated

# Connectionist Machines

- Connectionist machines emulate brain structure
  - A computation network of processing elements
- All world *knowledge* is stored in the *connections* between the elements
- Neural networks are connectionist machines



- The machine has many non-linear processing units
- Cyclic Connections may also define memory

# Turing's Connectionist Machines

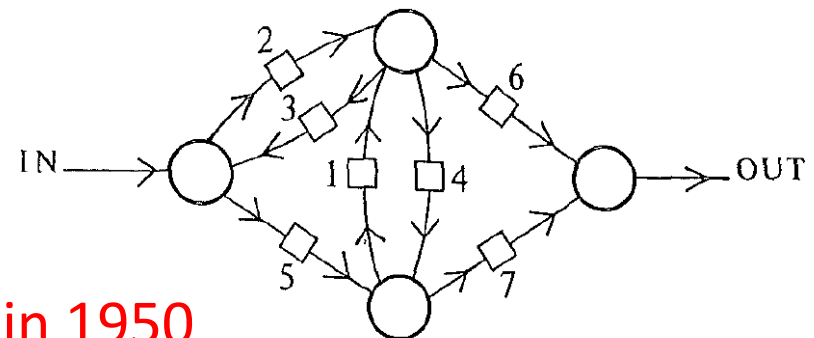
- B-type machine
  - A unit is a NAND node (NOT+AND)
  - An edge can modify the signal
  - Learning: edge types are determined by an A-type machine
- Intelligent Machinery
  - Alan Turing, as early as 1948
- Unfortunately, it was dismissed by his employer and went unpublished until 1968, 14 years after his death.

## Intelligent Machinery

A. M. Turing  
[1912—1954]

### Abstract

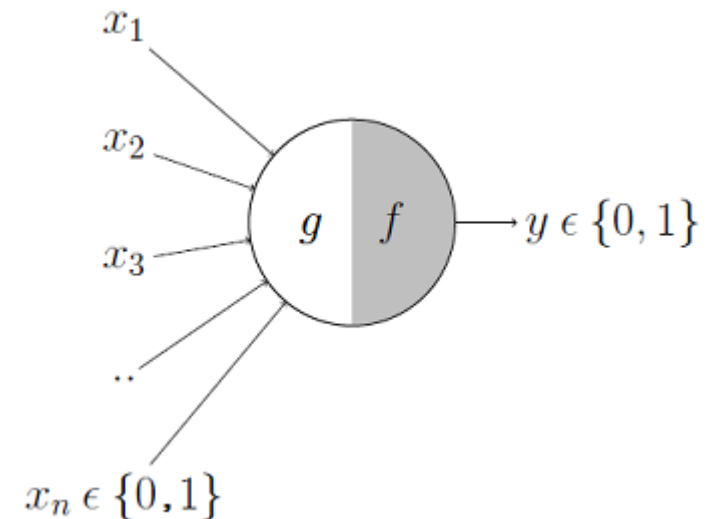
The possible ways in which machinery might be made to show intelligent behaviour are discussed. The analogy with the human brain is used as a guiding principle. It is pointed out that the potentialities of the human intelligence can only be realized if suitable education is provided. The investigation mainly centres round an analogous teaching process applied to machines. The idea of an unorganized machine is defined, and it is suggested that the infant human cortex is of this nature. Simple examples of such machines are given, and their education by means of rewards and punishments is discussed. In one case the education process is carried through until the organization is similar to that of an ACE.



Before his "Turing Test" paper in 1950

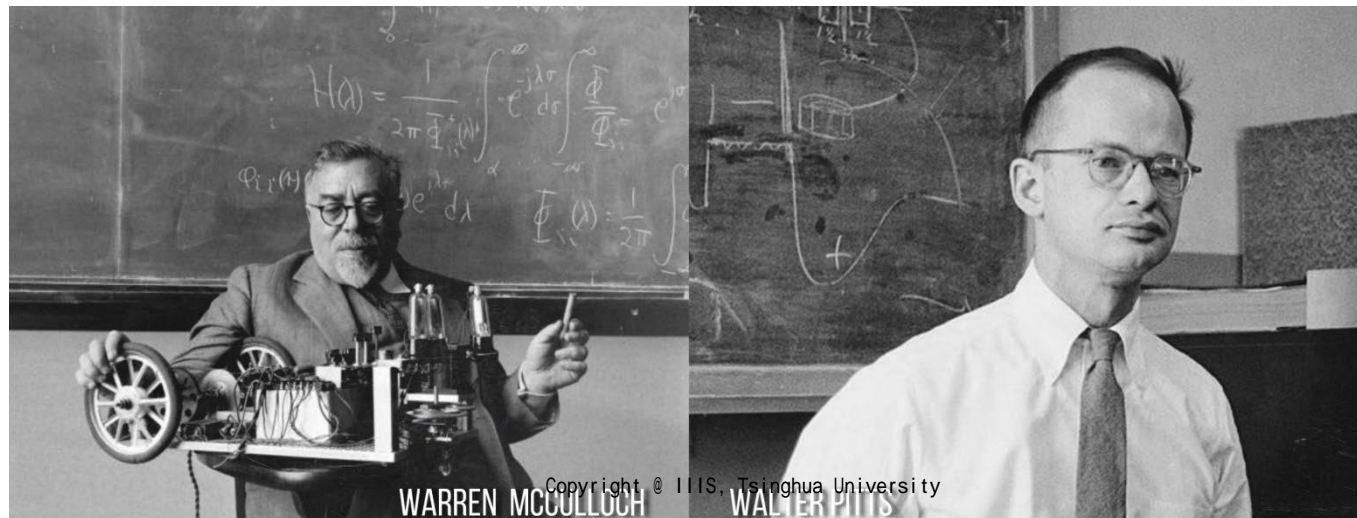
# McCulloch-Pitts Neuron

- The first mathematical model of neuron (1943)
- The model was specifically targeted as a computational model of the "nerve net" in the brain.
- Boolean unit
  - $x_n \in \{0,1\}$
  - $g(x) = \sum_i x_i$
  - $y = f(g(x)) = \mathbb{I}[g(x) > \theta]$  (or flip)
- Represent any Boolean functions



# McCulloch-Pitts Neuron

- The first artificial neuron.
  - *A Logical Calculus of the Ideas Immanent in Nervous Activity, 1943*
  - McCulloch: Neurophysiologist, Prof at U. Illinois at Chicago
  - Pitts: 20 year-old homeless wannabe logician who arrived at his door
  - After five years of unofficial studies, the University of Chicago awarded Pitts an Associate of Arts (his only earned degree) for his work on the paper



# Hebbian Learning

- The first learning algorithm for neural networks
  - “Organization of behavior”, Donald Hebb, 1949
- Principle: *Cells that fire together wire together.*
  - *If neuron  $x_i$  repeatedly triggers neuron  $y$ , the synaptic knob (神经突触) connecting  $x_i$  to gets larger*
- Mathematical model:
$$w_i = w_i + \eta x_i y$$

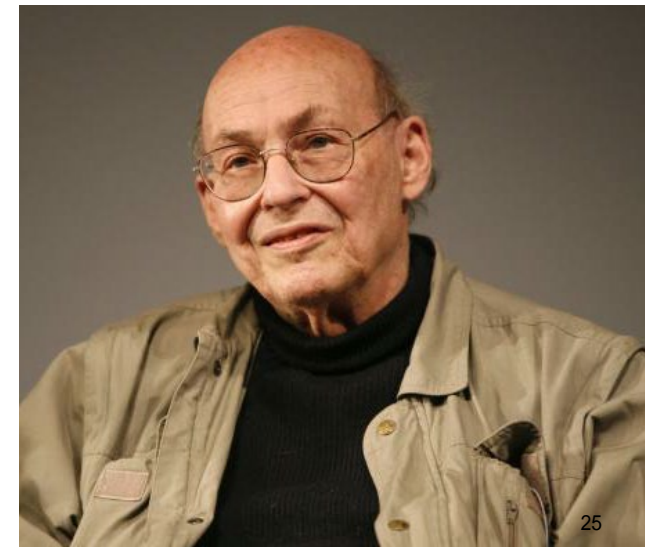
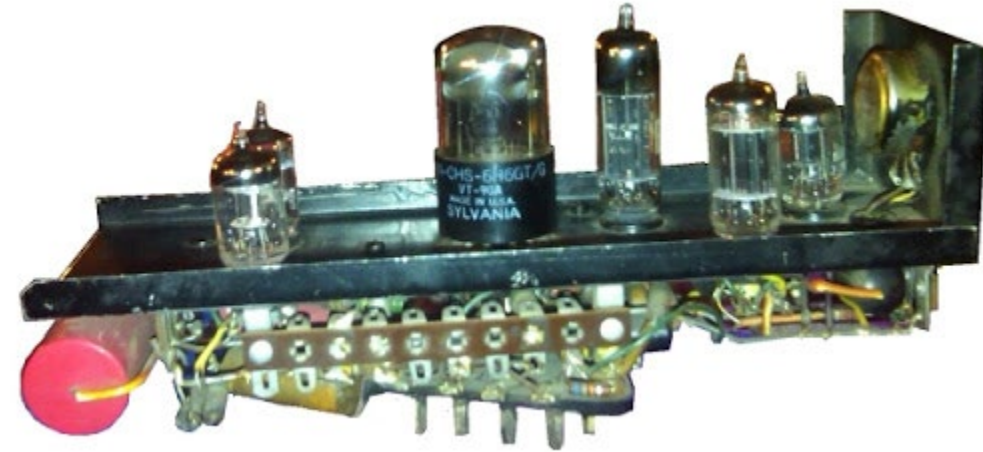
$w_i$  is the weight between  $x_i$  and  $y$
- The basis of many learning algorithms in Machine Learning!





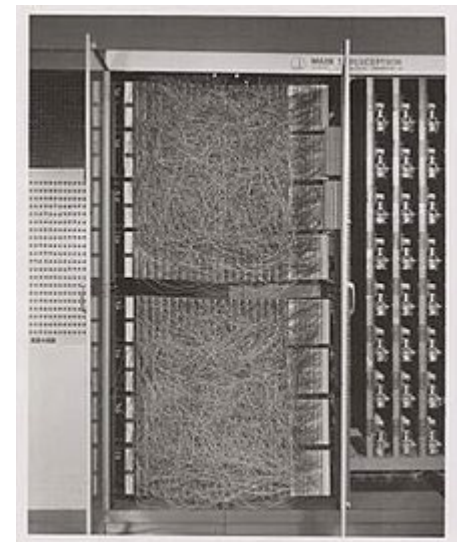
# The First Implementation Attempt

- SNARC (Stochastic Neural Analog Reinforcement Calculator)
  - 1951 summer
  - 40 units with random wires
  - An operator will press a button to give reward to the machine
  
- Marvin Minsky
  - 1969 Turing Award
  - Founder of MIT CSAIL



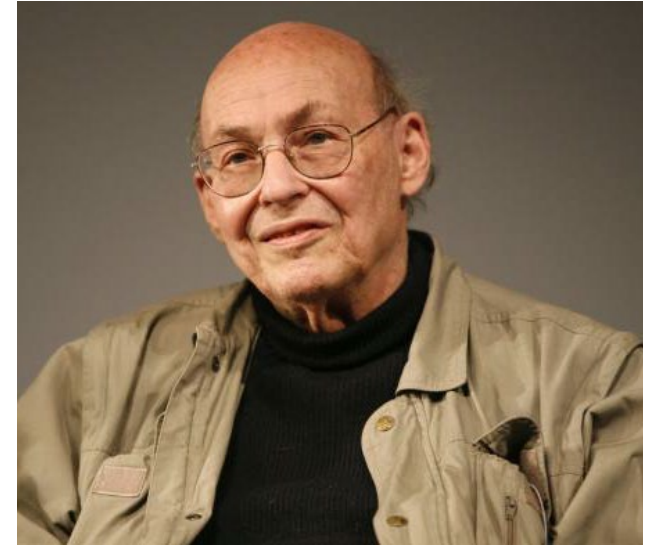
# The First Spring of AI

- Perceptron, 1958 by Frank Rosenblatt
  - First practical implementation of learning neural networks
- Mark 1 perceptron: a machine for image recognition
  - 400 photocells, randomly connected to the "neurons".
  - Weights were encoded in potentiometers (继电器)
  - Weight updates during learning
- *“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”* --- New York Times, 1958



# The MIT Summer Vision Project

- Marvin Minsky
  - 1969 Turing Award
  - Founder of MIT CSAIL
- The project at 1966 summer
  - *“The summer vision project is an attempt to **use our summer workers effectively in the construction of a significant part of a visual system.** The particular task was chosen partly **because it can be segmented into sub-problems** which allow individuals to **work independently** and yet participate in the construction of a system complex enough to be real **landmark in the development of ‘pattern recognition’**”*
- Summary
  - “奋斗一个夏天，解决模式识别”



A Failed Attempt 😞

# The Perceptron

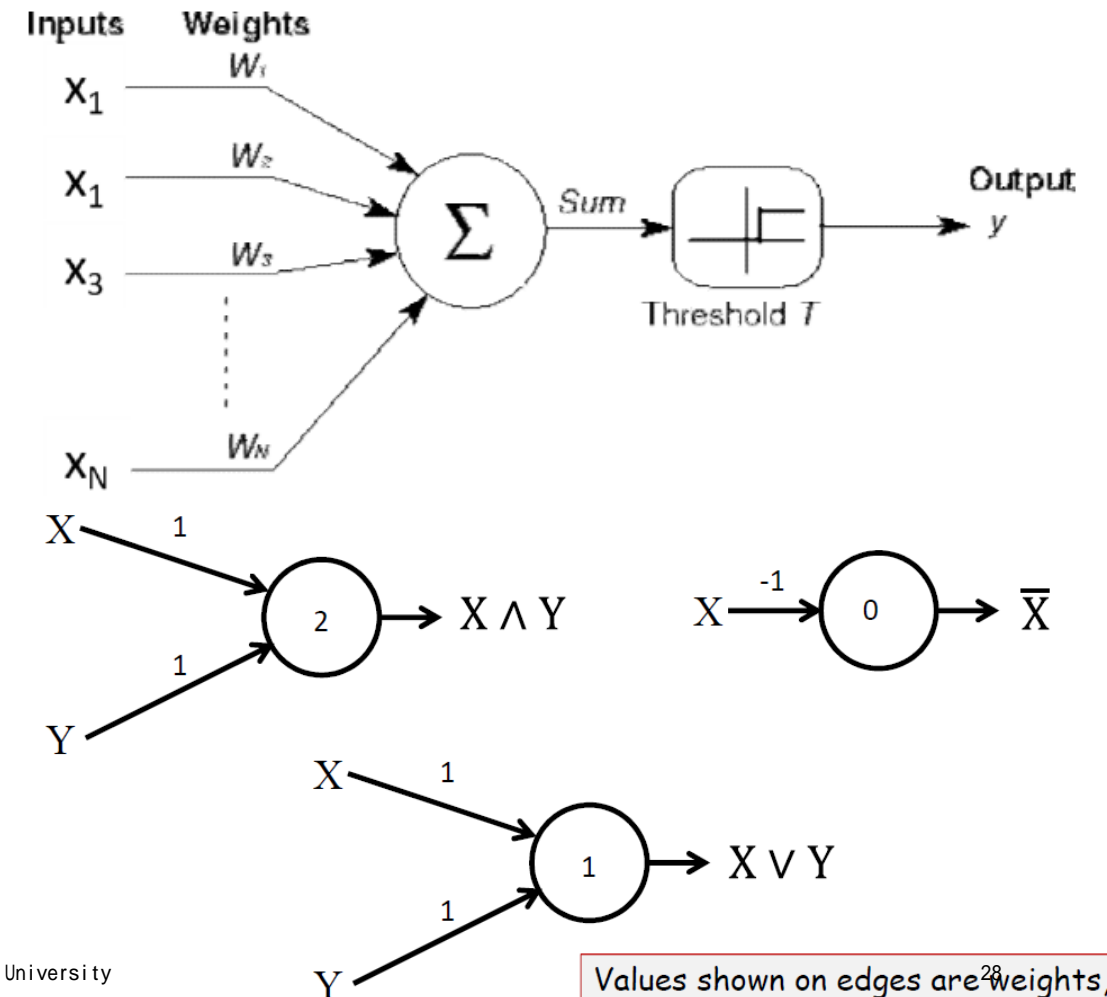
- Simplified Mathematical Formula

- $y = \begin{cases} 1 & \text{if } \sum_i w_i x_i - T > 0 \\ 0 & \text{else} \end{cases}$

- $w$  weights;

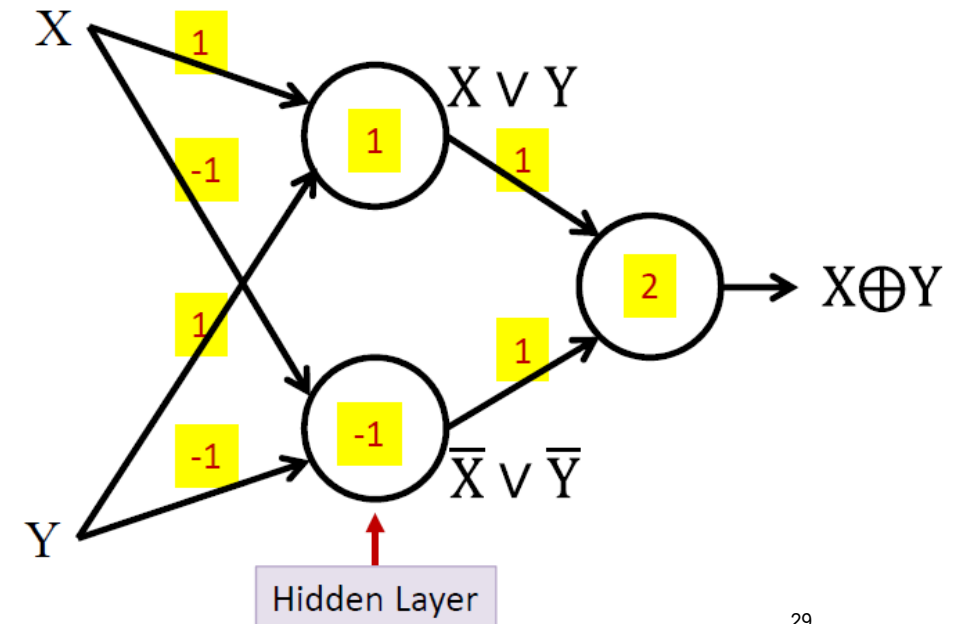
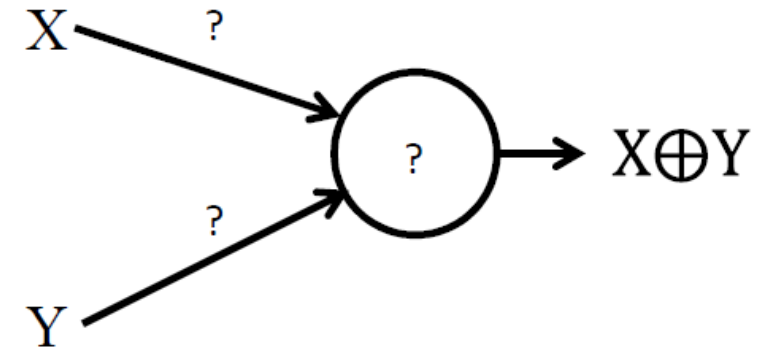
- Learning rule

- $w = w + \eta(d(x) - y(x))x$
  - Update the weights whenever the perceptron output is wrong
  - Convergence on linearly-separable case
  - It can mimic any Boolean gates (AND, OR, NOT)



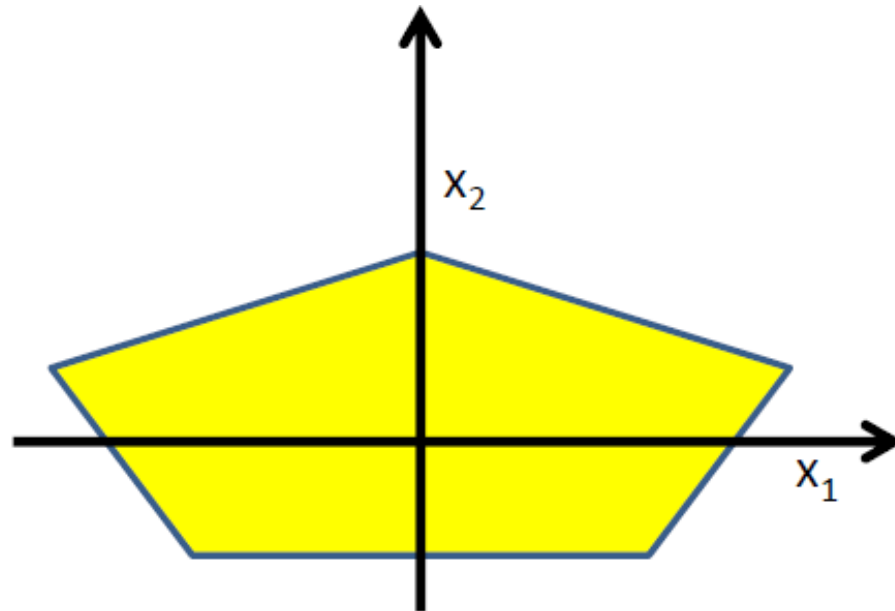
# The Perceptron

- The XOR problem
  - The book, *Perceptrons*, at 1969 by Marvin Minsky
- Multi-layer Perceptron!
  - Also suggested by Marvin Minsky at 1969
- A Multi-layer perceptron can compose arbitrarily complicated Boolean functions!



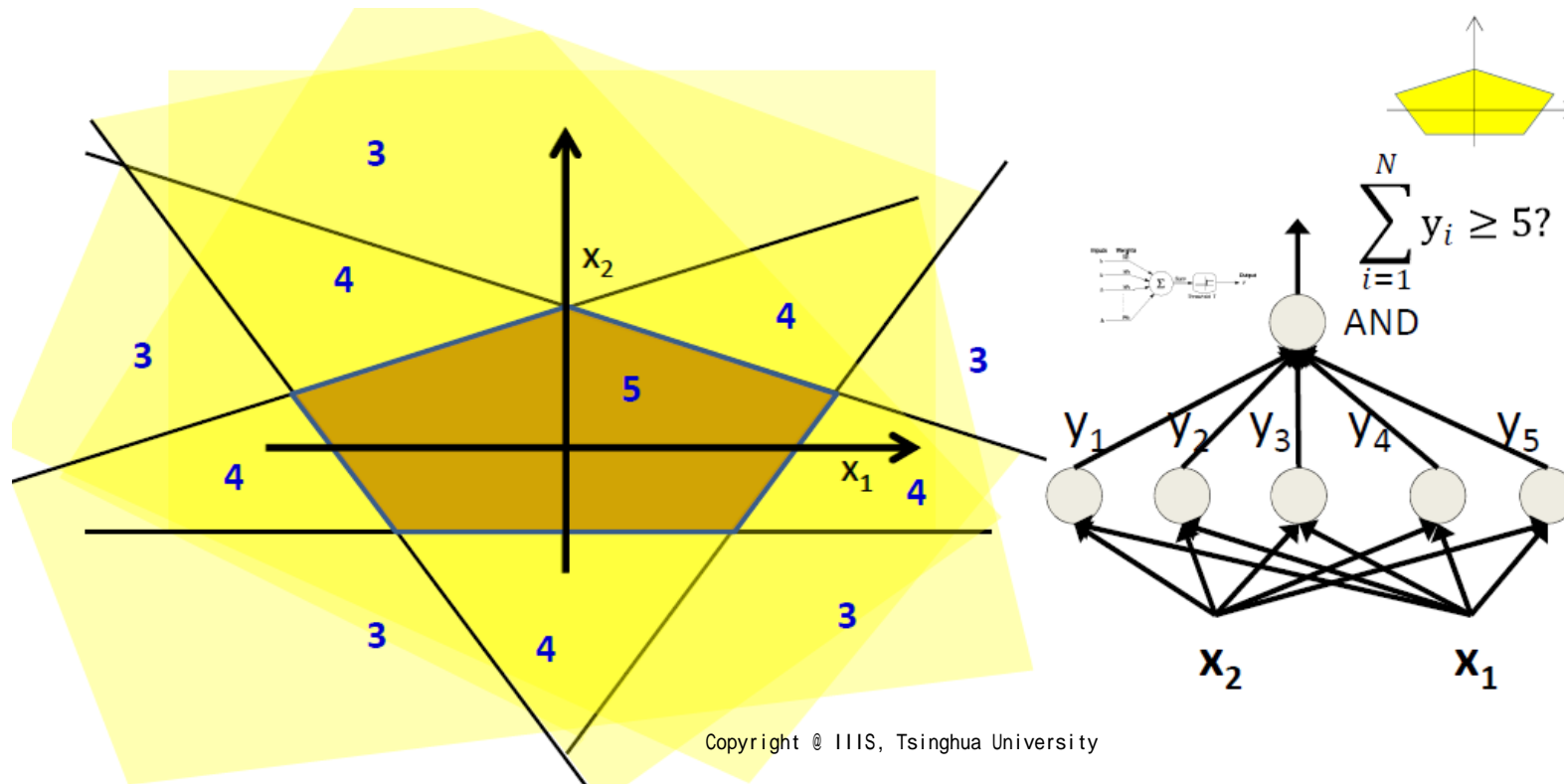
# The Perceptron

- Perceptron on real inputs
  - Construct a perceptron for the following decision boundary



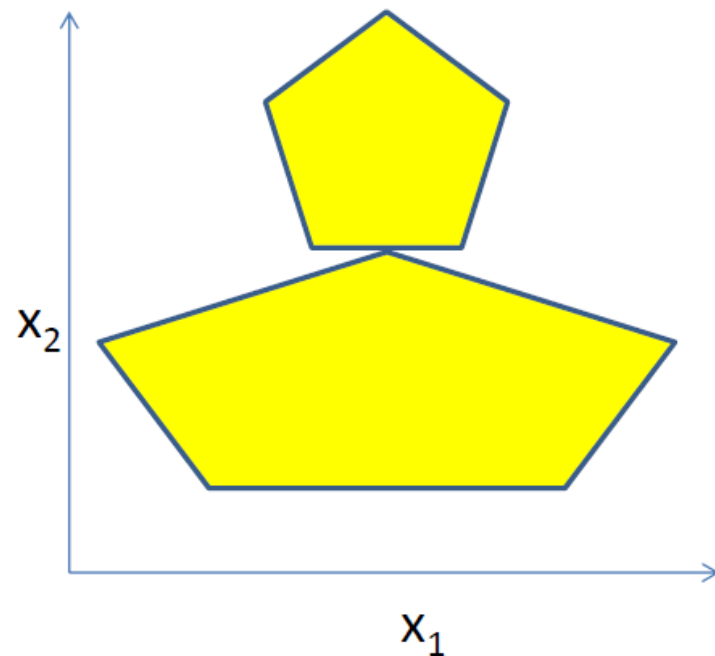
# The Perceptron

- Perceptron on real inputs
  - Construct a perceptron for the following decision boundary

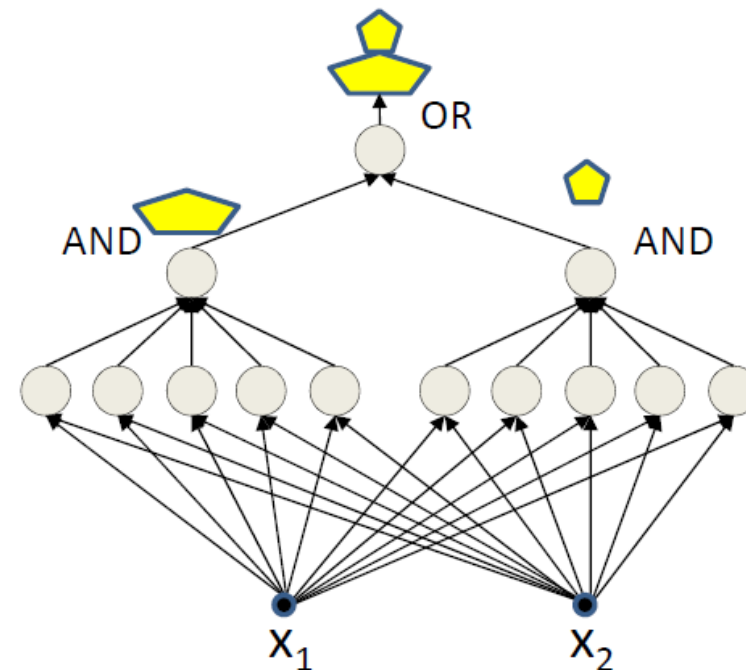


# The Perceptron

- Perceptron on real inputs
  - A more complex case



MLPs can represent arbitrarily complex decision (classification) boundaries





# The Perceptron

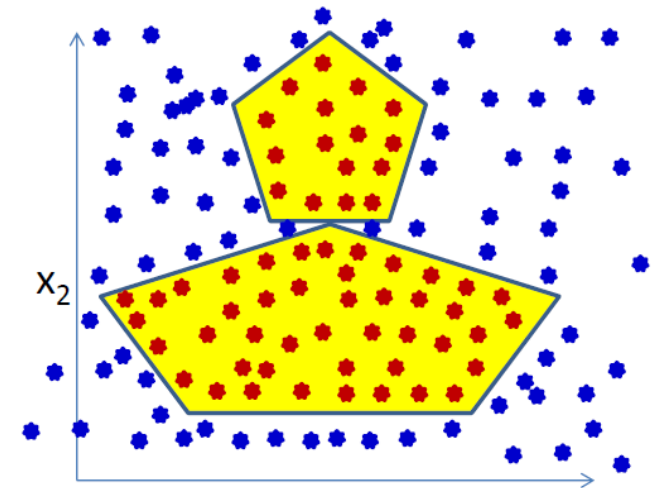
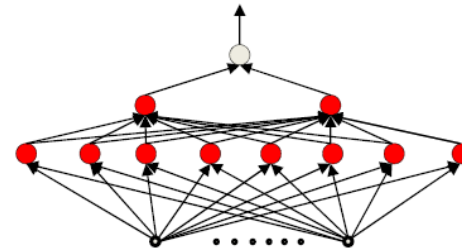
- How many layers are necessary for Boolean functions?
  - *Two-layered MLPs are universal Boolean functions*
- How many perceptrons in hidden layers are required for a  $N$ -variable function?
  - The XOR function over  $N$  variables
  - $O(2^{N-1})$  perceptrons
- Lower-bound for the parity function (Håstad, 1987)
  - depth- $k$  circuits require size  $\exp\left(\Omega\left(N^{\frac{1}{k-1}}\right)\right)$
- A *deep* network is preferred!

YZ \ WX	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

$$O = W \oplus X \oplus Y \oplus Z$$

# The Perceptron

- How to learn the weights of a **multi-layer** perceptron?
  - An intuitive example
  - Low-level: linear boundaries
  - Hidden-level: convex regions
  - Exponential search?



- **Training a 3-node neural network is NP-complete**
  - *Avrim L. Blum & Ronald L. Rivest, Neural Networks, 1992*
  - Assumption: a 2-layered network with  $n$ -variable input and thresholding functions

# The First AI Winter

- The perceptron cannot represent the XOR problem
  - Marvin Minsky, 1969
- The notion of NP-completeness
  - Levin-Cook Theorem, 1971
  - Karp's 21 NP-complete problems, 1973
- Sad Fact: Frank Rosenblatt died in July 1971 on his 43rd birthday 😞

# The First AI Winter

- The Lighthill report, 1973
  - *Artificial Intelligence: A General Survey* by James Lighthill
  - Evaluation of the state of AI research in the United Kingdom

## 3 Past disappointments

Most workers in AI research and in related fields confess to a pronounced feeling of disappointment in what has been achieved in the past twenty-five years. Workers entered the field around 1950, and even around 1960, with high hopes that are very far from having been realised in 1972. In no part of the field have the discoveries made so far produced the major impact that was then promised.

The disappointment felt may be analysed into two kinds: work in the categories A and C of [section 2](#) has some respectable achievements to its credit (and achievement in such categories of work with rather clear aims is clearly discernible), but to a disappointingly smaller extent than had been hoped and expected, while progress in category B has been even slower and more discouraging, tending (as explained in [section 2](#)) to sap confidence in whether the field of research called AI has any true coherence. In the meantime, claims and predictions regarding the potential results of AI research had been publicised which went even farther than the expectations of the majority of workers in the field, whose embarrassments have been added to by the lamentable failure of such inflated predictions.

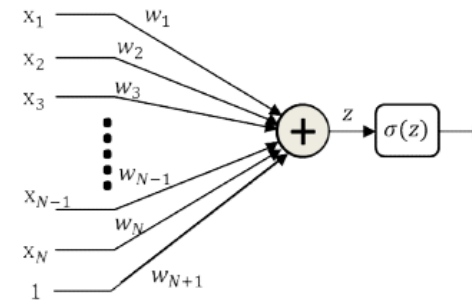
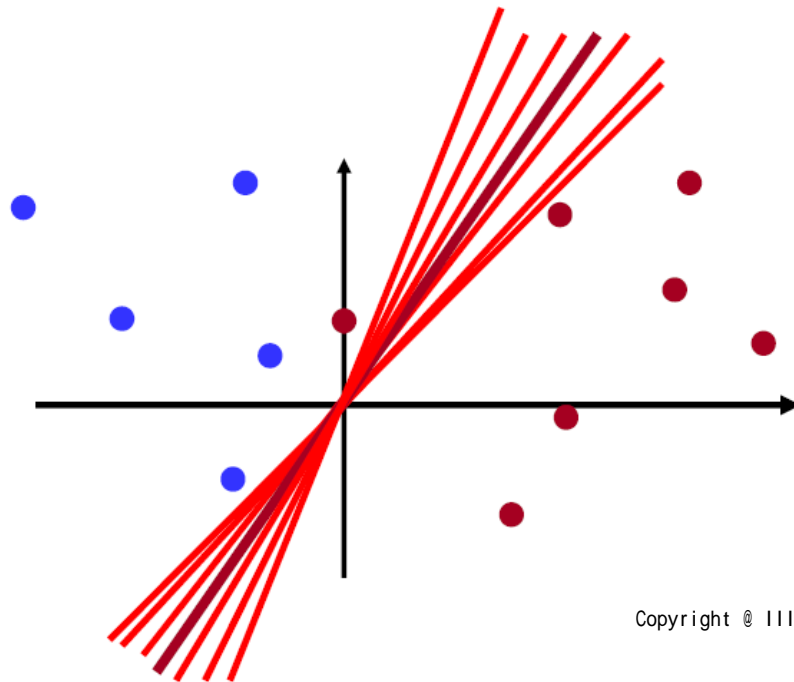
These general statements are expanded in a little more detail in the rest of [section 3](#), which has been influenced by the views of large numbers of people listed in [section 1](#) but which like the whole of this report represents in the last analysis only the personal view of the author. Before going into such detail he is inclined, as a mathematician, to single out one rather general cause for the disappointments that have been experienced: failure to recognise the implications of the **combinatorial explosion**. This is a general obstacle to the construction of a self-organising system on a large knowledge base which results from the explosive growth of any combinatorial expression, representing numbers of possible ways of grouping elements of the knowledge base according to particular rules, as the base's size increases.

# The First AI Winter

- The Lighthill report, 1973
    - *Artificial Intelligence: A General Survey* by James Lighthill
    - Evaluation of the state of AI research in the United Kingdom
  - DARPA Funding Cuts, 1974
    - DARPA's own study (the American Study Group) suggested that most AI research was unlikely to produce anything truly useful in the foreseeable future
  - A decade-long AI winter until the success of expert system in 1980s
    - And then followed by the second (longer) AI winter in 1990s ☹️
- Summary:**
- **MLP is Universal Function Approximator**
  - **But hopeless to find the *optimal* solution**

# Differentiable Functions

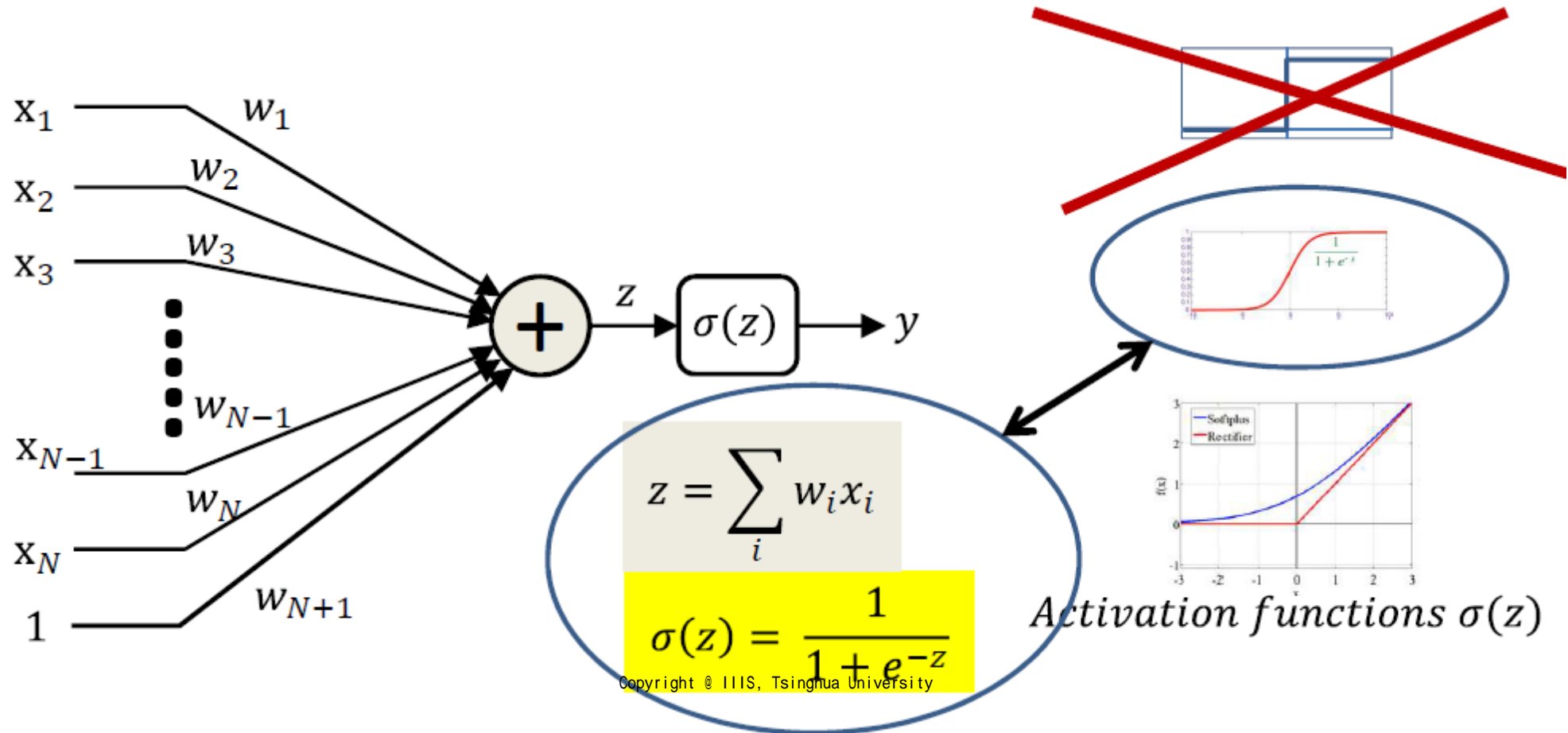
- Why is training so hard in classical multi-layer perceptrons?
  - Perceptron training: adjust weights when error
  - You can alter weights a lot without changing the error
  - No indication of improvement  $\rightarrow$  combinatorial optimization



$$\sigma(z) = \begin{array}{|c|c|} \hline & \\ \hline \end{array}$$

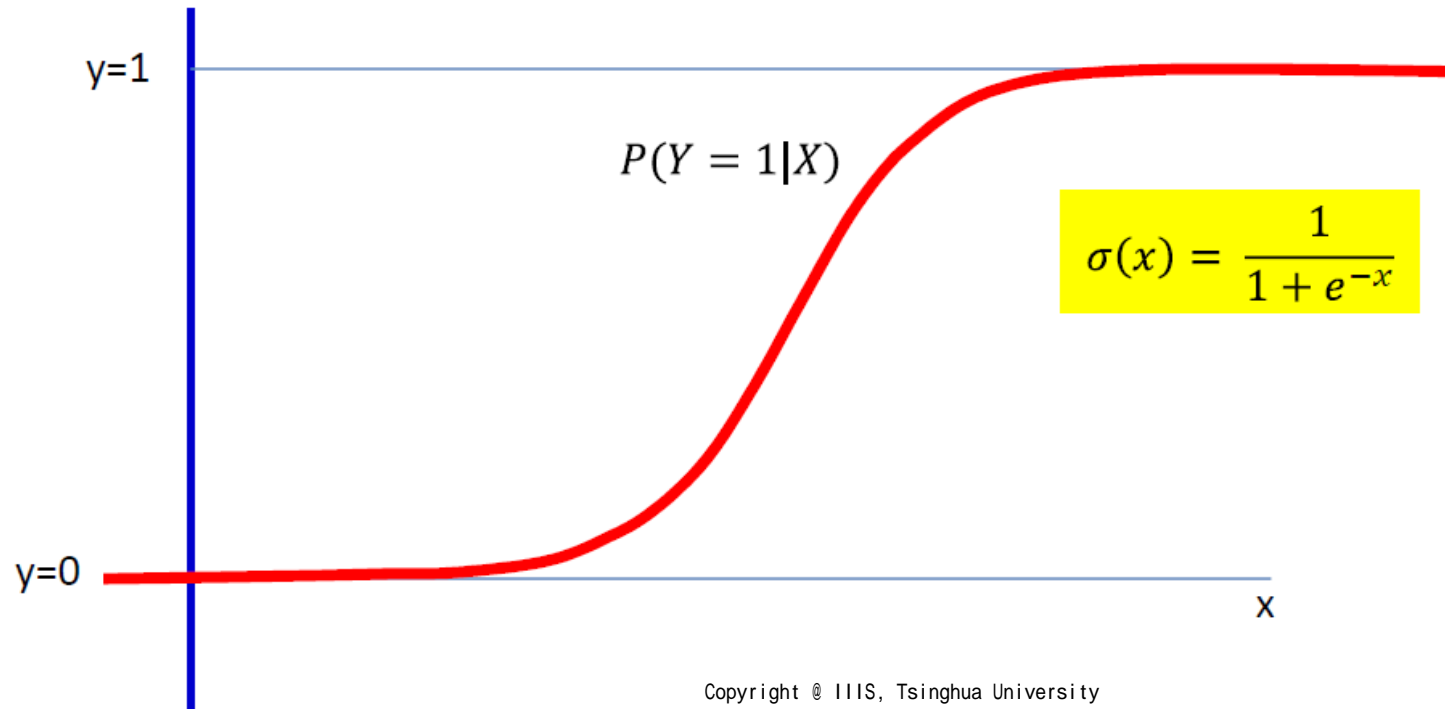
# Differentiable Functions

- The differentiable approximation of thresholding



# Differentiable Functions

- A probability interpretation of sigmoid function
  - Differentiable function  $\rightarrow$  we now have gradient!
  - The direction of improvement





# Differentiable Functions

- The *Backpropagation* Algorithm
  - *Learning representations by back-propagation errors, Nature, 1986*
  - *By David Rumelhart, Geoffrey Hinton, Ronald Williams*
  - Key idea: update weights by back-propagating gradients

# Learning representations by back-propagating errors

David E. Rumelhart\*, Geoffrey E. Hinton†  
& Ronald J. Williams\*

\* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA

† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

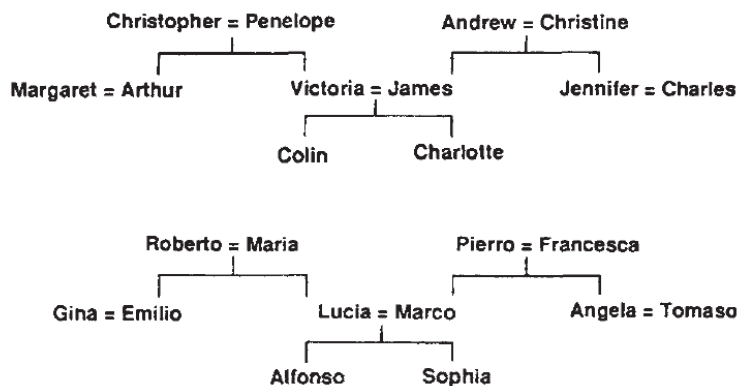
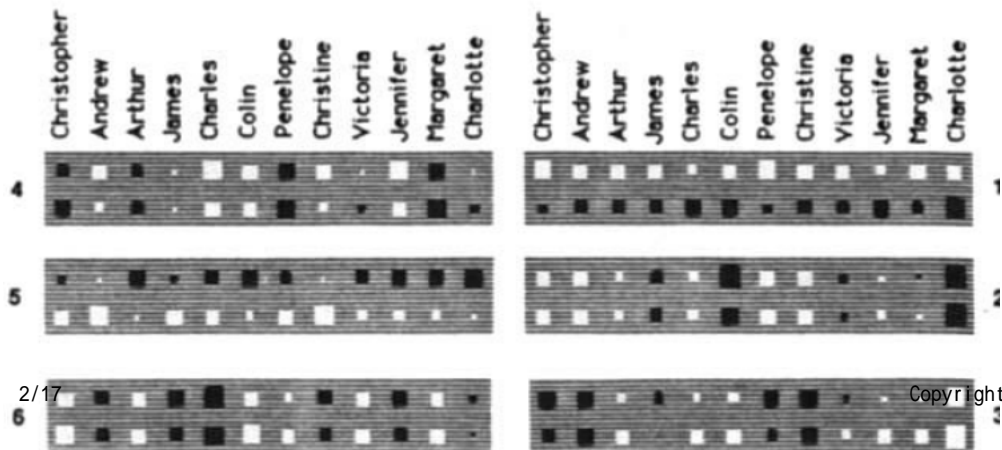


Fig. 2 Two isomorphic family trees. The information can be



**Fig. 4** The weights from the 24 input units that represent people to the 6 units in the second layer that learn distributed representations of people. White rectangles, excitatory weights; black rectangles, inhibitory weights; area of the rectangle encodes the magnitude of the weight. The weights from the 12 English people are in the top row of each unit. Unit 1 is primarily concerned with the distinction between English and Italian and most of the other units ignore this distinction. This means that the representation of an English person is very similar to the representation of their Italian equivalent. The network is making use of the isomorphism between the two family trees to allow it to share structure and it will therefore tend to generalize sensibly from one tree to the other. Unit 2 encodes which generation a person belongs to, and unit 6 encodes which branch of the family they come from. The features captured by the hidden units are not at all explicit in the input and output encodings, since these use a separate unit for each person. Because the hidden features capture the underlying structure of the task domain, the network generalizes correctly to the four triples on which it was not trained. We trained the network for 1500 sweeps, using  $\epsilon = 0.005$  and  $\alpha = 0.5$  for the first 20 sweeps and  $\epsilon = 0.01$  and  $\alpha = 0.9$  for the remaining sweeps. To make it easier to interpret the weights we introduced 'weight-decay' by decrementing every weight by 0.2% after each weight change. After prolonged learning, the decay was balanced by  $\partial E / \partial w$ , so the final magnitude of each weight indicates its usefulness in reducing the error. To prevent the network needing large weights to drive the outputs to 1 or 0, the error was considered to be zero if output units that should be on had activities above 0.8 and output units that should be off had activities below 0.2.

# Differentiable Functions

- The *Backpropagation* Algorithm
  - *Learning representations by back-propagation errors, Nature, 1986*
  - *By David Rumelhart, Geoffrey Hinton, Ronald Williams*
  - Key idea: update weights by back-propagating gradients
- For the first time, it shows such a backward gradient computation method can automatically learn features
  - Although the idea can be traced back to Paul Werbos's dissertation at 1974
  - *Remark: Still **NO** guarantee for optimal solution*
  - ***It is particularly important to make an idea work!***

# Revival of Connectionism

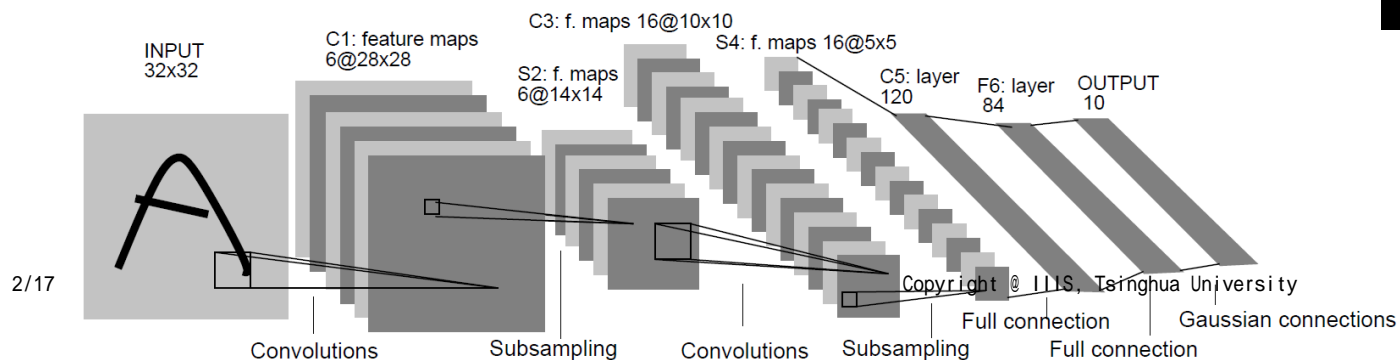
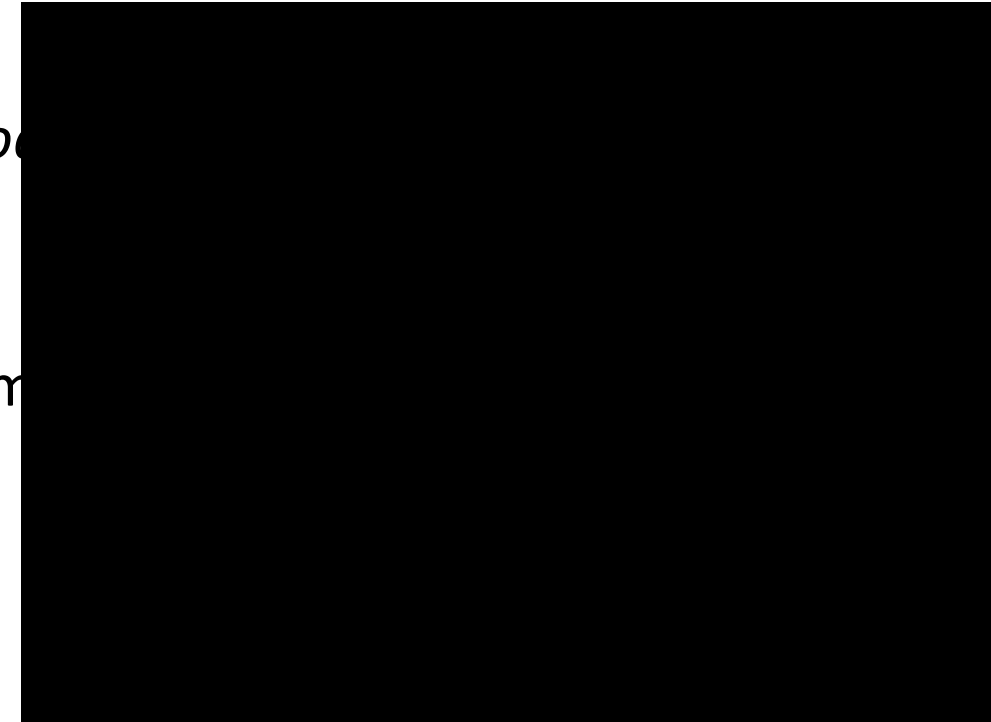
- Geoffrey E. Hinton
  - PhD in AI at 1978, U. Edinburgh (short of funding)
  - Worked at UCSD and CMU (backprop)
  - Professor Emeritus at U. Toronto
  - Inventor of Boltzmann machine, distributed representations, mixtures of experts
  - The beginning of DL era in 2012
  - Turing Award at 2018; Nobel Prize at 2024
- Comments
  - 1978 ~ 2012 → two AI winter and multiple waves
  - Many ideas has been invented for decades
  - The perspective of doing great science



- Fun story: Geoffrey **Everest** Hinton
- great-great-grandson of logician George Boole
  - middle name is from another relative, George Everest

# Revival of Connectionism (1990s)

- The convolutional neural network (CNN)
  - *Backpropagation applied to handwritten zip code*
  - *Neural Computation 1989*
  - *By Yann LeCun, et al., from AT&T Bell Lab*
  - Remark: idea can be traced back to K. Fukushima
- LeNet-5 at 1998
  - The first commercial application of CNN
  - The foundation of modern computer vision



## LeNet-1 at 1993

Remark: 32-year-old Yann LeCun  
Fun fact: A pos-doc of Hinton

**10-year effort to make CNN work!**

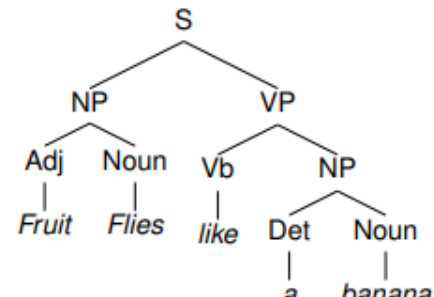
# Revival of Connectionism (2000s)

- Neural Networks for Natural Language Processing
  - *A neural probabilistic language model, NIPS 2000 (neural language model)*
  - *By Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent*
  - *Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015 (attention)*
  - *By Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio*

### a simple PCFG

- 1.0 S  $\rightarrow$  NP VP
- 0.3 NP  $\rightarrow$  Adj Noun
- 0.7 NP  $\rightarrow$  Det Noun
- 1.0 VP  $\rightarrow$  Vb NP
- 
- 0.2 Adj  $\rightarrow$  fruit
- 0.2 Noun  $\rightarrow$  flies
- 1.0 Vb  $\rightarrow$  like
- 1.0 Det  $\rightarrow$  a
- 0.4 Noun  $\rightarrow$  banana
- 0.4 Noun  $\rightarrow$  tomato
- 0.8 Adj  $\rightarrow$  angry

### Example



$$1 * 0.3 * 0.2 * 0.7 * 1.0 * 0.2 * 1 * 1 * 0.4 = 0.0033$$



# Revival of Connectionism (2000s)

- Neural Networks for Natural Language Processing
  - *A neural probabilistic language model, NIPS 2000* (neural language model)
  - *By Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent*
  - *Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015* (attention)
  - *By Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio*
- The foundation of deep NLP models
  - Prof. Yoshua Bengio, Turing Award 2018
  - 15 years of works, finally revolutionize the NLP field

# Revival of Connectionism: The New Spring

## • Speech Recognition, 2012

- Led by G. Hinton; Joint work with Google, MSR and IBM
- Outperforms existing methods with a huge gap (1.3% ~ 10%)

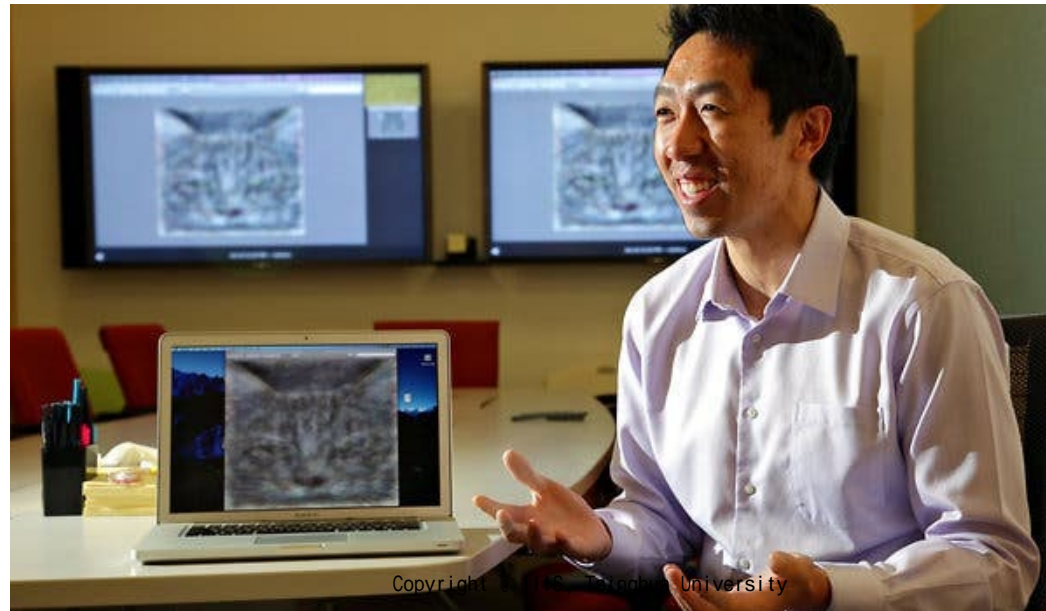
**[TABLE 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.**

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	



# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- **Google's Brain Recognizes Cat Videos, 2012**
  - Led by Andrew Ng and Jeff Dean, at Google X Lab
  - A neural network of 16,000 CPUs trained on YouTube videos



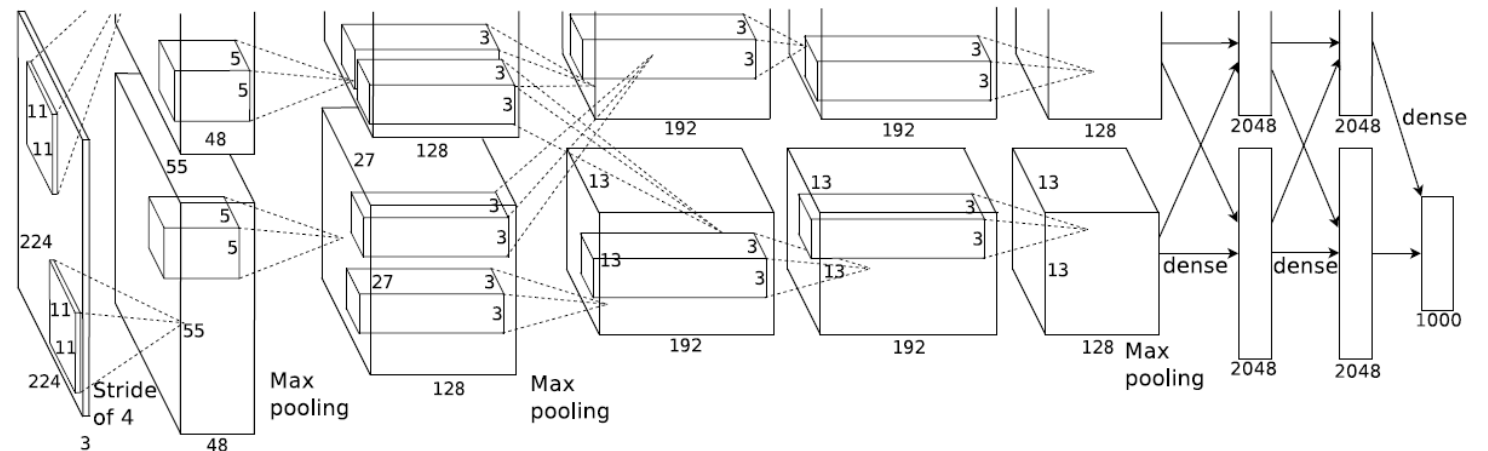
# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat Videos, 2012
- **AlexNet in ImageNet Challenge, 2012**
  - By Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton

**2012 is a magical year!**

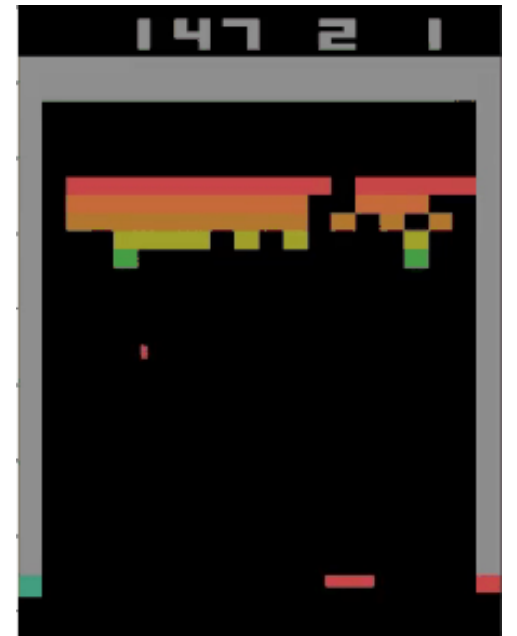
Model	Top-1	Top-5
<i>Sparse coding [2]</i>	47.1%	28.2%
<i>SIFT + FVs [24]</i>	45.7%	25.7%
<b>CNN</b>	<b>37.5%</b>	<b>17.0%</b>

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.



# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat Videos, 2012
- AlexNet in ImageNet Challenge, 2012
- **Deep Q-Learning, 2014**
  - By DeepMind
  - First time apply deep learning in reinforcement learning



# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat Videos, 2012
- AlexNet in ImageNet Challenge, 2012
- Deep Q-Learning, 2014
- **AlphaGo, 2016**
  - by DeepMind
  - Defeat 李世石 by 4:1, March 2016



# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat \
- AlexNet in ImageNet Challenge
- Deep Q-Learning, 2014
- AlphaGo, 2016
- **Google's Machine Translation System, 2016**
  - *It reduces translation errors by 60% compared to Google's phrase-based production system*
  - Only 2~3 years of development

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi  
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat Videos, 2012
- AlexNet in ImageNet Challenge, 2012
- Deep Q-Learning, 2014
- AlphaGo, 2016
- Google's Machine Translation System, 2016
- **OpenAI's RL Projects, 2019**
  - **OpenAI Five defeated OG in Dota II (2:0)**



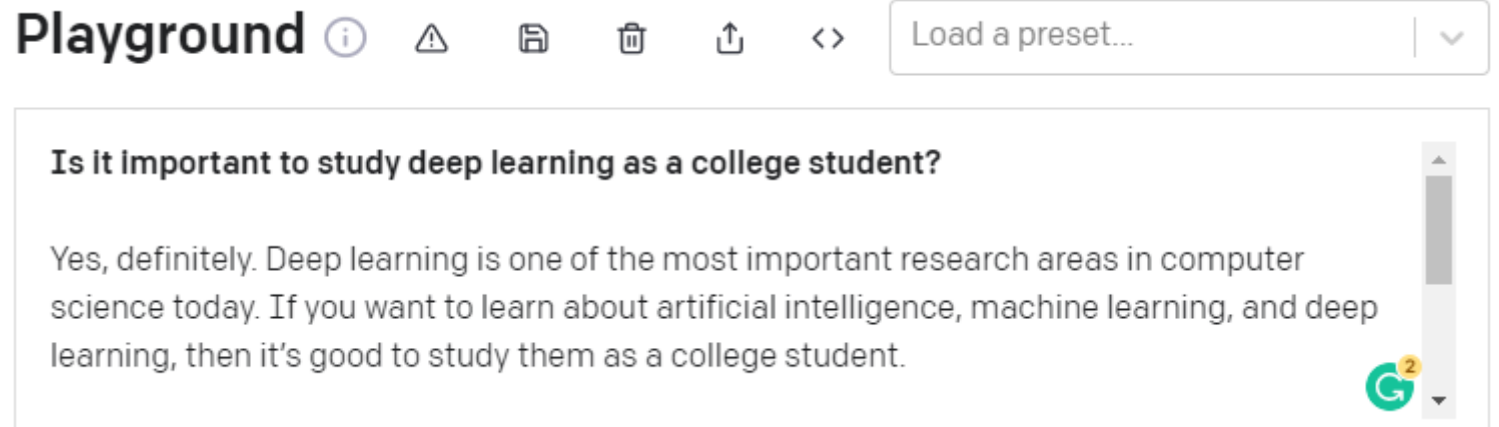
# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat Videos, 2012
- AlexNet in ImageNet Challenge, 2012
- Deep Q-Learning, 2014
- AlphaGo, 2016
- Google's Machine Translation System, 2016
- **OpenAI's RL Projects, 2019**
  - **OpenAI Five defeated OG in Dota II (2:0)**
  - **First time to use pure deep RL to solve Rubik's cube manipulation**



# Revival of Connectionism: The New Spring

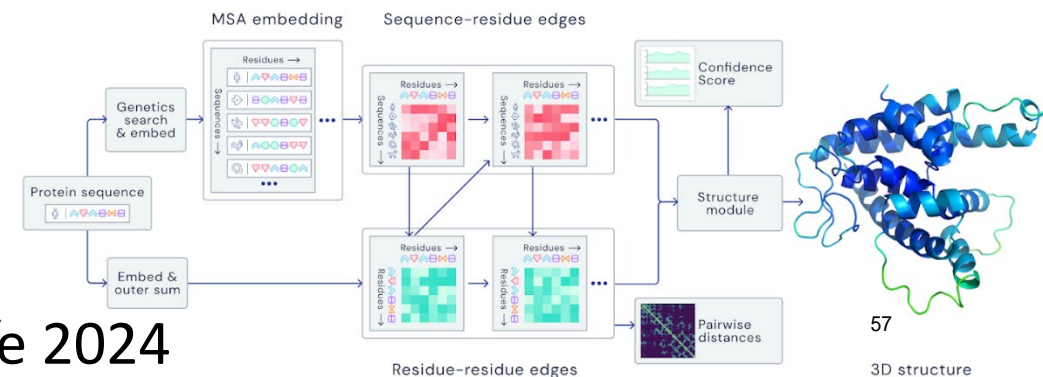
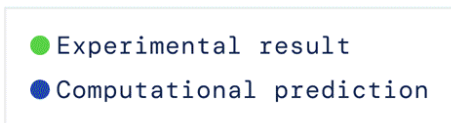
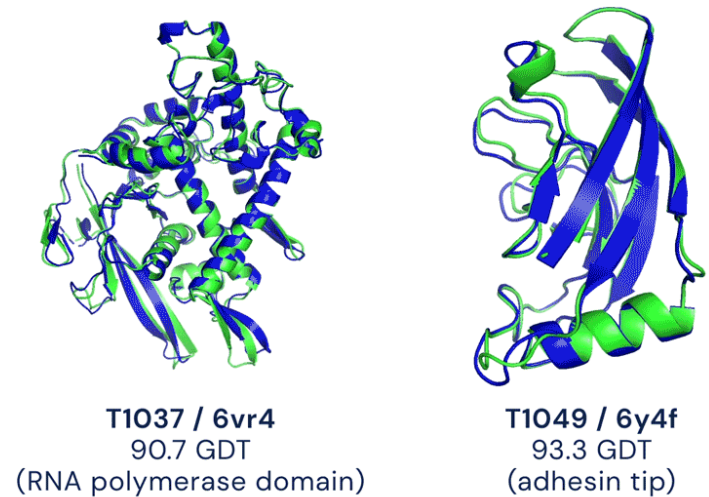
- Speech Recognition, 2012
- Google’s Brain Recognizes Cat Videos, 2012
- AlexNet in ImageNet Challenge, 2012
- Deep Q-Learning, 2014
- AlphaGo, 2016
- Google’s Machine Transla
- OpenAI’s RL Projects, 201
- **GPT-3, 2020**
  - First time to show big language models can be few-shot learner
  - The start of the era of “big pretrained models”





# Revival of Connectionism: The New Spring

- Speech Recognition, 2012
- Google's Brain Recognizes Cat Videos, 2012
- AlexNet in ImageNet Challenge, 2012
- Deep Q-Learning, 2014
- AlphaGo, 2016
- Google's Machine Translation System, 2016
- OpenAI's RL Projects, 2019
- GPT-3, 2020
- **AlphaFold, 2020**
- A new trend of "AI for science" & Nobel Prize 2024



# Revival

- Speech Recognition
- Google's Embeddings
- AlexNet in 2012
- Deep Q-Learning
- AlphaGo, 2016
- Google's Neural Machine Translation
- OpenAI's GPT-1
- GPT-3, 2020
- AlphaFold
- AlphaCode, 2021

1553\_H. XOR and Distance C++ pass Layer 18 play

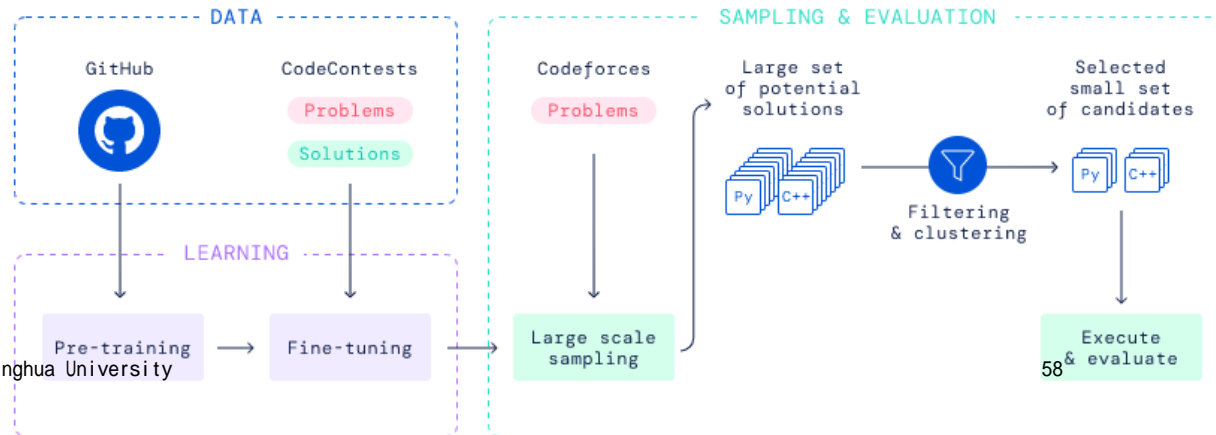
Head 1 Head 2 Head 3 Head 4 Head 5 Head 6 Head 7 Head 8 Head 9 Head 10 Head 11 all none

## Problem Description

```
// RATING: 1200
// TAGS: brute force,greedy
// LANGUAGE IS cpp
// CORRECT SOLUTION
// You are given an array a consisting of n distinct elements and an integer k.
// Each element in the array is a non-negative integer not exceeding 2^k-1.
// Let's define the XOR distance for a number x as the value of
// $$$f(x) = \min_{i = 1}^n \min_{j = i + 1}^n |(a_i \oplus x) - (a_j \oplus x)|$$$
// where  $\oplus$  denotes [the bitwise XOR operation](https://en.wikipedia.org/wiki/Bitwise_operation#XOR).
// For every integer x from 0 to 2^k-1, you have to calculate f(x).
// Input
// The first line contains two integers n and k (1 ≤ k ≤ 19; 2 ≤ n ≤ 2^k).
// The second line contains n integers a_1, a_2, ..., a_n (0 ≤ a_i ≤ 2^k-1). All
// these integers are distinct.
// Output
// Print 2^k integers. The i-th of them should be equal to f(i-1).
// Examples
// Input
// 3 3
// 6 0 3
// Output
// 3 1 1 2 2 1 1 3
```

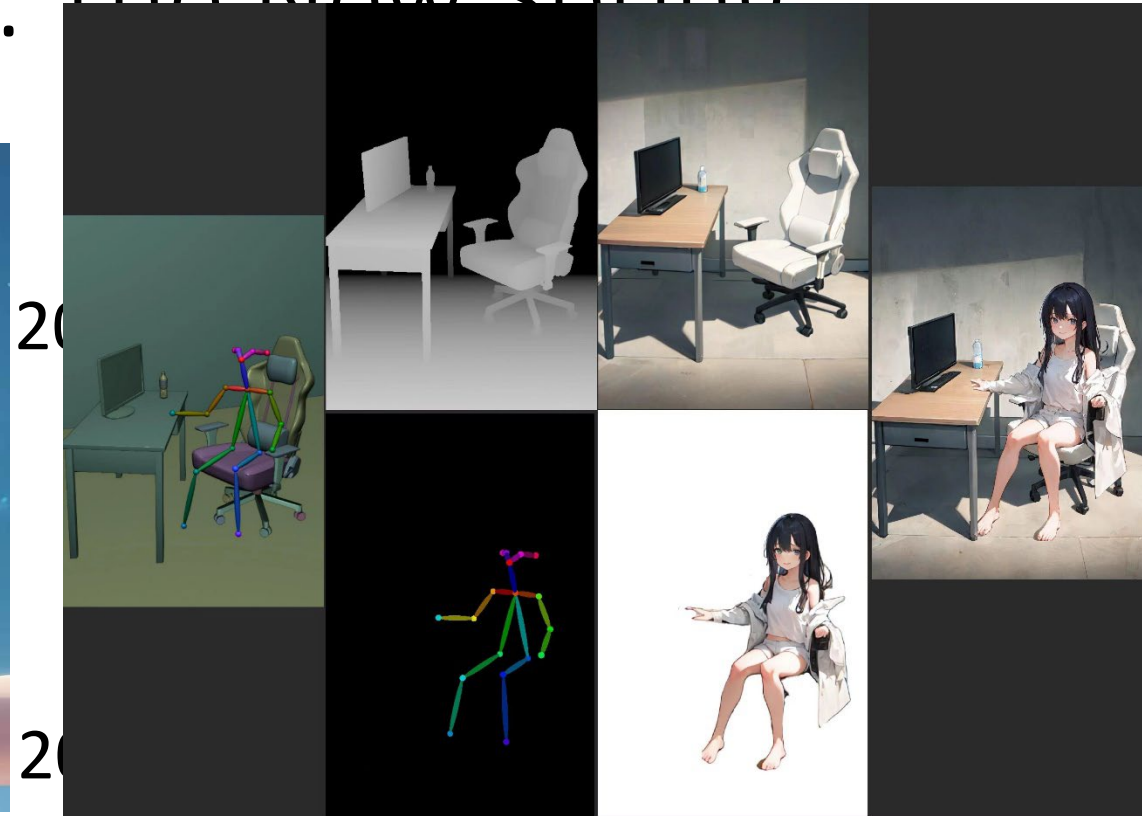
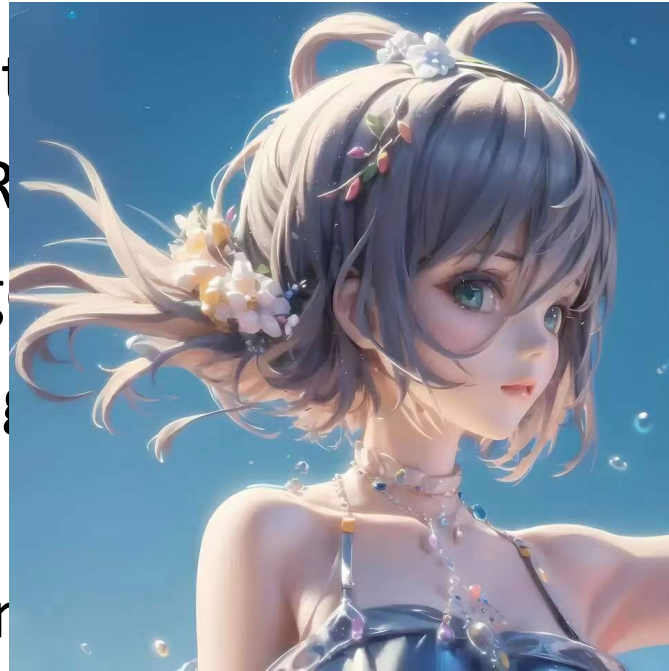
## Solution

```
#include <bits/stdc++.h>
using namespace std;
int n, k;
int a[1 << 19];
int main() {
    cin >> n >> k;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    for (int x = 0; x < (1 << k); x++) {
        int ans = (1 << 30);
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                int t = a[i] ^ x;
                int t2 = a[j] ^ x;
                ans = min(ans, abs(t - t2));
            }
        }
        cout << ans << " ";
    }
    return 0;
}
```



# Revival of Connectionism: The New Spring

- Speech Recognition
- Google's Brain Research
- AlexNet in Image Classification
- Deep Q-Learning
- AlphaGo, 2016
- Google's Machine Learning
- OpenAI's RL Projects, 2019
- GPT-3, 2020
- AlphaFold, 2020
- AlphaCode, 2021



• **AIGC, 2022**

# Revival of Connectionism: The New Spring

User help me write a short note to introduce myself to my neighbor

ChatGPT Hi there,

My name is [Your Name] and I just moved in next door. I wanted to reach out and introduce myself. I'm looking forward to getting to know you and the neighborhood. If you ever need anything or just want to chat, please don't hesitate to stop by or give me a call.

Best,  
[Your Name]

User can you make it more formal?

ChatGPT Dear Neighbor,

I am writing to introduce myself. My name is [Your Name] and I have recently moved in next door. I wanted to take a moment to reach out and say hello. I am looking forward to getting to know you and the neighborhood.

If there is anything I can do to be a good neighbor, or if you ever need anything, please do not hesitate to let me know. I am happy to help in any way that I can.

Sincerely,  
[Your Name]

User can you send it for me?

ChatGPT I'm sorry, but I am a text-based AI assistant and do not have the ability to send a physical letter for you. I can only provide suggestions and advice on what to write. It is up to you to take that advice and send the letter yourself.

User this code is not working like i expect — how do i fix it?

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
    defer cancel()
    resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
    return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```

ChatGPT It's difficult to say what's wrong with the code without more context. Can you provide more information about what the code is supposed to do and what isn't working as expected? Also, is this the entire code or just a part of it?

## • ChatGPT, 2022



Is it important to study deep learning as a college student?

< 2/2 >

Yes, studying deep learning as a college student can be very important, especially if you're interested in fields like artificial intelligence, computer vision, natural language processing, robotics, and even interdisciplinary applications in science, healthcare, and finance. Here are a few reasons why it's beneficial:

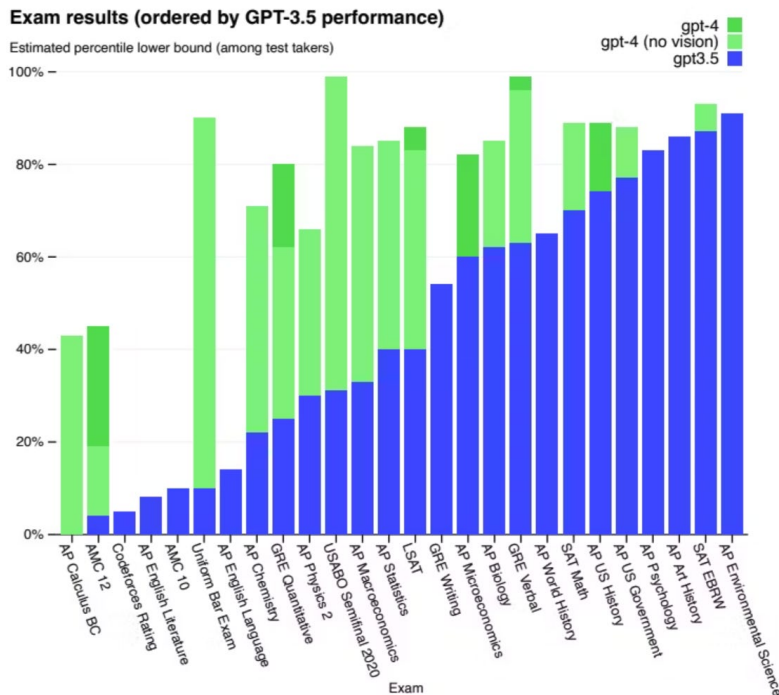
## 1. High Demand in the Job Market

Deep learning is widely used in industries such as tech, healthcare, finance, and robotics. Companies like Google, Tesla, and OpenAI are actively hiring people with deep learning expertise.

## 2. Research Opportunities

# Revival of Connectionism: The New Spring

## Speech Recognition



- GPT-3, 2020
- AlphaFold, 2020
- AlphaCode, 2021

2/17

identify and count the number of humans in the image

There are approximately **137** humans in the image.

(Note: This is a rough estimation based on visible individuals in the provided image. Individuals may be obscured or partially visible, potentially affecting the accuracy count.)

- **GPT4-V, 2023**

Copyright © IIIS, Tsinghua University

identify all the food items in the image.

Certainly!

Here are the food items identified in the image:

- Orange juice
- Avocado (halved)
- Kiwi (sliced)
- Blackberries
- Banana (sliced)
- Chia seeds
- Strawberries (sliced)
- Whole wheat bread (slices)
- Grapes
- Orange

62

Re

- Sp
- G
- A
- D
- A
- G
- O
- G



Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She walks confidently and casually. The street is damp and reflective, creating a mirror effect of the

• AlphaFold, 2020

• GPT4-V, 2023

• AlphaCode, 2021

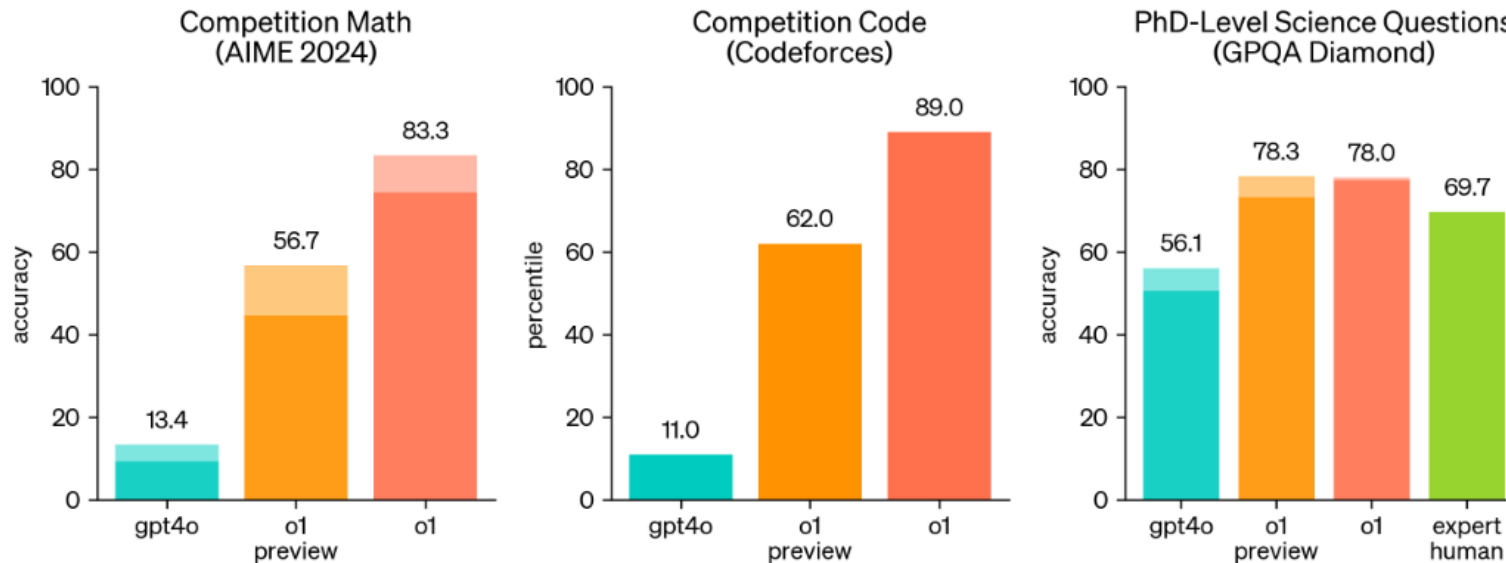
• Sora, 2024

oyfjdnisdr rtqwainr acxz myn:  
> Think step by step

Use the example above to dec

oyekaijzdf aaptcg suaokybha

aqht myznvaatzacdfoulxxz



o1 greatly improves over GPT-4o on challenging reasoning benchmarks. Solid bars show pass@1 accuracy and the shaded region shows the performance of majority vote (consensus) with 64 samples.



Thought for 5 seconds ▾

To decode the given ciphertext using the example provided, we'll use the pattern identified in example.



ystem, 2010

AGC, 2022

ChatGPT, 2022

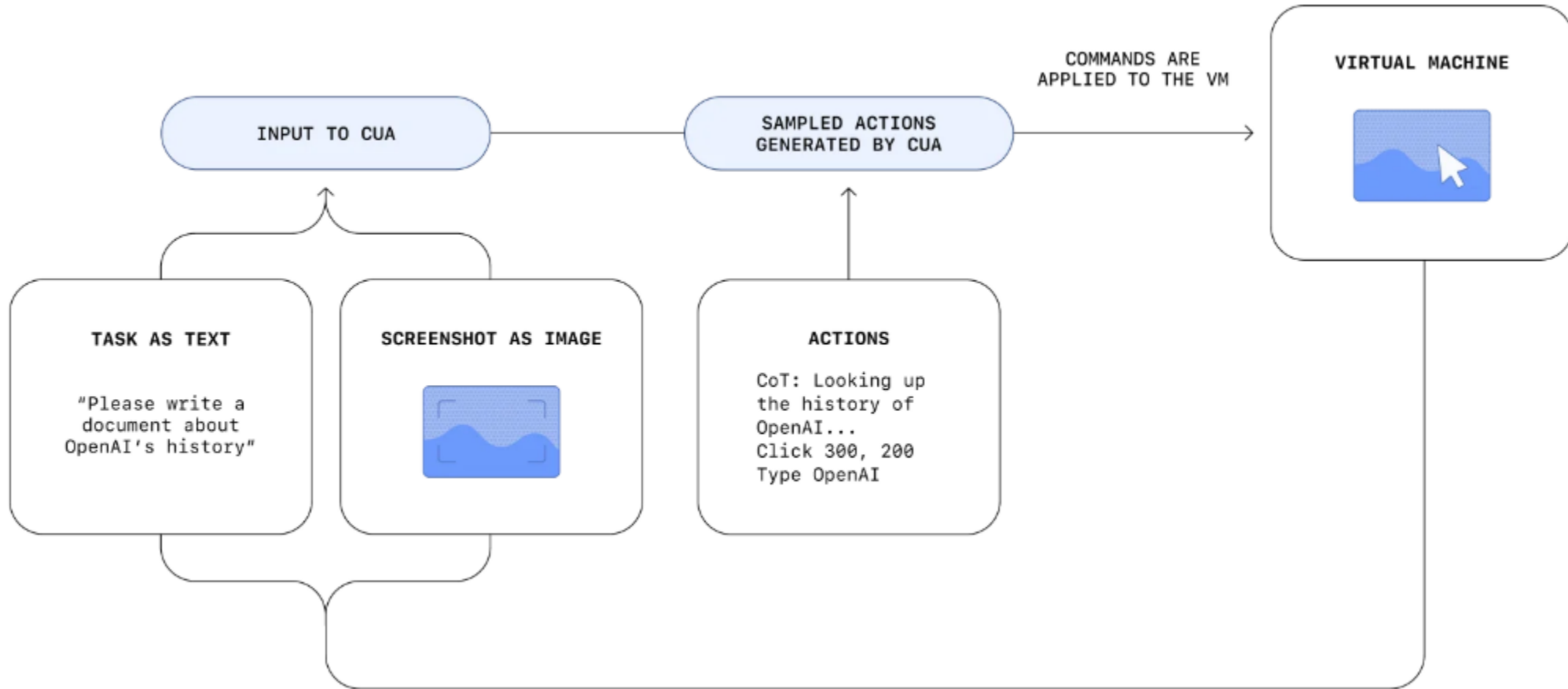
GPT4-V, 2023

Gemini, 2024

• **O1, 2024**

- AI can think
- The beginning of **Reasoning Model**
- O3 released later





Operator OpenAI

Starting Soon

24

- AlphaFold, 2020
- AlphaCode, 2021

- GPT4-V, 2023
- Sora, 2024

- *The beginning of general autonomous agent*

# Revival of Connectionism: The New Spring

- Speech Recognition, 2012



2/17

- AlphaCode, 2021

Copyright © IIIS, Tsinghua University

- Sora, 2024

- O1, 2024
- OpenAI Operator, 2024
- **DeepSeek R1, 2024**
  - *First open-source LLM on par with OpenAI*

## 晚上吃啥？回答不许超过三个字！



Thought for 32 seconds ^

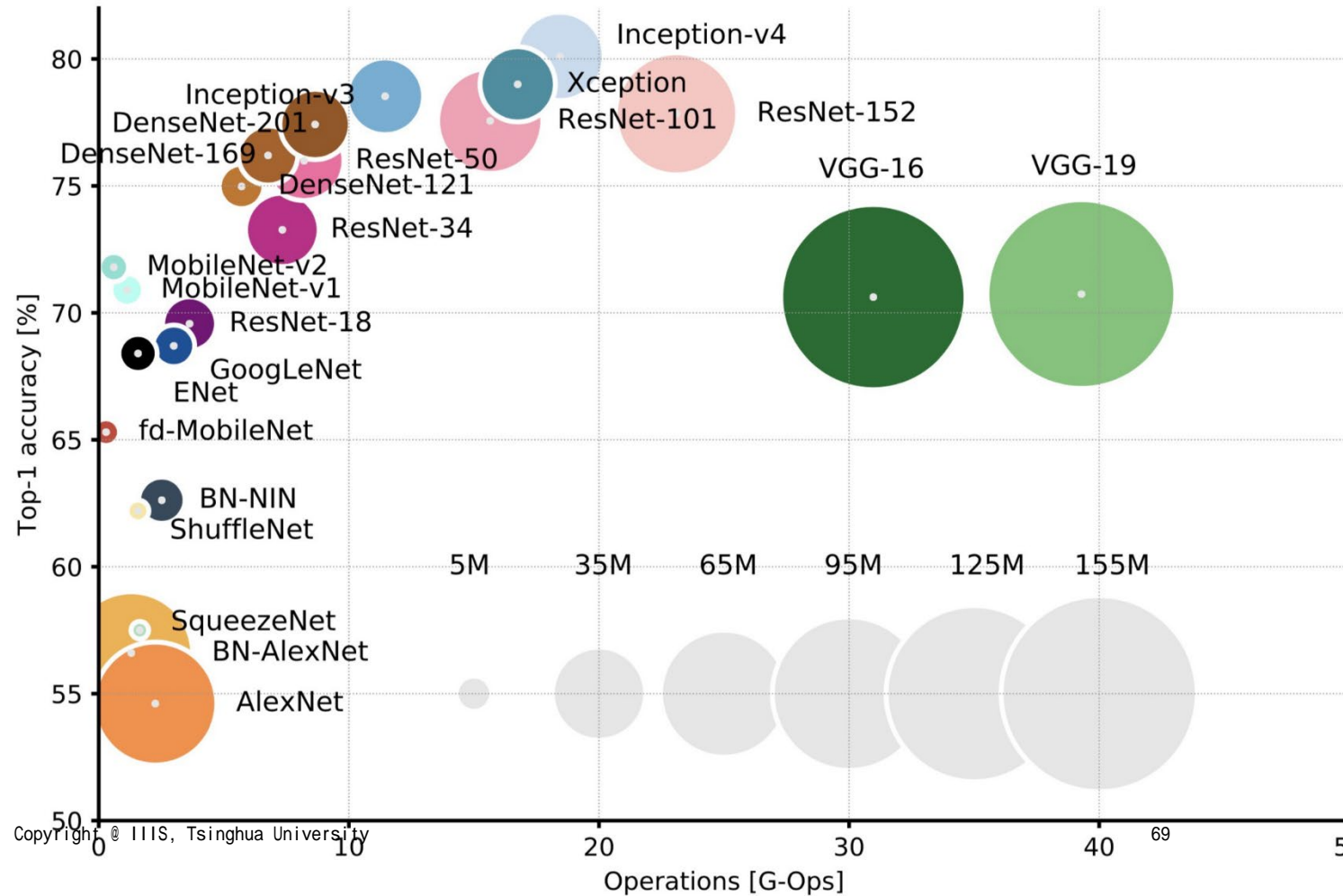
嗯，用户问“晚上吃啥？”，而且要求回答不能超过三个字。首先，我需要确定用户的意图。他们可能是在寻找简单快速的晚餐建议，或者只是随便问问，想要一个简短的回答。因为回答要简短，必须直接给出选项，不需要解释。

●●●●●

所以可能的回答包括：“饺子”、“拉面”、“炒饭”、“盖饭”、“麻辣烫”、“三明治”、“烤鱼”、“牛肉面”、“炸酱面”、“酸辣粉”等等。需要确保每个选项都是三个字，并且常见易懂。比如“麻辣烫”比较流行，可能是个好选择。或者“酸菜鱼”也是三个字。需要选最常用的几个，然后挑一个作为回答。或者随机选一个，但用户可能需要多样的答案，不过这里只回复一个。所以最终决定选哪个？比如“饺子”比较普遍，或者“麻辣烫”更有个性。根据用户可能的喜好，如果回答简洁且三个字，选常见的比较好。例如“牛肉面”或者“炒饭”可能更稳妥。或者“火锅”虽然两字，但可能用户允许？但用户明确要求三个字，必须严格遵守。所以最终可能选“麻辣烫”或者“酸菜鱼”之类的。或者“烤肉”不行，三个字的话“烤羊肉”可能太长。或者“寿司”不行。或者“三明治”三个字。综合考虑，可能“麻辣烫”是比较合适的三个字回答，或者“炒饭”更简单。但用户可能需要更多选项，不过只能回答一个。因此，最终回答可能是“炒饭”、“拉面”或“麻辣烫”中的一个。

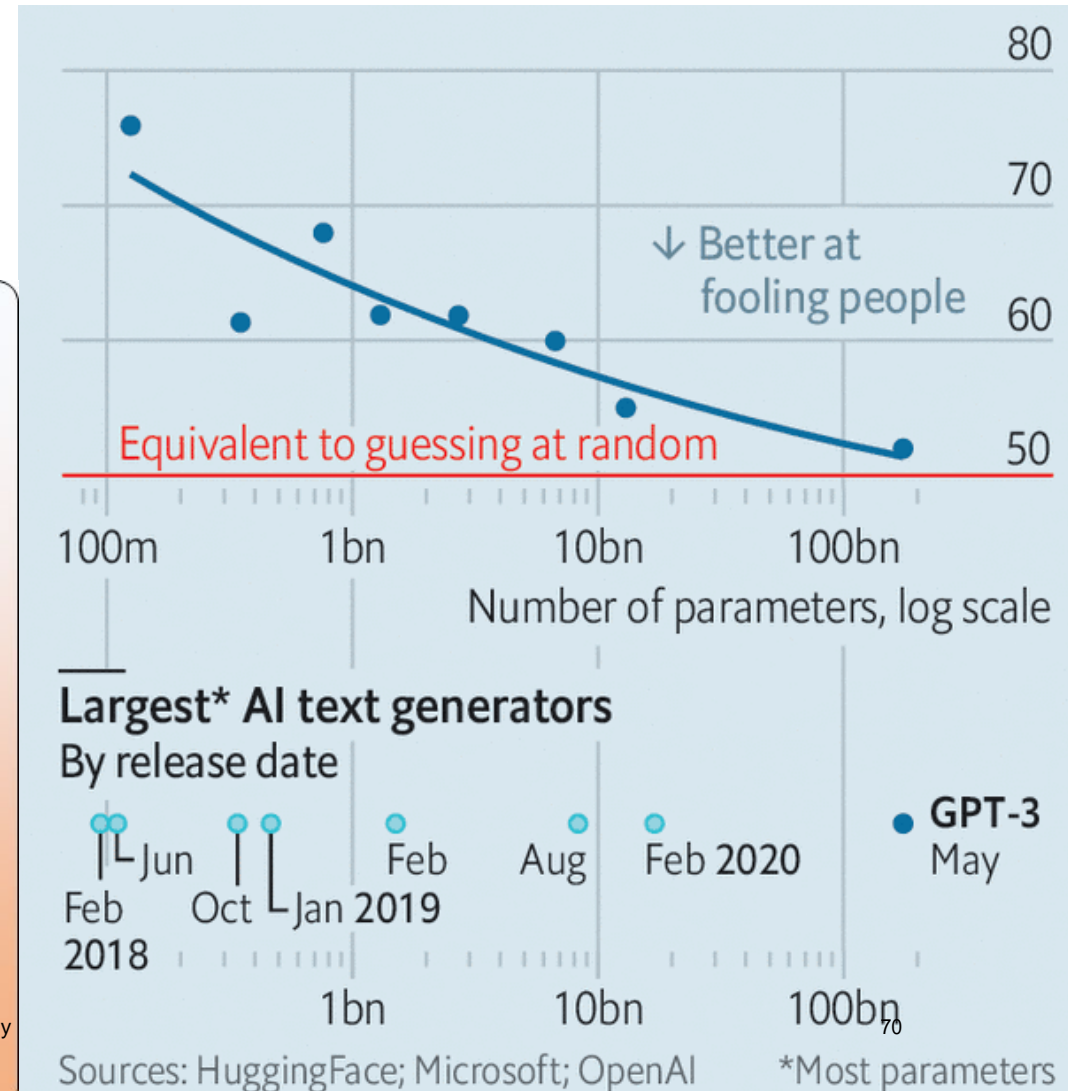
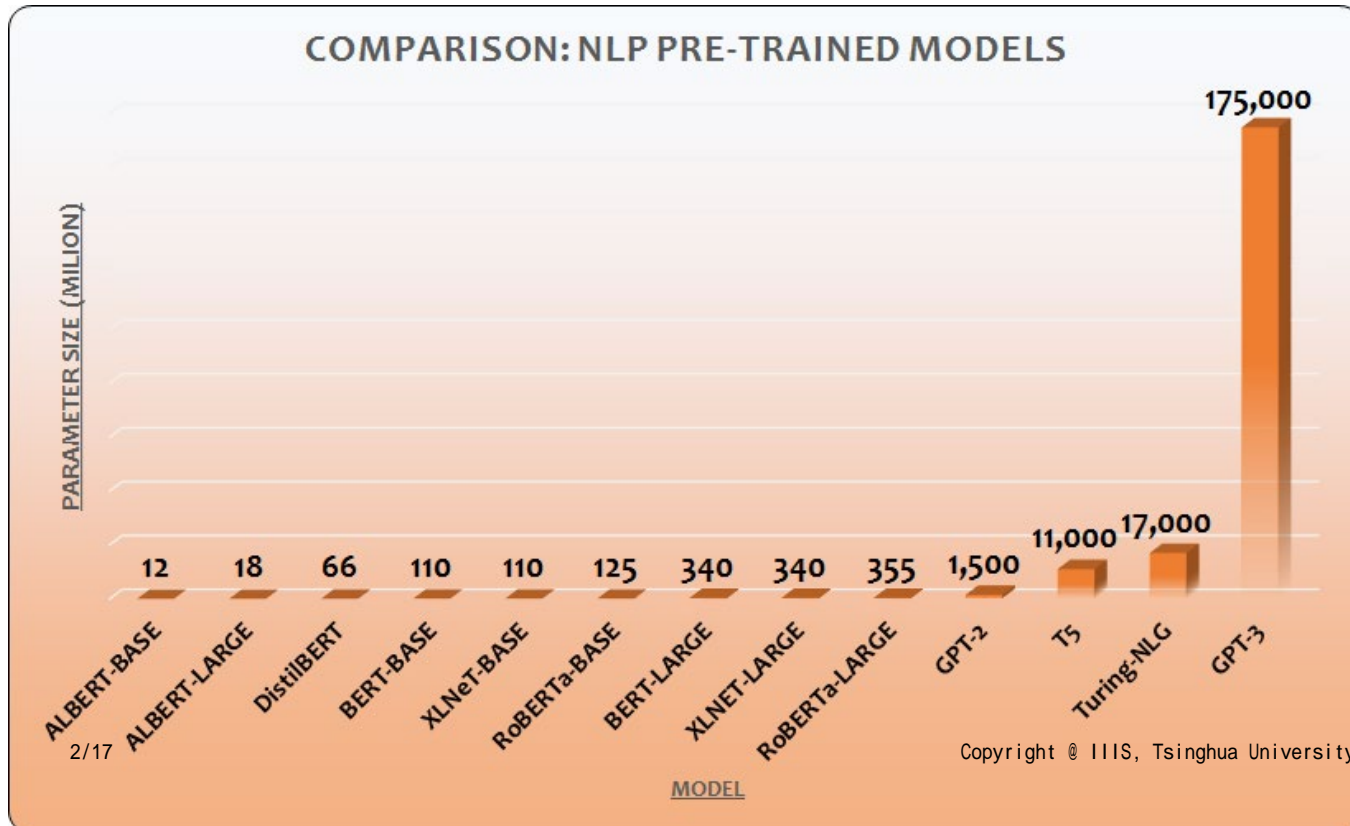
# The Era of Deep Learning

- Rapid Advances in Vision
  - Image Classification
  - Larger/Deeper models



# The Era of Deep Learning

- BIG data + Huge Model in NLP
  - Without any use of human knowledge



# The Era of Deep Learning

- Turing Award at 2018
  - Geoffrey Hinton, inventor of modern backpropagation and many fundamental ideas
  - Yann LeCun, Inventor of convolutional neural networks, foundation of modern pattern recognition
  - Yoshua Bengio, Inventor of neural language model and attention



# The Era of Deep Learning

---

## lanç The Nobel Prize in Physics 2024

---

John J. Hopfield

“for foundational discoveries and inventions that enable machine learning with artificial neural networks”



Geoffrey Hinton

“for foundational discoveries and inventions that enable machine learning with artificial neural networks”





# The Era of Deep Learning

## The Nobel Prize in Chemistry 2024

David Baker

“for computational protein design”



© Nobel Prize Outreach. Photo: Clément Morin

Demis Hassabis

“for protein structure prediction”



© Nobel Prize Outreach. Photo: Clément Morin

John Jumper

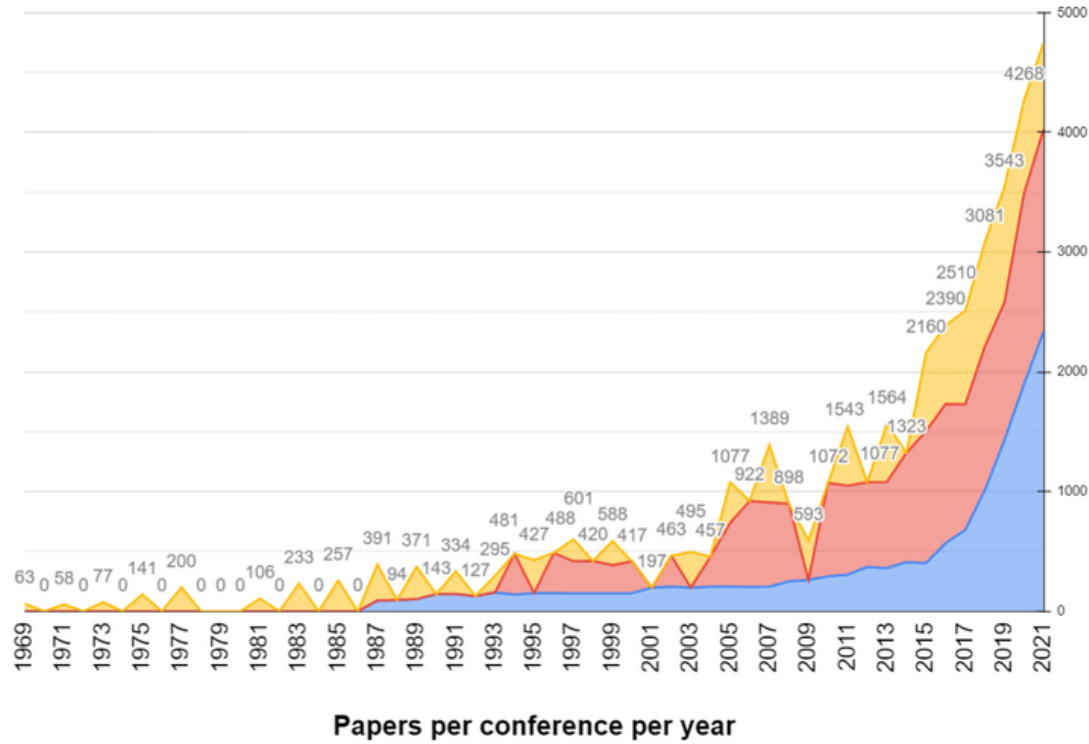
“for protein structure prediction”



© Nobel Prize Outreach. Photo: Clément Morin

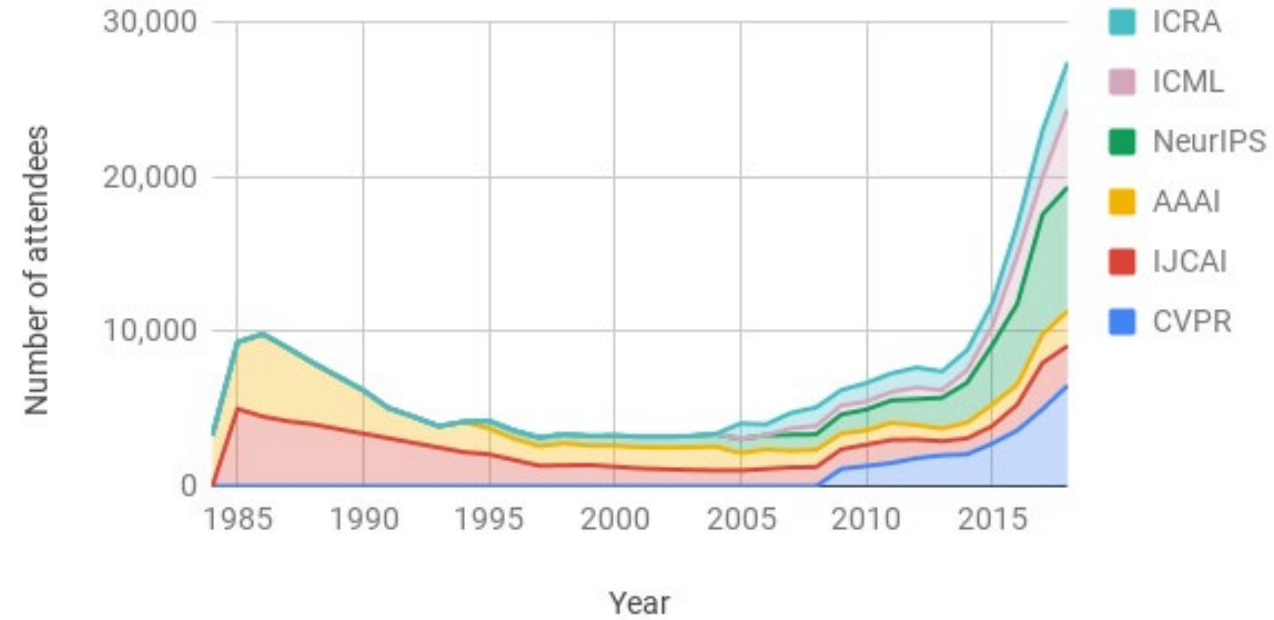
# The Era of Deep Learning

- IJCAI
- AAI
- NeurIPS



Attendance at large conferences (1984-2018)

Source: Conference provided data



# A Historical Perspective

- The history of deep learning
  - Original over-optimism in 1960s and then AI winter
  - A “cold” research area for decades (1970s ~ 2000s)
  - Magic year of 2012 ---- but “Rome wasn't built in a day”
- What's **Be ready to solve real-world challenges &**
- We can we learn from **Get your hands dirty!!!**
  - Be determined for great science!
  - Be respectful to prior works ---- most ideas have been explored
  - Be aware of entire literature beyond those hot topics!
  - Deep Learning is the science of engineering ...
  - **and it is critical to make things work!**
    - Idea is cheap. Show me the code/model!

# Coding Time!

- You need to
  - Use Google & ChatGPT! (so get your network access ready)
  - Be ready to use Python and PyTorch
  - Be prepared to write a well-written report in English (Grammarly recommended! Or ask your AI to help check it first ...)
  - Learn to use cloud compute (no need for HW1, more instructions coming)
- Computation
  - Your laptop for debugging and coding
  - Google CoLab for light tuning (this lecture)
  - AutoDL for heavy training (in future lecture)
    - Each enrolled IIIS student will have 800 RMB compute budget

# Google Colab Tutorial

**Deep Learning 2025 Spring**

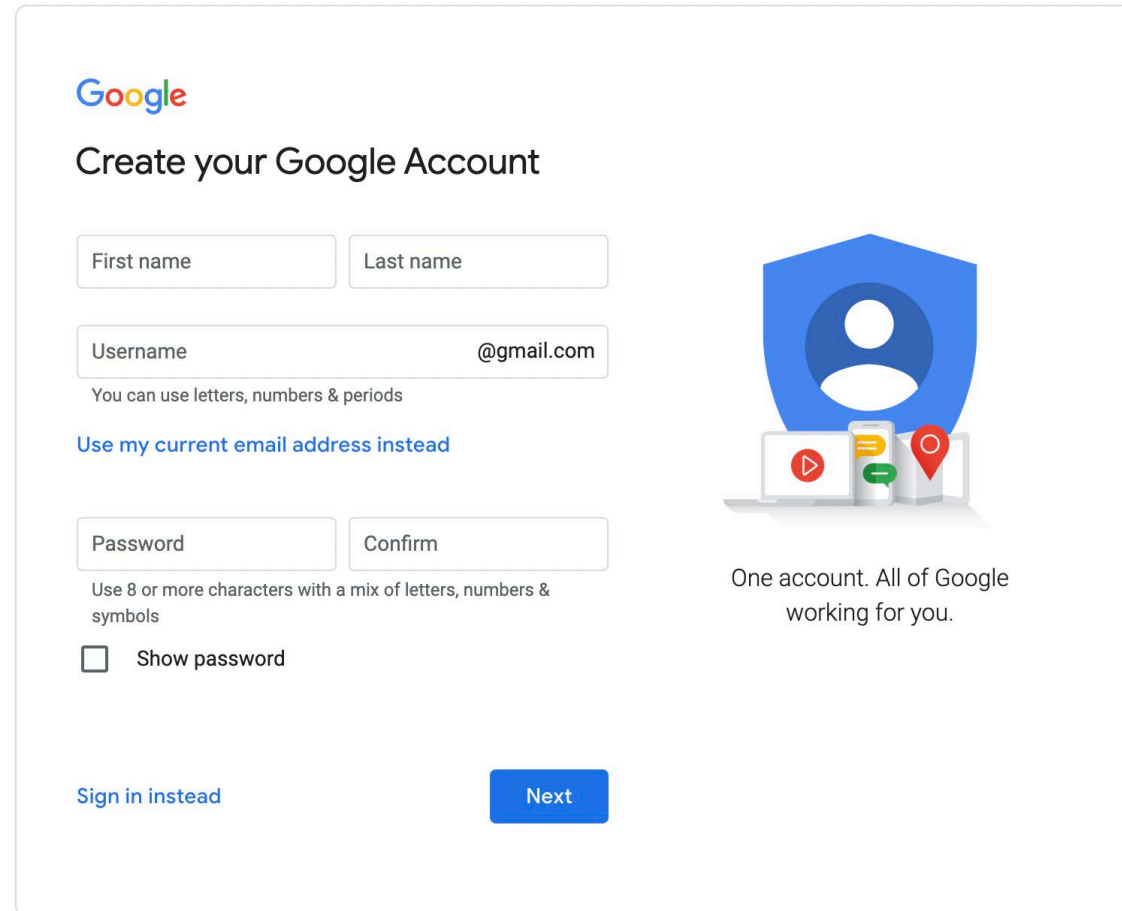
Jingzhi Cui

# Purpose

- Introduce the usage of Google Colab
- Help the ones who are not familiar with Python or PyTorch

# Google Account

- <https://accounts.google.com/SignUp?hl=en>



The screenshot shows the Google Account creation interface. At the top left is the Google logo, followed by the heading "Create your Google Account". The form consists of several input fields: "First name" and "Last name" (two separate boxes), "Username" (with "@gmail.com" as a placeholder) and "Password" (with a "Confirm" box next to it). Below the username field is a note: "You can use letters, numbers & periods". There is a link "Use my current email address instead" in blue. Below the password fields is a checkbox labeled "Show password". At the bottom left is a link "Sign in instead" and at the bottom right is a blue "Next" button. To the right of the form is a graphic of a blue shield with a white person icon, and below it, icons for YouTube, Gmail, and Maps. Below the graphic is the text "One account. All of Google working for you."

# Google Colab

- Colaboratory is a Google research project created to help disseminate machine learning education and research.
- It's a **Jupyter** notebook environment that requires no setup to use and runs entirely in the cloud.
- It allows you to use **free** Tesla T4 GPU it also gives you a total of 16GB of RAM, and you can use it **up to 12 hours** in row (You need to restart the session after 12 hours)





目录

- 使用入门
- 数据科学
- 机器学习
- 更多资源
  - 精选示例
- 部分



+ 代码 + 文本 复制到云端硬盘

连接 修改

## 欢迎使用 Colab!

如果您已经熟悉 Colab，请观看此视频，了解互动表格、已执行代码的历史记录视图和命令面板。



## 什么是 Colab?

借助 Colaboratory (简称 Colab)，您可在浏览器中编写和执行 Python 代码，并且：

- 无需任何配置
- 免费使用 GPU
- 轻松共享

无论您是一名学生、数据科学家还是 AI 研究员，Colab 都能够帮助您更轻松地完成工作。您可以观看 [Colab 简介](#) 了解详情，或查看下面的入门指南！

### 使用入门

您正在阅读的文档并非静态网页，而是一个允许您编写和执行代码的交互式环境，称为 **Colab 笔记本**。

例如，以下代码单元格包含一个简短的 Python 脚本，该脚本会计算值、将其存储在变量中并输出结果：

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400



# How to Upload Your Code

- From local (our coding projects)
  - File -> Upload Notebook -> Choose File from local
- From Google Drive (or other existing Colab notebooks)
  - Don't run codes directly!
- From GitHub
  - File -> Open Notebook

目录

- 使用入门
- 数据科学
- 机器学习
- 更多资源
- 精选示例

+ 部分

+ 代码 + 文本 复制到云端硬盘

连接 Colab AI

## 欢迎使用 Colab!

(新功能) 试用 Gemini

- [Generate a Gemini API](#)
- [Talk to Gemini with the](#)
- [Compare Gemini with](#)
- [More notebooks](#)

如果您已经熟悉 Colab, 请观

[ ] 开始借助 AI 编写或生成代码

## 什么是 Colab?

借助 Colaboratory (简称 Colab)

- 无需任何配置
- 免费使用 GPU
- 轻松共享

无论您是一名 **学生**、**数据科学家** 还是 **AI 研究员**, Colab 都能够帮助您更轻松地完成工作。您可以观看 [Colab 简介](#) 了解详情, 或查看下面的入门指南!

## 使用入门

### 打开笔记本

搜索笔记本

标题	上次打开时间 ↓	首次打开时间 ↑	
欢迎使用 Colaboratory	22:06	2022年10月27日	🔗
Basic_MetaDrive_Usages.ipynb	1月24日	1月24日	🔍 🔗
Kaggle.ipynb	2023年12月27日	2023年12月26日	📄 🔗
tensorboard_in_notebooks.ipynb	2023年12月26日	2023年12月26日	🔍 🔗
word_embeddings_tutorial.ipynb	2023年12月21日	2023年12月21日	🔍 🔗
TPUs in Colab	2023年10月27日	2023年10月27日	🔗
Bot Of Legends Failed To Download Could N...	2023年10月7日	2023年10月7日	📄 🔗

+ 新建笔记本 取消



# Set Up Environment

- Hardware environment
  - Connect to hosted runtime by clicking “Connect”
  - Enable GPU (TPU):
    - Edit -> Notebook Settings
- You can run shell command with a “!” prefix

```
[1] !nvidia-smi

Wed Feb 16 12:26:29 2022

+-----+
| NVIDIA-SMI 460.32.03   Driver Version: 460.32.03   CUDA Version: 11.2   |
+-----+
| GPU  Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|====+=====+====+=====+=====+=====+=====+=====+=====+
|  0  Tesla K80         Off      | 00000000:00:04.0 Off  |    0%      Default  |
| N/A   40C    P8     27W / 149W |  0MiB / 11441MiB |           |      MIG M.     |
|====+=====+====+=====+=====+=====+=====+=====+
| Processes:                                                       GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage   |
|====+=====+=====+=====+=====+=====+=====+=====+
| No running processes found                                           84         |
+-----+
```

相当于连接到了一台服务器上

+ 代码 + 文本

开始借助 AI 编写或生成代码。

### 版本说明

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

#### 2024-02-21

- Try out Gemma on [Colab!](#)
- Allow unicode in form text inputs
- Display documentation and link to source when displaying functions
- Display image-like ndarrays as images
- Improved UX around quick charts and execution error suggestions
- Released Marketplace image for the month of February ([GitHub issue](#))
- Python package upgrades
  - bigframes 0.19.2 -> 0.21.0
  - regex 2023.6.3 -> 2023.12.25
  - spacy 3.6.1 -> 3.7.4
  - beautifulsoup4 4.11.2 -> 4.12.3
  - tensorflow-probability 0.22.0 -> 0.23.0
  - google-cloud-language 2.9.1 -> 2.13.1
  - google-cloud-aiplatform 1.39.0 -> 1.42.1
  - transformers 4.35.2 -> 4.37.2
  - pyarrow 10.0.1 -> 14.0.2

#### 2024-01-29

- New [Kaggle Notebooks <-> Colab updates!](#) Now you can:
  - Import directly from Colab without having to download/re-upload
  - Upload via link, by pasting Google Drive or Colab URLs
  - Export & run Kaggle Notebooks on Colab with 1 click
- Try these notebooks that talk to Gemini:
  - [Gemini and Stable Diffusion](#)
  - [Learning with Gemini and ChatGPT](#)
  - [Talk to Gemini with Google's Speech to Text API](#)
  - [Sell lemonade with Gemini and Sheets](#)
  - [Generate images with Gemini and Vertex](#)
- Python package upgrades
  - google-cloud-aiplatform 1.38.1 -> 1.39.0
  - bigframes 0.18.0 -> 0.19.2
  - polars 0.17.3 -> 0.20.2
  - gdown 4.6.6 -> 4.7.3 ([GitHub issue](#))
  - tensorflow-hub 0.15.0 -> 0.16.0
  - flax 0.7.5 -> 0.8.0
- Python package inclusions
  - sentencepiece 0.1.99

#### 2024-01-08

# 选择使用 GPU 还是 CPU

开始借助 AI 编写或生成代码。

版本说明 ×

Please follow our [blog](#) to see more info as [Analyzing a Bank Failure with Colab](#).

**2024-02-21**

- Try out Gemma on [Colab!](#)
- Allow unicode in form text inputs
- Display documentation and link t
- Display image-like ndarrays as im
- Improved UX around quick charts
- Released Marketplace image for
- Python package upgrades
  - bigframes 0.19.2 -> 0.21.0
  - regex 2023.6.3 -> 2023.12.25
  - spacy 3.6.1 -> 3.7.4
  - beautifulsoup4 4.11.2 -> 4.12.3
  - tensorflow-probability 0.22.0 -> 0.23.0
  - google-cloud-language 2.9.1 -> 2.13.1
  - google-cloud-aiplatform 1.39.0 -> 1.42.1
  - transformers 4.35.2 -> 4.37.2
  - pyarrow 10.0.1 -> 14.0.2

**2024-01-29**

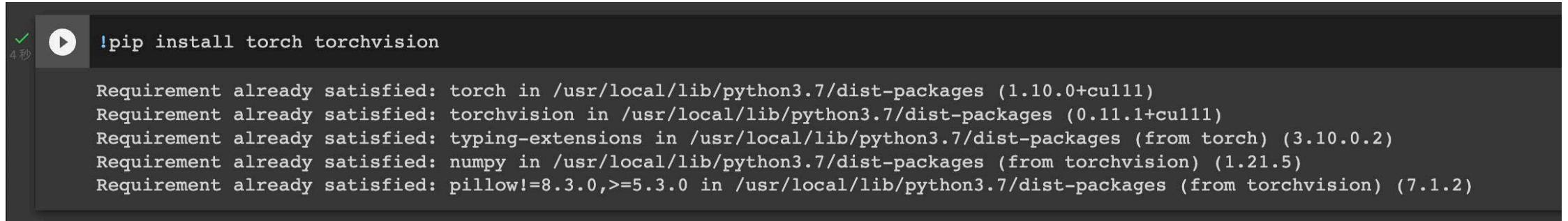
- New [Kaggle Notebooks <-> Colab updates!](#) Now you can:
  - Import directly from Colab without having to download/re-upload
  - Upload via link, by pasting Google Drive or Colab URLs
  - Export & run Kaggle Notebooks on Colab with 1 click
- Try these notebooks that talk to Gemini:
  - [Gemini and Stable Diffusion](#)
  - [Learning with Gemini and ChatGPT](#)
  - [Talk to Gemini with Google's Speech to Text API](#)
  - [Sell lemonade with Gemini and Sheets](#)
  - [Generate images with Gemini and Vertex](#)
- Python package upgrades
  - google-cloud-aiplatform 1.38.1 -> 1.39.0
  - bigframes 0.18.0 -> 0.19.2
  - polars 0.17.3 -> 0.20.2
  - gdown 4.6.6 -> 4.7.3 ([GitHub issue](#))
  - tensorflow-hub 0.15.0 -> 0.16.0
  - flax 0.7.5 -> 0.8.0
- Python package inclusions
  - sentencepiece 0.1.99

**2024-01-08**

连接到托管的运行时  
更改运行时类型  
连接到自定义 GCE 虚拟机  
连接到本地运行时  
查看资源  
管理会话  
断开连接并删除运行时  
显示代码执行历史记录  
将光标放在上次运行的单元格上

# Set Up Environment

- Code environment
  - Install necessary packages with “!pip install”



```
✓ 4 秒 !pip install torch torchvision  
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (1.10.0+cu111)  
Requirement already satisfied: torchvision in /usr/local/lib/python3.7/dist-packages (0.11.1+cu111)  
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch) (3.10.0.2)  
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torchvision) (1.21.5)  
Requirement already satisfied: pillow!=8.3.0,>=5.3.0 in /usr/local/lib/python3.7/dist-packages (from torchvision) (7.1.2)
```

# Set Up Environment

- Code environment

- Where am I?

```
[2] !pwd  
/content
```

- Upload/Mount necessary files/data

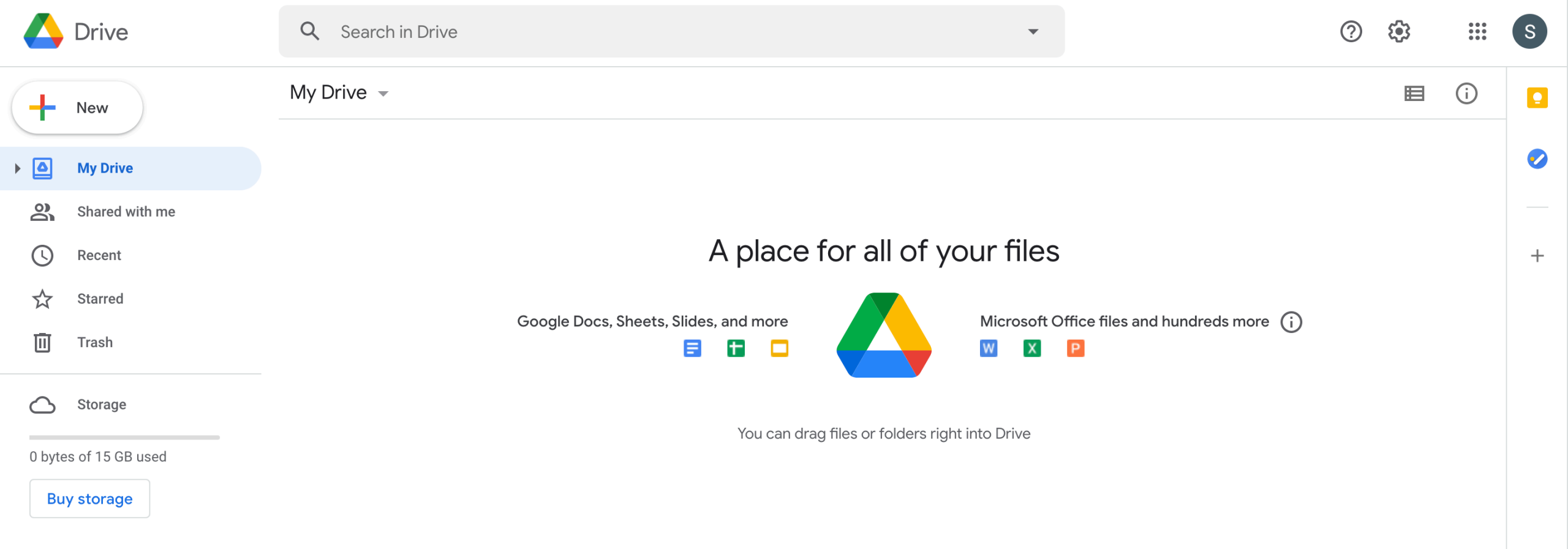
The screenshot shows a code editor interface with a file explorer on the left and a terminal window on the right. The file explorer displays a directory structure with a folder named 'sample\_data' containing several files: README.md, anscombe.json, california\_housing\_test.csv, california\_housing\_train.csv, mnist\_test.csv, and mnist\_train\_small.csv. The terminal window shows the command '!nvidia-smi' being executed, resulting in the following output:

```
Wed Feb 16 12:30:00  
+-----+  
| NVIDIA-SMI 418.66.03 Driver Version 418.66.03  
+-----+  
| GPU   Name               Fan    Temp    Perf     Pwr   Usage    |  
| Fan  Temp    Perf     Pwr   Usage    |  
+-----+  
| 0   Tesla K80              34C    34C    P0     34W   100%    |  
| N/A   40C                    |  
+-----+  
+-----+  
| Processes: |  
| GPU   GI    PID    Type  Use    Memory |  
| Fan  Temp    Perf     Pwr   Usage    |  
+-----+  
| No running processes found  
+-----+
```



# Mount your google drive in Colab notebook

## 1. Open your Google drive

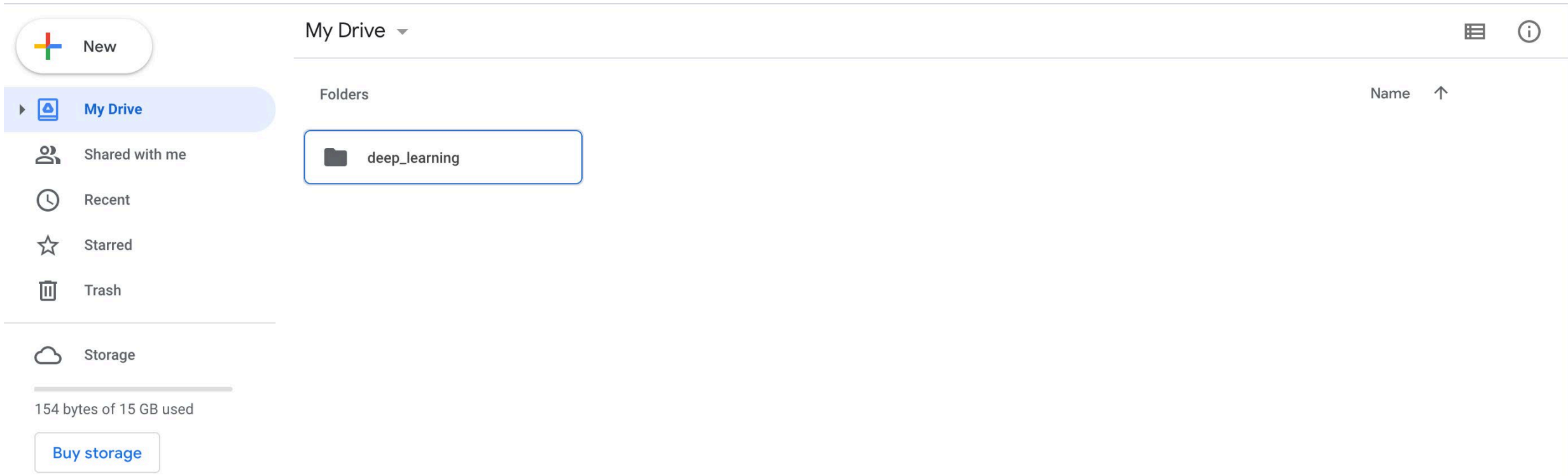


<https://drive.google.com/drive/my-drive>

2/17

# Mount your google drive in Colab notebook

## 2. Create a new folder in My Drive



# Mount your google drive in Colab notebook

## 3. Run the code in your colab

```
[ ] from google.colab import drive
drive.mount('/content/drive')
```

## 4. Click the link and enter your authorization code

```
from google.colab import drive
drive.mount('/content/drive')
```

Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&red](https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&red)

Enter your authorization code:

## 5. success

```
[1] from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

## 6. Now you can access your google drive in Colab notebook in /content/drive/MyDrive

```
[ ] !ls /content/drive/MyDrive
```

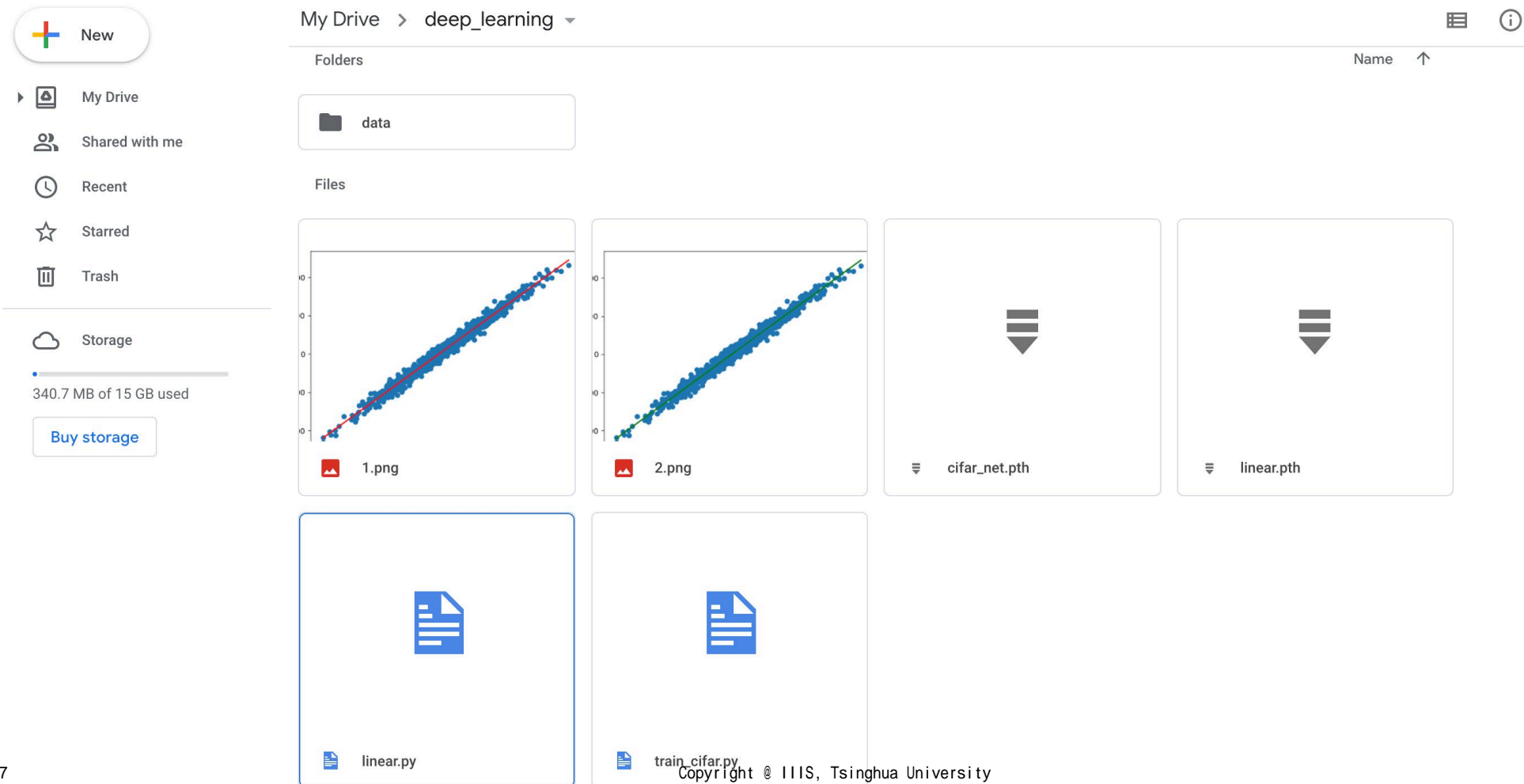
```
2/17
'Colab Notebooks'  deep_learning
```

# Run Your Code

- Fill out the code in your homework
- Execute the code one-by-one in the notebook
  - When you select any other code segments, then you should be able to see a number in [ ] area
  - If there is no number within [ ], then you need to re-run that segment

# Run your code in google drive

## 1. Upload your code to you google drive (linear.py)



# Run your code in google drive

## 2. Mount your google drive

```
[1] from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ] !ls /content/drive/MyDrive
```

```
'Colab Notebooks'  deep_learning
```

```
[ ] !nvidia-smi
```

```
Mon Feb 22 08:03:05 2021
```

```
+-----+
| NVIDIA-SMI 460.39          Driver Version: 460.32.03   CUDA Version: 11.2   |
+-----+-----+-----+
| GPU  Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|====+=====+====+=====+=====+=====+=====+=====+
|    0  Tesla T4             Off   | 00000000:00:04:0 Off |                    0 |
| N/A   33C    P8      9W / 70W   |  0MiB / 15109MiB |      0%      Default |
+-----+-----+-----+-----+-----+-----+
|
+-----+
| Processes:
| GPU  GI  CI           PID  Type  Process name          GPU Memory
|   ID  ID                                     Usage
+-----+-----+-----+-----+-----+-----+
| No running processes found
+-----+
```

# Run your code in google drive

## 3. run your code in the google drive

```
[ ] !python /content/drive/MyDrive/deep_learning/linear.py
```

```
cuda:0
(1000, 1) (1000,) 2.6731389771201597
target paremeters, weight 2.673, bias: 10.000:
[1, 1] loss: 118208.547, weight: 0.643, bias: -0.978
[1, 2] loss: 108463.602, weight: 0.646, bias: -0.957
[11, 1] loss: 103885.477, weight: 0.828, bias: 0.410
[11, 2] loss: 94150.102, weight: 0.841, bias: 0.502
[21, 1] loss: 86701.719, weight: 1.082, bias: 2.234
[21, 2] loss: 78251.109, weight: 1.094, bias: 2.320
[31, 1] loss: 74544.781, weight: 1.315, bias: 3.803
[31, 2] loss: 67236.242, weight: 1.325, bias: 3.874
[41, 1] loss: 66530.805, weight: 1.517, bias: 5.076
[41, 2] loss: 60108.305, weight: 1.526, bias: 5.133
[51, 1] loss: 61304.531, weight: 1.690, bias: 6.098
[51, 2] loss: 55559.105, weight: 1.698, bias: 6.144
[61, 1] loss: 57895.461, weight: 1.839, bias: 6.917
[61, 2] loss: 52669.000, weight: 1.846, bias: 6.953
[71, 1] loss: 55665.801, weight: 1.965, bias: 7.573
[71, 2] loss: 50839.309, weight: 1.971, bias: 7.602
[81, 1] loss: 54202.312, weight: 2.072, bias: 8.099
[81, 2] loss: 49685.758, weight: 2.077, bias: 8.122
[91, 1] loss: 53237.641, weight: 2.163, bias: 8.520
[91, 2] loss: 48962.387, weight: 2.167, bias: 8.538
[101, 1] loss: 52598.625, weight: 2.240, bias: 8.857
```

# Download Your Checkpoints

- Save checkpoints in Google Drive frequently
- Download your files immediately after finishing the experiment
  - File -> Download -> Download .ipynb



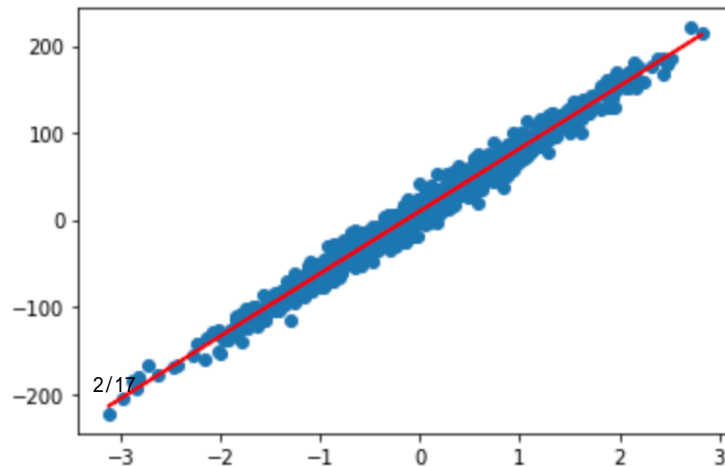
# Generate training data

```
[1] import torch
import numpy as np
from sklearn.datasets import make_regression
import matplotlib.pyplot as plt
import math
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
```

cuda:0

```
[2] bias = 10
X, y, coef = make_regression(n_samples=1000, n_features=1, n_informative=1,
                             coef=True, bias=bias, noise=10)
X, y = X.astype(np.float32), y.astype(np.float32)
plt.scatter(X, y)
plt.plot(X, X.reshape(-1) * coef + bias, color='red')
X_all, y_all = torch.from_numpy(X).to(device), torch.from_numpy(y).to(device)
print(X.shape, y.shape, coef)
```

(1000, 1) (1000,) 71.81313920366837



# Define a Network

```
[3] import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc = nn.Linear(1, 1)

    def forward(self, x):
        predict = self.fc(x)
        return predict

net = Net()
net.to(device)

Net(
  (fc): Linear(in_features=1, out_features=1, bias=True)
)
```

# Train the network

```
[5] bsz = 500
print("target parameters, weight %.3f, bias: %.3f:" % (coef, bias))
for epoch in range(301): # loop over the dataset multiple times

    running_loss = 0.0
    for i, data in enumerate(zip(torch.split(X_all, bsz), torch.split(y_all, bsz))):
        # get the inputs; data is a list of [inputs, labels]
        # inputs, labels = data
        inputs, labels = data[0], data[1]

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        # print(outputs.shape, outputs)
        loss = criterion(outputs.view(-1), labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if epoch % 10 == 0:
            weight_ = net.fc.weight.item()
            bias_ = net.fc.bias.item()
            print('[%d, %5d] loss: %.3f, weight: %.3f, bias: %.3f' % (epoch + 1, i + 1, running_loss, weight_, bias_))
            running_loss = 0.0

print('Finished Training')
```

```
[6] PATH = './linear.pth'
torch.save(net.state_dict(), PATH)
```

# The training progress

```

↳ target parameters, weight 71.813, bias: 10.000:
[1, 1] loss: 2705857.500, weight: -0.320, bias: -0.436
[1, 2] loss: 2699893.000, weight: -0.183, bias: -0.415
[11, 1] loss: 2113961.250, weight: 8.660, bias: 0.927
[11, 2] loss: 2084677.250, weight: 9.260, bias: 1.018
[21, 1] loss: 1409867.000, weight: 20.470, bias: 2.712
[21, 2] loss: 1399013.750, weight: 21.024, bias: 2.796
[31, 1] loss: 918827.750, weight: 30.618, bias: 4.231
[31, 2] loss: 921905.625, weight: 31.073, bias: 4.299
[41, 1] loss: 600871.125, weight: 38.840, bias: 5.449
[41, 2] loss: 611270.500, weight: 39.205, bias: 5.502
[51, 1] loss: 397890.875, weight: 45.431, bias: 6.413
[51, 2] loss: 411304.188, weight: 45.724, bias: 6.455
[61, 1] loss: 268777.438, weight: 50.704, bias: 7.175
[61, 2] loss: 282736.000, weight: 50.938, bias: 7.208
[71, 1] loss: 186803.750, weight: 54.921, bias: 7.776
[71, 2] loss: 199999.219, weight: 55.109, bias: 7.803
[81, 1] loss: 134855.828, weight: 58.294, bias: 8.252
[81, 2] loss: 146672.781, weight: 58.444, bias: 8.273
[91, 1] loss: 102010.633, weight: 60.991, bias: 8.627
[91, 2] loss: 112233.250, weight: 61.111, bias: 8.643
[101, 1] loss: 81303.500, weight: 63.147, bias: 8.923
[101, 2] loss: 89936.211, weight: 63.243, bias: 8.936
[111, 1] loss: 68297.453, weight: 64.872, bias: 9.157
[111, 2] loss: 75456.961, weight: 64.949, bias: 9.167
[121, 1] loss: 60167.836, weight: 66.251, bias: 9.341
[121, 2] loss: 66020.109, weight: 66.313, bias: 9.349
[131, 1] loss: 55118.484, weight: 67.354, bias: 9.487
[131, 2] loss: 59842.617, weight: 67.404, bias: 9.493
[141, 1] loss: 52008.531, weight: 68.236, bias: 9.601
[141, 2] loss: 55777.508, weight: 68.276, bias: 9.606
[151, 1] loss: 50114.676, weight: 68.942, bias: 9.692

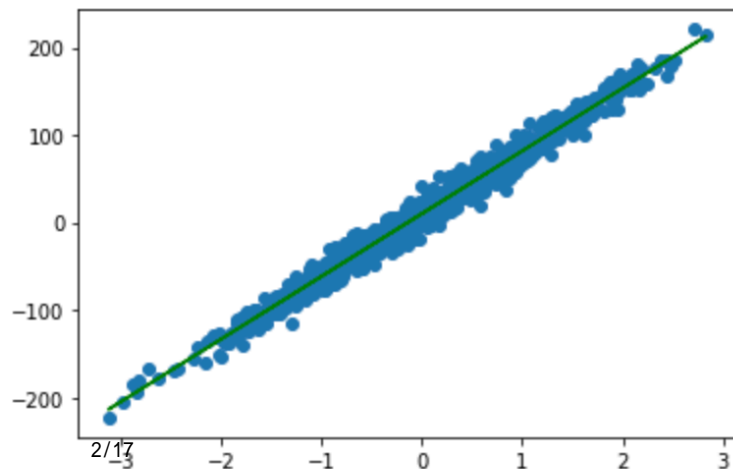
```

# Visualize the weight and bias of trained model

```
[7] net = Net()
net.load_state_dict(torch.load(PATH))
net.to(device)
weight_ = net.fc.weight.item()
bias_ = net.fc.bias.item()
print("parameters in model:")
print("weiths: ", weight_, "bias: ", bias_)
print("target paremeters :")
print("weiths: ", coef, "bias: ", bias)

fig2 = plt.figure()
plt.scatter(X, y)
plt.plot(X, X.reshape(-1) * weight_ + bias_, color='green')
```

```
parameters in model:
weiths: 71.66061401367188 bias: 10.018959999084473
target paremeters :
weiths: 71.81313920366837 bias: 10
[<matplotlib.lines.Line2D at 0x7ff322becf98>]
```



# PyTorch Example

- Load dataset

```

✓ 1秒 #load the data
train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('../data', train=True, download=True,
                  transform=transforms.Compose([
                      transforms.ToTensor(),
                      transforms.Normalize((0.1307,), (0.3081,))
                  ])),
    batch_size=args['batch_size'], shuffle=True, **kwargs)
test_loader = torch.utils.data.DataLoader(
    datasets.MNIST('../data', train=False, transform=transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))
    ])),
    batch_size=args['test_batch_size'], shuffle=True, **kwargs)

```

Downloading <http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz>  
 Downloading <http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz> to ../data/MNIST/raw/train-images-idx3-ubyte.gz  
 ██████████ 9913344/? [00:00<00:00, 18851363.58it/s]  
 Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading <http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz>  
 Downloading <http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz> to ../data/MNIST/raw/train-labels-idx1-ubyte.gz  
 ██████████ 29696/? [00:00<00:00, 484374.71it/s]  
 Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

# PyTorch Example

- Define the network

```
[11]
✓ 0 秒

class Net(nn.Module):
    #This defines the structure of the NN.
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d() #Dropout
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        #Convolutional Layer/Pooling Layer/Activation
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        #Convolutional Layer/Dropout/Pooling Layer/Activation
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        x = x.view(-1, 320)
        #Fully Connected Layer/Activation
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        #Fully Connected Layer/Activation
        x = self.fc2(x)
        #Softmax gets probabilities.
        return F.log_softmax(x, dim=1)
```

# PyTorch Example

- Define the dataflow

```
def train(epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        if args['cuda']:
            data, target = data.cuda(), target.cuda()
        #Variables in Pytorch are differentiable.
        data, target = Variable(data), Variable(target)
        #This will zero out the gradients for this batch.
        optimizer.zero_grad()
        output = model(data)
        # Calculate the loss The negative log likelihood loss. It is useful to train a classification p
        loss = F.nll_loss(output, target)
        #dloss/dx for every Variable
        loss.backward()
        #to do a one-step update on our parameter.
        optimizer.step()
        #Print out the loss periodically.
        if batch_idx % args['log_interval'] == 0:
            print('Train Epoch: {} [{} / {}] ( {:.0f}%) \t Loss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.data[0]))
```



# Notice:

- Save your data and model in google drive, otherwise, they may be lost when you disconnect.
- If you want to train your model using Colab for a long time, make sure you save your training states frequently and you can recover your training from these training states.
- Try it out yourself in HW0!

# The example links

- PyTorch linear:
  - [https://colab.research.google.com/drive/1yS5kvHC\\_TQuWesPmon7W7hrD978tOJ2i?usp=sharing](https://colab.research.google.com/drive/1yS5kvHC_TQuWesPmon7W7hrD978tOJ2i?usp=sharing)
- Mount and run your code in google drive:
  - <https://colab.research.google.com/drive/163LrsNRHC48tCCGHhBfgqCyGCODXRmaR?usp=sharing>
- linear.py
  - <https://drive.google.com/file/d/1jdwTgo6Qf4PoPz5JEQwy-KJoPncb30yt/view?usp=sharing>
- PyTorch MNIST
  - <https://colab.research.google.com/github/rpi-techfundamentals/fall2018-materials/blob/master/10-deep-learning/04-pytorch-mnist.ipynb>

# Thanks!