

AI @ UChicago

Introductory Course Meeting 1

Overview

What is Machine Learning?

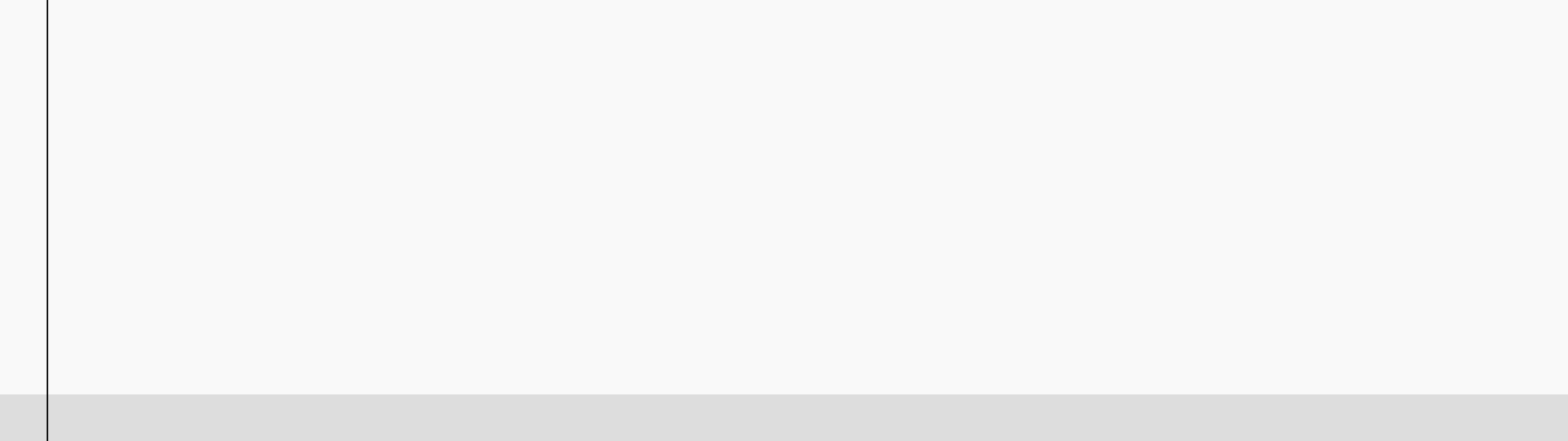
About AI @ UChicago

- Aim to provide education.
- 4 lectures this quarter: week 3, 4, 7, 8.
- **Introductory course** and **Language Models course (Transformers)**
- New club! So let us know what you want.

Course Overview

- **Meeting 1: What is Machine Learning (ML)? The goals of supervised learning.**
- Meeting 2: Classification, Regression and the basic ML workflow.
- Meeting 3: Overfitting, regularization, validation.
- Meeting 4: Neural networks and the Multi-Layer Perceptron (MLP) part 1.

Course Overview

- Interrupt me whenever – “I have a question”.
 - I might be wrong.
 - Only assume basic familiarity with python.
- 
- A thin vertical line is positioned on the left side of the slide, extending from the middle to the bottom. A solid grey horizontal bar spans the entire width of the slide at the bottom.

01

What is Machine Learning?

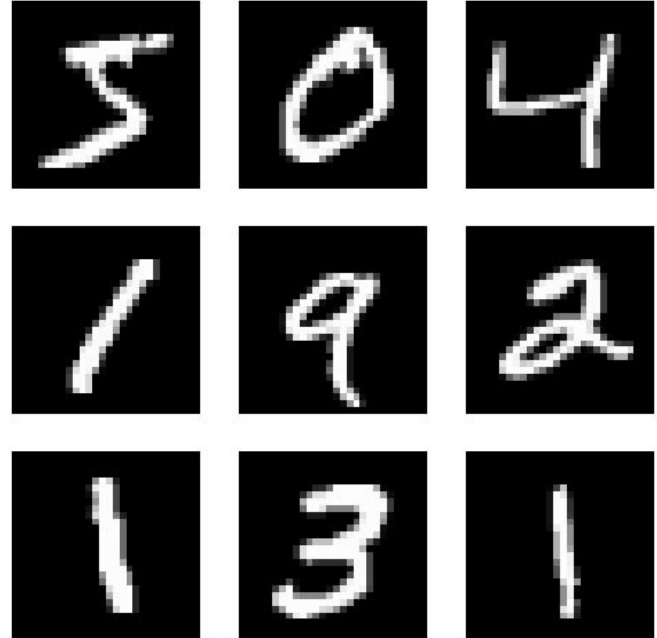
Machine Learning

- Using **data and examples**, instead of **explicit instructions**, to automatically create systems that perform **complex tasks**.
- Example of tasks:
 - Price prediction (house prices)
 - Detecting anomalies (malignant or benign tumor?)
 - Object recognition (what digit is this image?)
 - Sentiment analysis (is the course feedback positive or negative?)

Example: Digit Recognition

- Given an image, can we recognize which digit is it of?

First 9 images from the MNIST dataset



Example: House Price Prediction

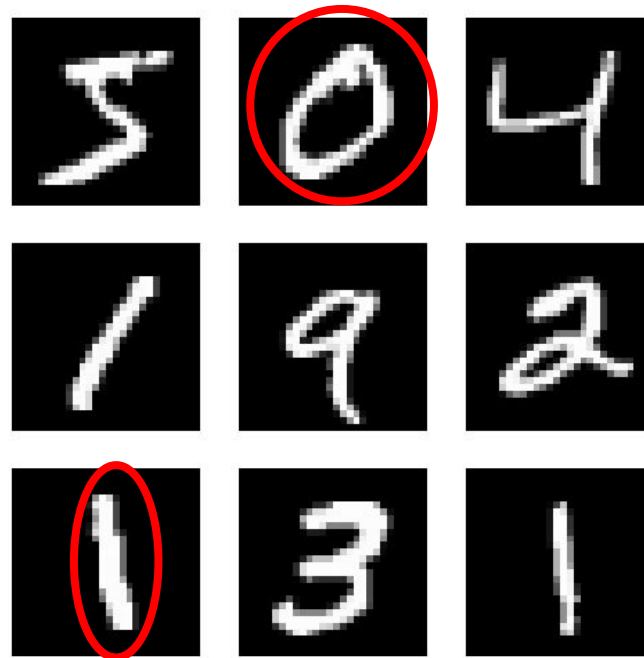
Index	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

- Given the above statistics of the houses in a block, can we predict the median price of these houses?

Machine Learning

- These are complex tasks!
- One way to do it would be to use explicit instructions.

First 9 images from the MNIST dataset



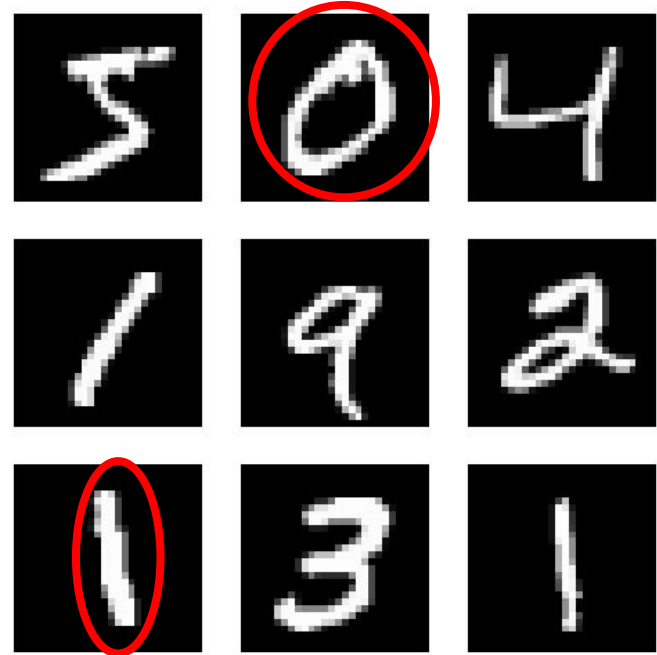
Machine Learning

- These are complex tasks!
- One way to do it would be to use **explicit instructions**.
- But this is problematic!

- Hence our goal:

Using **data and examples**, instead of **explicit instructions**, to automatically create systems that perform **complex tasks**.

First 9 images from the MNIST dataset

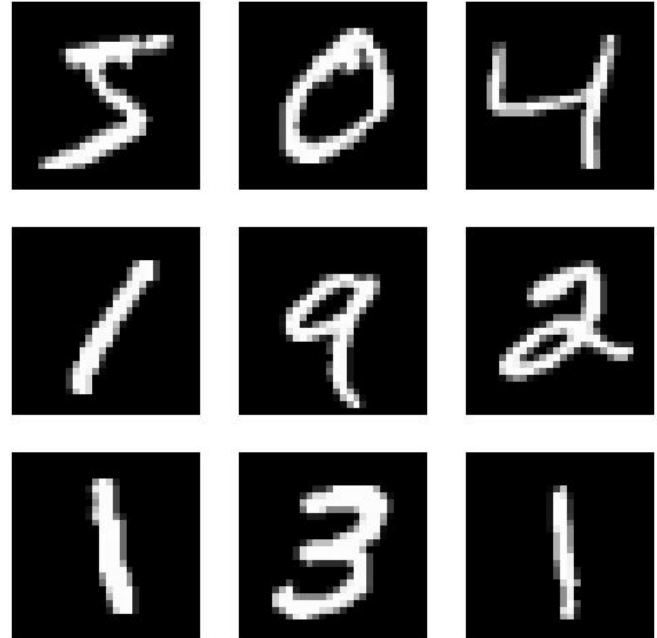


Machine Learning

The MNIST dataset has 70,000 labeled pairs of a photo of a digit and its corresponding digit.

How do we best **use this dataset** to build the best machine that **predicts** the digit from the image?

First 9 images from the MNIST dataset



Supervised Machine Learning

- In this course, we focus on **supervised machine learning**.
- **Given** a **labeled** dataset S of m pairs (x, y) of some input x and its corresponding **true label** y
- Example: Digit recognition on MNIST dataset:
 - $m = 70000$
 - x : image of a digit (or any image in general) (how is an image represented digitally?)
 - y : the digit
- **Task:** Build a machine (= function) h that takes in input x and spits out **prediction** $h(x)$ so that it is as “correct” (“close” to true label y) as possible, **in reality** (why?)

Supervised Machine Learning

- **Task:** Build a machine (= function) h that takes in input x and spits out **prediction** $h(x)$ so that it is as “correct” (“close” to true label y) as possible **in reality**.
- **In reality:**
 - Your machine h = what you think the relationship between x and y is like
 - Your machine h has to **generalize** beyond the given dataset.
 - Clearly, S is not “reality”.
 - Getting it right on S is not everything!
 - For now: maybe h_1 making some mistakes on S so that it performs better “in reality”, is better than h_2 making no mistakes on S .

Example: Height to wingspan

- **Task:** build a machine learning model that **predicts** the **wingspan** of UChicago students **from** their **height**.
- You are given: the height and wingspan of 50 UChicago students, both in inches.
- S consists of (height1, wingspan1), (height2, wingspan2), ..., (height50, wingspan50).
- These are $(x_1, y_1), (x_2, y_2), \dots, (x_{500}, y_{500})$.
- $m = 50$
- x = height in inches. A real number, e.g., 70
- y = wingspan in inches. A real number, e.g., 72
- h : takes in a real number and spits out a real number, e.g., $h(x) = 0.9x + 10$
- “**Reality**”: the relationship between height and wingspan of **all UChicago** students.
- Say, there are 8000 of us. **Question:** Is getting it perfect on your given 50 data points good?
- What if you have 500? 1000?

Example: House Price Prediction

Index	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

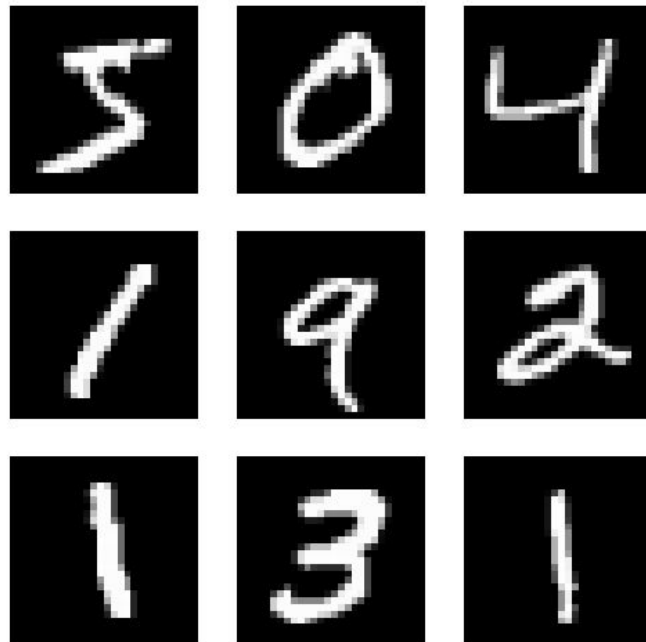
- California housing dataset. $m = 20640$.
- x : statistics of houses in a block in California. Vector of 8 real numbers.
- y : median price of houses in that block. A real number.
- h : takes in a vector of 8 real numbers and spits out a real number,
- e.g., $h(x) = 20x_1 - 4x_2x_3 + \sin(x_4)\cos(x_5) + 123x_6x_7x_8 + 1$.

Example: Digit Recognition

- MNIST dataset. $m = 70000$.
- x : 28 x 28 matrix of 0-255, high = bright.
- y : The digit. An integer from 0-9.
- h : takes in a 28x28 matrix and spits out an integer 0-9.

e.g. $h(x) = ??????????$

First 9 images from the MNIST dataset



A lot of other examples!

- Spam filtering
- Face detection/recognition; object detection, search
- Machine translation, summarization, chat agents
- Typing completion, search, recommendation systems
- Protein fold prediction, imaging, diagnosis

02

The history of the hype

Are 21st century computer scientists that smart?

Some ML History

- **1956:** Dartmouth Summer Research Project on Artificial Intelligence

“The study is to proceed on the basis of the conjecture **that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.** An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.”

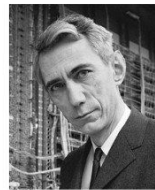
1956 Dartmouth Conference: The Founding Fathers of AI



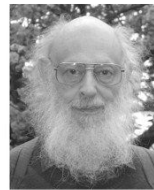
John McCarthy



Marvin Minsky



Claude Shannon



Ray Solomonoff



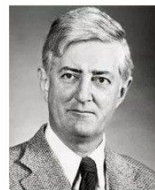
Alan Newell



Herbert Simon



Arthur Samuel



Oliver Selfridge



Nathaniel Rochester



Trenchard More

Some ML History

■ 1956: Dartmouth Summer Research Project on Artificial Intelligence

1. Automatic Computers

If a machine can do a job, then an automatic calculator can be programmed to simulate the machine. The speeds and memory capacities of present computers may be insufficient to simulate many of the higher functions of the human brain, but the major obstacle is not lack of machine capacity, but our inability to write programs taking full advantage of what we have.

2. How Can a Computer be Programmed to Use a Language

It may be speculated that a large part of human thought consists of manipulating words according to rules of reasoning and rules of conjecture. From this point of view, forming a generalization consists of admitting a new word and some rules whereby sentences containing it imply and are implied by others. This idea has never been very precisely formulated nor have examples been worked out.

3. Neuron Nets

How can a set of (hypothetical) neurons be arranged so as to form concepts. Considerable theoretical and experimental work has been done on this problem by Uttley, Rashevsky and his group, Farley and Clark, Pitts and McCulloch, Minsky, Rochester and Holland, and others. Partial results have been obtained but the problem needs more theoretical work.

4. Theory of the Size of a Calculation

If we are given a well-defined problem (one for which it is possible to test mechanically whether or not a proposed answer is a valid answer) one way of solving it is to try all possible answers in order. This method is inefficient, and to exclude it one must have some criterion for efficiency of calculation. Some consideration will show that to get a measure of the efficiency of a calculation it is necessary to have on hand a method of measuring the complexity of calculating devices which in turn can be done if one has a theory of the complexity of functions. Some partial results on this problem have been obtained by Shannon, and also by McCarthy.

5. Self-Improvement

Probably a truly intelligent machine will carry out activities which may best be described as self-improvement. Some schemes for doing this have been proposed and are worth further study. It seems likely that this question can be studied abstractly as well.

6. Abstractions

A number of types of "abstraction" can be distinctly defined and several others less distinctly. A direct attempt to classify these and to describe machine methods of forming abstractions from sensory and other data would seem worthwhile.

7. Randomness and Creativity

A fairly attractive and yet clearly incomplete conjecture is that the difference between creative thinking and unimaginative competent thinking lies in the injection of a some randomness. The randomness must be guided by intuition to be efficient. In other words, the educated guess or the hunch include controlled randomness in otherwise orderly thinking.

Some ML History

- **1958:** Rosenblatt's Perceptron - the simplest “neural network”

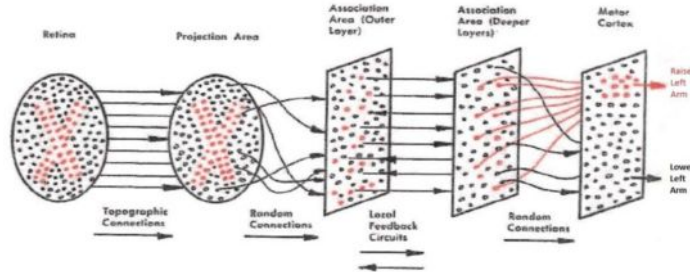


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

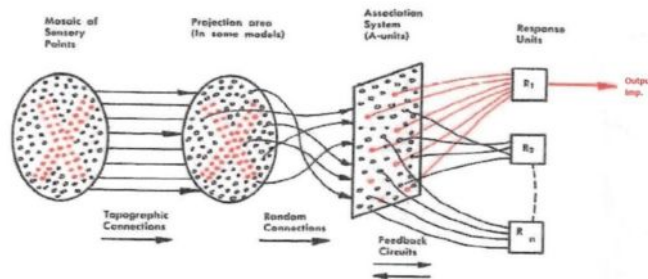
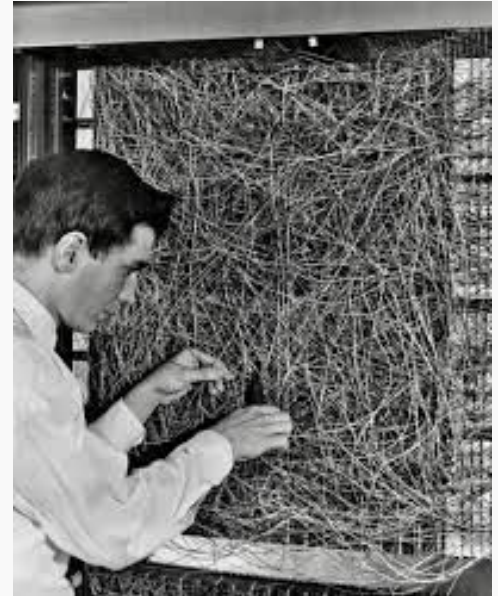


FIG. 2 — Organization of a perceptron.



Some ML History

- **1959:** Game-playing ML

Some Studies in Machine Learning Using the Game of Checkers

Arthur L. Samuel

Abstract: Two machine-learning procedures have been investigated in some detail using the game of checkers. Enough work has been done to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program. Furthermore, it can learn to do this in a remarkably short period of time (8 or 10 hours of machine-playing time) when given only the rules of the game, a sense of direction, and a redundant and incomplete list of parameters which are thought to have something to do with the game, but whose correct signs and relative weights are unknown and unspecified. The principles of machine learning verified by these experiments are, of course, applicable to many other situations.

Some ML History

- **1989:** Yann LeCun (now Chief AI Scientist @ Meta), Optical Character Recognition (OCR)

Handwritten Digit Recognition with a Back-Propagation Network

Y. Le Cun, B. Boser, J. S. Denker, D. Henderson,
R. E. Howard, W. Hubbard, and L. D. Jackel
AT&T Bell Laboratories, Holmdel, N. J. 07733

ABSTRACT

We present an application of back-propagation networks to handwritten digit recognition. Minimal preprocessing of the data was required, but architecture of the network was highly constrained and specifically designed for the task. The input of the network consists of normalized images of isolated digits. The method has 1% error rate and about a 9% reject rate on zipcode digits provided by the U.S. Postal Service.

Some ML History

- **1959:** Game-playing ML

Some Studies in Machine Learning Using the Game of Checkers

Arthur L. Samuel

Abstract: Two machine-learning procedures have been investigated in some detail using the game of checkers. Enough work has been done to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program. Furthermore, it can learn to do this in a remarkably short period of time (8 or 10 hours of machine-playing time) when given only the rules of the game, a sense of direction, and a redundant and incomplete list of parameters which are thought to have something to do with the game, but whose correct signs and relative weights are unknown and unspecified. The principles of machine learning verified by these experiments are, of course, applicable to many other situations.

Some ML History



■ 2016: AlphaGo

“defeated the human European Go champion by 5 games to 0.”

Less “feature-engineering”

Article | Published: 27 January 2016

Mastering the game of Go with deep neural networks and tree search

[David Silver](#) , [Aja Huang](#), [Chris J. Maddison](#), [Arthur Guez](#), [Laurent Sifre](#), [George van den Driessche](#), [Julian Schrittwieser](#), [Ioannis Antonoglou](#), [Veda Panneershelvam](#), [Marc Lanctot](#), [Sander Dieleman](#), [Dominik Grewe](#), [John Nham](#), [Nal Kalchbrenner](#), [Ilya Sutskever](#), [Timothy Lillicrap](#), [Madeleine Leach](#), [Koray Kavukcuoglu](#), [Thore Graepel](#) & [Demis Hassabis](#) 

[Nature](#) **529**, 484–489 (2016) | [Cite this article](#)

486k Accesses | **9009** Citations | **3066** Altmetric | [Metrics](#)

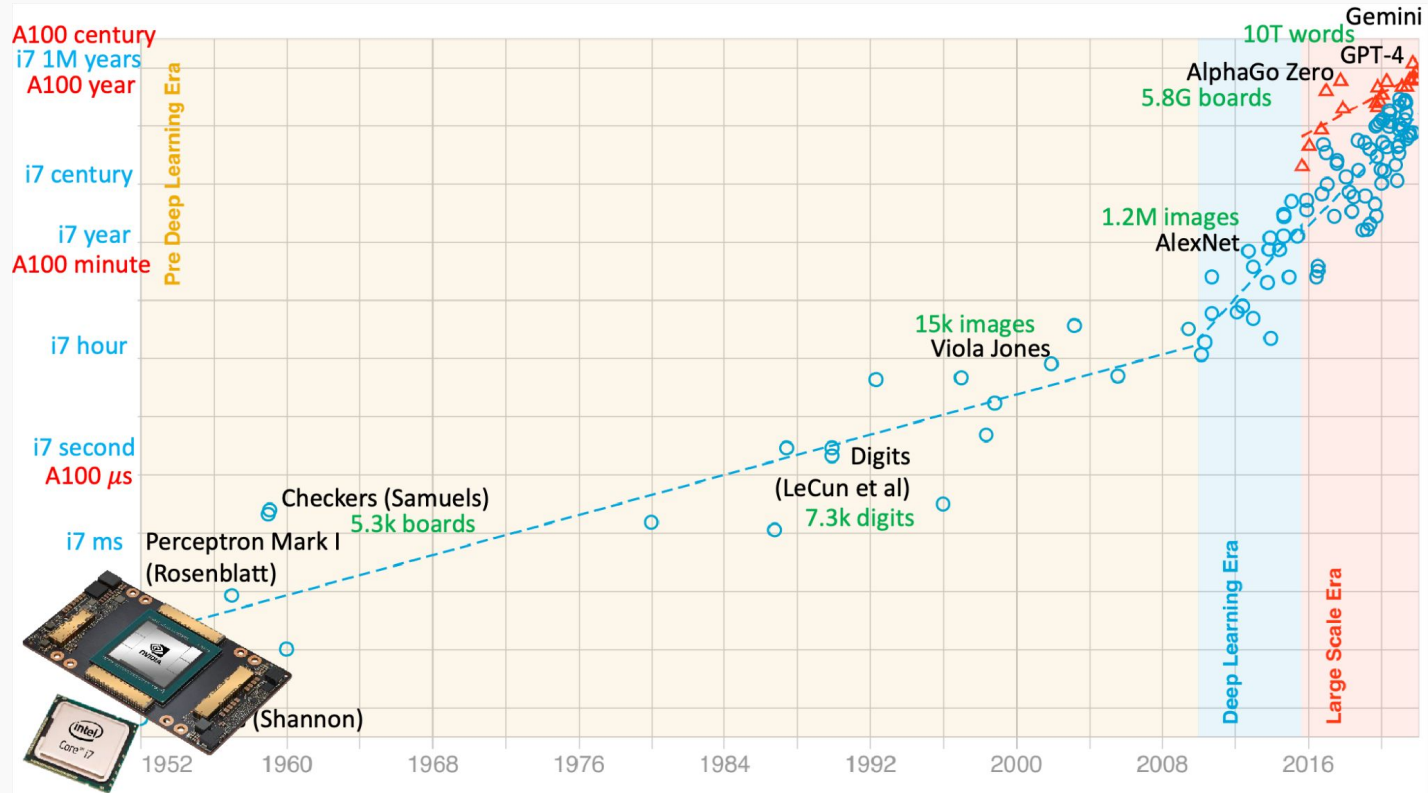
Abstract

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

Some ML History

- **Now:**
 - Chatbots: ChatGPT, Claude, Gemini
 - Image generation: DALL-E 2, Midjourney
 - Game-playing bots: Stockfish (Chess)
 - Face ID

Why... now?



Why... now?

- **2012: AlexNet**
 - Neural network to classify images
 - 60M parameters
 - Took a lot of compute, but used GPUs

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Why... now?

- Lots of compute (faster runtime with GPUs)
- Lots of data
- During AI Winter (1970~ ish): Did not have compute nor data

Punchline: Though the groundwork of modern machine learning (neural networks) has already been laid out during the second half of the 20th century, it was then infeasible due to the lack of computational power and data. But this is exactly what the 2010s-20s has.

A caveat: this is not obvious!

- **1964:** Weizenbaum's ELIZA
- Rules-based chatbot
- "ELIZA itself **examined the text for keywords**, applied values to said keywords, and transformed the input into an output"

```
Welcome to

EEEEEE LL      IIII  ZZZZZZ  AAAAA
EE      LL      II    ZZ    AA  AA
EEEEEE LL      II    ZZZ    AAAAAA
EE      LL      II    ZZ    AA  AA
EEEEEE LLLLLL  IIII  ZZZZZZ  AA  AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

A caveat: this is not obvious!

- **1984-present:** Lenat's CYC Knowledge Base
- “long-term artificial intelligence project that **aims to assemble a comprehensive ontology and knowledge base that spans the basic concepts and rules about how the world works**”
- Knowledge + (Logical) Inference Engine

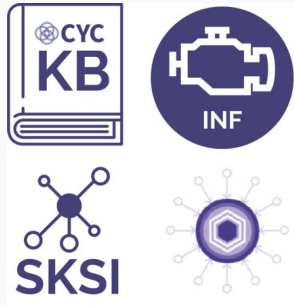
<https://cyc.com/platform/>

Knowledge base [\[edit \]](#)

The [knowledge base](#) is divided into *microtheories*. Unlike the knowledge base as a whole, each microtheory must be free from monotonic contradictions. Each microtheory is a first-class object in the Cyc ontology; it has a name that is a regular constant. The concept names in Cyc are CycL *terms* or *constants*.^[6] Constants start with an optional `#$` and are case-sensitive. There are constants for:

- Individual items known as *individuals*, such as `#$BillClinton` or `#$France`.
- *Collections*, such as `#$Tree-ThePlant` (containing all trees) or `#$EquivalenceRelation` (containing all [equivalence relations](#)). A member of a collection is called an *instance* of that collection.^[1]
- *Functions*, which produce new terms from given ones. For example, `#$FruitFn`, when provided with an argument describing a type (or collection) of plants, will return the collection of its fruits. By convention, function constants start with an upper-case letter and end with the string `Fn`.
- *Truth functions*, which can apply to one or more other concepts and return either true or false. For example, `#$siblings` is the sibling relationship, true if the two arguments are [siblings](#). By convention, truth function constants start with a lowercase letter.

For every instance of the collection `#$ChordataPhylum` (i.e., for every [chordate](#)), there exists a female animal (instance of `#$FemaleAnimal`), which is its mother (described by the predicate `#$biologicalMother`).^[1]



Rules v. Probabilistic/Statistical

- Seems like the probabilistic/statistical approach to capture “intelligence” is working a lot better (for now?)
- This was a divide at AI@50 in 2006, 50 years from the 1956 Dartmouth Conference.
- What's next?
 - Deeper and deeper networks, with more and more compute and data?
 - Or an alternative architecture/model?
- We'll start from the beginning and build up to here.

Next for us

- Meeting 1: What is Machine Learning (ML)? The goals of supervised learning.
- Meeting 2: Classification, Regression and the basic ML workflow.
- Meeting 3: Overfitting, regularization, validation.
- Meeting 4: Neural networks and the Multi-Layer Perceptron (MLP) part 1.

Time: 8 p.m. Thursday Week 4

Location: CSIL 1

Join us on:

Questions/curiosities? @ **Hung**, conghunglt@uchicago.edu.