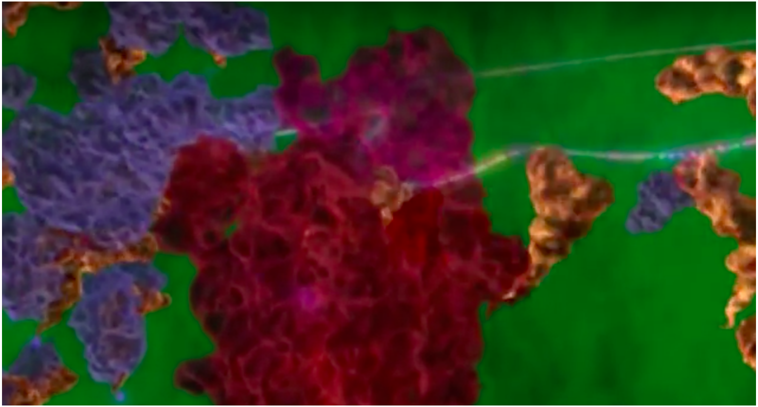
BIO LOGIC: Biological Computation

Kay Kirkpatrick, Math 490

2019

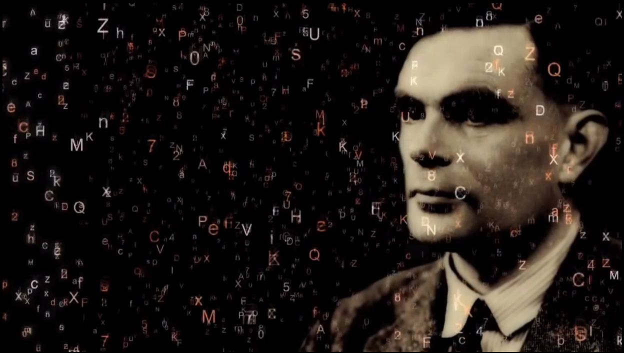
BIO LOGIC: Biology and Computation



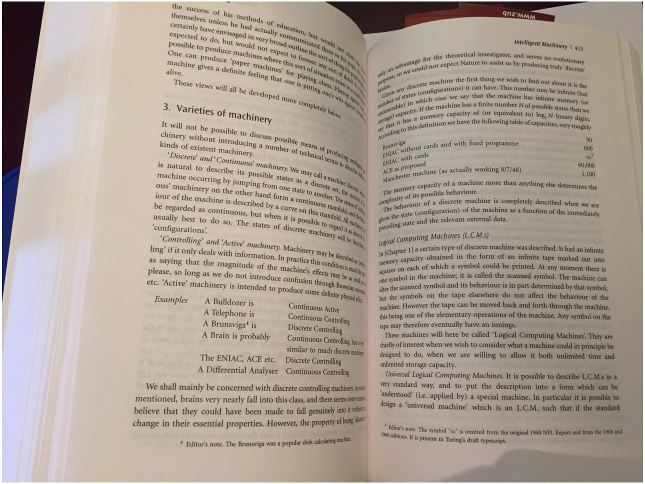
Kay Kirkpatrick, Math 490

Urbana, Miami and Mascouten lands

Movie still from the Yonath lab; next slide from Cambridge U.



Turing 1948: Intelligent Machinery


the success of his methods of education, but would not claim the results themselves unless he had actually communicated them to his pupil. He would certainly have envisaged in very broad outline the sort of thing his pupil might be expected to do, but would not expect to foresee any sort of detail. It is already possible to produce machines where this sort of situation arises in a small degree. One can produce 'paper machines' for playing chess. Playing against such a machine gives a definite feeling that one is pitting one's wits against something alive.

These views will all be developed more completely below.

3. Varieties of machinery

It will not be possible to discuss possible means of producing intelligent machinery without introducing a number of technical terms to describe different kinds of existent machinery.

'Discrete' and 'Continuous' machinery. We may call a machine 'discrete' when it is natural to describe its possible states as a discrete set, the motion of the machine occurring by jumping from one state to another. The states of 'continuous' machinery on the other hand form a continuous manifold, and the behaviour of the machine is described by a curve on this manifold. All machinery can be regarded as continuous, but when it is possible to regard it as discrete it is usually best to do so. The states of discrete machinery will be described as 'configurations'.

'Controlling' and 'Active' machinery. Machinery may be described as 'controlling' if it only deals with information. In practice this condition is much the same as saying that the magnitude of the machine's effects may be as small as we please, so long as we do not introduce confusion through Brownian movement, etc. 'Active' machinery is intended to produce some definite physical effect.

Examples	A Bulldozer is	Continuous Active
	A Telephone is	Continuous Controlling
	A Brunsviga[4] is	Discrete Controlling
	A Brain is probably	Continuous Controlling, but is very similar to much discrete machinery
	The ENIAC, ACE etc.	Discrete Controlling
	A Differential Analyser	Continuous Controlling

We shall mainly be concerned with discrete controlling machinery. As we have mentioned, brains very nearly fall into this class, and there seems every reason to believe that they could have been made to fall genuinely into it without any change in their essential properties. However, the property of being 'discrete' is

[4] Editor's note. The Brunsviga was a popular desk calculating machine.

only an advantage for the theoretical investigator, and serves no evolutionary purpose, so we could not expect Nature to assist us by producing truly 'discrete' brains.

Given any discrete machine the first thing we wish to find out about it is the number of states (configurations) it can have. This number may be infinite (but enumerable) in which case we say that the machine has infinite memory (or storage) capacity. If the machine has a finite number N of possible states then we say that it has a memory capacity of (or equivalent to) $\log_2 N$ binary digits. According to this definition we have the following table of capacities, very roughly

Brunsviga	90
ENIAC without cards and with fixed programme	600
ENIAC with cards	∞[5]
ACE as proposed	60,000
Manchester machine (as actually working 8/7/48)	1,100

The memory capacity of a machine more than anything else determines the complexity of its possible behaviour.

The behaviour of a discrete machine is completely described when we are given the state (configuration) of the machine as a function of the immediately preceding state and the relevant external data.

Logical Computing Machines (L.C.M.s)

In [Chapter 1] a certain type of discrete machine was described. It had an infinite memory capacity obtained in the form of an infinite tape marked out into squares on each of which a symbol could be printed. At any moment there is one symbol in the machine; it is called the scanned symbol. The machine can alter the scanned symbol and its behaviour is in part determined by that symbol, but the symbols on the tape elsewhere do not affect the behaviour of the machine. However the tape can be moved back and forth through the machine, this being one of the elementary operations of the machine. Any symbol on the tape may therefore eventually have an innings.

These machines will here be called 'Logical Computing Machines'. They are chiefly of interest when we wish to consider what a machine could in principle be designed to do, when we are willing to allow it both unlimited time and unlimited storage capacity.

Universal Logical Computing Machines. It is possible to describe L.C.M.s in a very standard way, and to put the description into a form which can be 'understood' (i.e. applied by) a special machine. In particular it is possible to design a 'universal machine' which is an L.C.M. such that if the standard

[5] Editor's note. The symbol '∞' is omitted from the original 1948 NPL Report and from the 1968 and 1969 editions. It is present in Turing's draft typescript.

Courtesy Copeland, Complete Turing

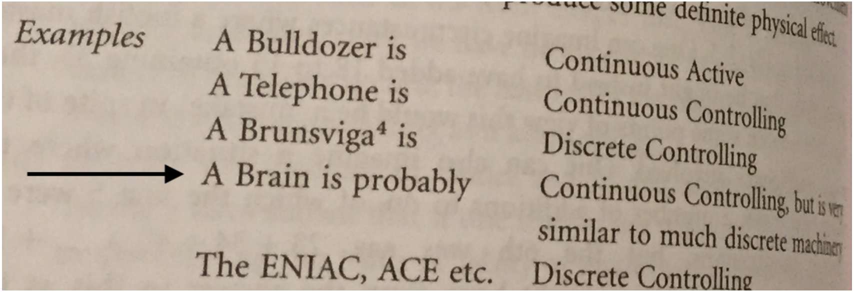
Turing 1948: Intelligent Machinery zoom-in

inery will be described

'Controlling' and 'Active' machinery. Machinery may be described as 'contro-ling' if it only deals with information. In practice this condition is much the same as saying that the magnitude of the machine's effects may be as small as we please, so long as we do not introduce confusion through Brownian movemen etc. 'Active' machinery is intended to produce some definite physical effect.

Examples	A Bulldozer is	Continuous Active
	A Telephone is	Continuous Controlling
	A Brunsviga[4] is	Discrete Controlling
	A Brain is probably	Continuous Controlling, but is very similar to much discrete machinery
	The ENIAC, ACE etc.	Discrete Controlling
	A Differential Analyser	Continuous Controlling

Turing's Mistake



some definite physical effect.

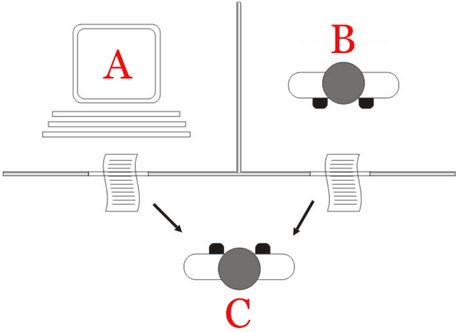
Examples	A Bulldozer is	Continuous Active
	A Telephone is	Continuous Controlling
	A Brunsviga[4] is	Discrete Controlling
	A Brain is probably	Continuous Controlling, but is very similar to much discrete machinery
	The ENIAC, ACE etc.	Discrete Controlling

The brain processes information AND is active:

ions, neurotransmitters, neuromodulators, hormones, remodeling synapses, movement, fields.

His 1950 Imitation Game relies on this mistake

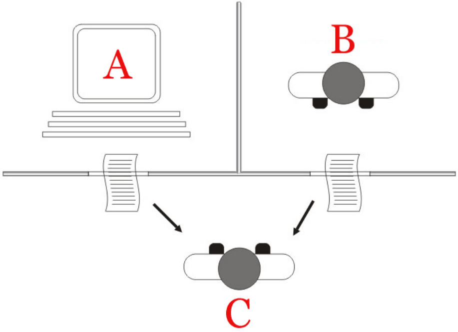
courtesy wikipedia



Turing Test involves just info input & info output.

His 1950 Imitation Game relies on this mistake

courtesy wikipedia



Turing Test involves just info input & info output.

Same with the Chinese Room Argument objection.

Need new: Active + Information Machines that...

Process info AND produce definite physical effects

Examples: Brains, organs, robots, ribosomes ...

Need new: Active + Information Machines that...

Process info AND produce definite physical effects

Examples: Brains, organs, robots, ribosomes ...

Computers? (require user)

Artificial neural networks? (some act)

A + I Machines may be better than Turing machines:

Brains, etc., compute and act simultaneously.

Robots have a communication bottleneck.

A + I Machines may be better than Turing machines:

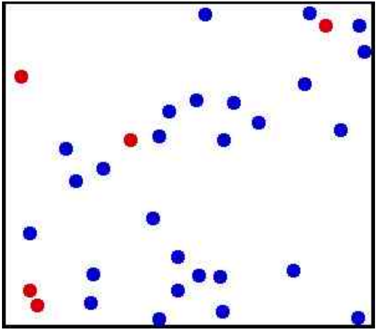
Brains, etc., compute and act simultaneously.

Robots have a communication bottleneck.

Statistical mechanics might help us figure out the brain

Statistical mechanics might help us figure out the brain

microscopic first principles $\rightsquigarrow$ zoom out $\rightsquigarrow$ MACROSCOPIC STATES



Courtesy Greg L and Digital Vision/Getty Images

Outline

1. What is the microscopic unit of computation?
2. Unit properties: information & computation.
3. Consequences of this new approach.

What are microscopic computational units in the brain?

Neurons are mesoscopic & complicated.

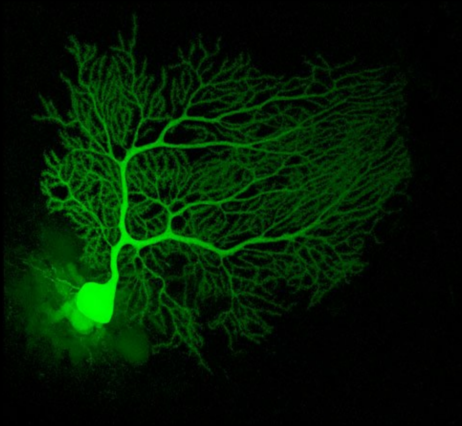
McCulloch-Pitts and von Neumann claimed:
neuron $\simeq$ vacuum tube (wrong)

What are microscopic computational units in the brain?

Neurons are mesoscopic & complicated.

McCulloch-Pitts and von Neumann claimed:
neuron $\simeq$ vacuum tube (wrong)

If they were right, the iPhone could model a brain.



S. Grant: a synapse is a computer $>$ vacuum tube

So a neuron $\geq 10^4$ synapses $\gg$ vacuum tube

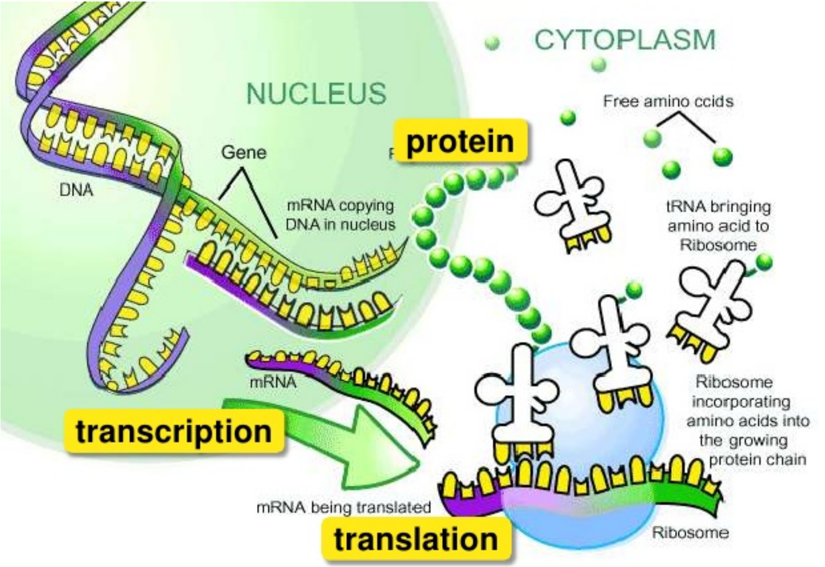
Instead?

S. Grant: a synapse is a computer $>$ vacuum tube

So a neuron $\geq 10^4$ synapses $\gg$ vacuum tube

Instead? Proteins have similar complexity to vacuum tubes, and ribosomes help make proteins.

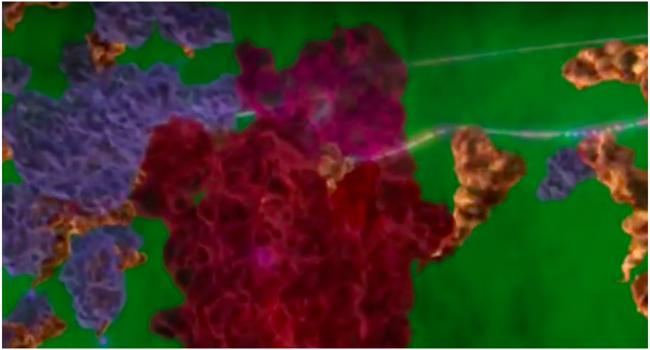
Ribosomes synthesize proteins from mRNA genes



Courtesy Jay Swan

How to model a ribosome mathematically? (Take 1)

Detailed: Molecular Dynamics. But movie:



Reductive: TASEP. No information. But theory.

How to model a ribosome, Take 2

Caetano-Anollés & Caetano-Anollés 2015:
As a Universal Turing Machine in the cell.

As a finite-state automaton.

How to model a ribosome, Take 2

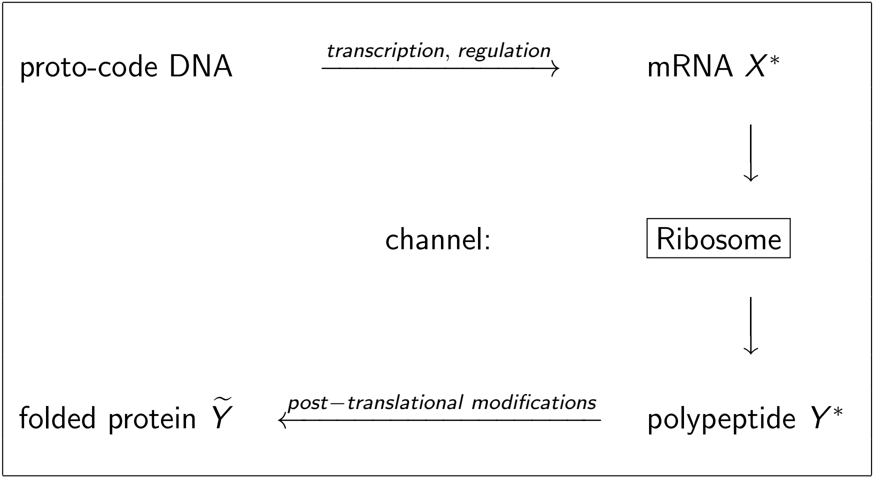
Caetano-Anollés & Caetano-Anollés 2015:
As a Universal Turing Machine in the cell.

As a finite-state automaton.

Reductive and info/computational, not active.

AIM above pure info, but first info details...

The ribosome as an information channel



The ribosome operates under its Shannon capacity

Theorem (Inafuku-K.-Osuagwu 2019):
Ribosome's Shannon capacity is ≥ 266 bits per second. Cf: observed rates $\simeq 24 to 120$ bits per second.

Proof uses the uniform distribution.

This is why ribosomes are $\sim 99.99\%$ accurate.

Enzymes operate under their Shannon capacities

Theorems (D. Inafuku 2019): Theoretical capacities & observed rates (units are bits/sec):

Enzyme name	**Cap. $\geq$**	**Actual rate**
DNA polymerase	$\mathbf{1554.2}$	1498
RNA II polymerase	$\mathbf{191.9}$	160
KcsA ion channel	$\mathbf{3 * 10^7}$	$2 * 10^7$

Notation to get beyond pure computation

Input mRNA tape X^* is string of codons in $\Sigma := \{A, C, G, U\}^3$

$\Sigma^* := \{$words with letters in Σ, incl. empty word $\epsilon\}$.

Notation to get beyond pure computation

Input mRNA tape X^* is string of codons in $\Sigma := \{A, C, G, U\}^3$

$$\Sigma^* := \{\text{words with letters in } \Sigma, \text{ incl. empty word } \epsilon\}.$$

Output polypeptide Y^* is a string of **physical blocks** in

$$\Delta^* := \{\text{strings of blocks in } \Delta := \{b_1, b_2, \ldots, b_{20}\}\}.$$

Notation to get beyond pure computation

Input mRNA tape X^* is string of codons in $\Sigma := \{A, C, G, U\}^3$

$$\Sigma^* := \{\text{words with letters in } \Sigma, \text{ incl. empty word } \epsilon\}.$$

Output polypeptide Y^* is a string of **physical blocks** in

$$\Delta^* := \{\text{strings of blocks in } \Delta := \{b_1, b_2, \ldots, b_{20}\}\}.$$

Y^* contains info and is also a physico-chemical structure.

Turing defined a 2-tape automatic a-machine

Consists of six parts: $M := \langle Q, \Sigma, \Delta, \delta, q_0, F \rangle$

- State space Q
- Σ input alphabet, and input program $X^* \in \Sigma^*$
- Δ output symbols, and output **tape** Y^*

Turing defined a 2-tape automatic a-machine

Consists of six parts: $M := \langle Q, \Sigma, \Delta, \delta, q_0, F\rangle$

- State space Q
- Σ input alphabet, and input program $X^* \in \Sigma^*$
- Δ output symbols, and output **tape** Y^*

- Initial state q_0
- Final/accepting states $F \subset Q$
- Transition function (rules about updating state/tape)
  $\delta : (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \Delta \times \{L, R\}$

We define an automatic biochemical abc-machine

An **abc-machine** $M_{T,A} = \langle Q, \Sigma, \Delta, \delta, q_0, F, T, A \rangle$ has:

- Q, Σ, q_0, F as before; input program $X^* \in \Sigma^*$
- State space Q, with initial state q_0, final states $F \subset Q$
- Transitions $\delta : (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \Delta \times \{L, R\}$

We define an automatic biochemical abc-machine

An **abc-machine** $M_{T,A} = \langle Q, \Sigma, \Delta, \delta, q_0, F, T, A \rangle$ has:

- Q, Σ, q_0, F as before; input program $X^* \in \Sigma^*$
- State space Q, with initial state q_0, final states $F \subset Q$
- Transitions $\delta : (Q \setminus F) \times \Sigma \to Q \times \Sigma \times \Delta \times \{L, R\}$

- Δ output **blocks**, and Y^* is **block-string** output
- **T** a random stopping time, when program X^* **halts**
- **A** an auxiliary machine, optimizing a bio-chemical energy functional, e.g., protein-folding

Diagram for the ribosome as an abc-machine $M_{T,A}$

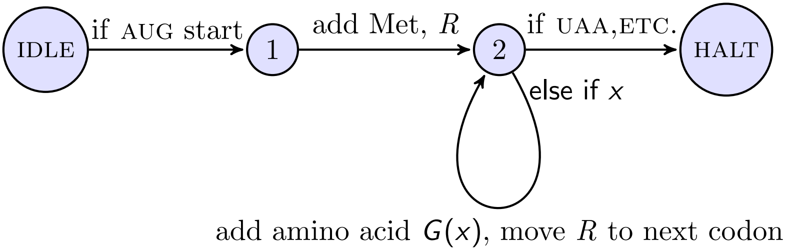


Figure: The (basic!) transition function δ diagram for a ribosome.

If the RNA degrades at time $\mathbf{T}(X^*, \omega)$, transition to HALT.

Auxiliary machine **A** folds the polypeptide: $Y^* \rightsquigarrow \widetilde{Y}$

The ribosome is more than just a Turing Machine (TM)

Observation (K.-Osuagwu 2018): $M_{T,A}$ is equivalent to a 2-tape TM with 2 oracles: protein-folding machine $\mathbf{A}$ for output $\widetilde{Y}$, and stopping time $\{t < \mathbf{T}\}$.

The ribosome $M_{T,A}$ is an A+I machine.

The ribosome is more than just a Turing Machine (TM)

Observation (K.-Osuagwu 2018): $M_{T,A}$ is equivalent to a 2-tape TM with 2 oracles: protein-folding machine $\mathbf{A}$ for output $\widetilde{Y}$, and stopping time $\{t < \mathbf{T}\}$.

The ribosome $M_{T,A}$ is an A+I machine.

Protein product $\widetilde{Y}$ is an A+I machine too.

(Modified input pieces can act too: e.g., ATP, GTP.)

Better than TMs on Decidability/Computability

Abc-machines avoid the Halting Problem by stopping at **T**.

Better than TMs on Decidability/Computability

Abc-machines avoid the Halting Problem by stopping at **T**.

A non-deterministic Turing Machine (NDTM) can approx. but not simulate **T**, which is truly random & non-computable.

Abc-machines are possibly higher complexity than TMs

If protein folder **A** is NP-complete, then it is in class **NP**.

Self-avoiding random walk model of **A** is **NP**-complete
(Berger and Leighton 1998)

Consequences for the brain

Individual computational units are more than TMs.

Consequences for the brain

Individual computational units are more than TMs.

Conjecture: Brain is more than a TM. Embodiment.

When can we model the whole brain? (Weak AI)

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^6$ ribosomes and $\sim 10^7$ proteins.

When can we model the whole brain? (Weak AI)

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^{6}$ ribosomes and $\sim 10^{7}$ proteins.

Total: $\sim 10^{18}$ units, ea. ~ 100 binary ops per sec.

When can we model the whole brain? (Weak AI)

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^{6}$ ribosomes and $\sim 10^{7}$ proteins.

Total: $\sim 10^{18}$ units, ea. ~ 100 binary ops per sec.

Need zettascale (10^{21} flops) or yottascale (10^{24}) computing, expected around 2030 and 2040.

When can we model the whole brain? (Weak AI)

The brain has $\sim 10^{11}$ cells (neurons and glia).

Each cell has $\sim 10^{6}$ ribosomes and $\sim 10^{7}$ proteins.

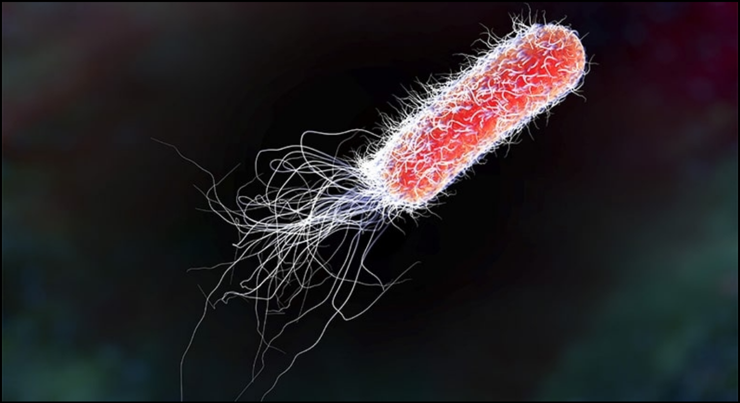
Total: $\sim 10^{18}$ units, ea. ~ 100 binary ops per sec.

Need zettascale (10^{21} flops) or yottascale (10^{24}) computing, expected around 2030 and 2040.

2-week weather modeling may require zettascale.

Why are artificial “neural networks” successful?

Maybe not because they’re like brains, but like something else...

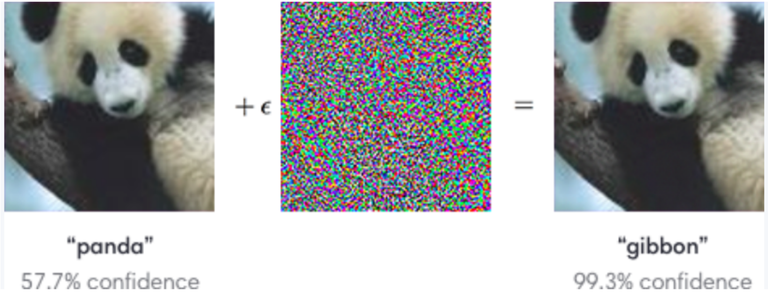


Maybe artificial neural networks are successful...

Because they're like biochemical networks in E. coli, which can basically do Stochastic Gradient Descent.

But artificial neural networks fail spectacularly

An adversarial example:



Courtesy Open AI

Key takeaway: “Neural network” is a misnomer

Nodes (neurones) are simpler than real neurons.

Better: “Biologically inspired” networks.

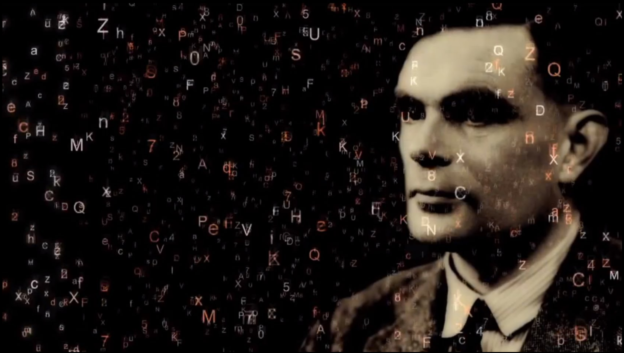
More takeaways

Biological computation is more sophisticated than we thought.

More takeaways

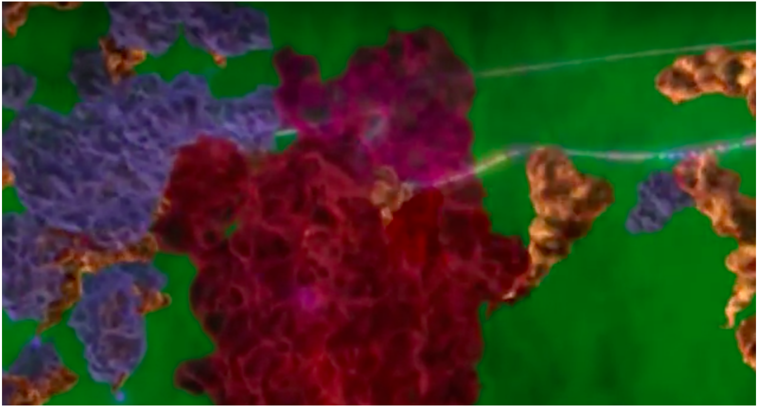
Biological computation is more sophisticated than we thought.

True AI may be harder than we thought.



Thanks

NSF CAREER Award DMS-1254791, Simons Sabbatical Fellowship



faculty.math.illinois.edu/∼kkirkpat/

What about DNA computing?