A Sample of Ethical Issues in AI (for Clinical Translational Seminar)

Daniel Rosiak

Case Western Reserve University

March, 08 2022

Outline of Contents

Intro

There are lots of powerful applications of AI today, including

- solving complex logistics problems that combine packing, routing, and scheduling problems
- accelerating clinical trials
- getting cars to drive themselves
- "deep fakes"
- analysis of medical images for cancer detection
- insight into group dynamics via social network analysis and sentiment classification
- determining when fruits are ripe and picking them

While it's exciting, there are many ethical risks as well.

Ethical Risk

Ethical risks can be broadly defined to include any choices made, or features of a system, that may cause substantial harm to persons or other entities with a moral status (such as animals, the environment, democratic institutions), or that may lead to considerable controversy or dispute for other reasons.

A few prominent examples of AI systems with notable ethical risk:

- **PredPol** (predictive policing): PredPol is an ML program for police departments that predicts hotspots where future crime might occur and helps determine how to distribute police presence. It has been shown to exhibit bias towards selecting low-income neighborhoods and locations with higher minority concentration (Temming, 2017). This leads to increased police presence in these areas, and by extension more recognized crime reports and active responses from these areas.
- **suicide prediction**: using some metadata (such as health records, billing data), a number of different research teams have used ML to predict suicide within the next 2 years (sometimes with shockingly high accuracy!)
- **AdFisher** (gender biases in Google search): more on this in a few slides

External vs. Internal Ethical Risks

When people think of ethical issues in AI, they often think of these sorts of "external" issues, which largely present issues on account of how they are *deployed* or how they might affect those who are impacted by their conclusions.

While there is of course overlap, these sorts of issues could be distinguished from **internal issues**, where this gets at core features of the AI models and algorithms themselves (prior to, or largely independent of, their use).

Continuity with non-AI tech

Especially with external issues (or issues of use), there seems to be a lot of continuity with what we can observe with many technologies, beyond algorithms and AI. But there is also continuity at the level of design and constitutive features. For instance,

- Winner (1980) describes how the extraordinarily low-hanging overpasses on Long Island were deliberately designed to achieve the effect of discouraging the presence of buses, which had the effect of limiting the entry of lower-income individuals using public transit
- Recently, more attention has been given to the safety disparities in the event of a car accident: the odds of a severe injury or death is 73 percent higher for women than for men, stemming mostly from the way car seats and seatbelts are designed (Forman et. al, 2019)

Replication of **(societal) bias** (and intensification of discrimination) by AI algorithms is an example of an issue that seems not to be unique to AI, but it is an issue that has garnered a lot of attention.

Replication of Societal Biases

There are a variety of ways machine learning (ML) systems can codify, automate, and exacerbate societal biases. Such systems reproducing already existing biases is something that often arises on account of using small or non-representative data sets for training, or through historical bias in the training data itself (some examples below).

There are numerous high-profile accounts of such cases, for example when gender biases in recruitment are replicated through the use of machine learning or when racial biases are perpetuated through machine learning in probation processes.

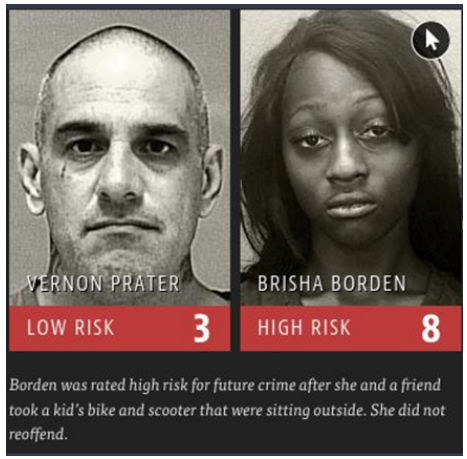
Historical Bias

In cases where little training data is available, it is generally difficult to form a training data set that accurately represents the population. Such training data commonly have **historical bias**, or bias created by selective targeting over a period of time.

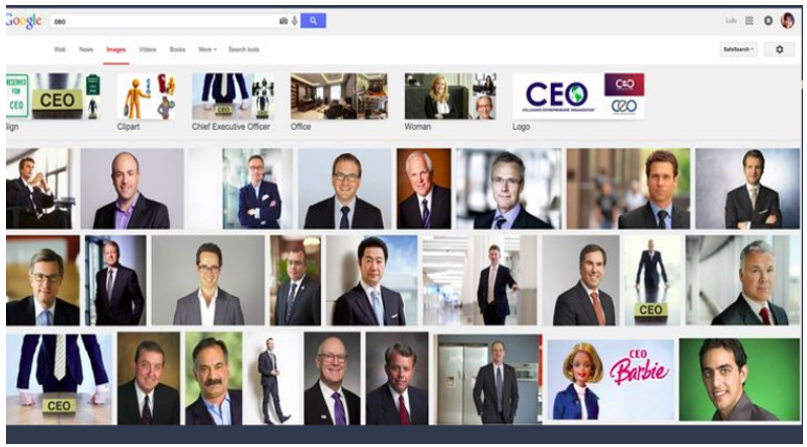
This problem frequently arises in ML implementations in the field of criminal justice, namely due to historical discrimination against minorities.

A notorious example of (racial) bias is provided by the **COMPAS** system, an ML risk assessment algorithm used to predict reoffending risk in convicted criminals (first used in legal courts by the state of Wisconsin). To train COMPAS, it is given a large set of crime reports as training data. The racial biases exhibited by COMPAS are likely learned from historical biases within the crime reports themselves, such as a disproportionate number of the reports being from low-income neighborhoods.

COMPAS frequently demonstrated a human-like bias towards race, wrongly predicting "that black defendants would reoffend nearly twice as often as it made that wrong prediction for whites" (Temming, 2017).



Another example comes from researchers at Carnegie Mellon, using a tool they called **AdFisher**, which revealed that simulated male users of Google were six times more likely than females to see Google ads for high paying jobs (i.e., were shown online ads promising to help get jobs paying more than 200k).



Ideas for Defending against such biases

1. making data sets and models come with annotations: declarations of provenance, security, conformity, and fitness for use.
2. de-bias the data: over-sample from minority classes, for instance, to defend against "sample size disparity" (where this refers to how, in most data sets there will be fewer training examples of minority class individuals than of majority class individuals – ML algos give better accuracy with more training data, so that means members of minority classes generally can expect to experience lower accuracy)
3. invent new ML models and algos that are more resistant to bias
4. let a system make initial recommendations that may be biased, but then train a second system to de-bias the recommendations of the first one

Beyond Bias

Examples of bias garner a lot of attention in the media and academy.

But one could argue that problems of bias are comparatively simple (it's often simply a matter of "garbage in-garbage out") and not unique to AI systems (arguably the same sorts of biases and discrimination can crop up in the design or implementation of other tech).

Moreover, there are a variety of important internal issues in AI (unrelated to bias) that often fly under the radar, especially among the wider public of non-experts, yet that seem to present equally risky ethical challenges.

Today we'll focus on a few of the most notable issues/risks/challenges – of potential ethical import – internal to AI models and algorithms.

Some Internal Features/Risks/Challenges that have Ethical Import

- opacity vs. transparency
- rigidity, goal blindness, and misalignment
- difficulty generalizing / brittleness
- quantifying uncertainty
- bad memory

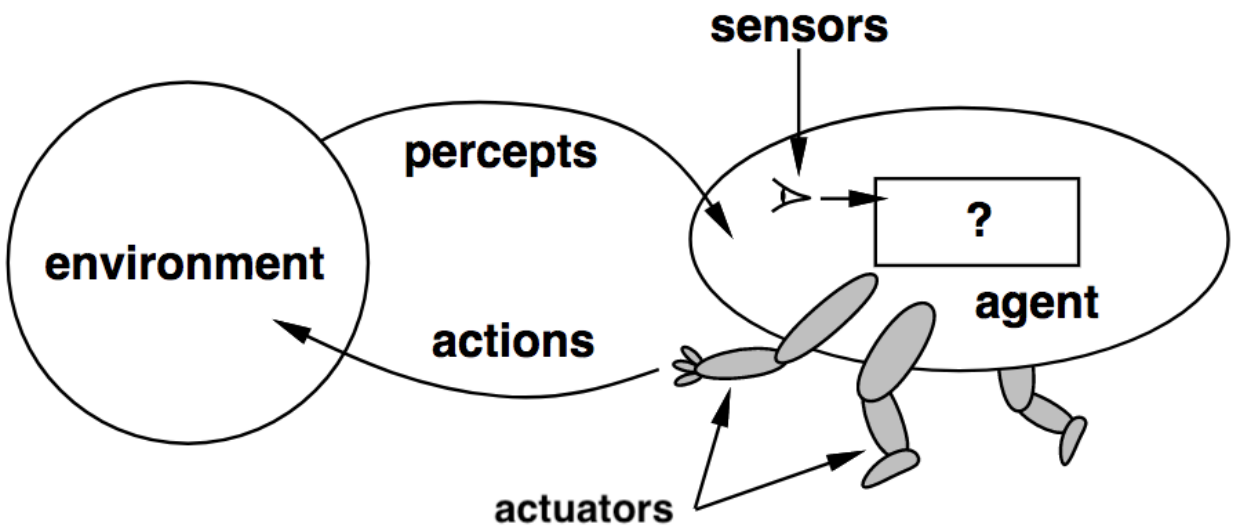
Brief Background on AI/ML

Background on Intelligent Agents

AI is fundamentally about building **intelligent agents**, where

Definition

An **agent** is anything that can be viewed as perceiving its *environment* through *sensors* and that acts upon that environment through *actuators*.



Background on Intelligent Agents

For instance,

- A human agent has eyes, ears, and other organs for its sensors, and has hands, legs, vocal tract, etc. for actuators
- A robotic agent may have cameras, infrared range finders, accelerometers, etc. for sensors, and motors, robotic arms, etc. for actuators
- A software agent may have file contents, network packets, and other user input (from keyboard/mouse/touchscreen/voice) as sensory inputs, and may act on its environment by writing files, sending network packets, displaying information, generating sounds, etc.

Background on Intelligent Agents

The world around us is accordingly full of agents – such as, thermostats, cellphones, and human beings.

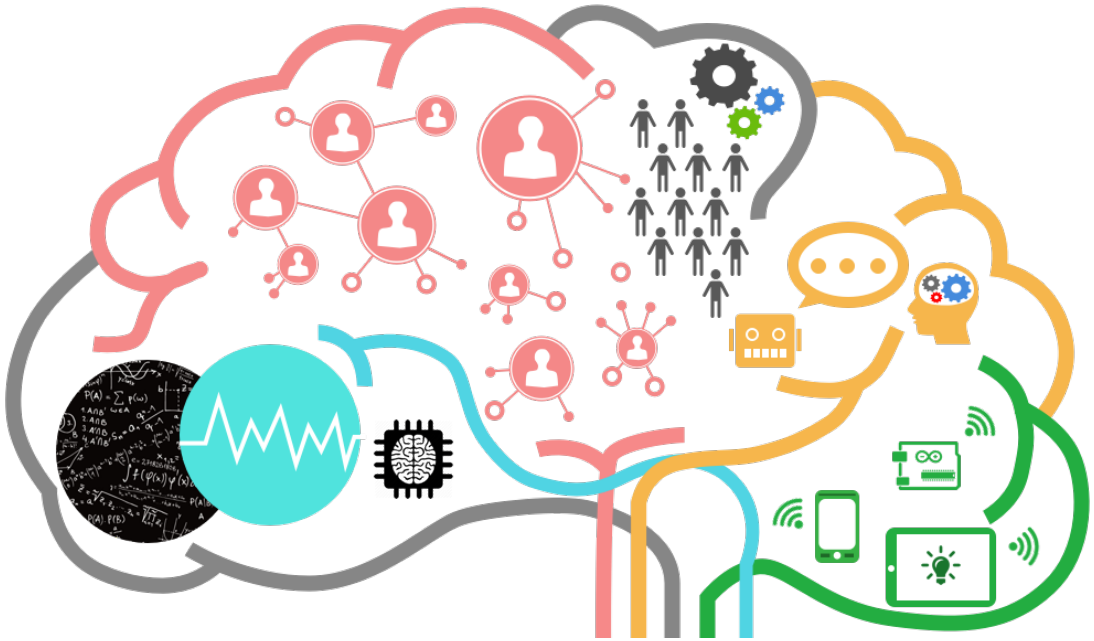


Image from Intelligent Agents Lab

Background on Intelligent Agents

Question: What makes an agent intelligent?

Background on Intelligent Agents

Well, minimally, we would like to construct an agent that is rational, where what is **rational** at any given time arguably depends on four things:

- the performance measure that defines the criterion of success
- the agent's prior knowledge of the environment
- the actions that are available to the agent
- the agent's percept sequence to date

Background on Intelligent Agents

Definition

A **rational agent** is an agent that, for each possible percept sequence (i.e., the complete history of everything the agent has ever perceived), selects an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has. (AIMA, 40)

Background on Intelligent Agents

As a practical matter, observe that rationality does not require omniscience and/or perfection.
However, it often will require

- **information gathering**: doing actions in order to modify future percepts (as in cases of exploring an environment)
- **learning**: an agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented
- **some degree of autonomy**: it should learn what it can to compensate for partial or incorrect prior knowledge; while an agent may have significant assistance from the prior knowledge given to it by a designer, after sufficient experience of its environment, the behavior of a rational agent can become effectively *independent* of its prior knowledge

More on Learning

Definition

An agent is said to be **learning** if it improves its performance after making observations about the world/its environment.

- Learning can range from the trivial and mundane, such as jotting down and updating a shopping list, to the profound, such as when Einstein inferred a new theory of the universe
- When the agent doing the learning is a computer – or when a chief component of the functioning of an agent program is that it learns – we call it **machine learning** (or **ML** for short): here, a computer
  - observes some data,
  - builds a **model** based on that data, and
  - uses that model as both a **hypothesis** about the world and a piece of **software** that can solve problems

Motivation

But *why* would we want a machine to learn? Why not just program it the right way to begin with?
There are two main reasons:

- **First**: the designers cannot anticipate all possible future situations – for instance, a robot designed to navigate mazes must learn the layout of each new maze it encounters
- **Second**: sometimes the designers have no idea how to program a solution themselves – for instance, while most people are good at recognizing the faces of family members, doing it subconsciously, the implicitness of this activity is such that even the best programmers won't know how to program a computer to accomplish that task (except by using ML algorithms).

General Observation

- The technology behind machine learning (ML) has become a standard part of software engineering.
- Today, any time you are building a software system, even if you don't think of it as an AI agent, components of the system can potentially be improved with ML – for example, older software to analyze images of galaxies under gravitational lensing was sped up by a factor of 10 million with a machine-learning model.
- ML is currently the backbone of cutting-edge AI applications

Forms of Machine Learning

There are **three main types of machine learning**, stemming ultimately from the three types of **feedback** that can accompany the inputs:

1. **Supervised Learning**: the agent observes input-output pairs and learns a function that maps from input to output.
   - For example, the inputs could be camera images, each one accompanied by an output saying "bus" or "pedestrian," etc. An output like this is called a **label**. Then the agent learns a function that, when given a new image, predicts the appropriate label.
   - In the case of braking actions, an input would be the current state (involving the speed and direction of the car, road conditions, etc.), and an output is the distance it took to stop. In this case, a set of output values can be obtained by the agent from its own percepts, after the fact – the environment is the teacher, where the agent is learning a function that maps states to stopping distances.
2. **Unsupervised Learning**: the agent learns patterns in the input without any explicit feedback.
   - The most common unsupervised learning task is **clustering**: detecting potentially useful clusters of input examples – e.g., shown millions of images taken from the internet, a computer vision system learns to identify a large cluster of similar images, each of which an English speaker would call "cats."
3. **Reinforcement Learning**: the agent learns from a series of reinforcements, i.e., rewards and punishments.
   - For example, at the end of a chess game, the agent may be informed that it has won (reward) or lost (a punishment). It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it, and then to alter its actions to aim towards actions that would obtain more rewards in the future.

Unsupervised Learning as gateway to Deep Learning

Unsupervised learning consists of the suite of techniques and algorithms used to find underlying patterns in data **without the need for humans to provide labeled (tagged) data**.

- The underlying aspiration of this approach is that learning will "emerge" through something resembling *mimicry*, where patterns are pulled out from probability densities and the method comes to self-organize around these emergent patterns
- As such, it will often find subgroups or hidden patterns within the dataset that a human may not see
- The two main subcategories of unsupervised learning are **neural networks** (inspired by the functioning of biological networks of neurons in the brain) and the **probabilistic methods**
- And neural networks are themselves the workhorse of another very important area: **deep learning**

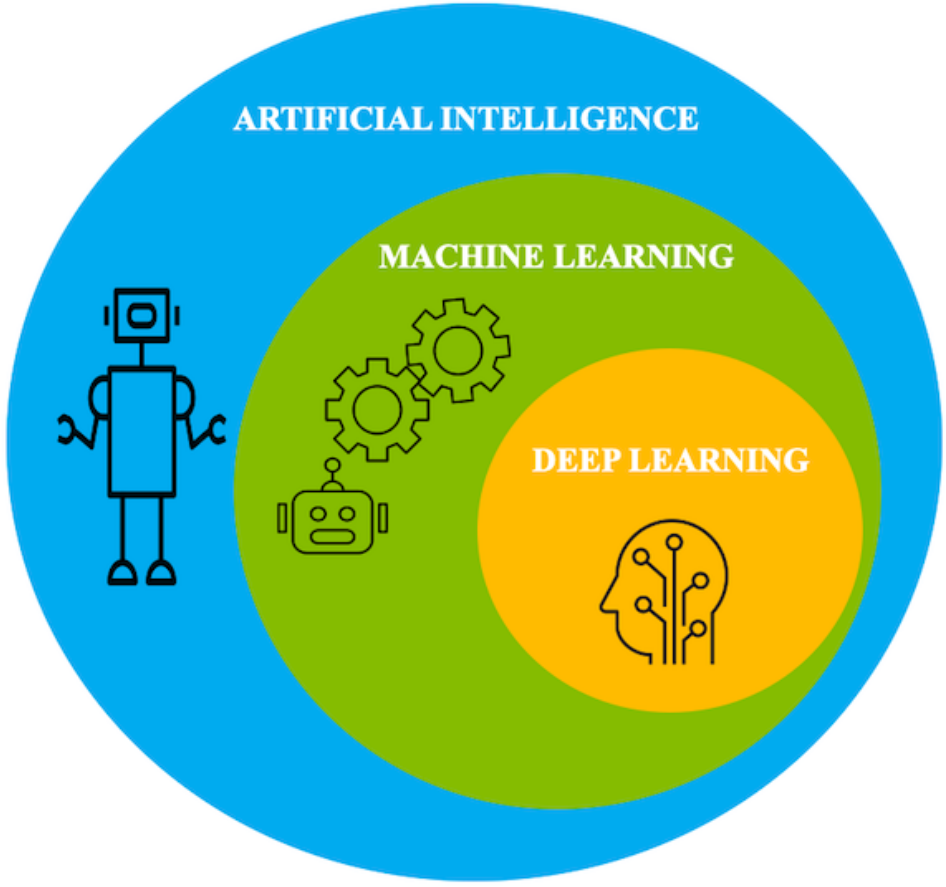
Deep Learning

Deep learning is a very important area of AI these days. The ability of deep learning to automate most of the process of *feature extraction* – eliminating most of the manual human intervention – derives a lot of its applicability from the fact that it works well on unstructured data, particularly as 80-90 percent of an organization's data is estimated to be unstructured (source: IBM).

While deep learning *can* leverage labeled datasets (i.e., fit into the supervised learning paradigm), it does not usually require a labeled dataset and is content to take in unstructured data (hence its general alignment with unsupervised learning, even though it really cuts across these two areas).

In sum,

- **Artificial Intelligence**: any technique enabling computers to mimic "intelligent" behavior
- **Machine learning**: any such technique where there is an ability to learn without being explicitly programmed
- **Deep learning**: learning such that patterns are extracted from data using neural networks


ARTIFICIAL INTELLIGENCE
MACHINE LEARNING
DEEP LEARNING

There are many other AI techniques and classes of problems – like search problems, using things like genetic algorithms – that present interesting ethical considerations but do not really fall under the heading of machine learning or deep learning.

Instead, in the interests of space, I'll tease out a few prominent internal issues by focusing on issues as they arise in the context of deep learning (via neural networks) on the one hand and reinforcement learning on the other.

Deep Learning: Neural Networks

Learning in general

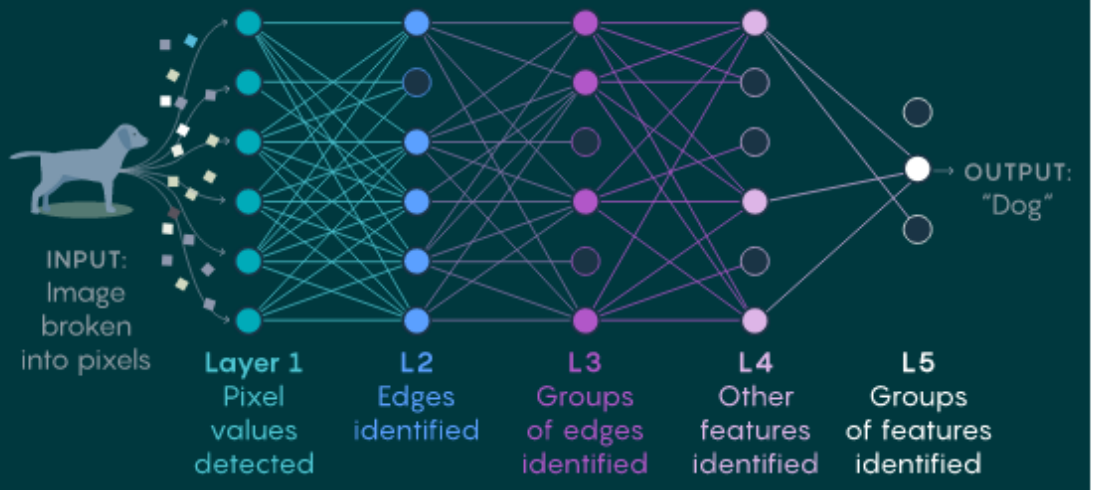
Definition

Deep learning is a broad class of techniques for machine learning, based ultimately on neural networks, in which

- the word 'deep' refers to the fact that the underlying network is typically organized into many layers (which means that computation paths from inputs to outputs have many sub-steps)
- these multiple layers are used to progressively extract higher-level features from raw input, having broken it up into many lower parts/layers
- the learning can be supervised, semi-supervised, or unsupervised

Perception in Neural Networks

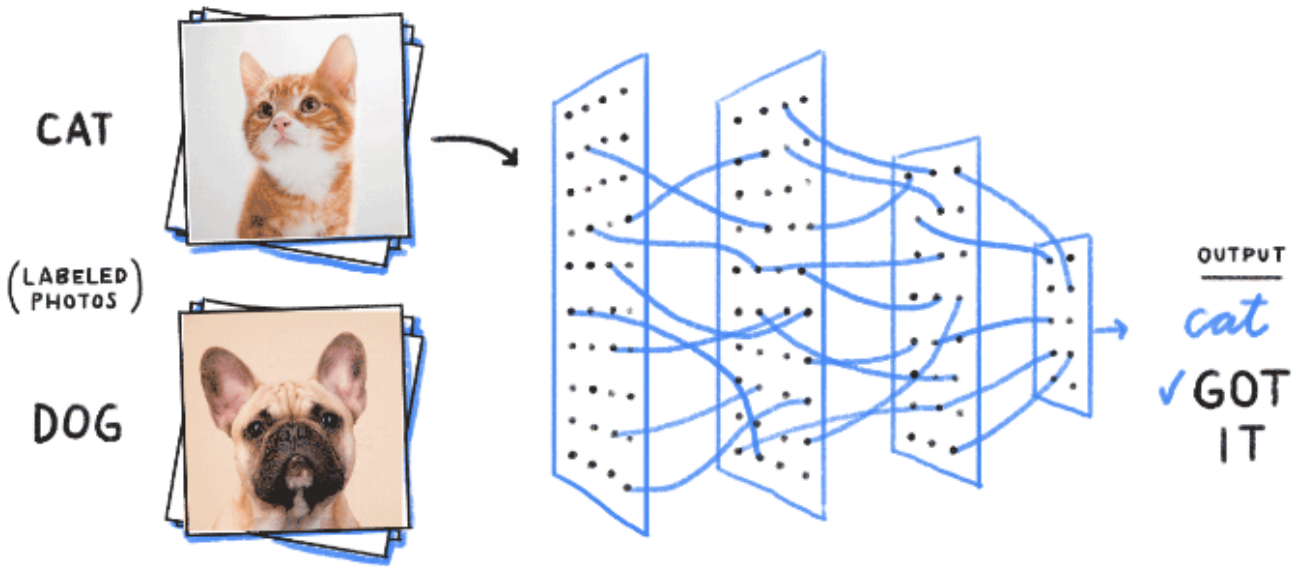
Neural networks pass an input, like an image, through multiple layers of digital neurons. Each layer can perform a "convolution" on the data to reveal a different feature of the input. The more layers between the input and the output, the deeper the network is.

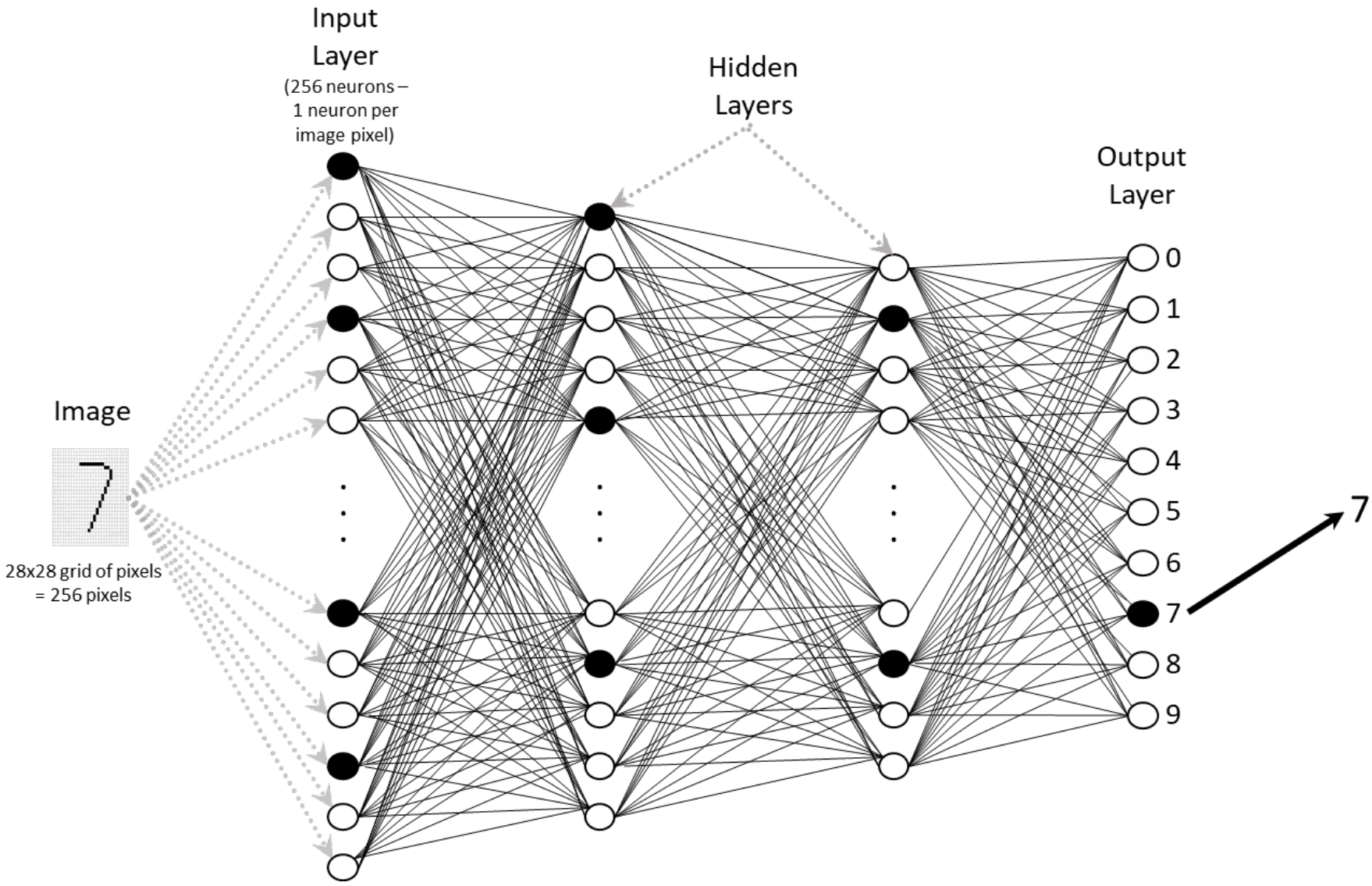


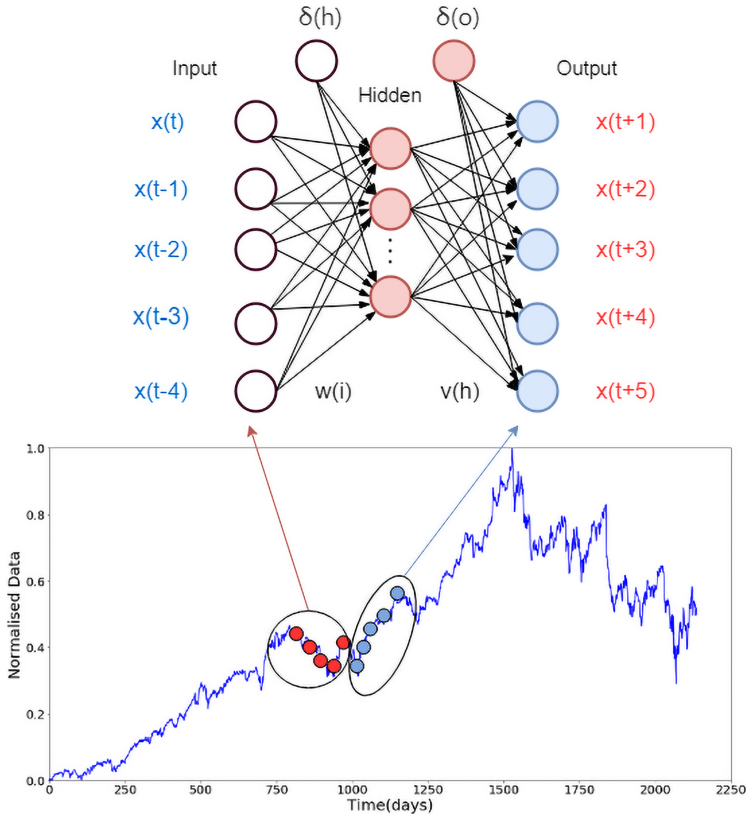
Research shows that deep nets often work best when early layers identify simple features and later layers identify increasingly complex ones. The brain perceives in a similar way.

—

Lucy Reading-Ikkanda/Samuel Velasco/Quanta Magazine


CAT
(LABELED PHOTOS)
DOG
OUTPUT
cat
✓GOT IT


Input Layer
(256 neurons – 1 neuron per image pixel)
Hidden Layers
Output Layer
Image
28x28 grid of pixels = 256 pixels
0
1
2
3
4
5
6
7
8
9
7

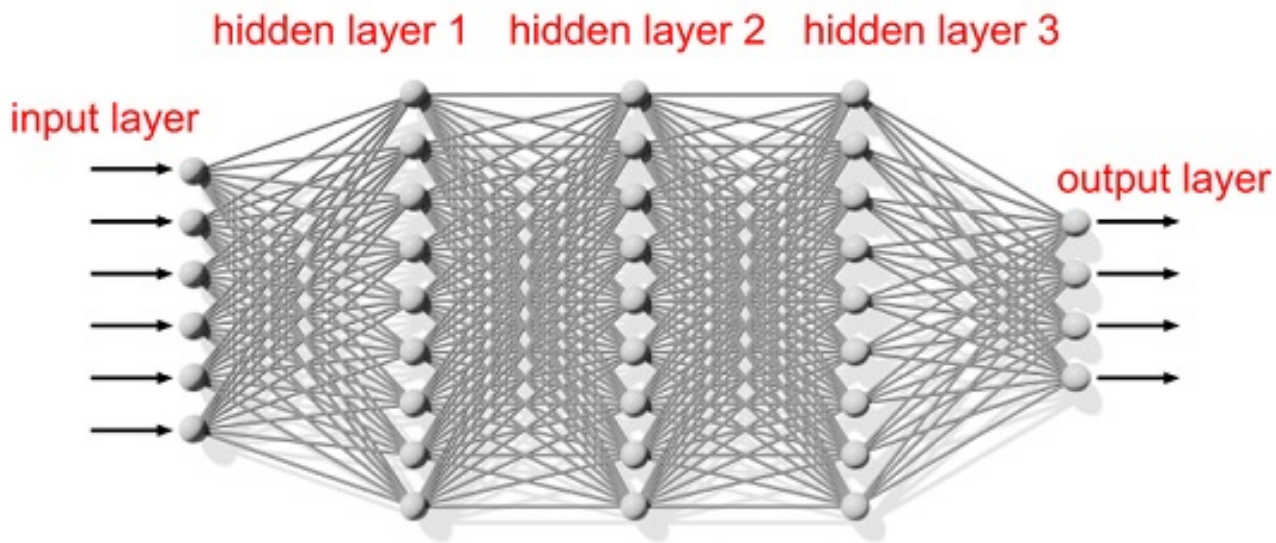

δ(h)
δ(o)
Input
Hidden
Output
x(t)
x(t-1)
x(t-2)
x(t-3)
x(t-4)
w(i)
v(h)
x(t+1)
x(t+2)
x(t+3)
x(t+4)
x(t+5)
Normalised Data
Time(days)
0.0
0.2
0.4
0.6
0.8
1.0
0
250
500
750
1000
1250
1500
1750
2000
2250

Deep Learning Motivation

- Deep learning works especially well for
  - visual object recognition
  - speech recognition
  - natural language processing
  - reinforcement learning tasks with complex environments

Deep Neural Networks

Deep neural networks are just neural networks with *many* (usually more than 3) layers between the input and output layers. These additional layers enable it to handle greater complexity.



deep neural network

Sample of Applications

- virtual assistants like Alexa
- image recognition – deep-learning based image recognition has attained very high levels, now consistently producing more accurate recognition levels (of faces, signs, etc.) than human contestants
- analysis of medical images for cancer detection
- social network analysis and sentiment classification

Another example: deep fakes via GANs

Deep fakes use deep learning – hence the name! – specifically using something called a **generative adversarial network (GAN)**, where two "competing" neural networks are trained simultaneously, the competition between the two being describable in the language of game theory.

Another example: deep fakes via GANs

A **GAN** consists of

- the **generator** network
- the **discriminator** network (used as a classifier trained to classify inputs as real (drawn from the training set) or fake (created by the generator))

where the two are trained simultaneously, with the generator learning to fool the discriminator and the discriminator learning to accurately separate real from fake data.

> *The idea is that in the equilibrium state of the "game" they are playing, the generator should ultimately reproduce the training distribution perfectly, such that the discriminator cannot perform better than random guessing.*

GANS have worked particularly well for image generation tasks – in particular, they are used in the creation of deep fakes.

See the Tom Cruise DeepFakes

Some Ethical Issues that arise with Deep Learning

General Observation

Automated decision systems – especially those employing AI tools – can bring a wide range of benefits, including more efficiency, consistency and even fairness.

- AI in particular can improve efficiency by enabling us to make sense of large amounts of data and, thereby make more informed decisions more quickly and responsively. It can not only make us more productive and knowledgeable, but also minimize bias and error: for instance, systems trained to diagnose cancer are reaching higher accuracy rates than a radiologist by collecting the knowledge that thousands of doctors have made judgment calls in the past. By filtering through all the images and only selecting the troubling ones, machines can relieve doctors from some cognitive load who do not have to sort them all and help correct their mistakes.
- The belief that algorithms can outperform expert judgment – and even help reduce harmful bias – is shared by Nobel laureate Daniel Kahneman, who made the argument at a conference on AI that the decision-making process of humans is too "noisy" and therefore should be replaced by algorithms whenever possible.

But they also give rise to new forms of harms and risks.

- A chief issue with automated decision systems is that the source of the harm, or the cause of the risk, may be much harder to identify and address, as compared to legacy human-operated systems.

Definition

Transparency (of an algorithm, application, or system) roughly concerns how much it is possible – *in theory* – to understand about the inner workings of the algorithm/application/system in question.

Definition

Opacity or **opaqueness** (of an algorithm, application, or system) refers to a marked *lack of transparency*. It is closely related to – and sometimes referred to under the label of – the **black box problem**, where a **black box** is a general concept from computing/engineering describing a system that can be viewed entirely in terms of its inputs and outputs *without any knowledge of its internal workings*.

- The generally opaque nature of many ML algorithms, and many techniques used in deep learning, seems to present us with a problem:

  While AI is coming to permeate more and more areas and is being used to address more and more social activities, it is generally the case that the algorithms/mechanisms underlying these tools are such that we cannot understand how and why a decision has been made.

- This feature would appear to undermine our capacity for guaranteeing fundamental social and ethical values.

How code is generally understood

Generally speaking, computer scientists typically have access to the code, the trained rules, and the underlying architectures of the systems that sustain their run-time operations.

- Keeping track of this is done using tools like using memory snapshots and dumps (outputting a snapshot of memory contents taken during program operation).
- Other approaches advantage sophisticated development environments which can step through (line by line of code) and monitor code execution, and the inputs, outputs, and activity of functions and methods in the program.

The problem for much of AI, but especially for Deep Learning

On the other hand, AI algorithms can be especially opaque. The logic guiding the process of an ML algorithm's production of a given output for certain input – for instance, of the trained hidden layer nodes of a deep neural network learning system – are not accessible in such a direct way.

- In many instances, they can only be *approximated* or *guessed* by using indirect means

An oft-cited problem – with ethical import – of neural networks is their black-box nature.

> *Imagine a bank that is using an ML algorithm to recommend mortgage applications for approval. Suppose an applicant is rejected and brings a lawsuit against the bank, alleging that the algo is discriminating racially. The bank answers that this is impossible, since the algo is deliberately blinded to race of the applicants. Even still, analysis shows that the bank's approval rate for black applicants has been steadily dropping. Submitting ten apparently equally qualified genuine applicants further shows that the algo accepts white applicants and rejects black applicants.*
>
> *Finding the answer to what is happening here might not be easy. If the ML algo is based on a deep neural network, for instance, it may be close to impossible to understand why, or how, the algo is judging applicants based on their race.*

Opacity Dilemma

Let's frame this opacity issue in terms of a related dilemma.

> **Dilemma**: In order for certain AI algorithms, especially those involving deep learning, to serve the purposes that we want them to serve by design, they tend to become more opaque to us (and, it may be, necessarily so), thus getting in the way of interpretability, communicability, and transparency.
>
> It is entirely possible that the very design and key features that gives deep learning the power and versatility that makes it so useful are what also makes it potentially unpredictable and dangerous.

A potential solution?

One natural solution might involve making normative ethical goals part of a deep learning neural network's training regimen.

- The idea would be to train the system towards ethical outcomes along with its other training.
- The ethical imperative would be enforced by the cost function that measures the efficacy of the network for each iteration.

Related issue

This issue of opacity is closely related to **explainability**.

- Why does an AI suspect a person might be a criminal or have cancer? The explanation for this and other high-stakes predictions can have many consequences. Simpler problems? What makes an image of a matchstick a matchstick? Is it the flame or the wooden stick?

The field of **explainable AI** concerns making AI systems that can explain themselves. A good explanation should generally have several properties

- it should be understandable and convincing to the user
- it should accurately reflect the reasoning of the system
- it should be complete
- it should be specific – in the sense that different users with different conditions or different outcomes should get different explanations

There are lots of open questions regarding what constitutes a fair system, a good explanation and what level of transparency is sufficient, as well as transparent to whom and for what purpose. There are additional concerns as well, including arguments to the effect that transparency may be neither feasible nor desirable.

- Too much transparency – up to the point of letting people know how decisions were made – can allow them to "game the system" and orient their data to be viewed favorably by the algorithm – which would both invite a number of seemingly unfair outcomes and also make the conclusions of the algorithm potentially less viable.
- There are broader issues with the very formulation of the notion of fairness: we don't want a risk assessment system to just recommend "detain" for 100 percent of immigrants in custody, as one system did.

One suggestion has been made that systems ought to offer *counterfactual explanations*, and provide the smallest change that can be made by a user to obtain a desired result.

- In the case of an algorithm refusing someone a home loan, for instance, the system should tell the person the reason, like too little savings, but also what he or she can do to reverse the decision – in that case, the minimum amount of savings needed to be approved.
- However, observe that providing explanations alone does not address the heart of the problem: knowing which features of the data are used by automated systems to make a decision – and whether or not they are appropriate for the decision in question.

Another Issue: Difficulty Generalizing/Brittleness

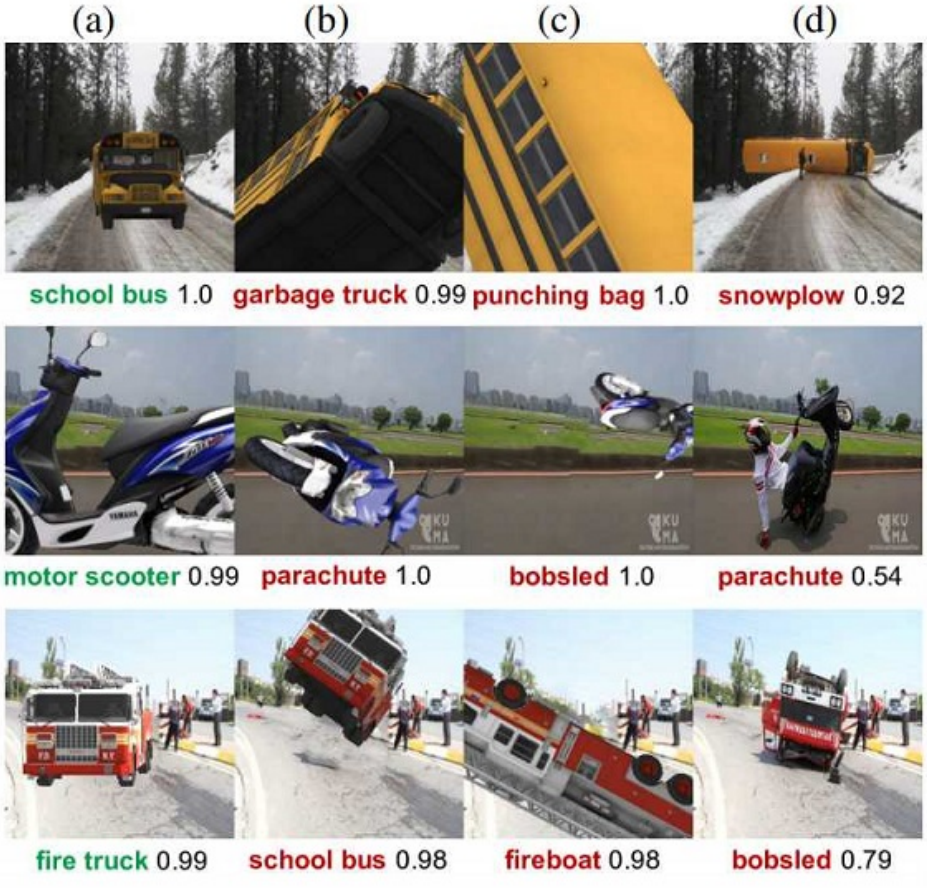
There are many examples – like with **imitation learning** – where the technique will mostly fail to generalize what it has learned to novel environments and inputs in a sensible way.

And this, in turn, is closely related to something called **brittleness**.

> For instance, neural networks are characteristically brittle: changing a single pixel in the many-pixeled picture of the organ of a sick patient can make the networks think the patient is healthy.

An Example of Brittleness

Take a picture of a school bus.



(Google Inception-v3 classifier.)

An Example of Brittleness

Flip it so it lays on its side, as it might be found in the case of an accident in the real world. A few studies have found that state-of-the-art AI that would normally correctly identify the school bus right-side-up failed to do so on average 97 percent of the time when it was rotated. They will say the school bus is a snowplow with very high confidence. The AIs are not capable of a task of mental rotation that even a 2 or 3 year old could do.

Lots of similar cases of AI brittleness. Another example commonly given is that neural networks can be close to 99.99 percent confident that colorized static is a picture of a lion.

This gets at what brittleness is about: the AI often can only recognize a pattern it has seen before; showing it a new pattern, it can be easily fooled.

One way to make AIs more robust against such failures is to expose them to as many confounding **adversarial examples** as possible. But even still, sometimes that isn't enough.

Another Issue: Bad Memory/Disastrous forgetting

Disastrous forgetting: The tendency of an AI to entirely and abruptly forget information it previously knew after learning new information, essentially overwriting past knowledge with new knowledge. A big issue people often isolate in deep neural networks is their generally "bad memory."

It was often said that one main weakness of deep neural network models is that, unlike humans, they are unable to learn multiple tasks sequentially.

At first it appears that catastrophic forgetting was an inevitable feature of these sorts of models, though in the last few years there have been attempts to overcome this limitation and train networks that can maintain expertise on tasks.

This issue of “bad memory” is closely related to the difficulty with generalizing: for instance, sticking with neural networks, one often needs to create a specialized network for each new task, to avoid having updated data cause “forgetting” – yet this is generally not scalable, as the number of networks increases linearly with the number of tasks.

Another Issue: Quantifying Uncertainty

Neural networks also generally lack a framework for assessing the level of *certainty* in the predictions they make: for instance, they can tell you that with high probability that a patient is likely to develop a disease, but they cannot tell you what the uncertainty around this probability is.

Is it 0.95 ± 0.050 *or* 95 ± 10.9*?*

The former would amount to a strong prediction, while the latter is a rather unreliable one.

Relates to **trustworthiness**...

> Models developed using ML and deep learning are widely used for all types of inference and decision making, meaning that it is increasingly important to evaluate the reliability and efficacy of AI systems *before* they are applied in practice, since the predictions made by such models are subject to noise and model inference errors. Accordingly, it is desirable to represent uncertainty in a trustworthy manner in any AI-based system.

Yet people have only just begun working on developing a widely accepted framework for this. There are a number of issues – like several types of uncertainty that need to be quantified – that need to be handled.

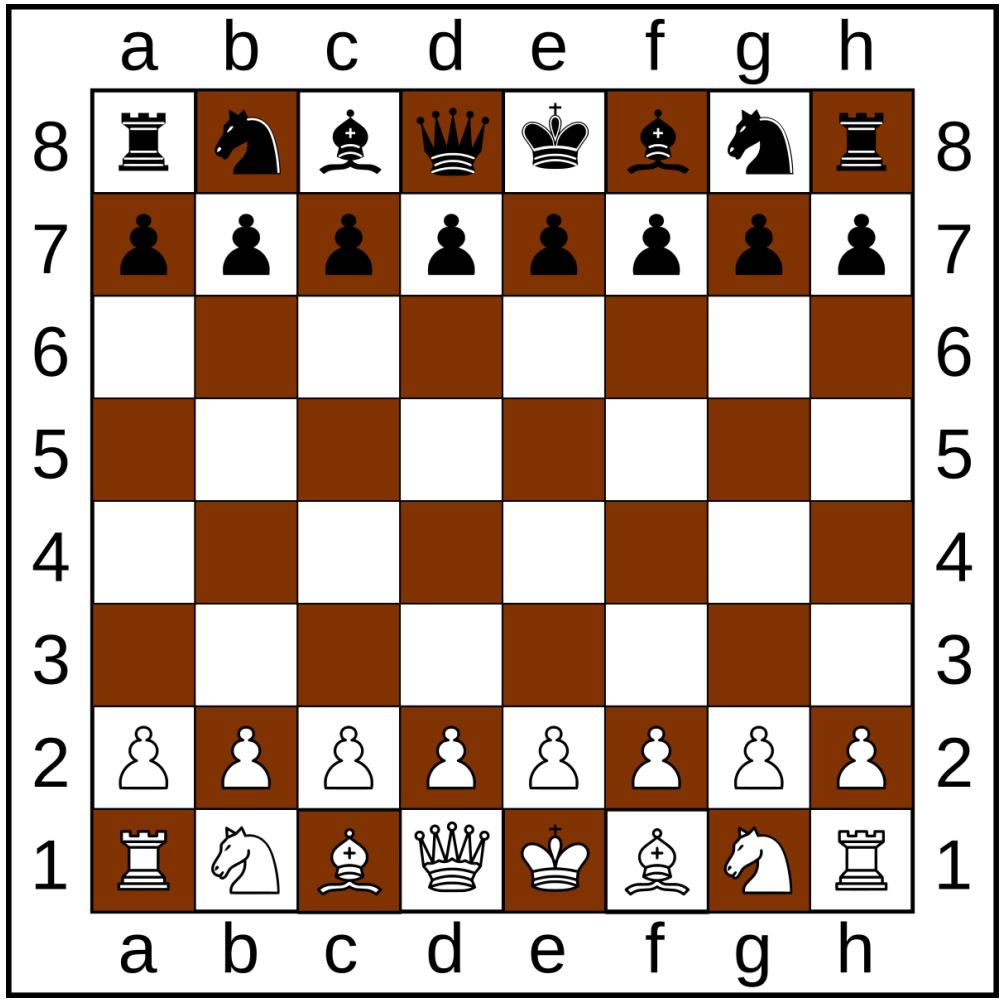
Reinforcement Learning

Reinforcement Learning Motivation

- Recall that with **supervised learning**, an agent learns by passively observing example input-output pairs, as supplied by a "teacher" labeling the data.
- With **unsupervised learning**, we relaxed the teacher requirement, and let the system discover for itself patterns in unlabeled data.
- In **reinforcement learning** – the topic of the remaining slides – agents can *actively learn* from their own experience, without a teacher, by considering their own success or failure, where this means experiencing rewards and punishments and letting these teach it how to maximize rewards in the future.

Reinforcement Learning Motivation

Take the problem of learning to play chess.



Reinforcement Learning Motivation

- Let's imagine we were to treat this as a **supervised learning problem**.
- Under the supervised paradigm, the chess-playing agent function would then take as **input** a board position and return as **output** a move – so we are effectively going to train this function by supplying examples of chess positions, each labeled with the correct move.
- We happen to have available databases of several million games from grandmasters of chess, each of which game consists of a finite sequence of positions and moves. In this collection of games, the moves made by the winner are – with the occasional exception – generally assumed to be good (after all, they led to winning against another chess master!).
- Using such a database, it appears we have at our disposal a decent **training set**.

Reinforcement Learning Motivation

But there's a snag!

- Even though a couple million (10^8, say) games from grandmasters might seem like a lot, the space of all possible chess positions is estimated to be closer to around 10^{120} – which is massively larger than the number of atoms in the observable universe (around 10^{80})
- Even if we restrict to the class of "sensible" games – not counting positions that involve obvious game-losing moves, or ridiculous move sequences – the result is closer to around 10^{40}. Still! That is *way* more than 10^8.
- Because of this, when confronted with a new game, it won't take long before they're encountering positions that are significantly different from any of those in the database, and in such instances the trained agent function is likely to fail miserably – not least because it simply has no idea what its moves are supposed to achieve or even the effect the moves have on future positions.

Reinforcement Learning Motivation

- But...chess is just a very small part of the real world – which of course involves problems and configuration spaces and environments *way* more complex than chess...
- And for such problems, using the same supervised approach, we would need the analogue of **much vaster "grandmaster" databases**.
- Yet, clearly, no such databases exist (or are even likely to be found anytime soon)!
- The **moral**? To quote Yann LeCun,

  "The AI revolution will not be supervised."

Reinforcement Learning Motivation

An alternative approach is supplied by **reinforcement learning (RL)**.

- With RL, an agent starts out not knowing anything about its environment – it knows only about which actions are available to it.
- It interacts with its environment (by, say, starting with random actions), periodically receiving **rewards** (alternatively called **reinforcements**) that reflect how well it is doing. In this way, with such feedback, **by trial and error it gradually learns** about its environment and which sets of actions in that environment enable it to reach a certain goal or obtain rewards.

Reinforcement Learning Motivation

- For instance, if we go back to the chess example: in playing chess the reward can be 1 for winning, 0 for losing, and $\frac{1}{2}$ for a draw.
- Ultimately, the goal with RL is to have an agent that **maximizes the expected sum of rewards**: an RL agent develops its behavior by interacting with its environment, weighing the punishments and rewards of its actions, and developing policies that maximize rewards.
- In reinforcement learning, the agent is *not* given to know in advance the transition model or even the reward function – it simply has to **act to learn more**.

Reinforcement Learning Motivated

Here, you can see an RL algo in action, applied to the very similar problem of a game of Pacman: Using Refinforcement Learning to Play Pacman

Reinforcement Learning Motivation

Stepping back, let's make some general observations:

- RL is about **understanding the world through action**, about **letting agents learn by trial and error**
- RL is inspired by intelligent behavior in animals and humans.

Reinforcement Learning Motivation

Reinforcement learning in a nutshell:
Imagine playing a new game whose rules you don't know, and the aim of which you don't know.
Suppose that after a hundred or so moves, a referee or scoreboard tells you

> *"You lose!"*

You then know that there was something suboptimal in your sequence of actions – so you play again and again and try to gradually hone in on those sequences of actions that arrive at the rewards and avoid the punishments.
That is reinforcement learning in a nutshell.

Reinforcement Learning Motivation

Why do we want to do this?

- For one thing, providing a reward signal to the agent is typically *much* easier than providing labeled examples of how to behave. In general, **the reward function is often very concise and easy to write down explicitly**: as for chess, it requires only a few lines of code to tell the chess-playing agent if it has won or lost the game
- Second, **we don't have to be experts**, capable of supplying the correct action for any possible situation (as we would have to if we approached things via supervised learning)

Reinforcement Learning Motivation

As it turns out, though, a *little bit of expertise* can go a very long way in reinforcement learning.

- When you have rewards like the win/loss rewards for chess or the win/loss rewards for a car race, as above, these are often called **sparse rewards**, since in the vast majority of states the agent can be in, they are given no informative reward signal at all.
- In other settings – like playing tennis – we can easily supply additional rewards for things done *along the way*, such as when a point is scored. Similarly, in car racing, we could reward the agent for each lap it does in the right direction around the track. An agent learning to walk can be rewarded whenever it is both standing up and doing any forward motion. These are called **intermediate rewards**, and their introduction can make learning much easier
- As long as we can provide the correct reward signal, reinforcement learning provides a very general and powerful way of building AI systems
- this is especially true for *simulated environments*, where there are plenty of relatively risk-free opportunities to gain experience and learn by acting

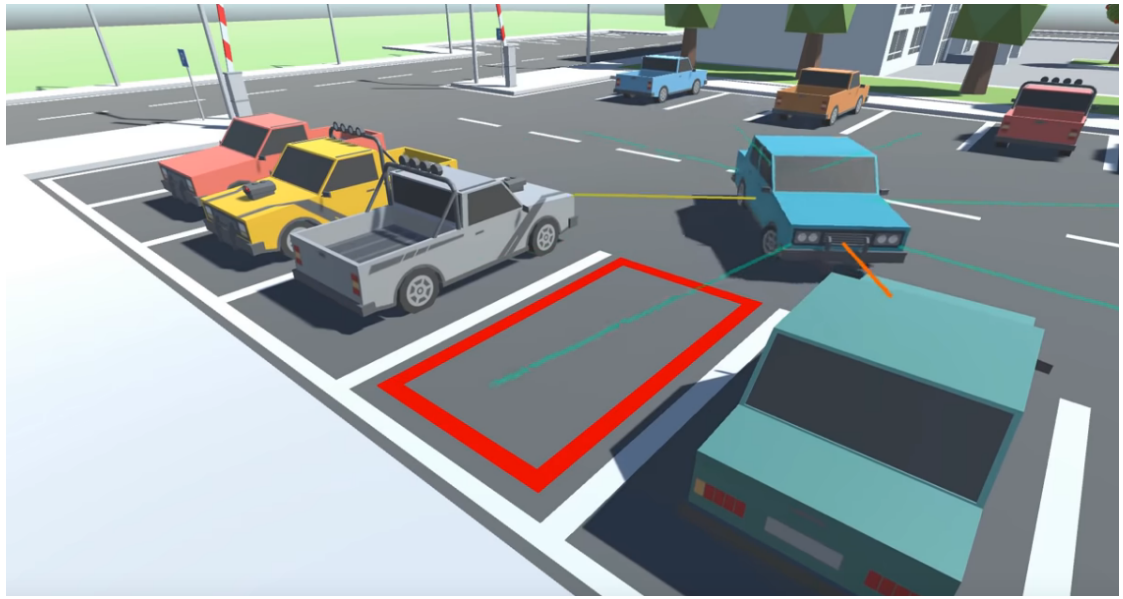
Reinforcement Learning Motivation

Reinforcement learning is one of the most active areas of machine learning (ML) research. Why?

- As just mentioned, providing a reward signal to the agent is typically easier than providing labeled examples of how to behave, and we don't have to be experts
- It frees us from manual construction of behaviors and from labeling the vast data sets required for supervised learning approaches
- It frees us from having to hand-code control strategies
- It's especially useful for training an agent to perform certain skills we don't fully understand ourselves, or at least how to teach it explicitly – like walking!

Learning to Park

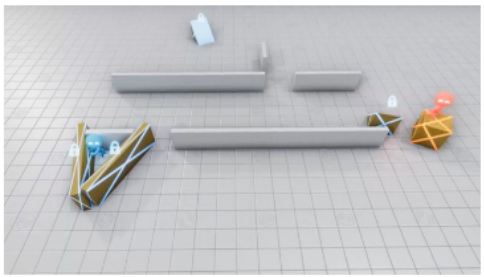
Watch Learning to Park with Neural Networks and Proximal Policy Optimization (a Reinforcement Learning approach)



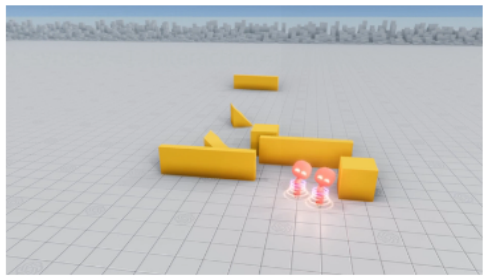
"Basically, the input of the Neural Network are the readings of eight depth sensors, the car's current speed and position, as well as its relative position to the target. The outputs of the Neural Network are interpreted as engine force, braking force and turning force. These outputs can be seen at the top right corner of the zoomed out camera shots.
The AI starts off with random behaviour, i.e. the Neural Network is initialized with random weights. It then gradually learns to solve the task by reacting to environment feedback accordingly. The environment tells the AI whether it is doing good or bad with positive or negative reward signals.
In this project, the AI is rewarded with small positive signals for getting closer to the parking spot, which is outlined in red, and gets a larger reward when it actually reaches the parking spot and stops there. The final reward for reaching the parking spot is dependent on how parallel the car stops in relation to the actual parking position. If the car stops in a 90 angle to the actual parking direction for instance, the AI will only be rewarded a very small amount, relative to the amount it would get for stopping completely parallel to the actual direction.
The AI is penalized with a negative reward signal, when it either drives further away from the parking spot or if it crashes into any obstacles."

Cool Example of Multi-Agent RL: Open AI's Hide-and-Seek

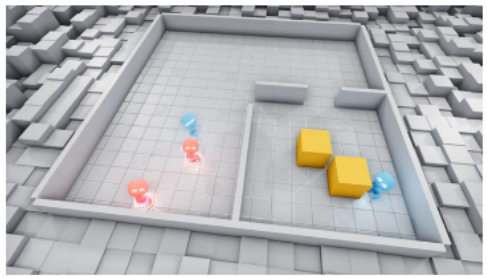
Check out Open Ai hide-and-seek!



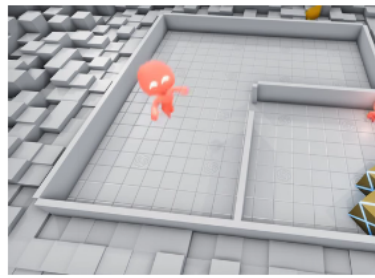
Box surfing Since agents move by applying forces to themselves, they can grab a box while on top of it and "surf" it to the hider's location.



Endless running Without adding explicit negative rewards for agents leaving the play area, in rare cases hiders will learn to take a box and endlessly run with it.



Ramp exploitation (hiders) Reinforcement learning is amazing at finding small mechanics to exploit. In this case, hiders abuse the contact physics and remove ramps from the play area.



Ramp exploitation (seekers) In this cas learn that if they run at a wall with a ram right angle, they can launch themselves

A nice video summary here "OpenAi plays hide and side...and breaks the game!"

Some prominent applications

- video games
- robotics
- self-driving cars
- content recommendations
- solving complex logistics problems that combine packing, routing, and scheduling problems
- accelerating clinical trials

Limitations and Challenges with RL

- **Manual specification**: An oft-discussed weakness of the RL approach is that researchers generally have to manually define a goal or reward function corresponding to an agent's goal (by at least specifying the features of the environment that give out rewards and those that punish), before the agent can learn the behaviors that help accomplish those goals.
- **Unsafe behavior**: For complex goals, this can be either too difficult or there can be another problem: misspecified rewards may not only result in bad performance, but also unsafe behavior!
- **Misalignment**: Because of this, some researchers have recently aimed to make the reward function part of the learning process as opposed to something that is specified before training. However, just because a goal is learned (rather than specified in advance) does not mean that it is aligned with our intentions! Moreover, there may be features of the environment that the programmers didn't think consciously of – and an agent may exploit these features in unexpected ways!

- In ML and RL, the workflow for solving a problem generally consists of two stages:
  1. The programmers define the objective and the environment that houses that objective.
  2. Then an optimization algorithm tries to find the best possible solution. In the case of RL, the objective and solution are given by the reward function and policy.
- This approach comes with the risk that the objective's definition may not accurately capture the human's intention. This could lead to an AI system that satisfies the objective to behave in undesirable ways, even if the algorithm that trained it was implemented flawlessly. Such a system is typically called **misaligned**.
- Moreover, what happens when a misspecified reward function encourages an RL agent to subvert its environment by prioritizing the acquisition of reward signals above other measures of goal achievement?
- Agent behavior that scores highly according to the reward function but is not aligned with the programmers' intention is often referred to as **specification gaming**.

Faulty Reward Functions

There are plenty of examples of specification gaming.
In one example, researchers at OpenAI trained an RL agent on the game **CoastRunners**, which is about a motorboat race in a simulated environment.

- The goal of the game - as understood by most humans - is to finish the boat race quickly and (preferably) ahead of other players. CoastRunners does not directly reward the players progression around the course, instead the player earns higher scores by hitting targets laid out along the route.
- Programmers (naturally) assumed the score the player earned would reflect the informal goal of finishing the race. However, it turned out that the targets were laid out in such a way that the reinforcement learning agent could gain a high score without having to finish the course. This led to some unexpected behavior when an RL agent was trained to play the game.
- In an example of misspecified reward, the game does not reward the agent for its progression along the track, but for hitting targets laid out along the track. This was exploited by the agent who found a strategy for hitting targets without finishing the race.

Coastrunners

Short video on Coastrunners

The RL agent finds an isolated lagoon where it can turn in a large circle and repeatedly knock over three targets, timing its movement so as to always knock over the targets just as they repopulate. Despite repeatedly catching on fire, crashing into other boats, and going the wrong way on the track, our agent manages to achieve a higher score using this strategy than is possible by completing the course in the normal way. Our agent achieves a score on average 20 percent higher than that achieved by human players.

While harmless and amusing in the context of a video game, this kind of behavior points to a more general issue with reinforcement learning: **it is often difficult or infeasible to capture exactly what we want an agent to do, and as a result we frequently end up using imperfect but easily measured proxies.** *Often this works well, but sometimes it leads to undesired or even dangerous actions. More broadly it contravenes the basic engineering principle that systems should be reliable and predictable.*

Extending the RL framework to Reward Learning

- A broad class of methods that have gained a lot of attention recently utilize the idea of a **human in the loop**.
- From "How learning reward functions can go wrong":

  The idea is simple: It is (presumably) easier to evaluate if observed behaviour is correct than to unambiguously specify what correct behaviour looks like. Hence, it stands to reason to expect that an evaluation of agent behaviour by humans will be less error-prone than by a reward function.

- However, in general, the possibility of agents manipulating the outcome of their reward learning process is still a big problem.

  The problem comes down to the simple fact that an agent has to infer its reward function from an environment that it can manipulate. As AI-safety researcher Stuart Armstrong put it, making the reward function part of the learning process is a large change akin to moving from "If you dont know what is right, look it up on this read-only list" to "If you dont know what is right, look it up on this read-write list."

Other technical issues

- **data-hungry**: it needs a ton of data or epochs (thousands of computing hours in a simulator), often to learn what humans can usually grasp in a few hours.
- **opaque**: in most cases, we can have only vague and high-level intuitions about what an RL algorithm learns and how it works. In general, we of course want the algorithms to be predictable and explainable; a neural net or RL algo that just learns whatever it wants from scratch given just the low level reward signal and maybe an environment model is low on the explainability scale.
- **issues with the reward function**: for RL to do the right thing, one must design a proper reward function. Such a function must capture exactly what the designer want the reinforcement learning agent to solve – this is hard and can come with a host of problems (hinted at above).
- **brittleness**: models generalize well to unseen inputs only after having large amounts of data, or epochs in a simulated environment, and are even then can still often be easily broken.

Deeper Dive into Ethical Issues with Reinforcement Learning

Task Specification Issues in the real world!

"Imagine training a household robot to make you a cup of coffee. To use RL, you might design a reward function that awards the agent for turning on the coffee machine, inserting the grounds, pouring a cup, and bringing you the finished product. You might also include a small penalty (negative reward) for each second that passes during the process, in order to encourage the swift delivery of your coffee. This simple reward function is already problematic, due to the fact that your household is a complicated place. Your home might include children, pets, fragile or sentimental belongings, a system of organization, and an unspoken code of conduct and culture. However, none of these were mentioned in the reward function above.

In this situation, to maximise its rewards, the household robot will prepare your coffee as fast as possible, even if this means running over pets, breaking glasses, and generally destroying your organised household along the way. On top of this, no matter how hard you try to write down all the rules a robot should obey, the real world is sufficiently complex that you will inevitably leave some out.

Task specification problems come in several flavours. The example of the household robot highlights the negative side effects that can accrue as an agent works to maximise its rewards. To name another, reward hacking can occur if the agent discovers a way to earn rewards that side-steps the task its human designers were aiming to achieve. For example, the household robot might discover that repeatedly switching the coffee maker off and back on earns much more reward than actually delivering any coffee." – The dangers of RL in the real world

Unsafe Exploration Risks

- Even with an accurately specified task, an agent's process for learning how to perform a given task can be dangerous.
- RL always involve trial-and-error learning: the agent takes random actions until it learns which behaviors earn the most total reward. This is generally problematic if learning is to take place in the real world, where errors can be expensive or deadly!
- Since an RL agent must explore a wide variety of actions before learning which behaviors to pursue and which to avoid, this can lead to issues known by the name **unsafe exploration**.
- The freedom to explore without consequence is not always present. If I want to build an autonomous vehicle using RL, how many thousands of times will the car crash itself before it can make even the simplest maneuvers? Sometimes – especially when training in simulated environments is not enough – we simply **can't afford mistakes**! In such cases, RL is not the best option!

Unexpected Risks

Unexpected risks! Recall the unexpected policies executed by the agents in the hide-and-seek world.

In discussing some of the surprising behaviors they observed, the team at OpenAi said:

> *Building environments is not easy and it is quite often the case that agents find a way to exploit the environment you build or the physics engine in an unintended way.*

Remember the Coastrunners example

where the reinforcement learning algorithm led the agent to get stuck in meaningless loops that maximized a simple reward at the expense of what we would deem a far more sensible greater goal (finishing the race)?

This gets at a problem: namely, that reinforcement-learning algos are notoriously very rigid. For instance, a reinforcement learning model that plays one game at championship level wont be able to play another game with similar mechanics. Related to this is the potential for

> **Goal blindness**: the tendency to maximize a reward in a subroutine at the expense of more "important" goals.

A Glimpse into an abstract issue: Wireheading

Wireheading dilemma:
Ring and Orseau (2011) consider the possibility of an agent sabotaging its own information source by placing itself in what they call a **delusion box** in which the agent controls its own inputs, decreasing the amount of information that its inputs give the agent about its environment.
These authors use it as an example of what can go wrong when one is not careful about how an agent is defined, arguing that many simple agents would put themselves in a delusion box. For instance, standard RL agents may distort their own perception to appear to receive high reward, rather than optimizing the objective in the external world that the reward signal was intended to encourage. In other words, a reinforcement-learning agent has a utility function that depends purely on its inputs, not on any other features or prior knowledge of the external world, so if a reinforcement-learning agent had the opportunity to replace its input with whatever input indicates maximum reward, it would do so, regardless of the effect on the external environment.