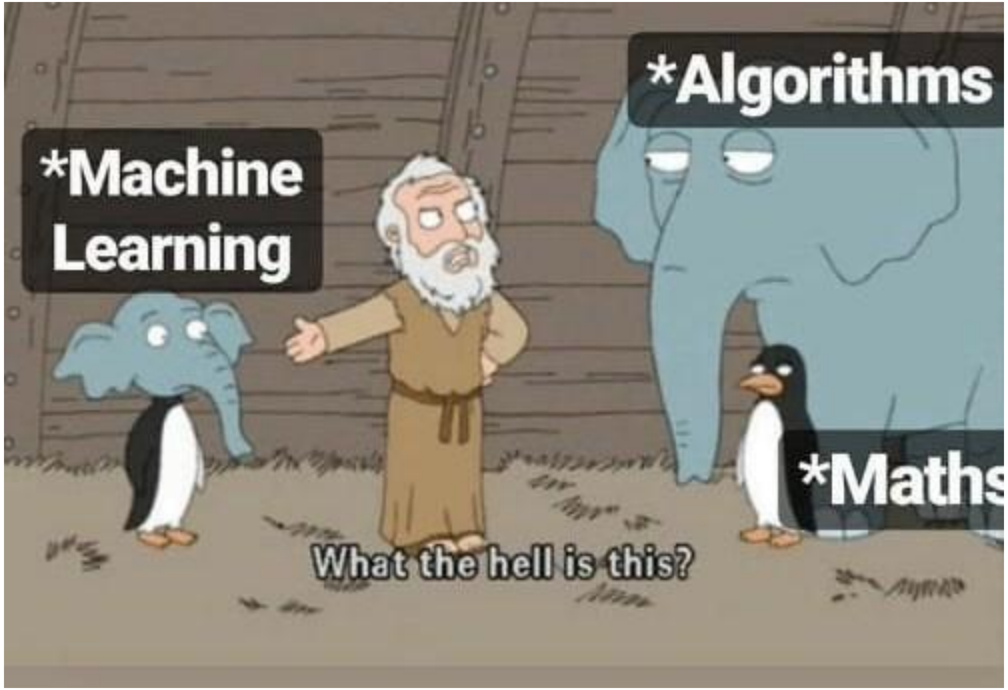
A Glimpse into the World of Artificial Intelligence

By Epoch Team, IIT Hyderabad

We don't do definitions, We do memes



What is Artificial Intelligence?

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.

Some standard applications of AI are:

Natural Language Processing (NLP)

Computer Vision (CV)

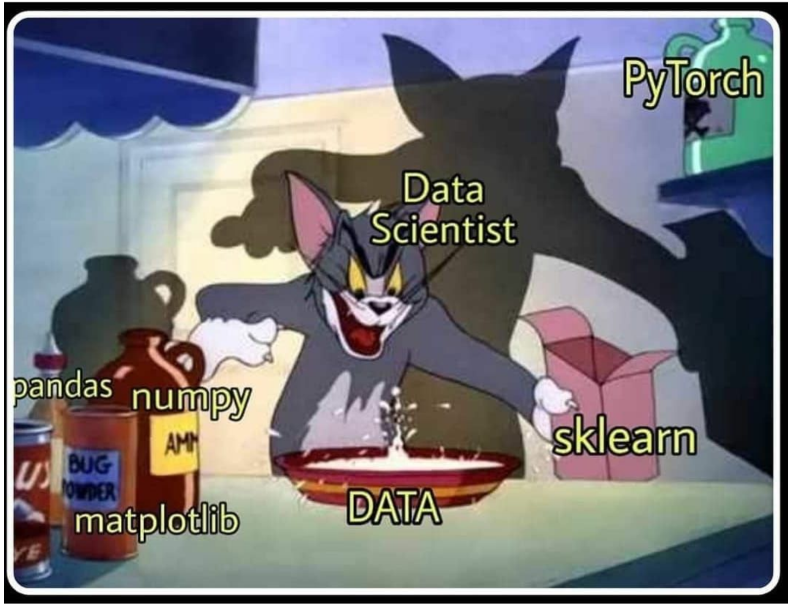
What is Machine Learning?

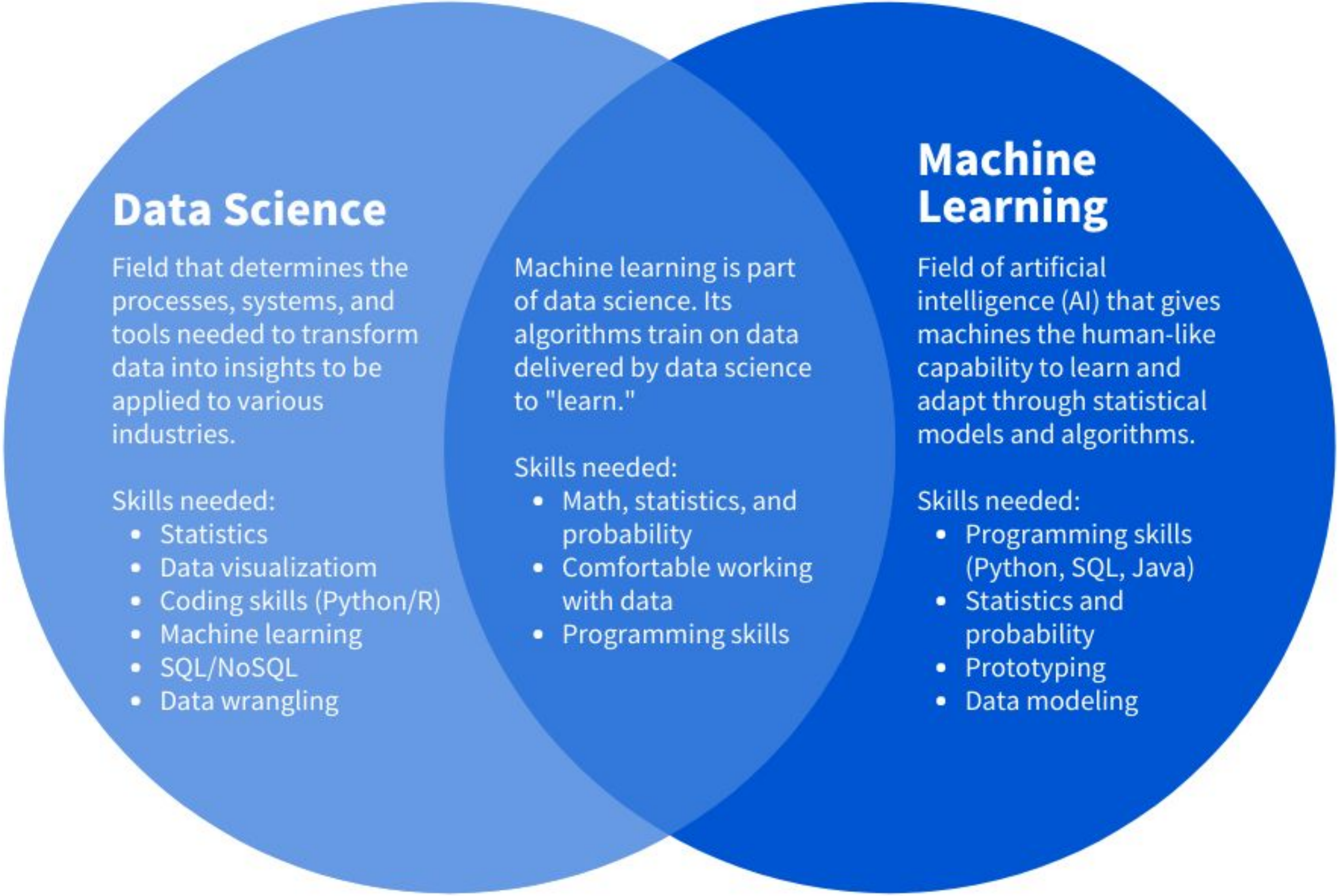
Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

"Machine Learning is the study of computer algorithms that improve automatically through experience." - Tom Mitchell

What is Data Science?

Data science is the field of applying advanced analytics techniques and scientific principles to extract valuable information from data for business, decision-making, strategic planning and other uses.




Data Science
Field that determines the processes, systems, and tools needed to transform data into insights to be applied to various industries.
Skills needed:
• Statistics
• Data visualizatiom
• Coding skills (Python/R)
• Machine learning
• SQL/NoSQL
• Data wrangling
Machine learning is part of data science. Its algorithms train on data delivered by data science to "learn."
Skills needed:
• Math, statistics, and probability
• Comfortable working with data
• Programming skills
Machine Learning
Field of artificial intelligence (AI) that gives machines the human-like capability to learn and adapt through statistical models and algorithms.
Skills needed:
• Programming skills (Python, SQL, Java)
• Statistics and probability
• Prototyping
• Data modeling

Brief insight of these libraries

Numpy - to work with arrays and vectors

Pandas - to work with the dataset (.csv files)

Matplotlib & Seaborn - for visualization

Tensorflow - to create deep learning models scikit learn

SymPy

seaborn

pandas

Why machine learning?

Image Recognition - Google lens

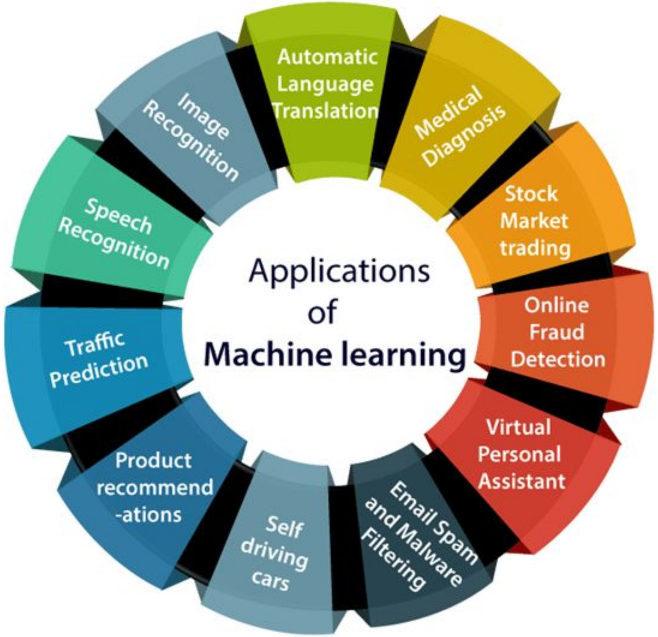
Medical Diagnostics - Medical image processing

Stock Market - Stock price prediction

Content Personalization - Netflix, Spotify, Youtube

Traffic Prediction - Google Maps

Speech Recognition - Virtual assistant like Alexa



Type of Machine Learning

Supervised Learning vs Unsupervised Learning

Supervised learning trains a model on known input and output data so that it can predict future outputs whereas unsupervised learning finds hidden patterns or intrinsic structures in input data.

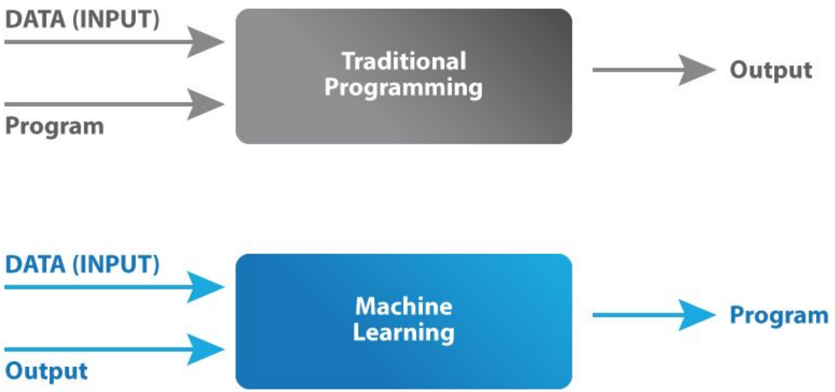
Or in other words, supervised learning make predictions based on labeled training data & unsupervised learning itself identifies relationships between the unlabeled training data.

Train/Test : The data set is splitted into two sets: a training set and a testing set. Say, 80% for training, and 20% for testing. The model is trained using the training set and tested using the testing set.

How exactly does a model learn?

Setting aside the mathematical jargon, a supervised model basically learns a function that maps the inputs X to the outputs Y.

In supervised learning, we give the model a certain set of parameters, that it continuously tune, so that it can learn an appropriate function that does well on the input dataset.



Loss function

Loss function predicts the deviation of the predicted outcome from the actual result. It measures how well the machine learning algorithm models the training dataset.

We convert the machine learning problem into an optimization problem, by defining the loss function and then gradually optimizing the algorithm so that the loss function is minimized.

One of the common loss functions is mean squared error (MSE).

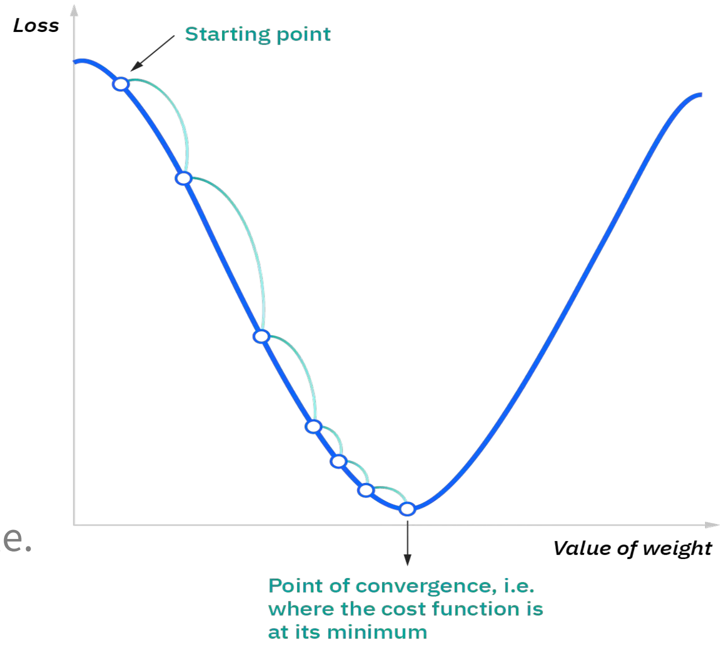
$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Now, what is gradient descent...

Gradient descent is an optimization algorithm/technique commonly used to train machine learning models.

The key idea behind this algorithm is to take the fastest route towards the minimum of the function from current point to converge fast.

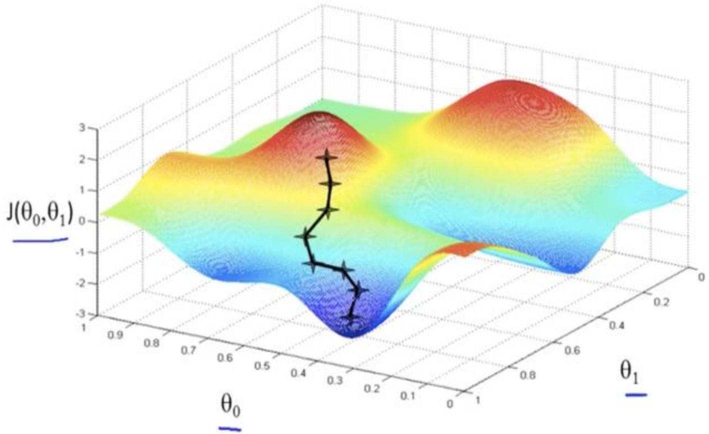
So, we take repeated steps in the opposite direction of the gradient of the function as that is the steepest descent possible.



Gradient descent

1. Compute the slope (gradient) that is the first-order derivative of the function at the current point
2. Move-in the opposite direction of the slope increase from the current point by the computed amount

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$




Gradient Descent
Global Optimum

Computer Vision - 101

No, it’s not about the webcam.

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.

People with no idea about AI saying it will take over the world:

My Neural Network:

Classification - The Holy Grail

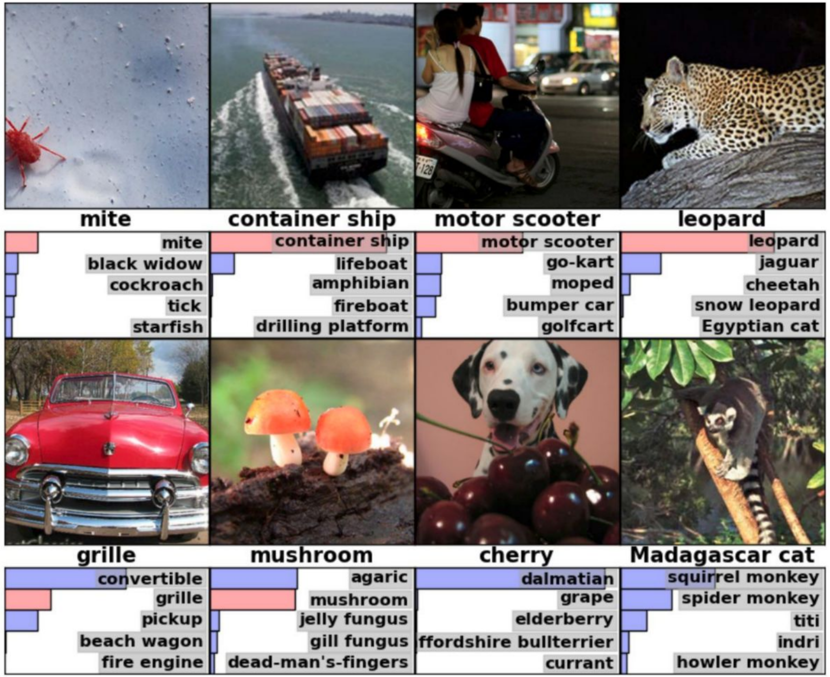
One of the first tasks that was tackled “Computer Vision” involved simple Classification of Hand-written digits.

Later on (after a lot of effort) the ImageNet dataset was created and it was being used for benchmarking models. There were many models working on it , each achieving only 40 -50 % accuracies at best.

During 2012, Alex Krizhevsky invented the AlexNet (You may call this the advent of the real “Deep learning”, it smashed the competition by nearly 10.8 % difference in the top-5 error rate.)

ImageNet Challenge

IMAGENET

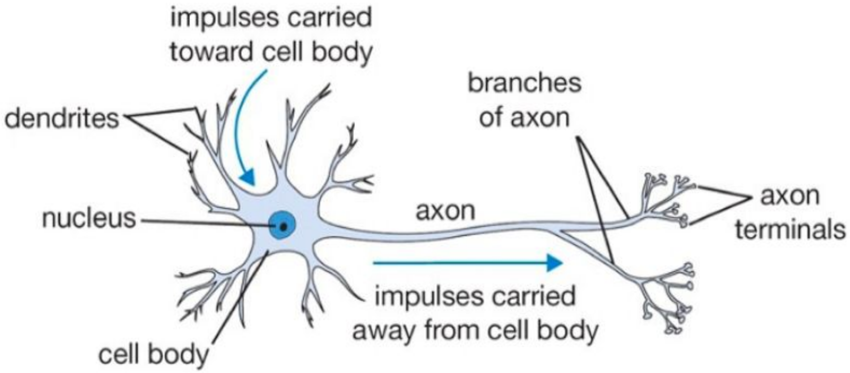


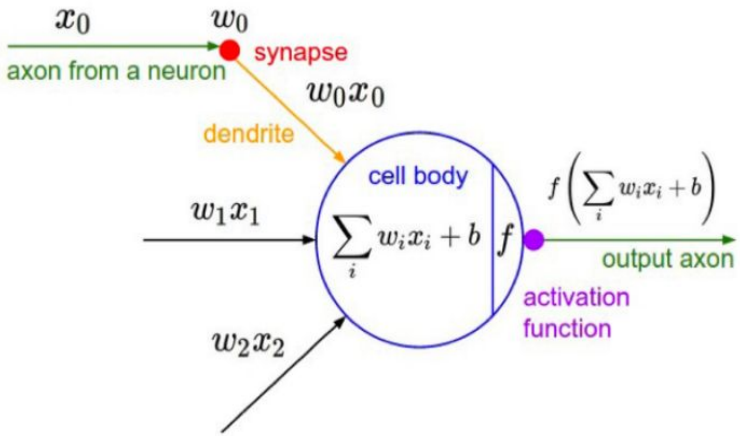
- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

The ImageNet dataset

Neural Networks

You know the story, These Machine Learning Models were “inspired” from how the brain actually works. However we aren’t even close to actually achieving human performance yet.



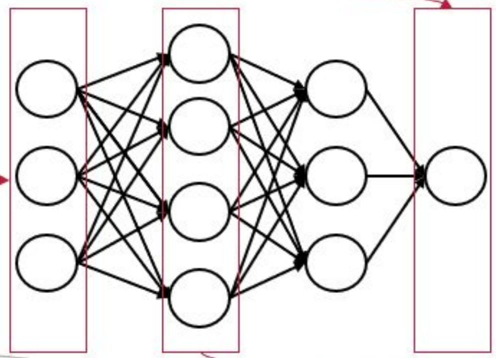


$$f\colon \mathbb{R}^n \to \mathbb{R}^p$$

$$f(x) = \boxed{g} \circ \boxed{f_K} \circ \ldots \boxed{f_2} \circ \boxed{f_1}(\boxed{x})$$

$$\boxed{f_i}\colon \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i-1}}$$

$$f_i(x) = \boxed{a}(w_i x + b_i)$$



$$z_i = w_i x_{i-1} + b_i$$
$$a_i = \sigma(z_i)$$

$$f(x) = g \circ f_K \circ \ldots f_2 \circ f_1(x)$$
$$= \boxed{g}(a(\ldots a(w_2 a(w_1 x + b_1) + b_2) \ldots + b_K))$$

$$\text{Regression} : g(x) = w_{K+1}x + b_{K+1}$$
$$\text{Binary classification} : g(x) = \frac{1}{1 + e^{-(w_{K+1}x + b_{K+1})}}$$
$$\text{Multiclass classification} : g(x)_j = \frac{e^{(w^j_{K+1}x + b^j_{K+1})}}{\sum_{k=1}^{N} e^{(w^k_{K+1}x + b^k_{K+1})}}$$

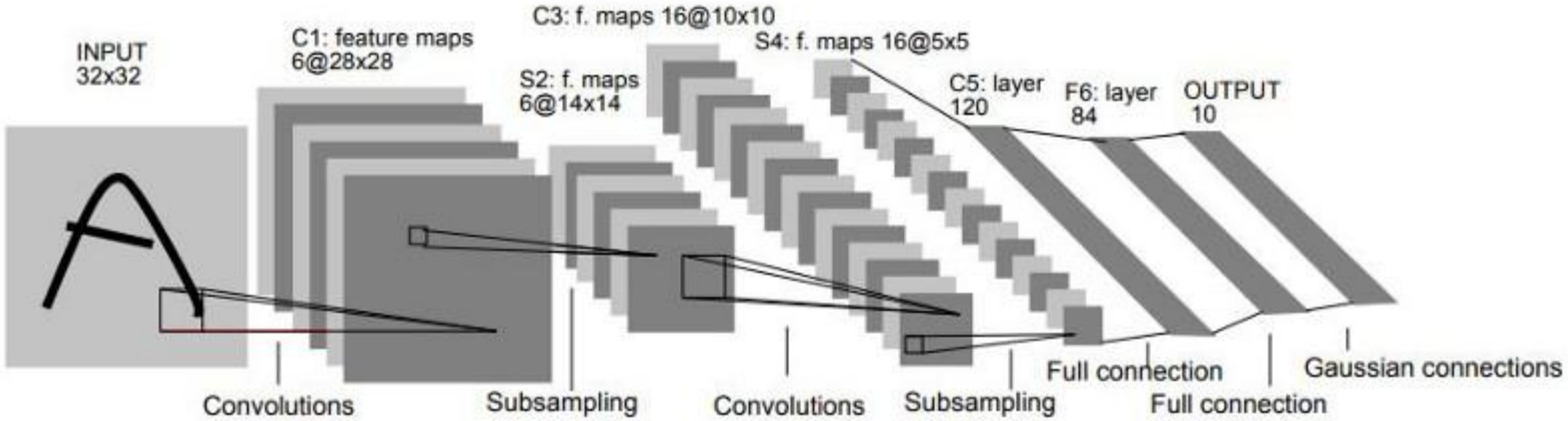
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$
$$\mathrm{ReLU}(x) = \max\{0, x\}$$

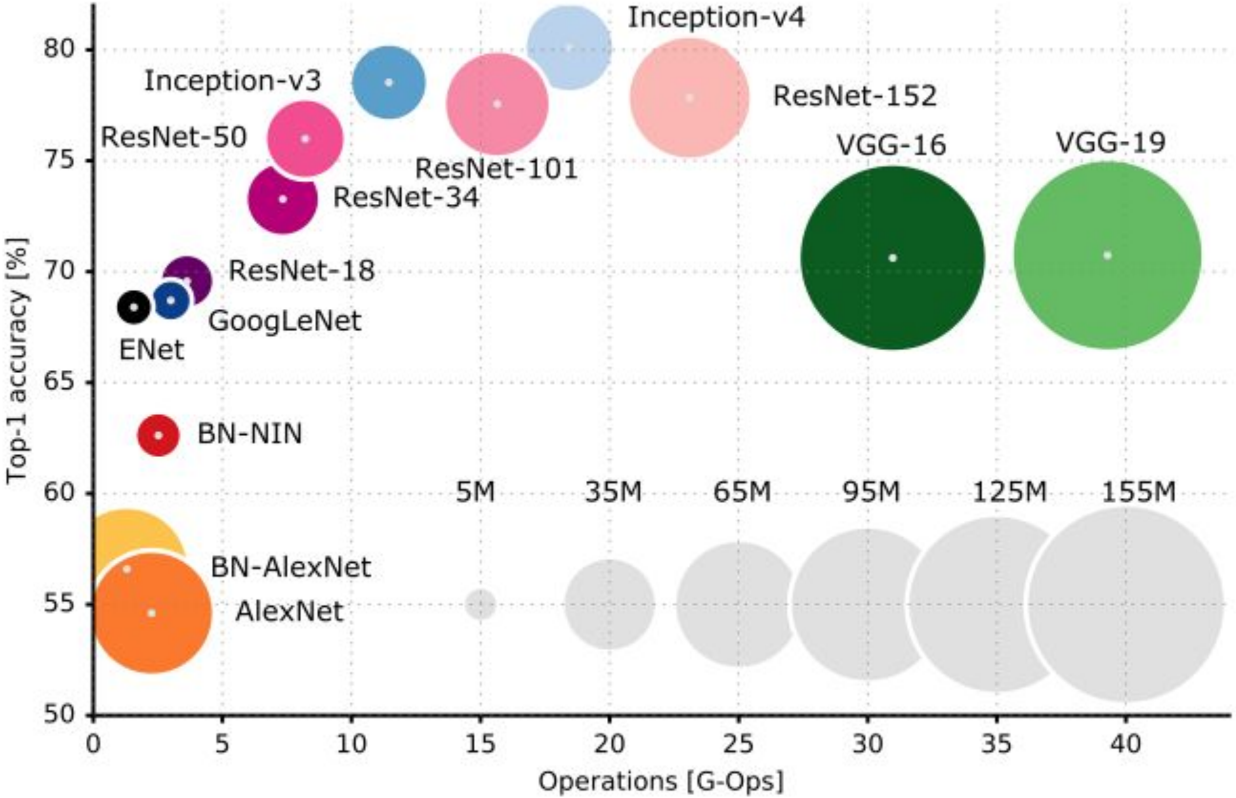
medium.com/@angela.shi

Structure of a Neural network

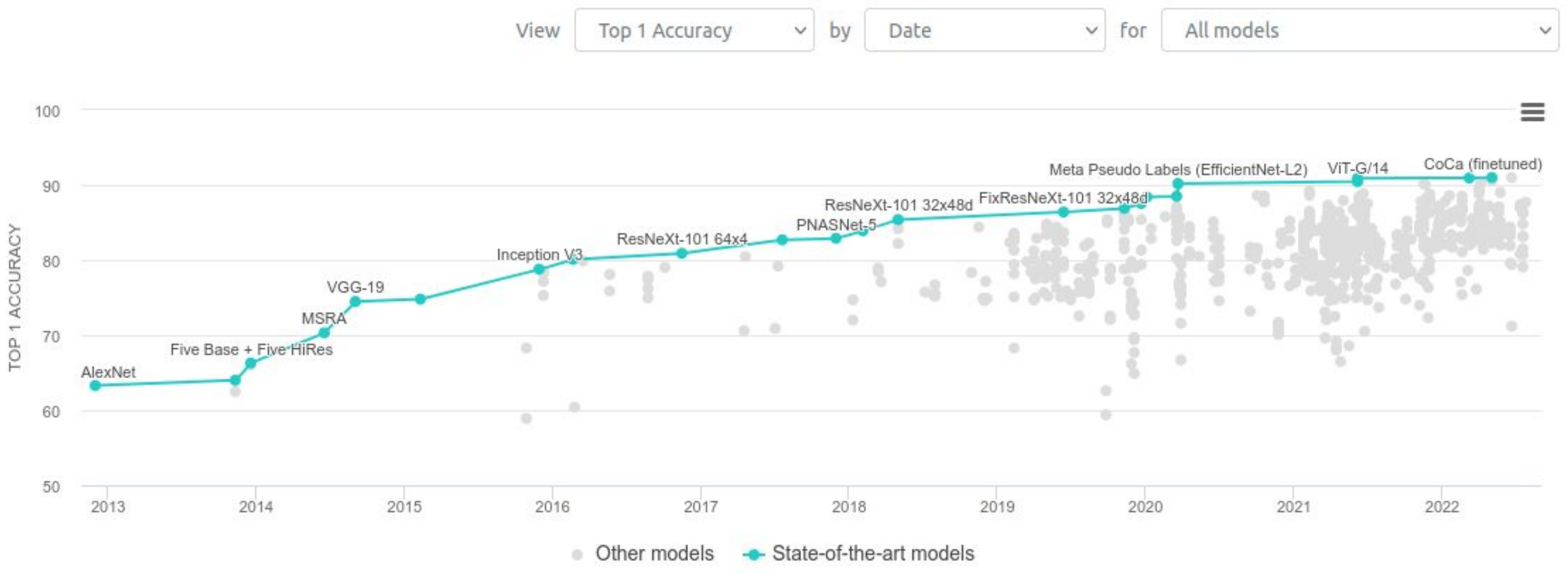
Convolutional Neural Networks

Sharing of parameters and the ability to **learn deeper aspects** of a particular image, like edges and even later, faces.



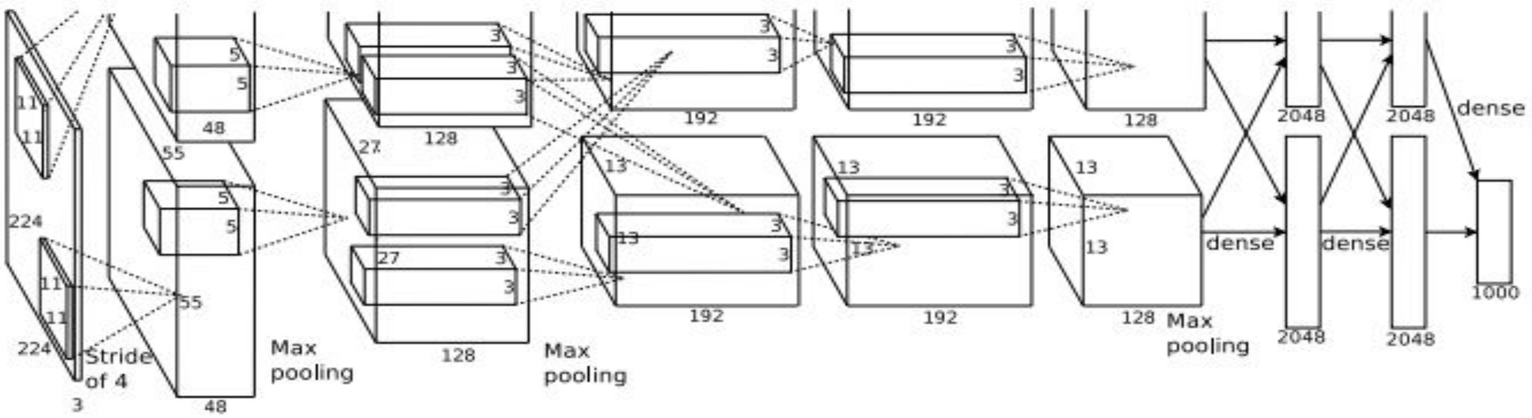


A few Different Model Names that fly around



ImageNet Performance charts over a few years

A few “Deep” Architectures



AlexNet - Highly optimized using GPU’s

Why was AlexNet so damn good!

The RELU non-linearity really became popular after the AlexNet paper.

GPU’s have the ability to perform computations in parallel with several cores,

Think about Matrix multiplication, things could speed if each row and column vector dot products happen simultaneously. This difference starts to become apparent. When the number of parameters become huge (Deep Neural Networks).

AlexNet had ~ 61M parameters ! : ()

It was nearly 4 times as fast compared to the CPU version.

Recognize the Celebs!

Sike , Image generation using GAN’s

Progressive GAN, Karras 2018.

These images are generated

Generative Adversarial Networks

Semantic Segmentation

The goal of Semantic Segmentation is to label every single pixel of an image with a particular class. This is used in Autonomous driving!

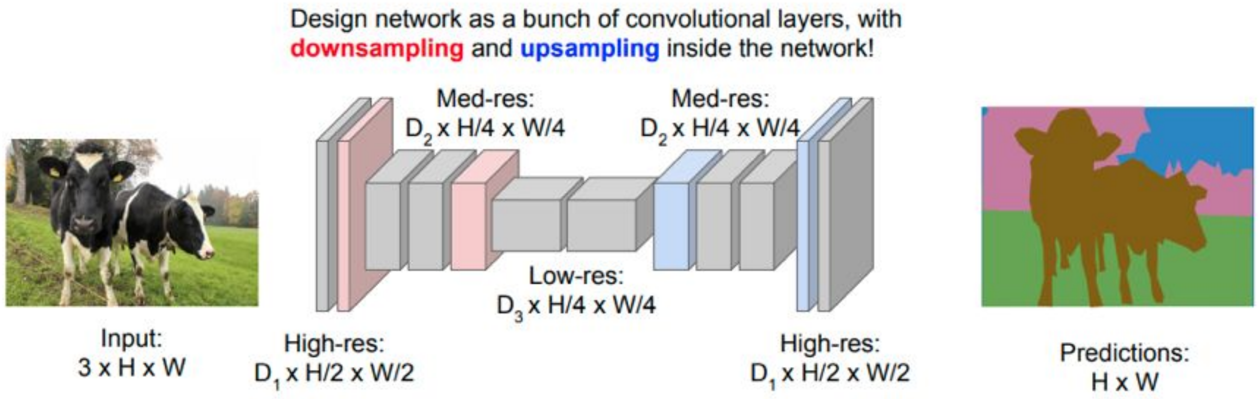


Image Captioning

Image captioning involves finding a “caption” that suits or describes a particular image, Microsoft COCO is the dataset that’s widely used.

Okay? But how do we go about this ?

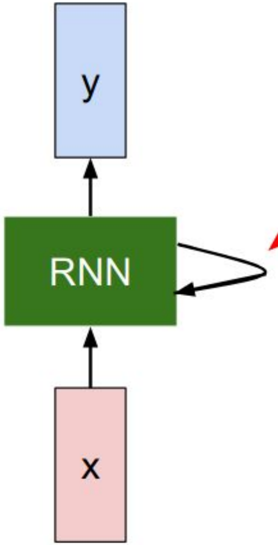
Ours: A man riding a wave in the ocean.
GT: A man riding a wave on a surfboard in the ocean.

Ours: A living room with a lot of furniture.
GT: Living room with furniture with garage door at one end.

Ours: A man riding a horse at a horse.
GT: A horse that threw a man off a horse.

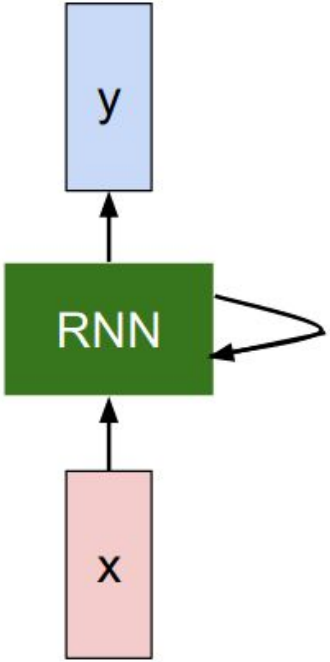
Ours: A man holding a frisbee in a field.
GT: The man is holding the string to fly his kite.

Recursive Neural Networks



Key idea: RNNs have an "internal state" that is updated as a sequence is processed

The structure of a “Vanilla” RNN



$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Sometimes called a “Vanilla RNN” or an “Elman RNN” after Prof. Jeffrey Elman

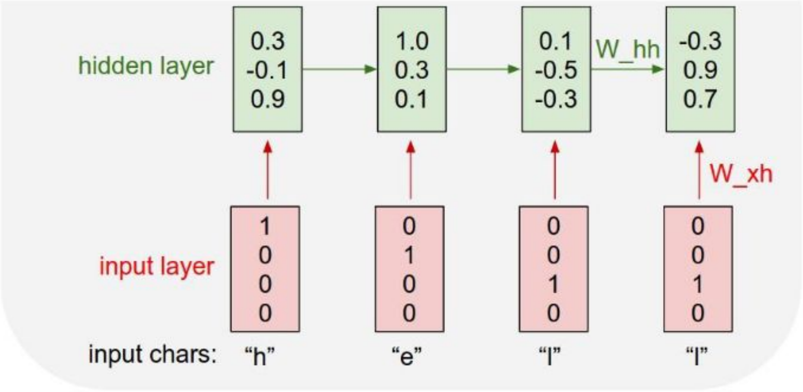
Character Level RNN's

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training
sequence:
"hello"

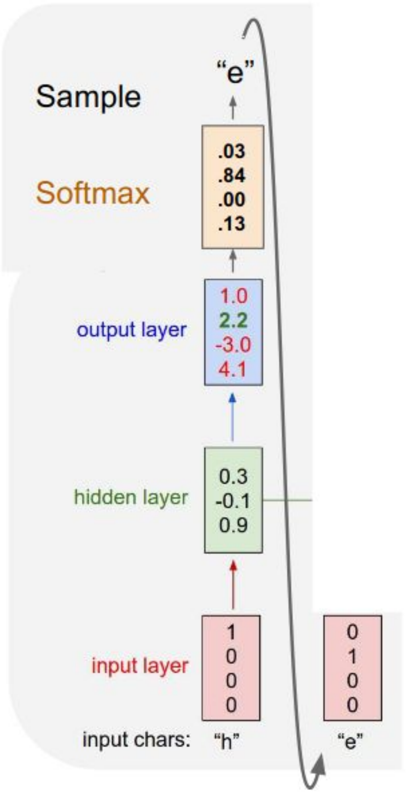
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



**Example:
Character-level
Language Model
Sampling**

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



Sequence to Sequence Sampling

Doing random Stuff with RNN’s

Since RNN’s work on sequenced data, you pretty much have the freedom to put any sort of sequenced data straight into the RNN’s. Including Shakespeare’s Sonnets, LaTeX Source code for a book on Algebraic Topology, or even the entire source code of Linux.

:)

The Sonnets

18.

SHall I compare thee to a Summers day?
Thou art more louely and more temperate:
Rough windes do ſhake the darling buds of Maie,
And Sommers leaſe hath all too ſhort a date:
Sometime too hot the eye of heauen ſhines,
And often is his gold complexion dimm'd,
And euery faire from faire ſome-time declines,
By chance,or natures changing courſe vntrim'd:
But thy eternall Sommer ſhall not fade,
Nor looſe poſſeſſion of that faire thou ow'ſt,
Nor ſhall death brag thou wandr'ſt in his ſhade,
When in eternall lines to time thou grow'ſt,
So long as men can breath or eyes can ſee,
So long liues this,and this giues life to thee,

Shakespeare-> RNN-> Output

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Courtesy:
CS231N

Algebraic Topology -> RNN -> Output

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

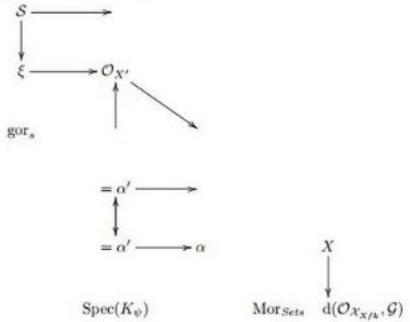
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

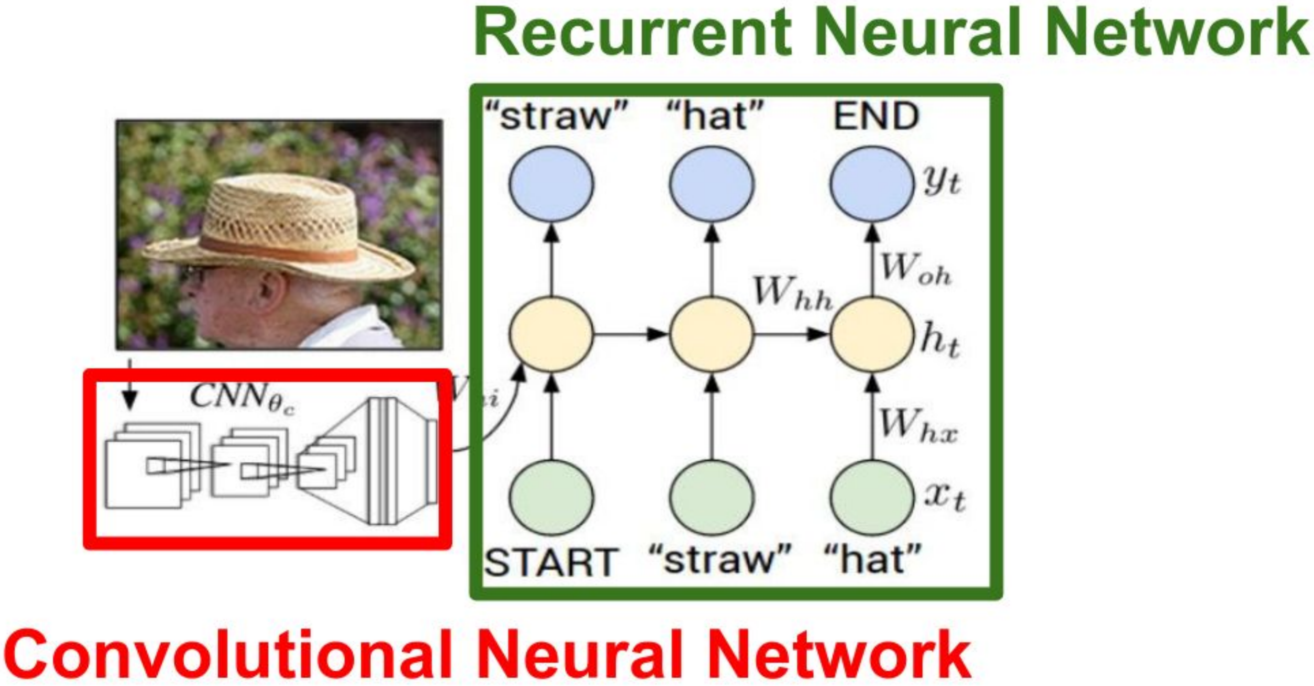
$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \text{-}1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Courtesy: CS231N (Stanford)

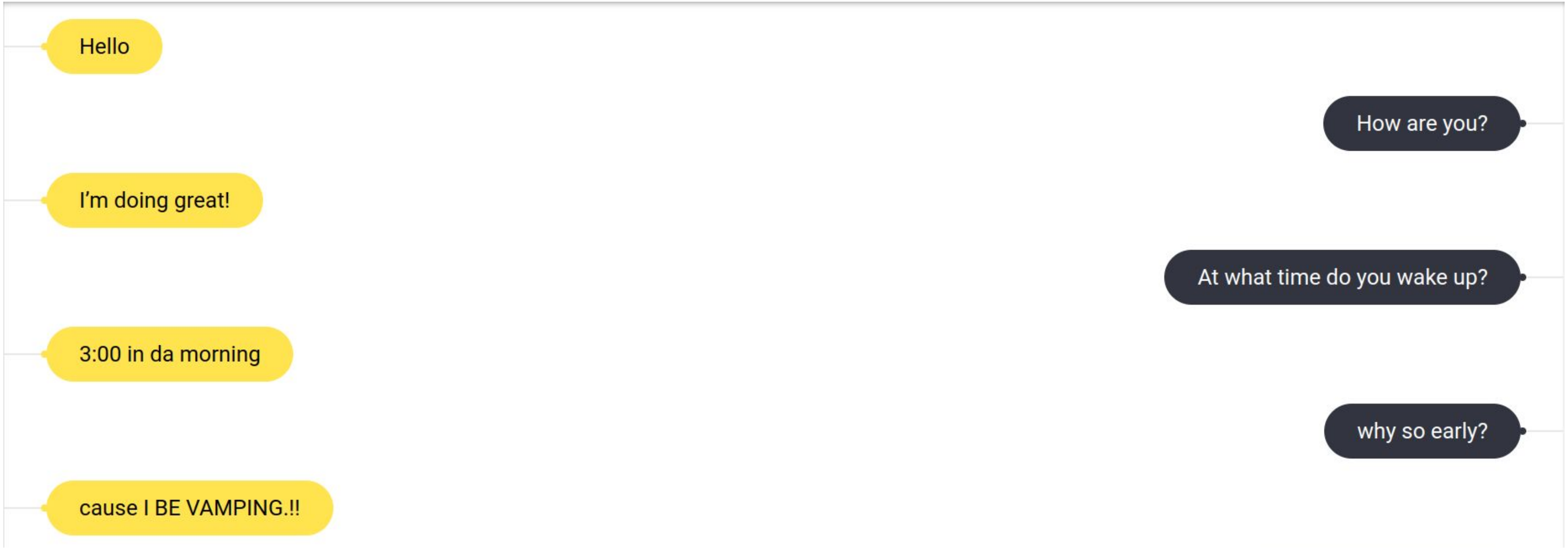
Finally! Image Captioning with RNN's



Natural Language Processing

Natural Language processing is the ability of a machine to interpret human language.

Chatbot


Hello
How are you?
I'm doing great!
At what time do you wake up?
3:00 in da morning
why so early?
cause I BE VAMPING.!!

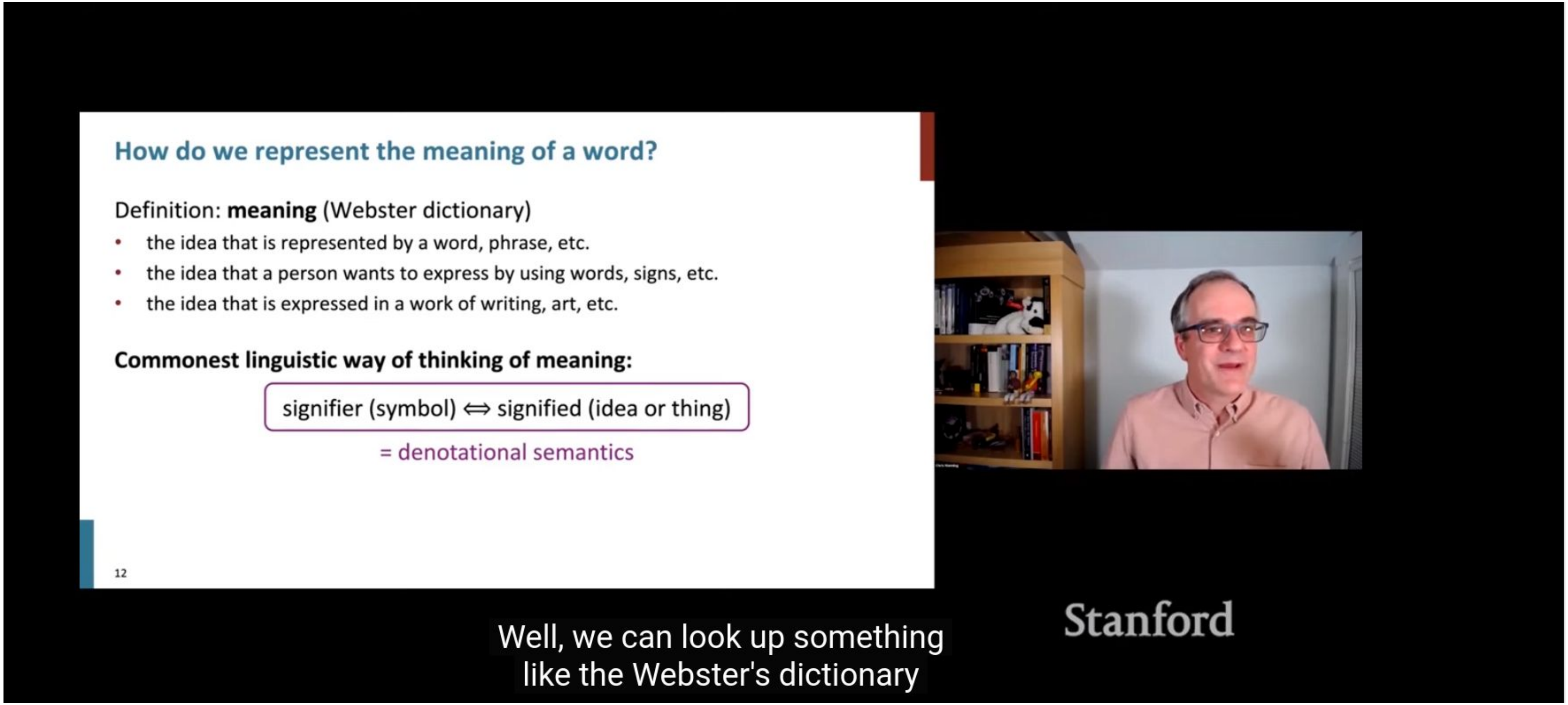
Tell Me This 20 hours ago (edited)

Human: What do we want!?
Computer: Natural language processing!
Human: When do we want it!?
Computer: When do we want what?

Reply · 203

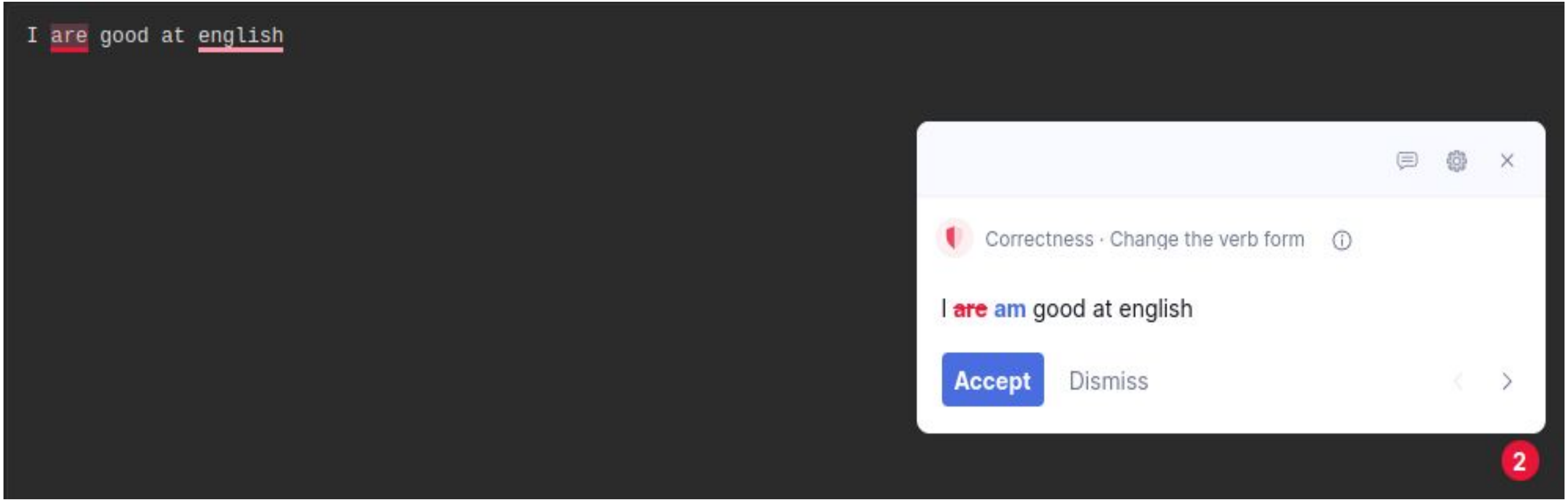
View reply

Speech to Text

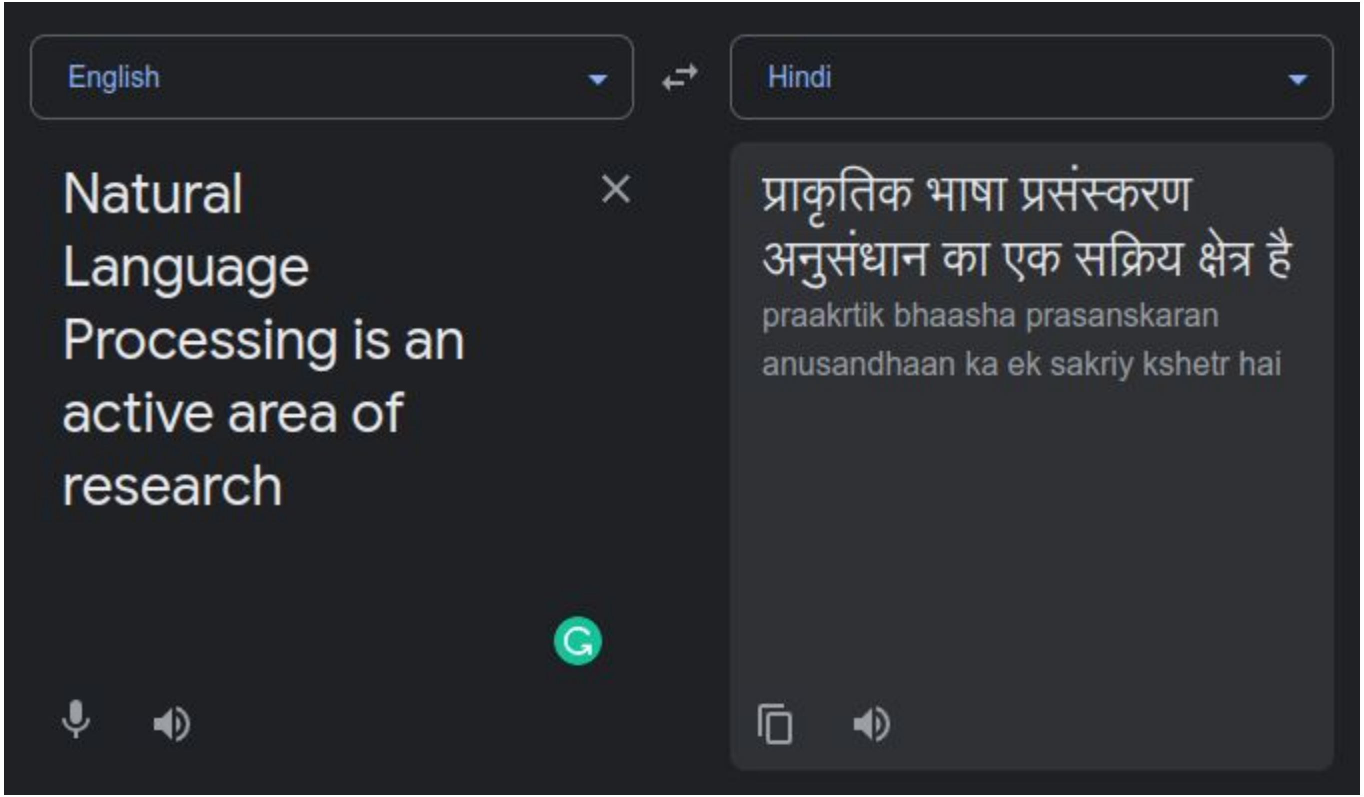


Question Answering

Spell Correction



Machine Translation

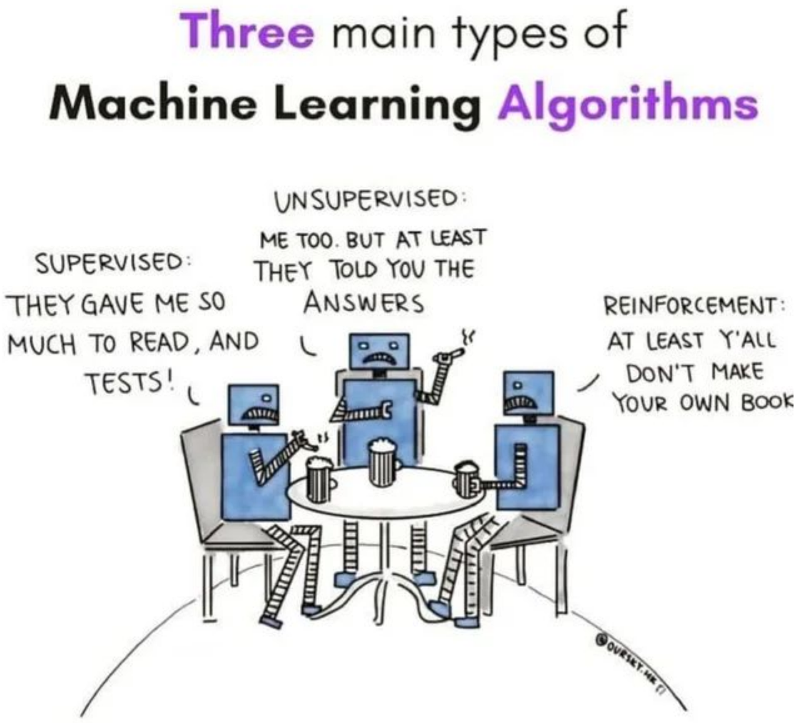


Closed Domain Question Answering

```
context = 'Alfred is a white cat and likes apples. Bobby is a black dog which loves mangoes.'
question = 'What does Alfred like?'
ground_truth = 'apples'
tsne_plot(context, question, ground_truth)
```

```
predicted answer :  apples
```

Reinforcement Learning



How did you learn to cycle

How did you learn to cycle

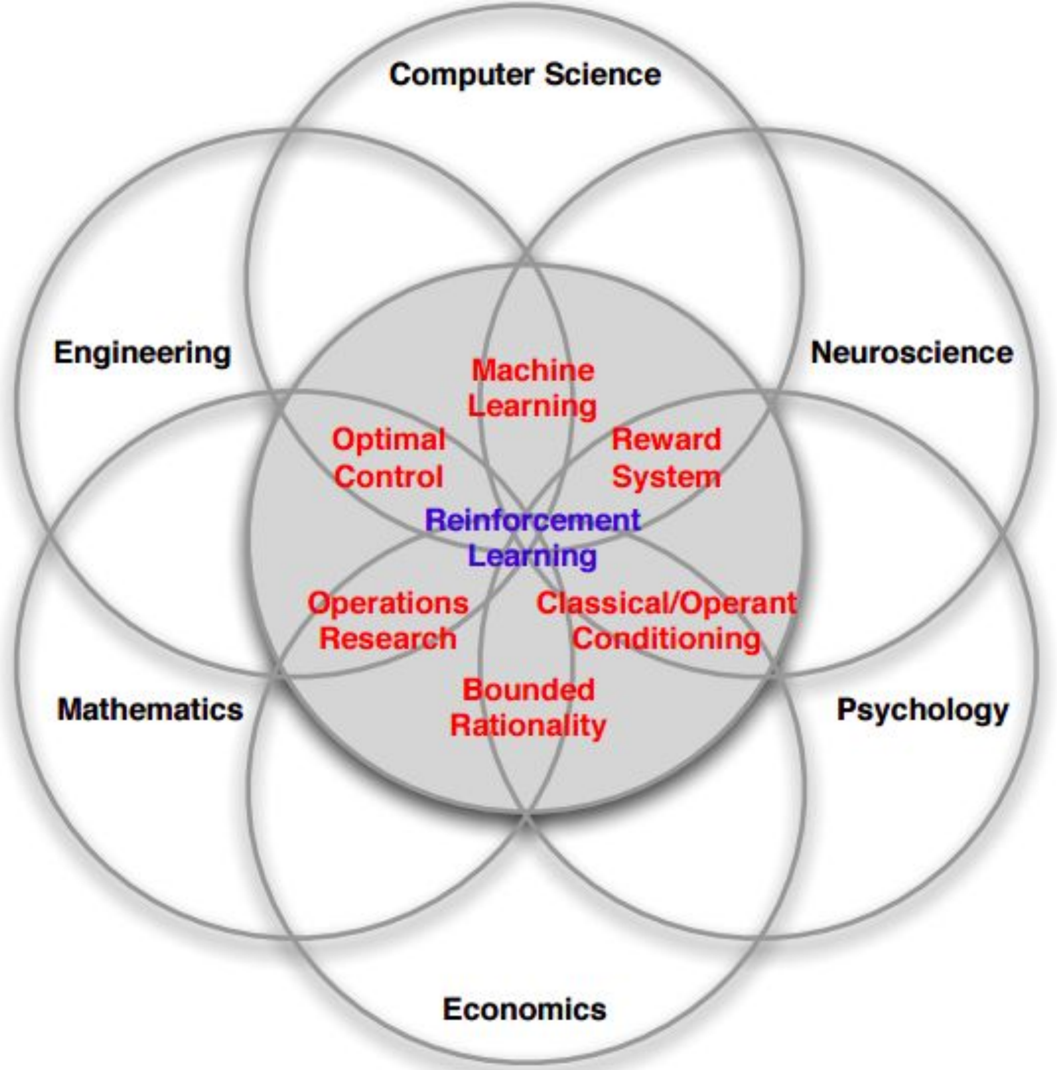
- Not supervised learning

How did you learn to cycle

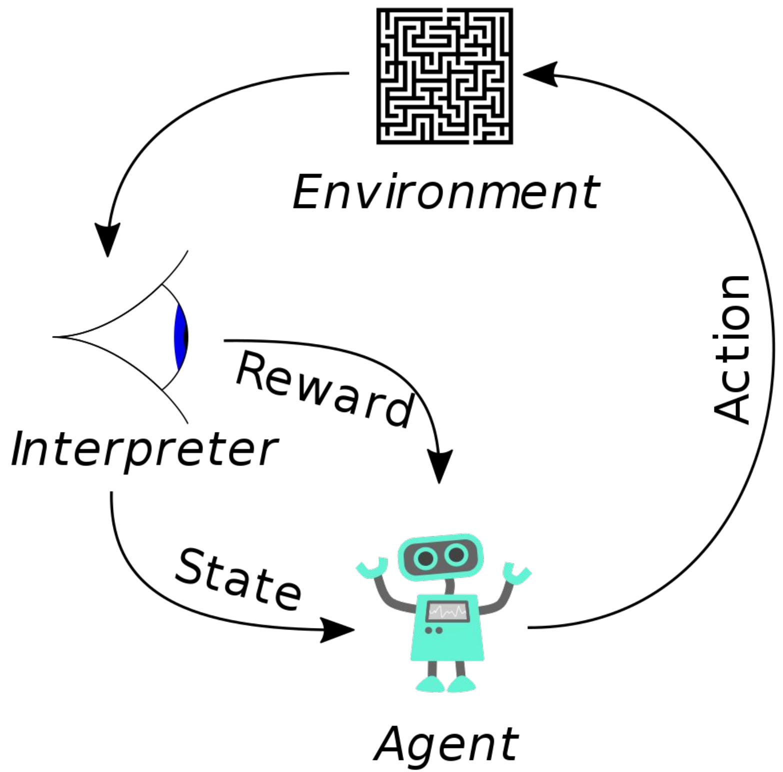
- Not supervised learning
- Not unsupervised learning

How did you learn to cycle

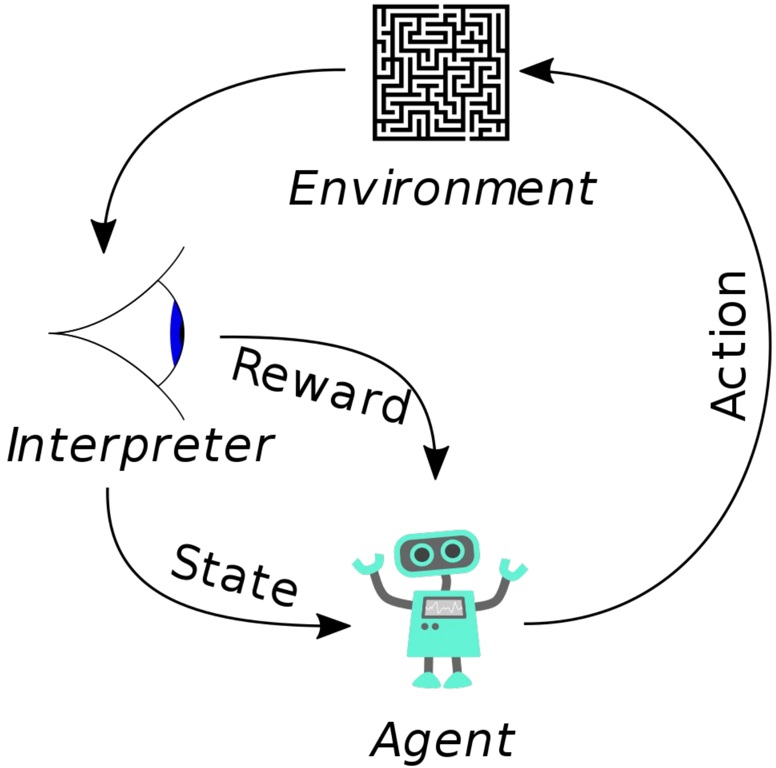
- Not supervised learning
- Not unsupervised learning
- Learning by Trial & Error


Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics

What is it?

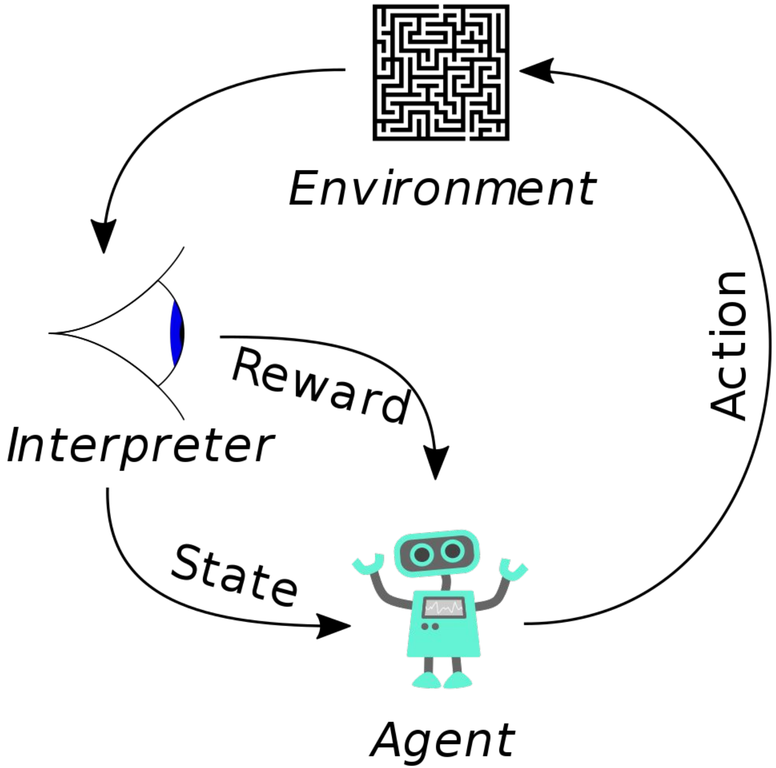


What is it?



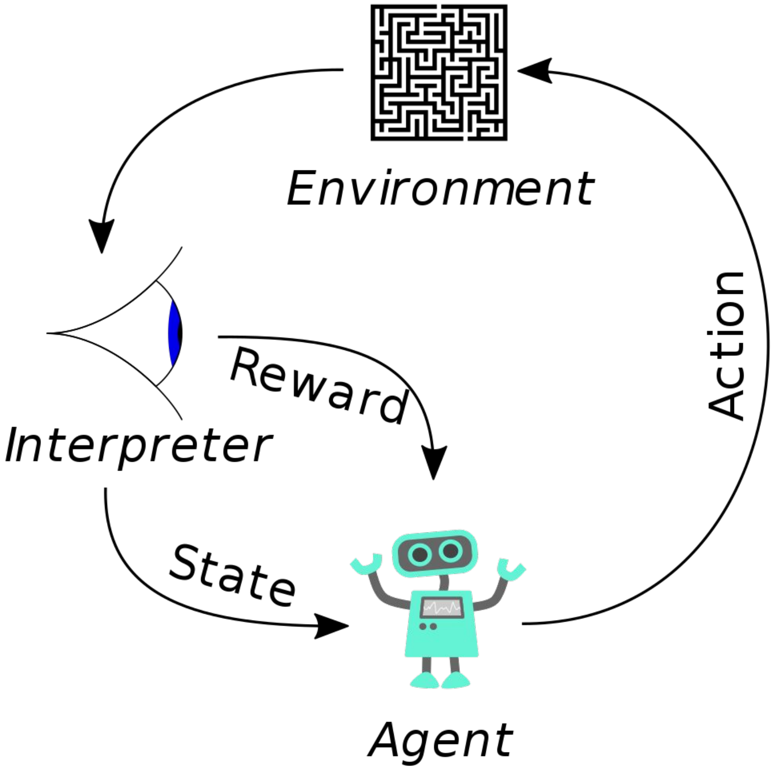
- A *policy* defines the learning agent's way of behaving at a given time

What is it?



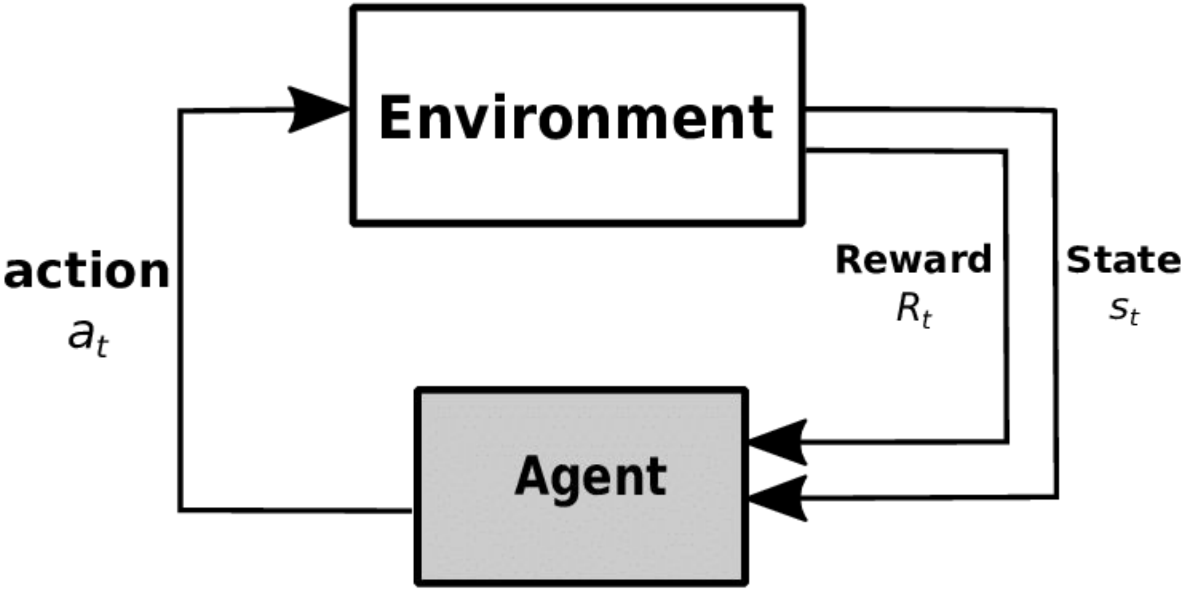
- A *policy* defines the learning agent's way of behaving at a given time
- A reward signal defines the goal of a reinforcement learning problem

What is it?

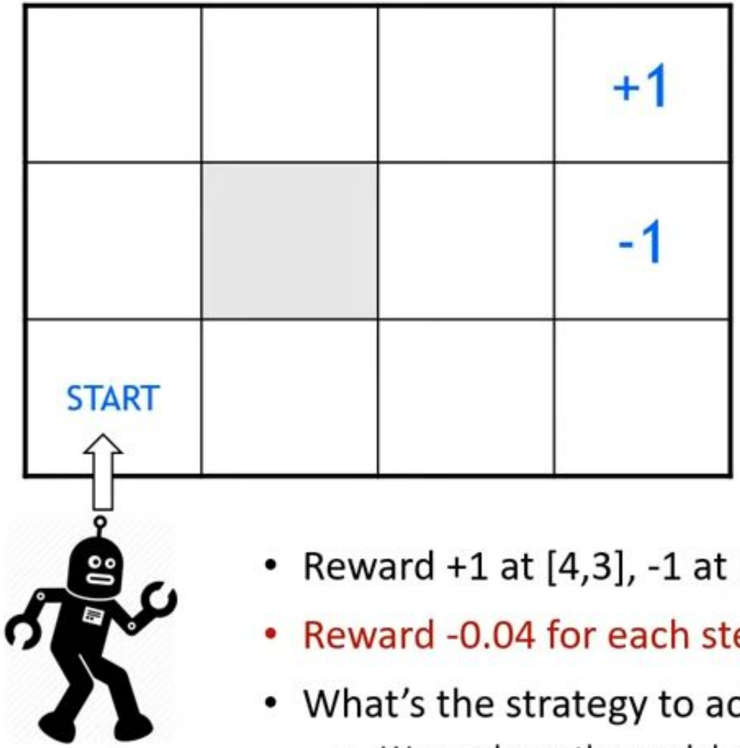


- A *policy* defines the learning agent's way of behaving at a given time
- A reward signal defines the goal of a reinforcement learning problem
- A *value function* specifies what is good in the long run

Reinforcement learning: An Example



Robot in a Room



actions: UP, DOWN, LEFT, RIGHT

(Stochastic) model of the world:

Action: **UP**

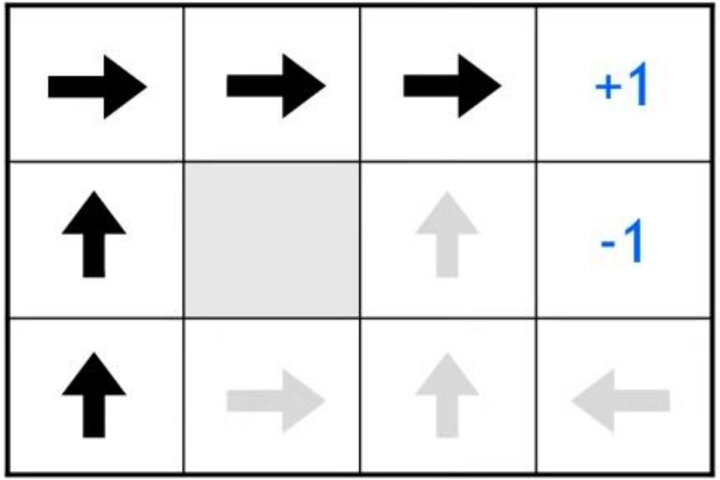
80% move UP
10% move LEFT
10% move RIGHT

- Reward +1 at [4,3], -1 at [4,2]
- Reward -0.04 for each step
- What's the strategy to achieve max reward?
  - We can learn the model and plan
  - We can learn the value of (action, state) pairs and act greed/non-greedy
  - We can learn the policy directly while sampling from it

Courtesy: MIT 6.S091: Introduction to Deep Reinforcement Learning (Deep RL)

Optimal Policy for a Deterministic World

Reward: -0.04 for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are deterministic:

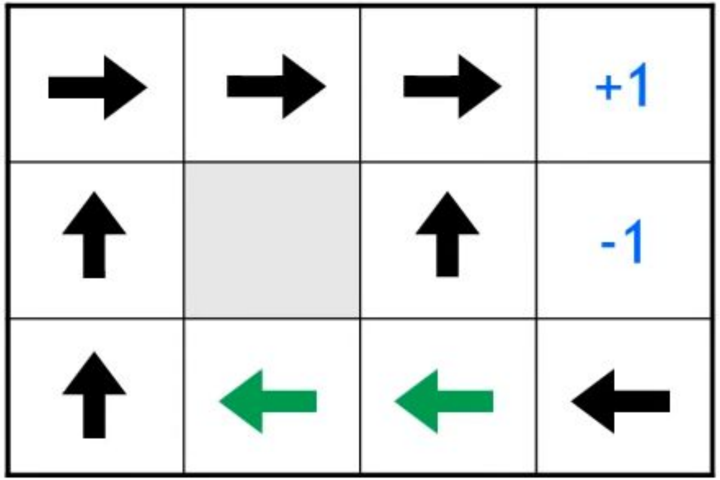
UP

100% move UP
0% move LEFT
0% move RIGHT

Policy: Shortest path.

Optimal Policy for a Stochastic World

Reward: **-0.04** for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

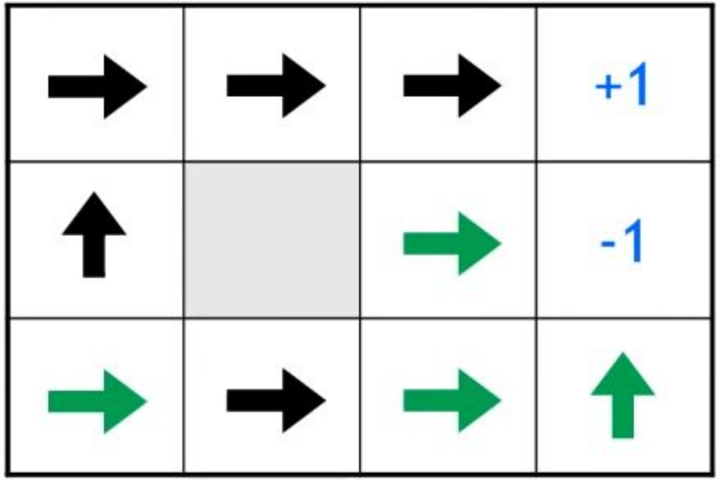
UP

80% move UP
10% move LEFT
10% move RIGHT

Policy: Shortest path. Avoid -UP around -1 square.

Optimal Policy for a Stochastic World

Reward: -2 for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

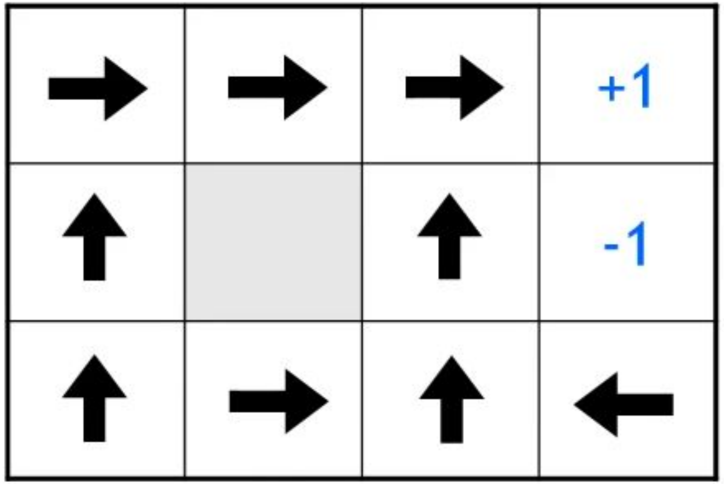
UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Shortest path.

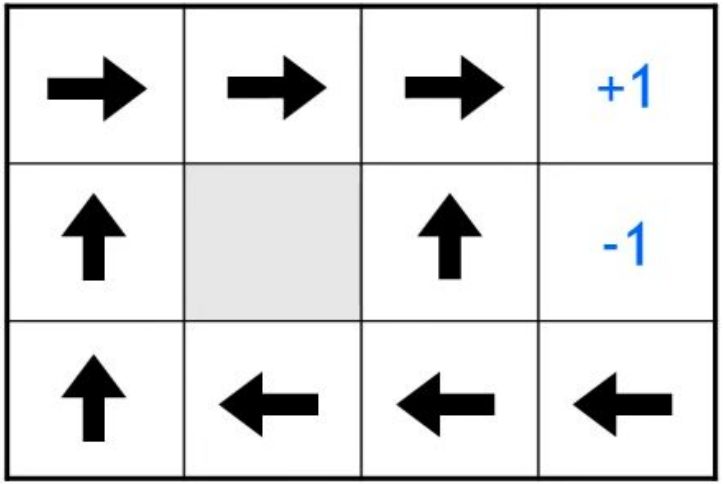
Optimal Policy for a Stochastic World

Reward: **-0.1** for each step



More urgent

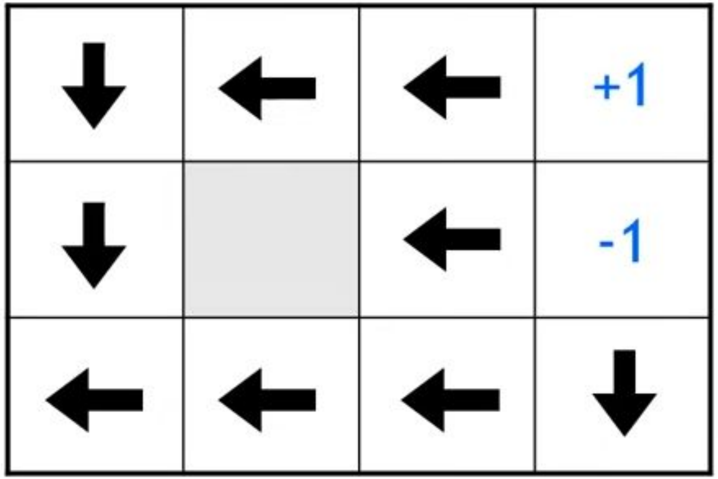
Reward: **-0.04** for each step



Less urgent

Optimal Policy for a Stochastic World

Reward: +0.01 for each step



actions: UP, DOWN, LEFT, RIGHT

When actions are stochastic:

UP

80%	move UP
10%	move LEFT
10%	move RIGHT

Policy: Longest path.

Conclusion

Important parameters

Conclusion

Important parameters

1) Environment

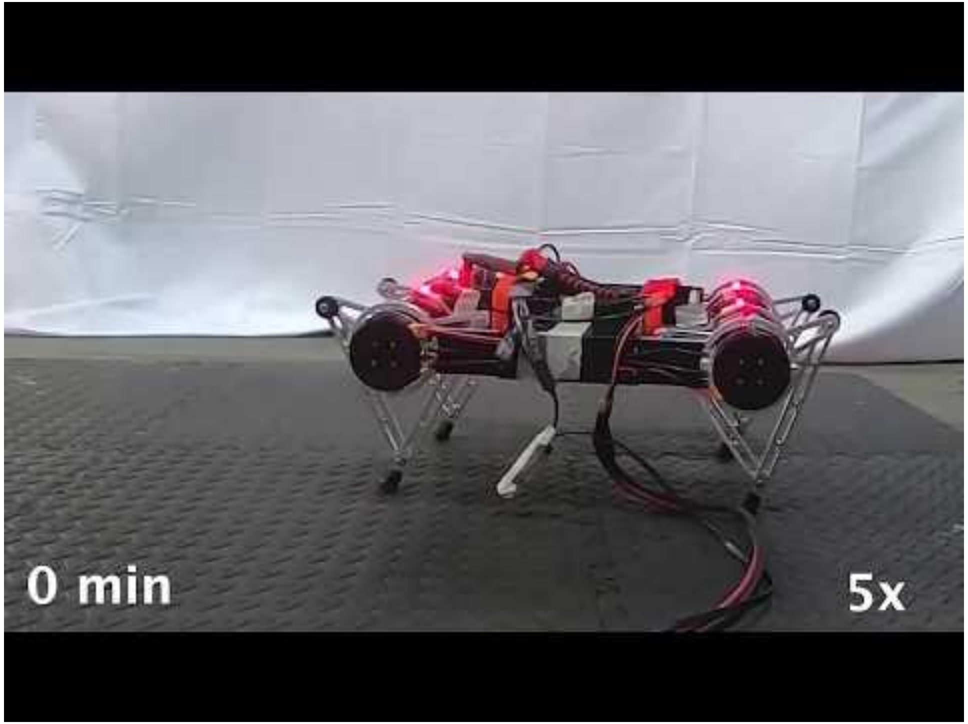
Conclusion

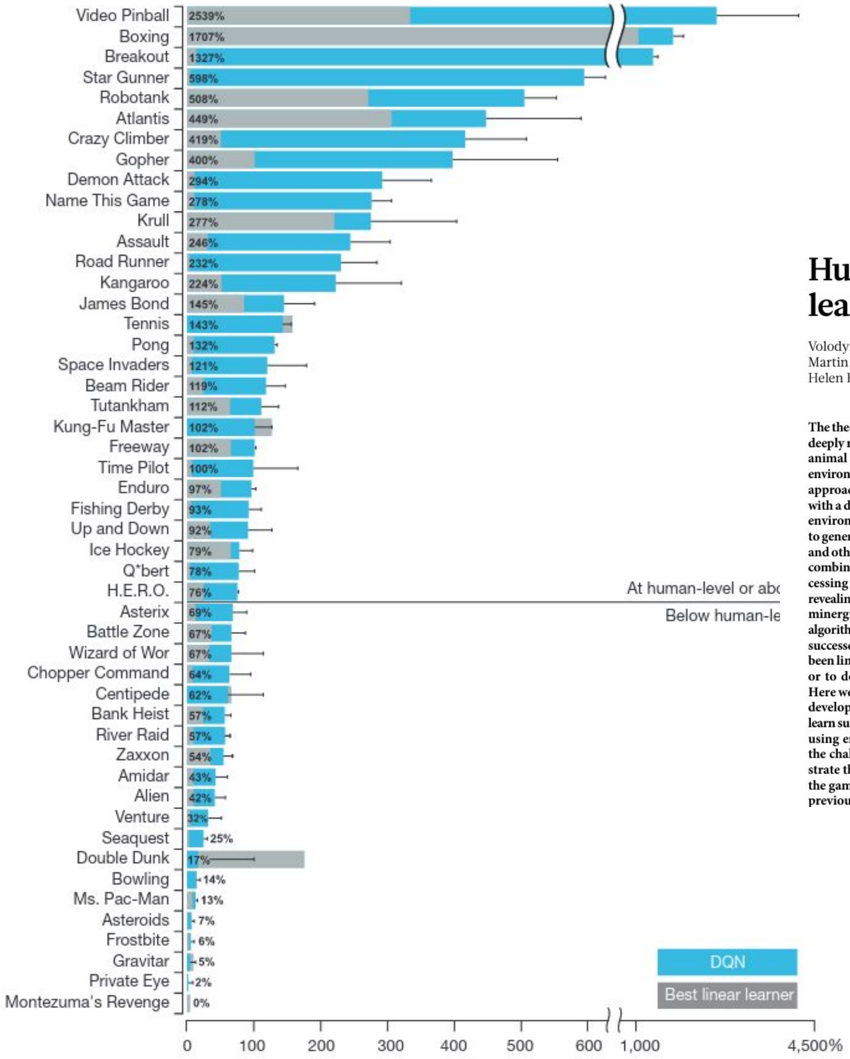
Important parameters

1) Environment
2) Reward

Applications

Applications





Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

The theory of reinforcement learning provides a normative account[1], deeply rooted in psychological[2] and neuroscientific[3] perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations. Remarkably, humans and other animals seem to solve this problem through a harmonious combination of reinforcement learning and hierarchical sensory processing systems[4,5], the former evidenced by a wealth of neural data revealing notable parallels between the phasic signals emitted by dopaminergic neurons and temporal difference reinforcement learning algorithms[3]. While reinforcement learning agents have achieved some successes in a variety of domains[6–8], their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. Here we use recent advances in training deep neural networks[9–11] to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. We tested this agent on the challenging domain of classic Atari 2600 games[12]. We demonstrate that the deep Q-network agent, receiving only the pixels and the game score as inputs, was able to surpass the performance of all previous algorithms and achieve a level comparable to that of a pro-

agent is to select actions in a fashion that maximizes cumulative future reward. More formally, we use a deep convolutional neural network to approximate the optimal action-value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s,\, a_t = a,\, \pi\right],$$

which is the maximum sum of rewards r_t discounted by γ at each time-step t, achievable by a behaviour policy $\pi = P(a|s)$, after making an observation (s) and taking an action (a) (see Methods)[19].

Reinforcement learning is known to be unstable or even to diverge when a nonlinear function approximator such as a neural network is used to represent the action-value (also known as Q) function[20]. This instability has several causes: the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and therefore change the data distribution, and the correlations between the action-values (Q) and the target values $r + \gamma \max_{a'} Q(s', a')$. We address these instabilities with a novel variant of Q-learning, which uses two key ideas. First, we used a biologically inspired mechanism termed experience replay[21–23] that randomizes over the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution (see below for details). Second, we used an iterative update that adjusts the action-values (Q) towards target values that are only periodically updated, thereby reducing correlations with the target.

While other stable methods exist for training neural networks in the

doi:10.1038/nature14236

Alpha Zero & Alpha Go

Why is this important?

Why is this important?

- IBM Deep Blue vs Gary Kasparov

Why is this important?

- IBM Deep Blue vs Gary Kasparov
- AlphaGo vs Lee Sedol. Is it any different?

Why is this important?

- IBM Deep Blue vs Gary Kasparov
- AlphaGo vs Lee Sedol. Is it any different?
- Ability to replicate intuitive pattern recognition is a big deal

Recent Progress in ML

Computer Vision / Image Recognition

  ImageNet, CNNs, Autonomous driving

Speech Recognition

  Voice assistants

Language Models

  GPT-3, PaLM

Game Playing / Deep Reinforcement Learning

  AlphaGo, AlphaZero

Text-to-Image Models

  Dall-E, ImaGen

How to approach Machine Learning?

1.) Python/R and libraries like NumPy, SciPy, Matplotlib, Seaborn, Scikit-Learn, Pandas, TensorFlow, PyTorch
2.) Mathematics for ML

    Probability & Statistics - Random variables, Expectations, Distributions

    Linear Algebra & Matrix Calculus - Gradients/Hessians, Eigenvalue/vector

3.) Understand the internals of ML algorithms
4.) Build ML applications to know which algorithms to use when
5.) Can read some ML research papers

Some good resources are mentioned at the last slide.

Resources!

1. 3b1b Series on Neural Network
2. CS229
3. CS244D
4. CS231N
5. CS224N
6. Pattern Recognition and Machine Learning: Bishop

Social Media

Follow our social media handles for regular updates

Instagram

Twitter

YouTube

WhatsApp