The Executives Guide To Artificial Intelligence

December 2020

Automatski

TABLE OF CONTENTS

INTRODUCTION

You must have heard that Artificial Intelligence is evolving exponentially fast and if allowed to go unchecked it will take over the world and "poses a fundamental risk to the existence of human civilization". And then there are others who call this fear mongering and baseless.

What is the reality?

Luckily for us A.I. has not yet been invented. Popular media and even some self proclaimed experts use the term A.I. to describe machines that use Machine Learning or Deep Learning. Or worse use the terms interchangeably as if they meant one and the same thing. The bottom line is that A.I. doesn't exist and has not been invented yet.

What we have today are machines, which have been taught to do some extremely specific niche tasks in a well-defined context. Moreover, the problem is that machines are not even able to do those tasks very well in real world situations. We believe this is contrary to what you might have heard in popular media. Which is drinking the A.I. kool aid.

Today A.I. is at the peak of its hype cycle. The next 100 or 1000 years of mankind will be defined by Quantum Computing, A.I. and the combination of the two.

This annual research report covers the entire evolution of the field and the reality of what we have today. What works, what doesn't and where we will go from here.

THE TERMINOLOGY

Classical A.I.

In the mid of the last century all efforts to create A.I. were based on solving problems like constraint solving, or logic processing using symbolic logic. Scientists had given up on such methods by mid of last century. It is only in the last 10 odd years that connectionist methods like neural networks have really taken the front row seat. The reason for the renewed interest in A.I.

Big Data

Big data is a field that treats ways to analyze, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with, by traditional data-processing application software. Data with many cases (rows) offer greater statistical power, while data with higher complexity (more attributes or columns) may lead to a higher false discovery rate. Big data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy, and managing and integrating data sources. Big data was originally associated with three key concepts: volume, variety, and velocity of data.

Data Engineering

The key to understanding what is data engineering lies in the "engineering" part. Engineers design and build things. "Data" engineers design and build pipelines that transform and transport data into a format wherein, by the time it reaches the Data Scientists or other end users, it is in a highly usable state. Data scientists focus on finding new insights from a data set, while data engineers are concerned with the production readiness of that data and all that comes with it: formats, quality, compression and storage, ingestion, latency and throughput, scaling, backups and archiving, merging, resilience, security, privacy and more.

Data Science

In data science, an algorithm is a sequence of statistical processing steps. That is used to model the data and the model is then used to make decisions about things of interest to us. Data Science is typically used in business to take decisions about product features, planning, allocation, customers etc.

Machine Learning

Machine learning is a branch of artificial intelligence (AI) focused on building applications that learn from data and improve their accuracy over time without being programmed to do so. Algorithms are 'trained' to find patterns and features in massive amounts of data in order to make decisions and predictions based on new but similar data. The better the algorithm, the more accurate the decisions and predictions will become as it processes more data.

Neural Networks

Neural Networks are networks of artificial neurons modelled after the neurons in the human brain. Such neural networks are trained on data to learn patterns between inputs and outputs. And then used in production scenarios to figure out the outputs given some similar input data.

Deep Learning

Deep Learning is another term for Neural Networks albeit with a lot of layers of neurons in it. Hence the origin of the word deep in deep learning. Compared to traditional neural networks which had 2 or 3 layers with just a few thousands of parameters. Deep Learning uses 50 or 100 or even 500 layers with billions of trainable parameters.

Reinforcement Learning

Reinforcement Learning is a set of techniques used to train agents in a defined environment to learn how to achieve a given goal. By training it by giving it rewards and punishments for its actions. Reinforcement Learning has been used to teach agents to play games like Chess and Go, to learn to walk etc.

Neurosymbolic Computing

Neither symbolic logic based methods, nor deep learning based methods have led to AI. Each of them has severe limitations. The current hype around Deep Learning is dying down faster than anyone realizes in the popular media, industry or social networks. The scientific community believes that a combination of Deep Neural Networks (Deep Learning) and Classical Symbolic A.I. will lead to AI. This is the crux of all efforts going forward.

Quantum - Machine Learning, Deep Learning, Neural Networks

This category of algorithms is a collection of algorithms from Machine Learning, Deep Learning, Neural Networks re-implemented over Quantum Computers in the hope of achieving quadratic or exponential acceleration over classical implementations so far.

Artificial Intelligence

Artificial Intelligence is defined as the entire universe of computing technology that exhibits anything remotely resembling human intelligence.

Leading AI textbooks define the field as the study of "intelligent agents". Any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.

The textbook definition above is the reason behind the confusion around AI. Since, that definition easily matches techniques in Neural Networks, Deep Learning and Machine Learning. While the former definition separates AI from all of those and creates a distinct separation between them and AI.

Artificial General Intelligence

AGI is defined as the holy grail. It is the hypothetical intelligence of a machine that has the capacity to understand or learn any intellectual task that a human being can.

We can say that machines with AGI would deserve to be called a separate species. And only these machines can be an existential threat to the human race. And nothing mentioned earlier can come close to it. While earlier techniques can be used to create weapons that can wipe out the human race. But the controlling entity behind those weapons would be other humans.

WHY IS EVERYONE TRYING TO CREATE A.I.?

It's a race for world domination. Pure and simple. Nothing less. And everyone knows that the country, organization or individual that creates and controls A.I. and Quantum Computing is going to rule the planet or mankind for the next centuries or even millennium. That's why.

That's the only reason everyone is spending billions of dollars and racing to create AI and Quantum Computers. And even trying to combine them both, the goal is world domination. Nothing less.

Elon Musk says – "If one company or a small group of people manages to develop godlike digital super-intelligence, they could take over the world."

The thing is that he is right on this one. And that's what this is all about.

HOW HAS THE LAST CENTURY TURNED OUT FOR A.I.?

In the history of artificial intelligence, an ***AI winter*** is a period of reduced funding and interest in artificial intelligence research. The term was coined by analogy to the idea of a nuclear winter. The field has experienced several hype cycles, followed by A.I. Winters or disappointment, criticism, and funding cuts, followed by renewed interest years or decades later.

There have been multiple A.I. Winters in mankinds journey to attempt to create A.I. so far.

Hype is common in many emerging technologies, such as the railway mania or the dot-com bubble. The AI winters earlier were a result of such hypes, due to over-inflated promises by developers, unnaturally high expectations from end-users, and extensive promotion in the media.

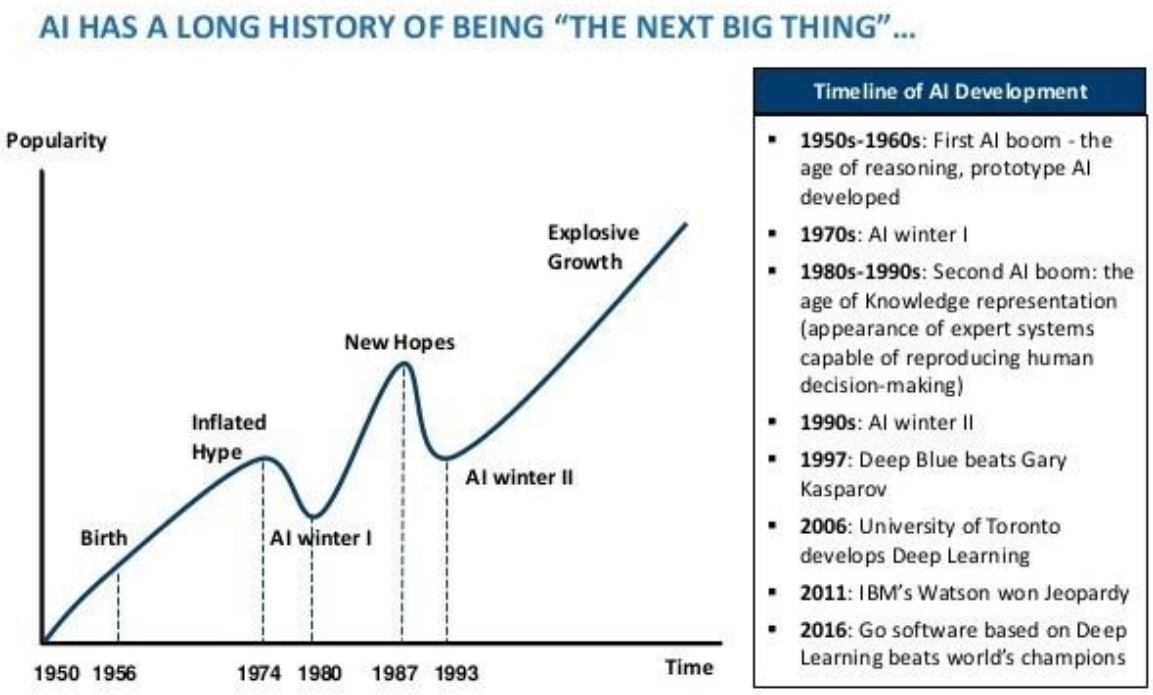


Figure 1- Two A.I. Winters In The Last Century

A.I. has seen multiple winters in the last century.

THE CURRENT HYPE CYCLE (AKA THE CURRENT A.I. SPRING)

Enthusiasm and optimism about AI has generally increased since its low point in the early 1990s. Beginning about 2010. Interest in deep learning and reinforcement learning from the research, investor and corporate communities led to a dramatic increase in funding and investment.

Deep Learning Breakthroughs

1. 2010 Speech Recognition
2. 2013 Image Recognition, Classification, Vision, Deep Fake Generation
3. 2015 Natural Language Processing, Translation, Text Generation, Q&A

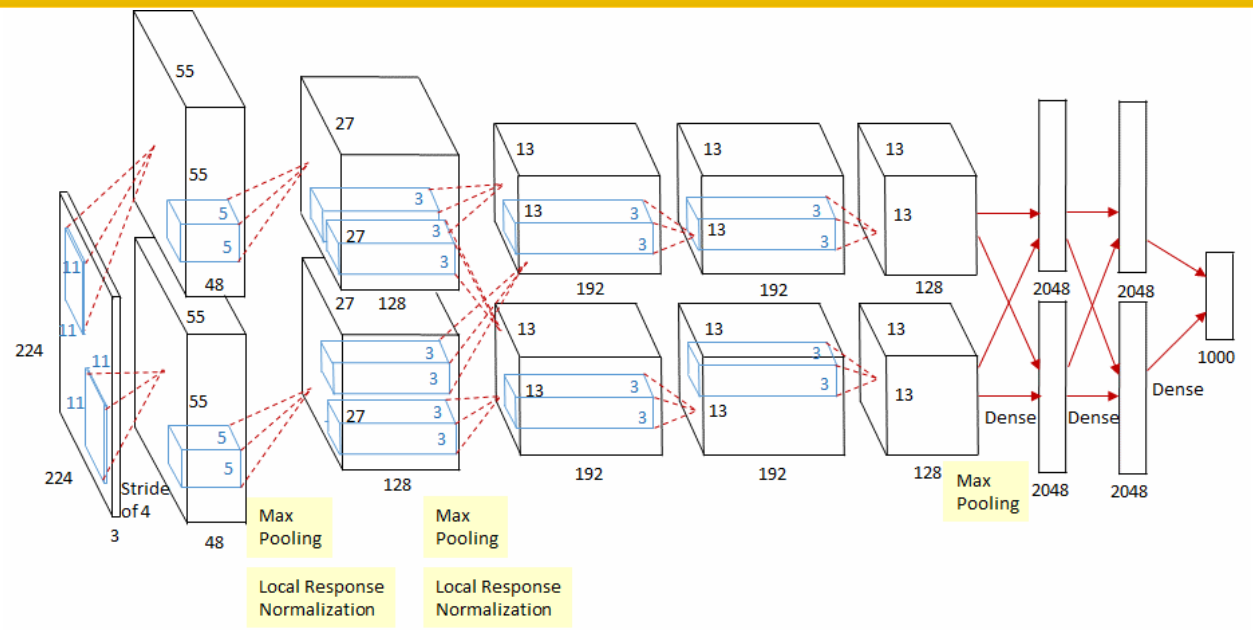


Figure 2- AlexNet

AlexNet Starts Deep Learning Boom (2012)

AlexNet, a GPU implemented CNN model designed by Alex Krizhevsky, wins Imagenet's image classification contest with accuracy of 84%. It is a huge jump over 75% accuracy that earlier models had achieved. This win triggers a new deep learning boom globally.

Reinforcement Learning Breakthroughs

1. 2015 Onwards - Playing The Game of GO and Atari Video Games, Walking or Flying Robots, Partially Autonomous Vehicles
2. 2020 – Game Generation

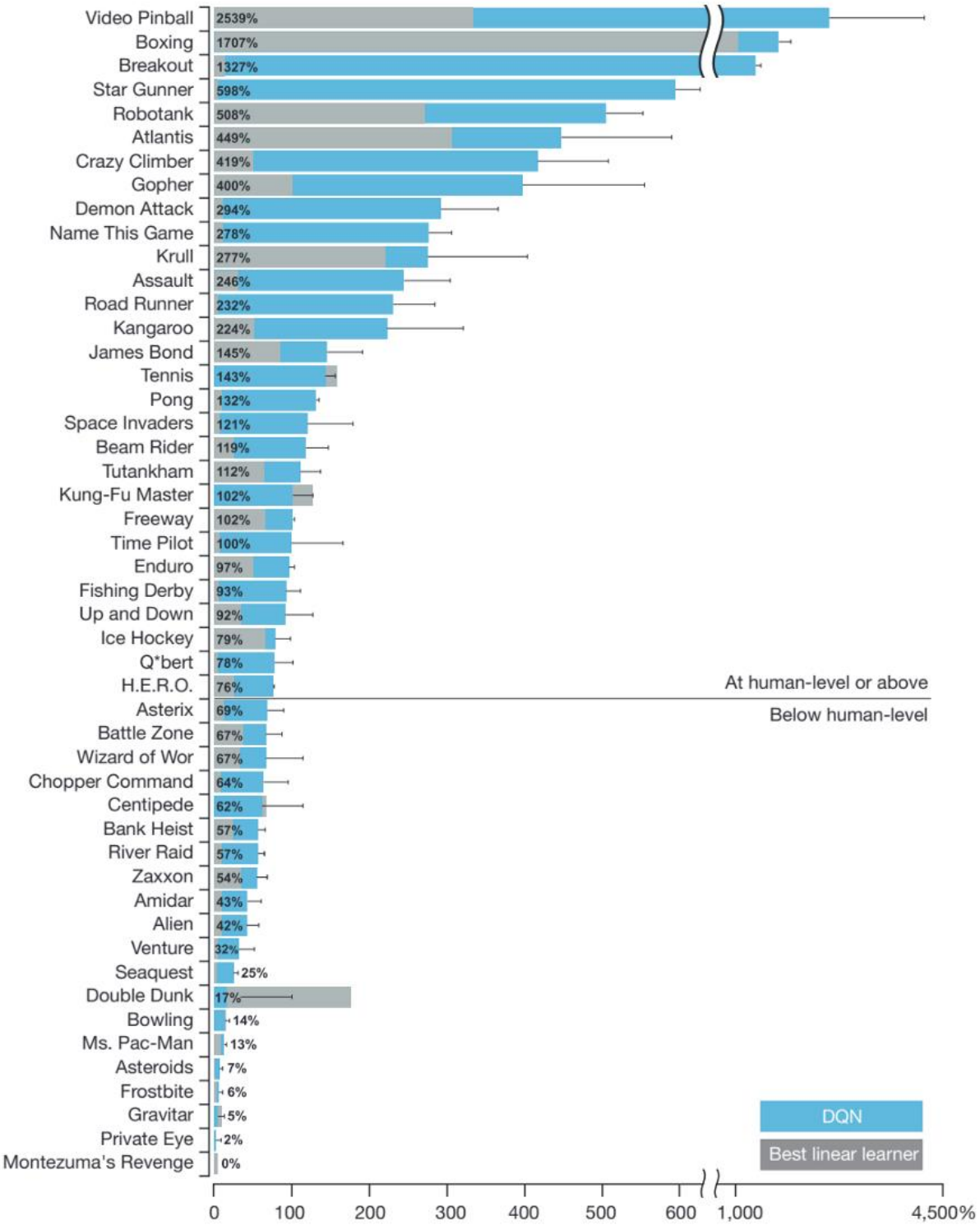


Figure 3- Atari Games Results - DeepMind DQN Nature Paper

AlphaGo Beats Human (2016)

Deepmind's deep reinforcement learning model beats human champion in the complex game of Go. The game is much more complex than chess, so this feat captures the imagination of everyone and takes the promise of deep learning to a completely new level.

Figure 4- AlphaGo Beats World Champion At GO

Deep Learning made a comeback in 2010 for a couple of reasons

- Availability of lots of data for training – Big Data
- Availability of lots of computational power – GPU's (Computer Graphics Cards) & Cloud Computing

The core techniques haven't changed that much in the last 30-40 years. Since 2010 there have been some incremental breakthroughs though, that built upon earlier core principles.

This has created a huge divide between the have's and have nots. Big firms have access to a lot of computational infrastructure and have been almost exclusively been able to train and create incrementally better and larger models. While in principle the techniques have been democratized and publicly available. The difference in budgets and computational infrastructure required to train models has created a monopoly like situation. Training language generator GPT-3 is estimated to have cost OpenAI $10 to $12 million—and that does not include cost of developing and training prototypes. Which is at least one or two orders of magnitude larger. OpenAI is funded in billions of dollars. Only a tiny handful of big tech firms can afford to do that kind of work. Most firms and startups cannot afford the exponential rising cost of AI models.

That said, there has been a stark difference in the way research is being done across the world since 2010.

- A lot of research now in Machine Learning, Deep Learning, Reinforcement Learning etc. is accompanied by code on github.com or university or company websites with free commercial use licensing – Open Source
- And in a lot of cases they are also accompanied also pre-trained models
- Almost all papers are published on arxiv.org or university or company websites – Open Access (Freely Accessible)
- A lot of presentations are free for all – Free Online Webinars or on Youtube.com
- A lot of cutting edge research and university courses are available and taught for free or a nominal price – Udemy, Coursera, Edx etc.

THE CULMINATION OF THE CURRENT HYPE CYCLE

…and the probable onset of A.I. Winter to conclude the current hype cycle.

Fathers of Deep Learning - Yoshua Bengio, Geoffrey Hinton, and Yan LeCun were awarded the Nobel Prize equivalent in Computing aka The Turing Award in 2019.

Figure 5- Yoshua Bengio, Geoffrey Hinton, Yann LeCun

Yoshua Bengio is a professor at the University of Montreal and Scientific Director at Mila and Quebec's Artificial Intelligence Institute;

Geoffrey Hinton is the VP and Engineering Fellow of Google, Chief Scientific Adviser of The Vector Institute, and a professor at the University Professor Emeritus at the University of Toronto; and

Yann LeCun is the Chief AI Scientist at Facebook and a professor at New York University.

The three scientists have known each other for over 30 years and sometimes have worked, both together and separately, in taking Deep Learning to the level as we see it today.

THE FAILURES OF AI

Even after billions of dollars spent on Deep Learning. The world still doesn't have any Self Driving Cars. There are no voice agents smart enough to assist us with specific things we want them to do for us, who really understand and execute our instructions. Even after so many breakthroughs in Vision and Image Recognition, there are still no reliable systems that don't make blunders or fatal mistakes. Our Robots are literally perfect when it comes to Kinematics (the ability to move) but they are still missing the one key ingredient i.e. 'Intelligence'

The Top Firms dedicated to Deep Learning have had little or no commercial successes so far in solving any real world production grade use case. Despite Global budgets for Deep Learning being in billions of dollars and salaries of researchers in millions of dollars.

Is an A.I. winter coming?

In all probability. Yes!

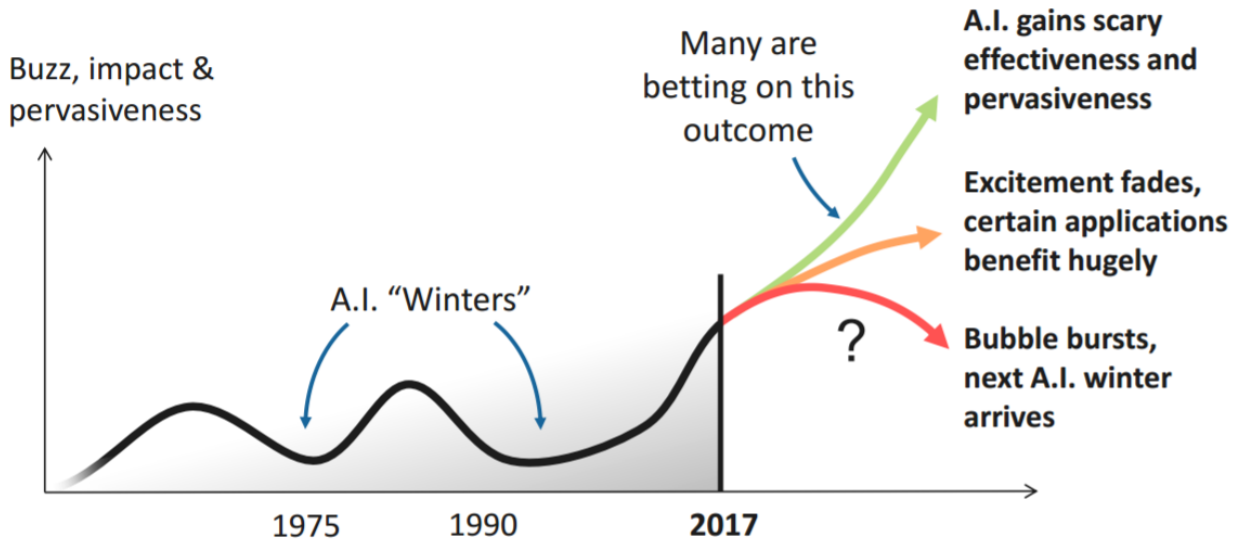


Figure 6- Is An A.I. Winter Coming?

Deep Learning and Reinforcement Learning has hit a plateau. Vendors have not been able to meet the inflated expectations of the industry and prospective consumers. Maybe an A.I. Winter is around the corner.

WHAT ARE THE FATHERS OF DEEP LEARNING SAYING NOW?

Yan LeCun is basically saying that the future of Deep Learning is in Unsupervised Learning (or what he calls Self Supervised Learning). Where no one has to train machines specifically. They will 'automagically' learn by themselves.

Obviously that is the ultimate dream of creating A.I. but how will it work is the ultimate question?

Yan LeCun says using Energy Based Methods in Deep Neural Networks, and something called Adversarial Training will deliver us unsupervised learning and consequently A.I.

Geoffrey Hinton has basically said on multiple occasions that the human brain doesn't work like Deep Neural Networks. And backpropagation which is at the core of training Deep Learning Networks has no counterpart in the human brain. There is no global end to end training in our human brain either.

Everything in our brain works locally and yet together in aggregate delivers intelligence. All this (Deep Learning) will never take us to A.I. and we will have to throw it all and start all over again, says Hinton.

Turing Award winner Geoffrey Everest Hinton is on a quest to understand learning in the brain and he sees unsupervised learning as the future of AI. From his development of Stacked Capsule Autoencoders and a Simple Framework for Contrastive Learning of Visual Representations, or SimCLR, to his exploration of Neural Gradient Representation by Activity Differences, or NGRADs, Dr. Hinton is on a quest to discover the algorithms that enable us to think, remember and learn.

Yoshua Bengio is also talking about deep unsupervised learning, which is using deep learning principles to learn in an unsupervised way, meaning without human guidance or labels on the data that is fed into AI systems to train them.

However he says, many hard problems remain open, like how to allow AI systems to automatically learn high-level semantics—in other words, how to represent more abstract concepts and the meaning of not just words but more complex thoughts, such as those we express in a phrase, a sentence or a paragraph.

Again, which is the entire dream of creating A.I. Nobody knows how to make all this work and create A.I.

A.I. COMPLETE

The reality is that nobody has a definition of Artificial Intelligence. All definitions fallback to making a vague comparison to Human Beings. It is A.I. if it is similar to Human Intelligence. It is A.I. if it can convince us that it is Human in a blind fold test.

The first proposed test for A.I. was the Turing Test…

> The Turing test, originally called the imitation game by Alan Turing in 1950, is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. Turing proposed that a human evaluator would judge natural language conversations between a human and a machine designed to generate human-like responses. The evaluator would be aware that one of the two partners in conversation is a machine, and all participants would be separated from one another. The conversation would be limited to a text-only channel such as a computer keyboard and screen so the result would not depend on the machine's ability to render words as speech. If the evaluator cannot reliably tell the machine from the human, the machine is said to have passed the test. The test results do not depend on the machine's ability to give correct answers to questions, only how closely its answers resemble those a human would give. (Ref: Wikipedia)

Another test proposed by Steve Wozniak (Co-Founder Apple) was The Coffee Test…

> In this test, a robot would be challenged to perform a very simple task – to enter your home, find the coffee machine, and brew a cup of coffee by pushing the right buttons.

Yet, another test was The Make Us Laugh Test.

> Which basically challenged a machine to try and make humans laugh.

With that same line of thought researchers have defined…

A.I. Complete

The most difficult problems in A.I. (which is basically almost all of them) are informally known as AI-complete or AI-hard, implying that the difficulty of these computational problems, assuming intelligence is computational, is equivalent to that of solving the central artificial intelligence problem—making computers as intelligent as people, or strong AI. To call a problem AI-complete reflects an attitude that it cannot be solved by a simple specific algorithm such as the ones we have today.

AI-complete problems are hypothesised to include computer vision, natural language understanding, and dealing with unexpected circumstances while solving any real-world problem. But its pretty much covers all of A.I. Efforts.

This definition is an analogy with Complexity Theory which has Complexity Classes like NP-Complete, NP-Hard.

WHAT IS A.I.?

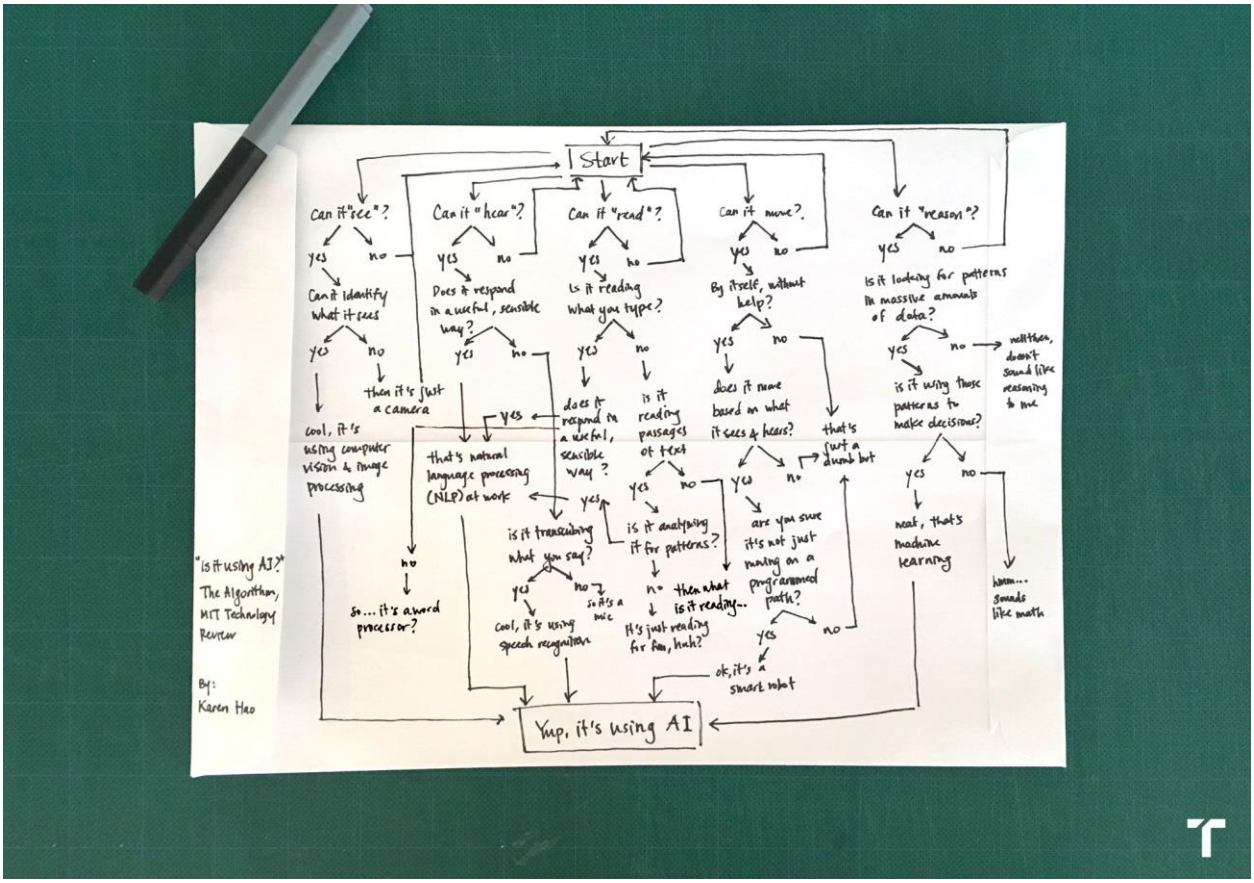


Figure 7- MIT's Definition Of A.I.

MIT starts with the Questions

1. Can it See?
2. Can it Hear?
3. Can it Read?

4. Can it Move?
5. Can it Reason?

If we sit back and think clearly. The ability to See, Hear, Read or Move is not Intelligence per se. But rather "Channels" of Input and Output for an Intelligent System. They are like tactical abilities that Intelligence exploits to deliver Intelligent Behavior. And in a sense are NOT a core Definition of Intelligence. We think that MIT erred in its definition.

Which leaves us with the last point "Can it Reason?"

So, what is the best definition of A.I.?

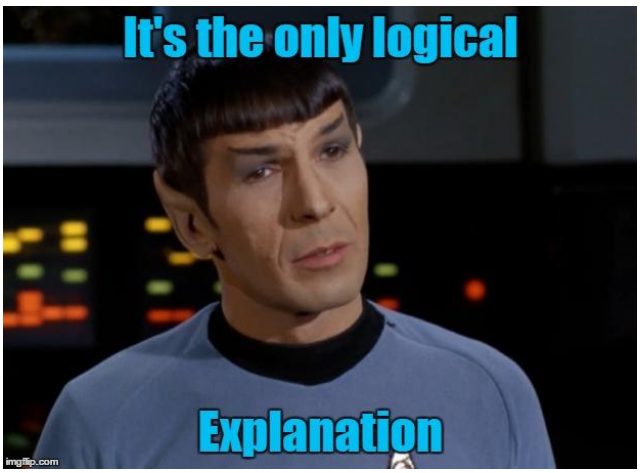


Figure 8- The Best Definition Of A.I.

A.I. whether implemented through connectionist mechanisms or symbolic logic or some cognitive algorithm or a mixture of them. Is A.I. only if it is Logically Perfect. That is what we think of A.I. at its core. And which when not achieved leads us to disappointment.

At Automatski the definition of A.I. we use is 'Logically Perfect' something like Mr. Spock from Star Trek (TV Series)

Critics jump to point out that based on experiments "Humans don't use Logic to do things and Solve Problems. So Logic shouldn't be a part of A.I."

There is a huge fallacy to that argument according to us. Firstly all the definitions of A.I. outside Automatski refer to Humans to define Intelligence. We don't. Secondly Humans are a mess, they are hardly logical or right most of the times. e.g. only 0.0001% of the human population is trained in Critical Thinking. The rest pretty much think and do whatever comes to their mind. Depending on anecdotes, emotional outbursts, temperament, ego or perhaps love.

Thirdly we are at that moment in history where we are defining Intelligence (esp. in Artificial Systems). We have a chance to define and build something near perfect. Fourthly, defining Intelligence as Logically Perfect simplifies the route to the elusive A.I. we all are hoping to create. And that's the reason behind our reasonable success at Automatski. We took a first principles based view of the entire problem called A.I.

ARTIFICIAL GENERAL INTELLIGENCE

Some people prefer to define and use subclassifications within A.I.

1. Artificial narrow intelligence (ANI) (Weak Intelligence), which has a narrow range of abilities like vision, speech etc.;
2. Artificial general intelligence (AGI) (String Intelligence), which is on par with human capabilities; or
3. Artificial superintelligence (ASI), which is more capable than a human.

Though it doesn't necessarily make sense to make yet another set of terminology in comparision with humans. And it might not be very useful either.

One of the most famous attempts at creating A.I. (or what some called A.G.I. We at Automatski make no such distinction. We simply call A.I. …. A.I.) is …

OpenCog

OpenCog is an architecture for AGI, based on a hypergraph knowledge store called the Atomspace. The system enables multiple AI algorithms, including neural nets, logic engines and evolutionary learning systems, to cooperate synergistically in learning and reasoning from and updating this knowledge graph.

It's based on a mathematical theory of general intelligence, which tells us how the general nature of general intelligence manifests itself in the specific case of human-like cognition. In the scope of AI approaches it would be considered a 'hybrid cognitive architecture' due to its integration of multiple AI algorithms based on different AI paradigms. However, it is different from other hybrid cognitive architectures in the depth of integration of the different algorithms and in its grounding in the mathematical theory of AGI.

At a high level OpenCog Prime's cognitive architecture is based on cognitive psychology and cognitive neuroscience. Most cognitive functions are distributed across the whole system, yet principally guided by some particular module.

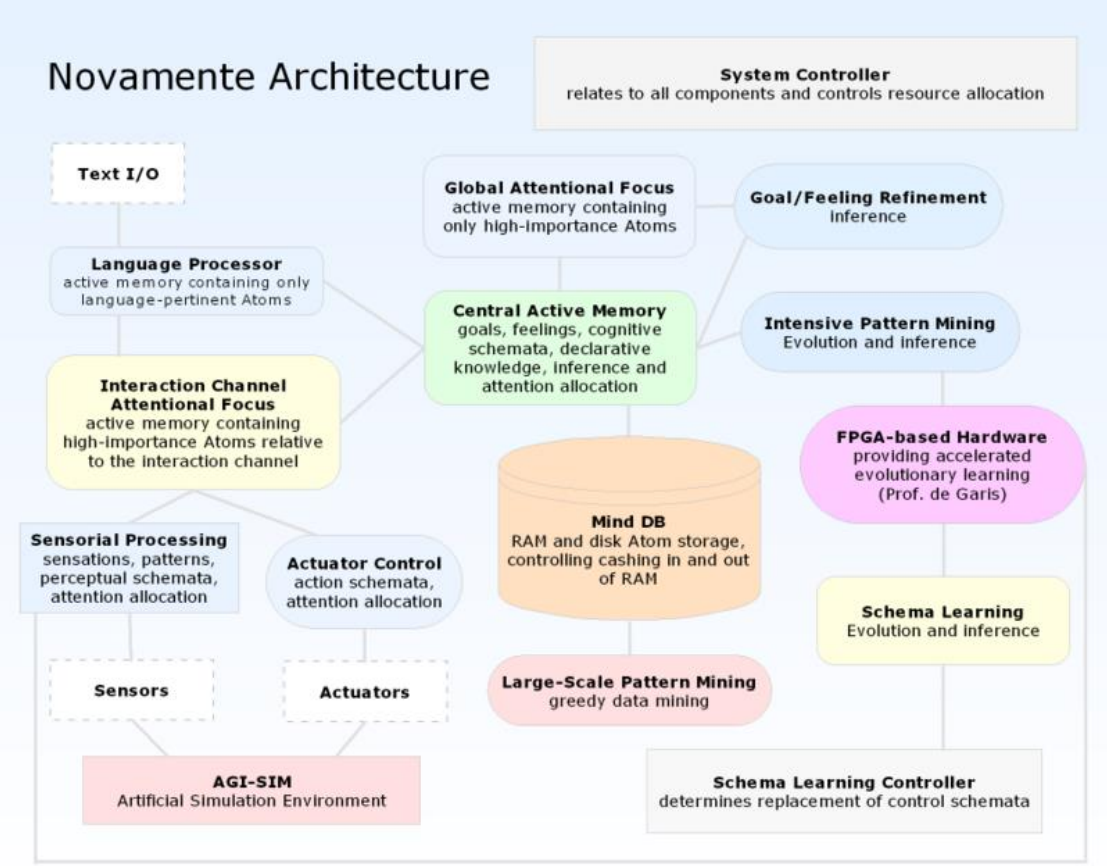


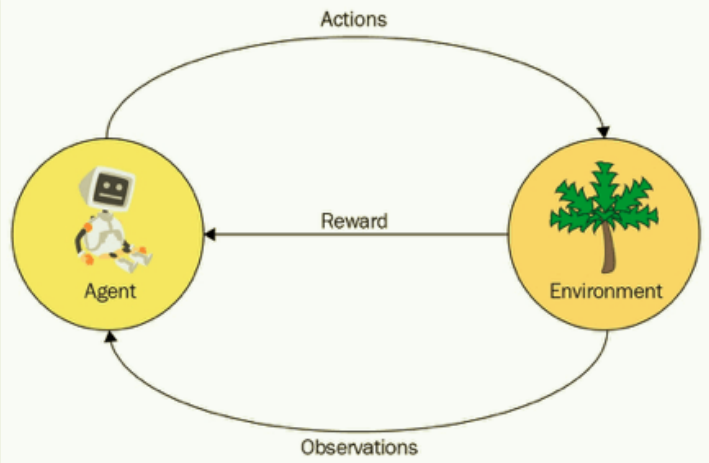
Figure 9- OpenCog - An Attempt At Creating A.I. (of A.G.I.) (Ref: OpenCog)

The meat of the design is in what lies inside the boxes, and how the insides of the boxes dynamically interact with each other and give rise to system-wide attractors and other emergent dynamic phenomena.

Researchers claim that OpenCog Hyperon (the latest version of OpenCog) can be taught and trained with their TrueAGI framework.

At the core OpenCog is basically an Ensemble of Algorithms. An attempt to solve A.I. using a collection of Approximate Methods.

THE LIMITATIONS AND FAILURES OF REINFORCEMENT LEARNING



Reinforcement learning (RL) continues to be **less** valuable for business applications than supervised learning, and even unsupervised learning. It is successfully applied only in areas where huge amounts of simulated data can be generated, like robotics and games.

The second problem with RL is that it requires an exponential number of samples to learn from. We say it is sample inefficient. But that's an understatement. In reality while RL is pretty generic and can be applied to almost any problem. It doesn't work except for simple scenarios. If you throw anything complicated or a Real World problem at it. You will have to wait for the universe to end before it can finish learning from gazillions of samples it requires to learn from. And since the Real World Time cannot go any faster. The learning process will not even get over but the universe might have come to an end before that.

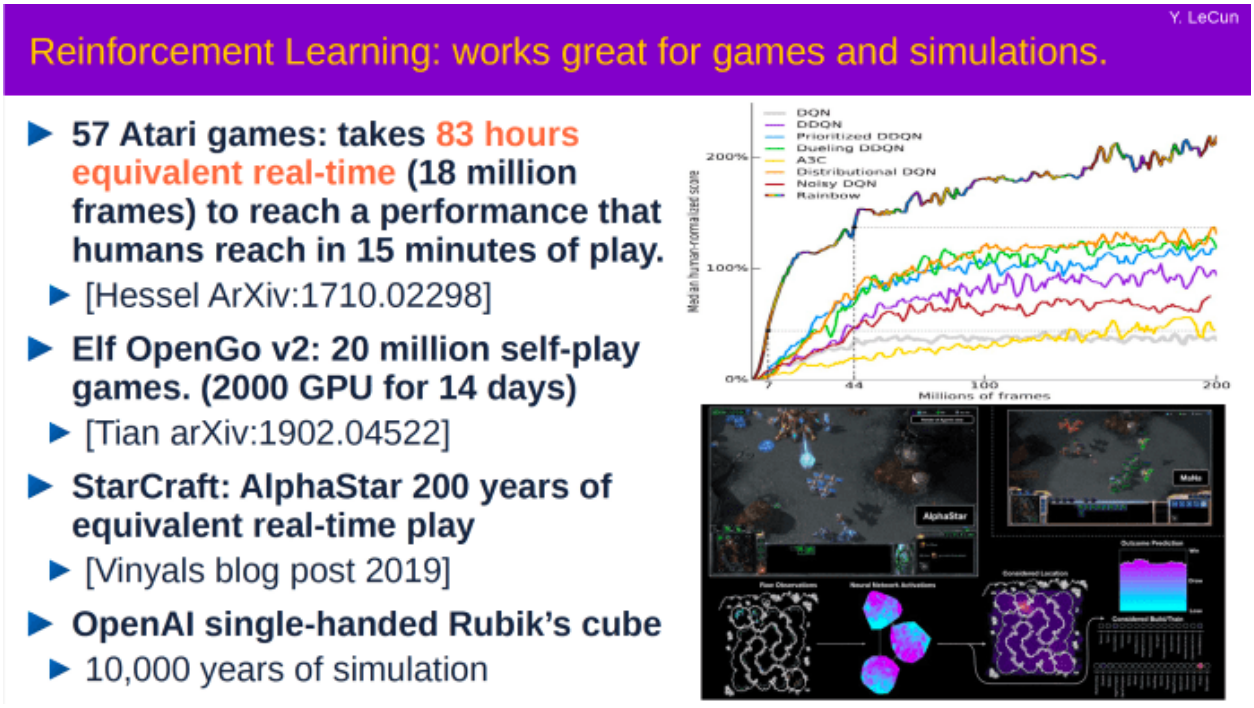


Figure 10- Reinforcement learning agents must be trained on hundreds of years' worth of session to master games, much more than humans can play in a lifetime (source: Yann LeCun).

Why does [model free] reinforcement learning fail in the real world?

Because the real world has millions of parameters the algorithm can learn from. You need millions/gazillions of samples to train it. And you can run a game on a computer very fast and train it. Thats why it works so well with games.

But you can't run the real world any faster than time. And some of the things you might try might kill you. And hence you cannot train a reinforcement learning system on the real world. - Yan lecun

The Possible Solution

Is to use Model based Reinforcement learning. The agent has a world simulator built inside it. Which it uses to learn from.

But this is easier said than done. Now the bigger problem with this is that the World Simulator is nearly impossible itself to train and create. As the world is very unpredictable and complex. And since we cannot do that very effectively we consequently cannot train Agents using Reinforcement Learning for Real World Problems.

THE LIMITATIONS AND FAILURES OF DEEP LEARNING

Deep Neural Networks are nothing but function approximators. They approximate an unknown non-linear function that maps inputs to outputs. 90% of all use cases of Deep Neural Networks either explicitly or implicitly under the hoods is Classification (Given an input which output class does it belong to)

Needless to say, contrary to any claims, hyperbole or marketing jargon. Deep Learning has and will fail miserably in 'all' situations that require one or few accurate globally optimum solutions. The problem gets exponentially compounded when the domain is large and the solution search space is in gazillions, which is the case with scientific problems.

Even in cases of problems where approximate solutions can work. Deep Neural Networks offer no reliability or assurance. They are prone to biases, adversarial attacks, errors and outright blunders.

The way Deep Learning works is that once you train them, you throw some (inputs) at them and they will throw a solution back at you. You simply have no other option but to take the solution at face value. If it says the person in the camera is John Doe, you don't know if its right or wrong and if its not Jane Doe, which it will be in reality a few times. If it says turn left you might as well turn left even if the road seems to be turning right. Or you can turn right. But how do you ever know.

Lastly, absolutely none of the Top Deep Learning Firms have had absolutely any commercial successes. They have made some exotic claims. But the researchers remain divided over such claims of breakthroughs. “These are just expensive advertisements of cool vendor technology. The claims are not replicable or verifiable” say critics.

The ultimate enthusiasm for Deep Learning will depend on what is delivered. And so far the situation doesn't look very positive.

If you think Vision was a solved problem, look at ObjectNet.

Almost all algorithms are trained on ImageNet which is a standard benchmark of sorts for vision algorithms. But it has a very serious limitation. It contains images in very convenient and similar viewing angles. Now why is that a problem?

Neural networks become confounded when they face objects from angles that are different from their training examples.

Enter ObjectNet. It extends ImageNet by including more classes and objects from different viewing angles. The best Neural Networks are completely flummoxed to see objects in ObjectNet. They simply don't work.

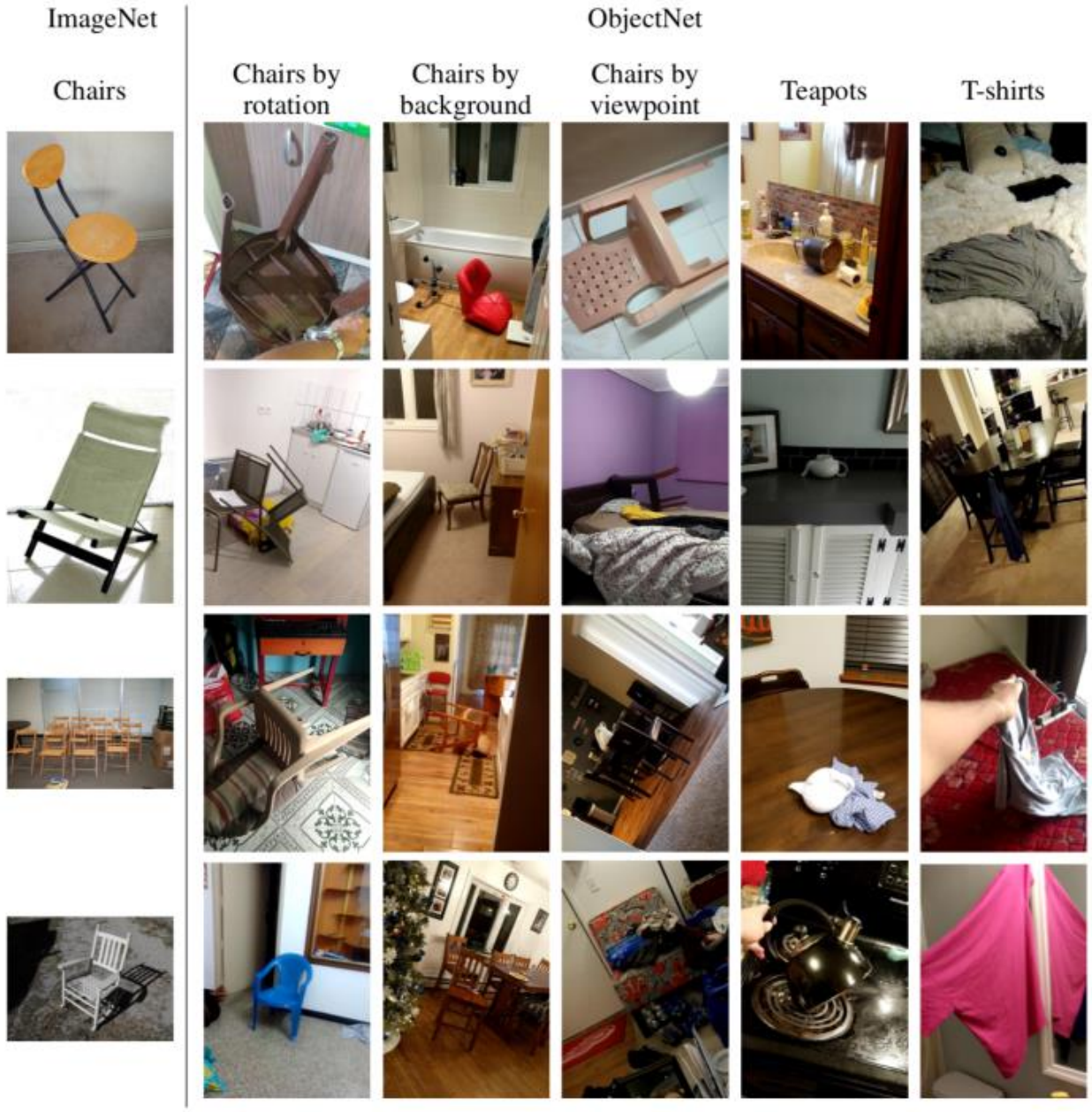


Figure 11- ImageNet vs ObjectNet

It seems we still are nowhere even in Vision algorithms. Which is the same story across the board.

The Story of Uber

Automobile vendors have been making extraordinary claims about the self driving capabilities of their vehicles without delivering much to match. Uber was pinning all its hopes of achieving profitability on its Self Driving efforts at their division called Advanced Technologies Group (ATG). They wanted to eliminate the human drivers and achieve profitability by reducing human costs and offering 24x7 taxi services. And this effort was of utmost importance to them.

San Francisco-based startup Aurora will purchase ATG from Uber for $4 billion. ATG was valued at $7.25 billion in the spring of 2019, the discounted sale of its self-driving car division represents both a relief and a setback for Uber. The company is cutting itself loose following years of effort spent on a project that has cost the company hundreds of millions of dollars and untold legal trouble. Without any success in sight.

Uber's decision to sell off its driverless cars unit highlights problems inherent to the self driving car efforts globally i.e. they don't work.

DEEP LEARNING FRAMEWORKS

Figure 12- Deep Learning Frameworks

ONNX

ONNX is an open format built to represent machine learning models. ONNX defines a common set of operators - the building blocks of machine learning and deep learning models - and a common file format to enable AI developers to use models with a variety of frameworks, tools, runtimes, and compilers.

ONNX is an essential ingredient to engineering machine learning and deep learning solutions.

DEEP LEARNING (AKA DEEP NEURAL NETWORKS)

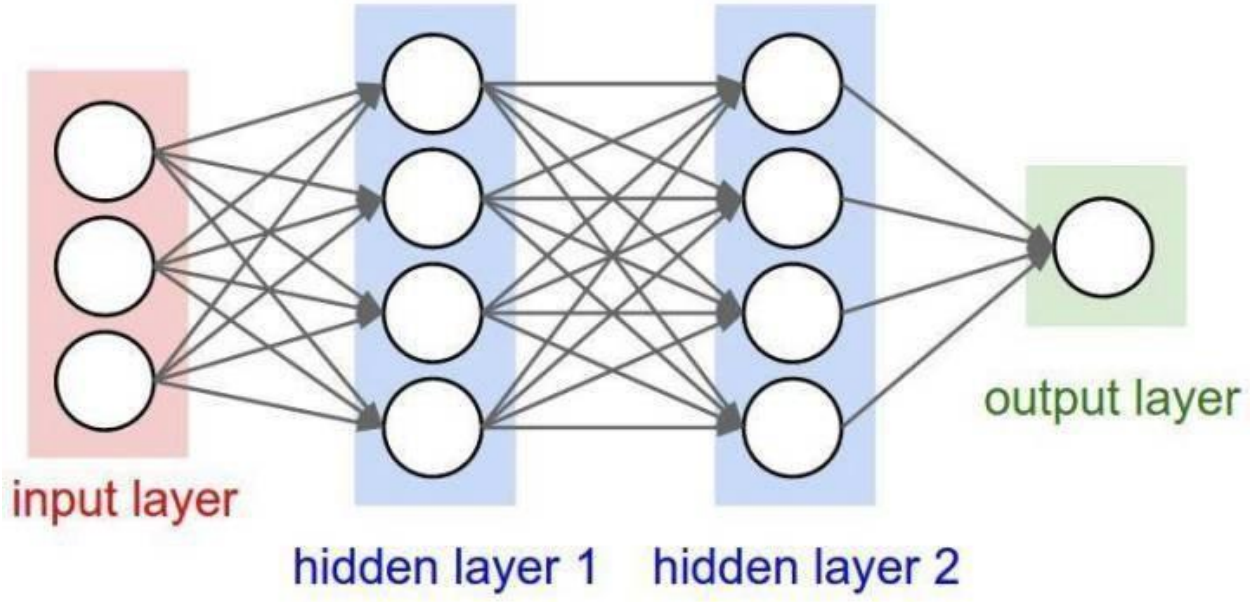


Figure 13- Neural Networks

Neural Networks are modelled after the Human Brain. They use a simple analogy of Neurons. A simple Neural Network has an input layer which takes the input parameters (e.g. colors of all pixels of an image). Then there are some hidden layers aka intermediate layers and then the output layer which outputs the value (e.g. a classification – cat, dog, mouse etc.)

Suppose you are an Airline Operator and have a lot of historical Flight Ticket Pricing Data. And want to train a Deep Neural Network that when given some parameters will calculate/output the price of the tickets.

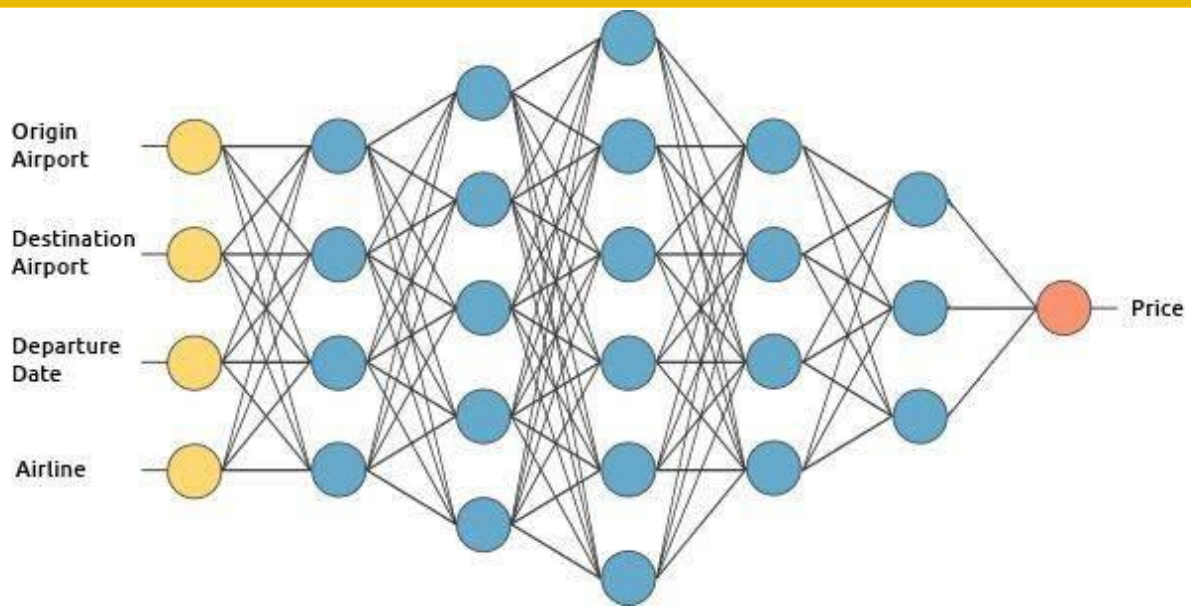


Figure 14- Flight Ticket Pricing Deep Neural Network

Each connection between neurons is closely related to weight, which determines the importance of the input values. When we start the initial weights are randomly set.

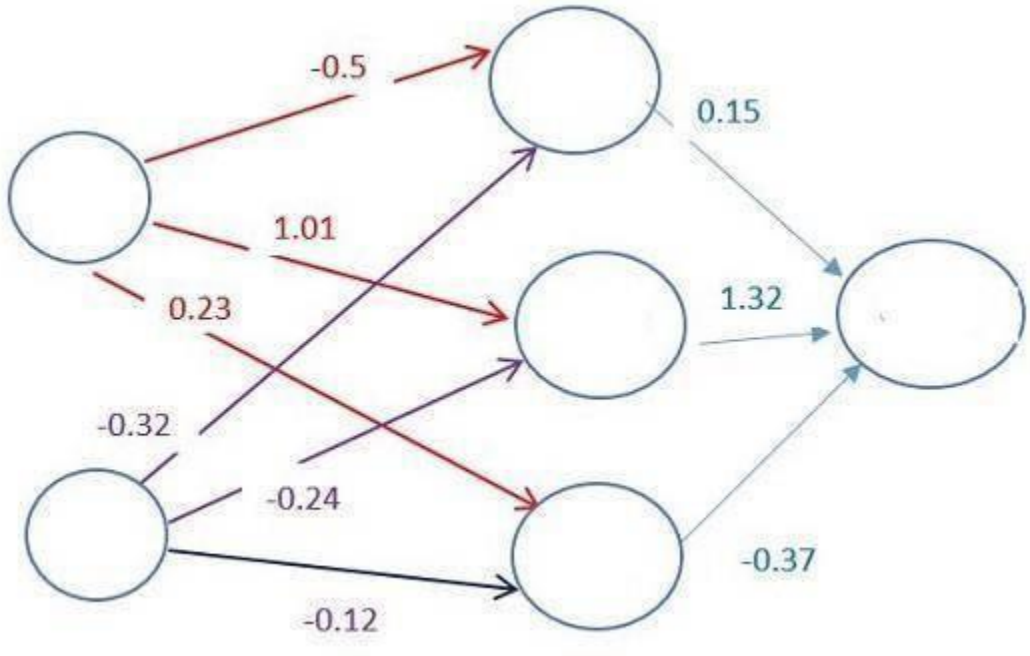


Figure 15- Weights In A Neural Network

Each Neuron also has an Activation Function which normalizes the neurons output between e.g. -1.0 and +1.0. There are many types of activation functions each with its own uniqueness and benefits.

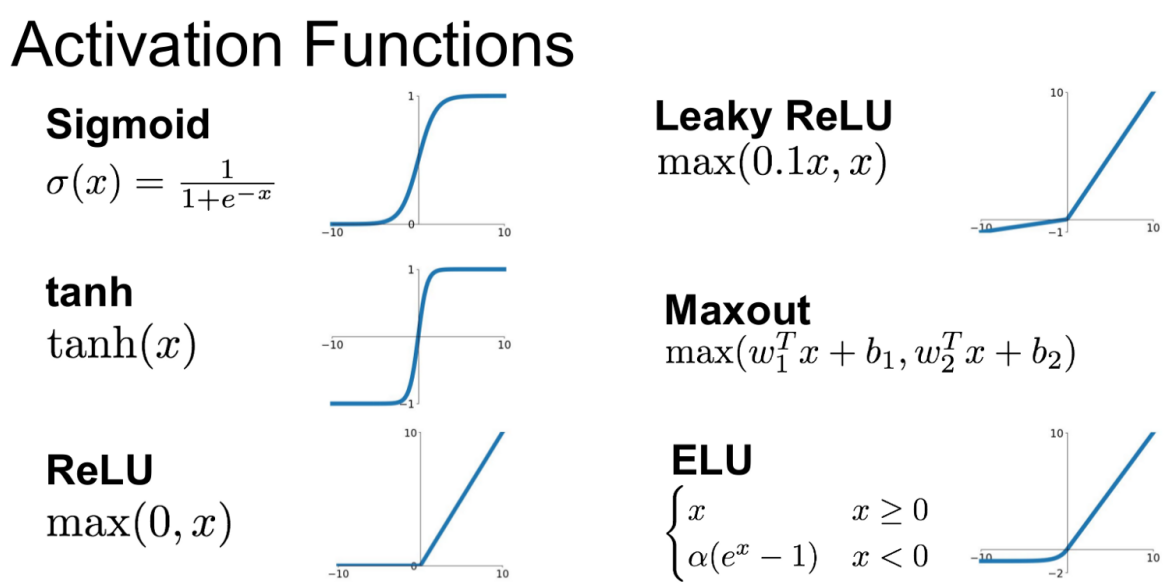
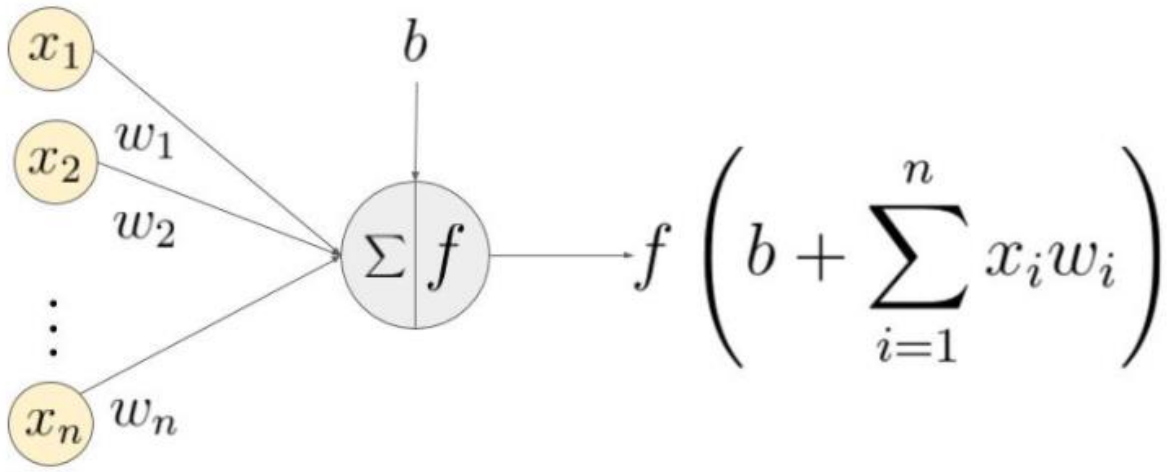


Figure 16- Activation Functions

There is also a bias which helps in controlling the value at which the activation function will trigger in a neural network.



An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Figure 17- How Everything Comes Together

HOW ARE NEURAL NETWORKS TRAINED?

Neural Networks have multiple layers. And once a forward pass is done we measure the error with the expected results. And then we move backwards to incrementally change the parameters, so that we

minimize the errors. And we do this minimization of errors by slightly moving each of the parameters in the direction opposite to their gradient (or derivative).

This is like an optimization problem. So the algorithm to do this is basically …

Stochastic Gradient Descent

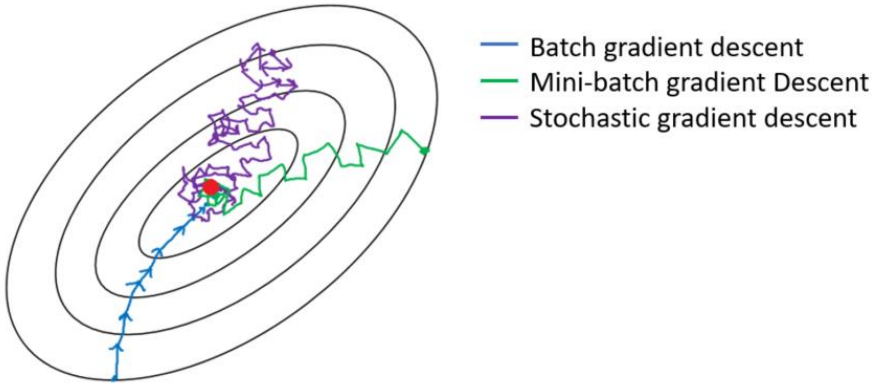


Figure 18- Gradient Descent

In gradient descent, a batch is the total set of examples you use to calculate the gradient in a single iteration. It is natural to assume that the batch has been the entire data set. But that is computationally prohibitive and disadvantageous to neural network performance. So for every batch we choose a small set of **'random'** input/output examples we train with. This leads to the term "Stochastic" in "Stochastic Gradient Descent"

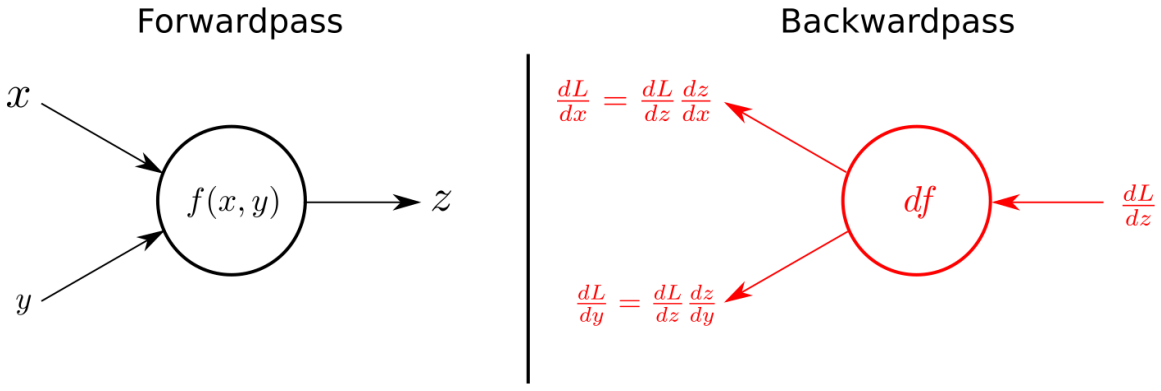


Figure 19- Backpropagation

The process is called **Backpropagation**.

And **Algorithmic Differentiation** is a generalization of Backpropagation. Which is why everyone is so excited about it.

For neural networks in which parts have connections with self, or in loops. Are trained with an algorithm called **"Backpropagation in time"**

> Sidenote: Training Spiking Neural Networks.
>
> To our knowledge nobody uses Spiking Neural Networks (SNN's) much because nobody knows how to directly train them. The only method which is successful in training SNN's is supervised training of a regular ANN (Artificial Neural Network) and then converting the model into SNN.

The underlying assumption behind all training methods is that the Neural Network is '**Differentiable'** end to end. Which means we can use algorithms/schemes/functions in our neural networks only if they are differentiable. (And even then we face the vanishing or exploding gradient numerical computational problem.)

This is one of the reasons the Fathers of Deep Learning say we have to throw everything away and start all over again. All this won't lead to A.G.I.

*** At Automatski we have fixed this. We have 'completely' eliminated Backpropagation. So we can create A.I. which can use arbitrary functions and schemes. Which may or may not be differentiable. This is exponentially more powerful, scalable and efficient compared to anything we have today. And we can finally also train Spiking Neural Networks with this.

MARKOV DECISION PROCESSES (MDP) & MARKOV REWARD PROCESSES (MRP)

The Markov Property States – Given (Just) The Present (State) The Future Is Independent Of The Past (States).

Once the current state in known, the history of information encountered so far may be thrown away, and that state is sufficient to gives us the same future as if we had all the history.

In mathematical terms, a state S_t has the Markov property, if and only if:

$$P[St+1 \mid St] = P[St+1 \mid S1, \ldots\ldots, St]$$

For a Markov state S_t and successor state S_{t+1}, the state transition probability function is a probability distribution over next possible successor states, given current state, i.e. the agent is in some state, there is a probability to go to the first state, and another probability to go to the second state and so on.

We can put this transition function in the form of a matrix, where each row sums to 1.

$$\mathcal{P} = \begin{matrix} \\ from \\ \\ \\ \end{matrix} \begin{matrix} to \\ \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \end{matrix}$$

A Markov process is a memory-less random process, i.e. a sequence of random states S1, S2, ….. with the Markov property. A Markov process or Markov chain is a tuple (S, P) on state space S, and transition function P. The dynamics of the system can be defined by these two components S and P. When we sample from an MDP, it's basically a sequence of states or as we call it a sample, or an observation or an episode.

An Markov Reward Process is a tuple (S, P, R, γ) where S is a finite state space, P is the state transition probability function, R is a reward function where,

$$R_s = \mathbb{E}[R_{t+1} \mid S_t = S]$$

It says how much immediate reward we expect to get from state S at the moment.

There is the notion of the return Gt, which is the total discounted rewards from time step t. This is what we care about, the goal is to maximize this return,

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

γ is a discount factor, where $\gamma \in [0, 1]$. It informs the agent of how much it should care about rewards now to rewards in the future. If ($\gamma = 0$), that means the agent is short-sighted, in other words, it only cares about the first reward. If ($\gamma = 1$), that means the agent is far-sighted, i.e. it cares about all future rewards. What we care about is the total rewards that we're going to get.

A Markov Decision Process is a Markov Reward Process with Decisions, it's an environment in which all states are Markov. This is what we want to solve. An MDP is a tuple (S, A, P, R, γ), where S is our state space, A is a finite set of actions, P is the state transition probability function

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

R is the reward function

$$\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

And γ is a discount factor $\gamma \in [0, 1]$

A policy π is a distribution over actions given states. A policy fully defines the behavior of an agent

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

REINFORCEMENT LEARNING

A reinforcement learning system consists of four main elements:

1. An Agent
2. A policy
3. A reward signal, and
4. A value function

The job of the Agent is to learn (a Policy) based on Value of the Rewards (or Punishments) it gets when it tries to do things in the Environment.

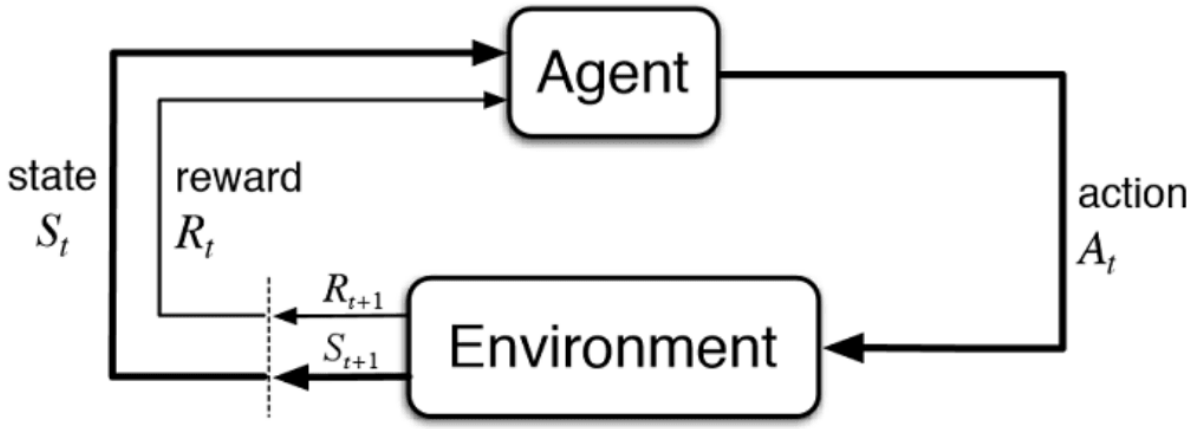


Figure 20- Reinforcement Learning (Moving From Time t to t+1)

How Does Reinforcement Learning Work?

The Agent is in a certain state S_t at time t. It performs an action on the Environment and receives a Reward R_{t+1} and moves into a new State S_{t+1}. By doing all this in its head it learns a Policy (which basically tells it what to do in which situations). The policy is what it needs to walk, fight, play games, solve problems etc.

Do you remember the famous Experiment in Psychology called Operant Conditioning?

B.F. Skinner is regarded as the father of Operant Conditioning. According to this principle, behavior that is followed by pleasant consequences is likely to be repeated, and behavior followed by unpleasant consequences is less likely to be repeated.

Skinner introduced a new term into the Law of Effect – Reinforcement. Behavior which is reinforced tends to be repeated (i.e., strengthened); behavior which is not reinforced tends to die out-or be extinguished (i.e., weakened).

This is the basic principle behind “Reinforcement” Learning Methods in A.I.

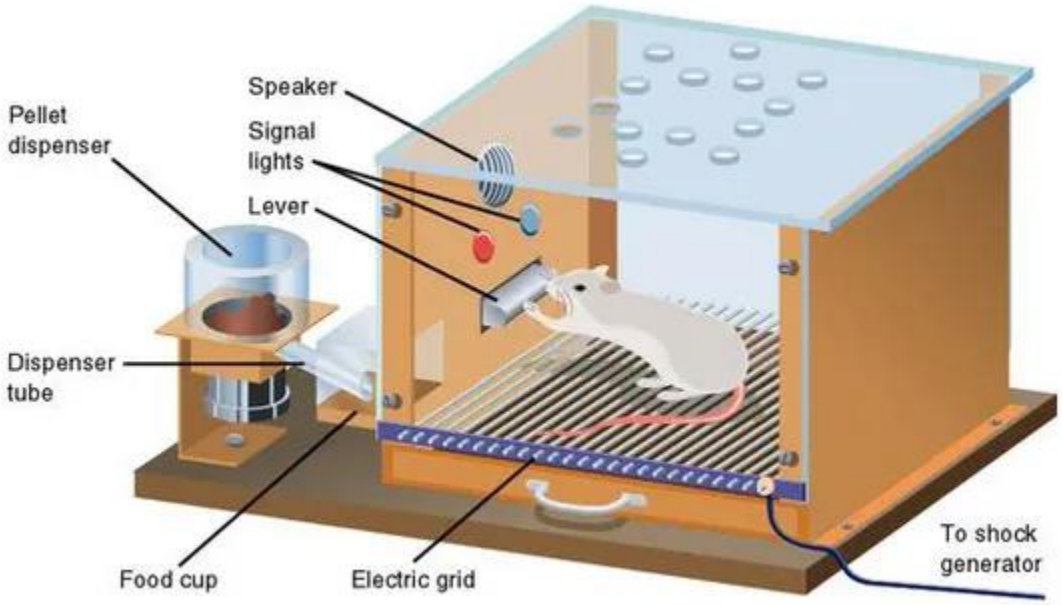


Figure 21- Operant Conditioning (aka Reinforcement)

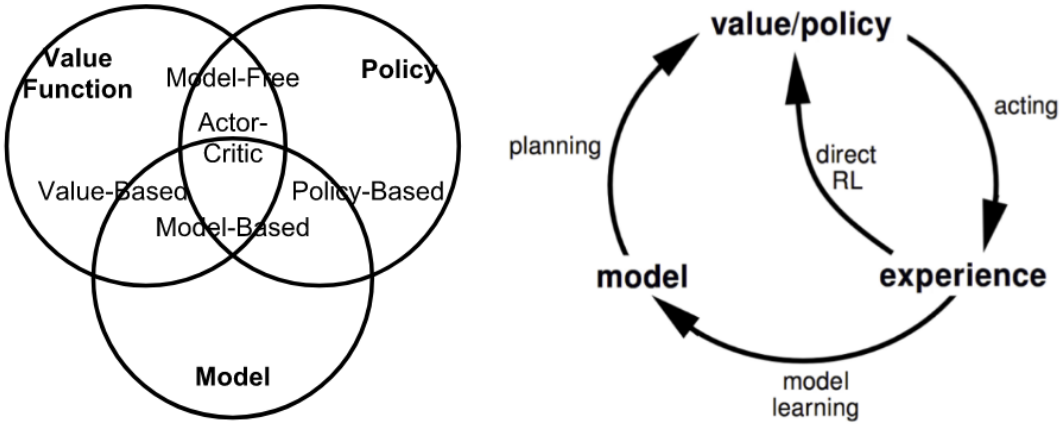


Figure 22- Summary Of Approaches In RL based On Whether We Want To Model The Value, Policy, Or The Environment. (Ref: David Silver’s RL Course)

Model-Free vs. Model-Based

1. Model-based: Rely on the model of the environment; either the model is known or the algorithm learns it explicitly.
2. Model-free: No dependency on the model during learning.

Value-Based vs. Policy-Based

1. Value-based means the agent is learning to maximize expected reward through learning different values than your parameters, examples of which are Q-learning and deep Q-learning.

2. Policy-based means that the agent is directly updating the weights of its policy instead of taking an intermediate value such as the Q-value.

Off-Policy vs. On-Policy

1. Off-policy means the agent can learn from historical data. This is the case when you have an experience replay memory. Or from a different policy rather than the target policy.
2. On-policy means the agent can only learn on new data, or on new observations. Deterministic outcomes or samples from the target policy itself.

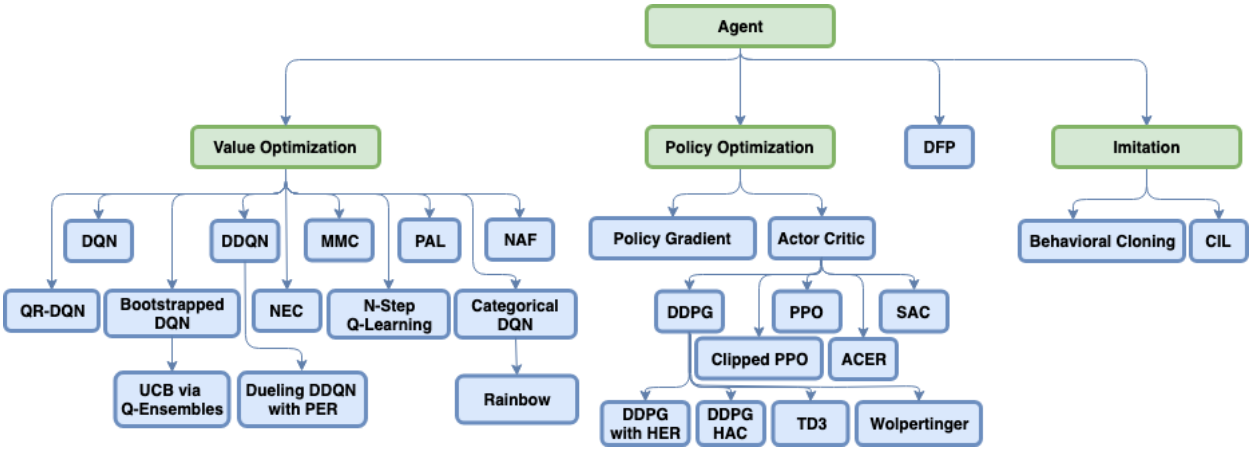


Figure 23- The Family Tree Of Reinforcement Learning Algorithms (Ref: Intel Labs/Nirvana Systems)

On the left are Value Based methods which update the value function of a policy and find policies that optimize said value function.

To the right of those are policy gradient methods which optimize for the policies directly without explicitly finding a value function.

To the far right is Imitation Learning whereby one wants to copy a policy given demonstrations of an existing but unacceptable policy.

And in the middle is Direct Future Prediction. Which outperformed the rest of the field (including A3C and variants of DQN) by more than 50%, in VizDoom AI Competition in 2016. All it took was a simple architecture with no additional supervisory signals! The trick, it used, is to reformulate the reinforcement learning (RL) problem as supervised learning (SL) problem. The ability to pursue complex goals is one of the major benefits of DFP.

Reinforcement Learning may be more efficient when the environment provides only a sparse scalar reward signal, whereas Supervised Learning can be advantageous when dense multidimensional feedback is available.

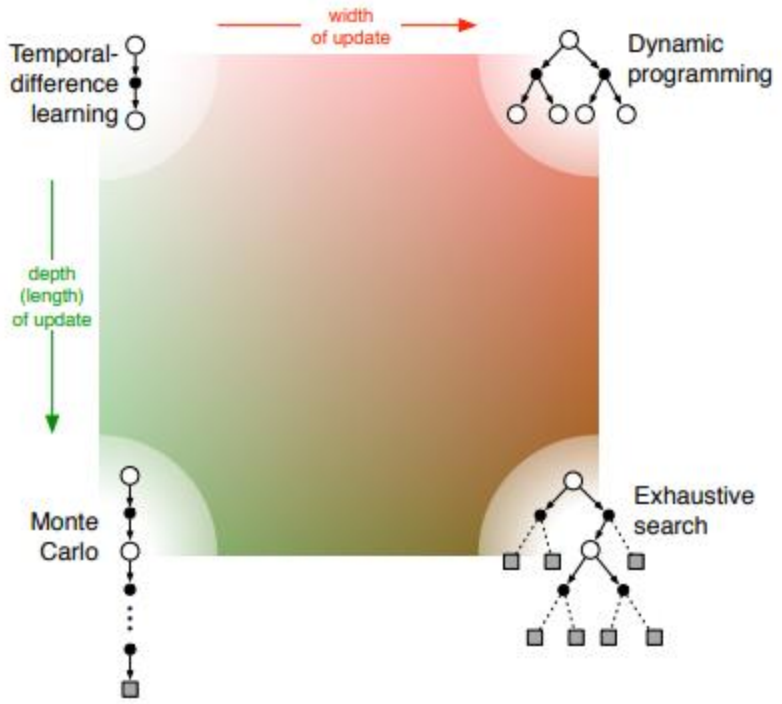


Figure 24- Reinforcement Learning Methods Based On Length & Width Of Updates

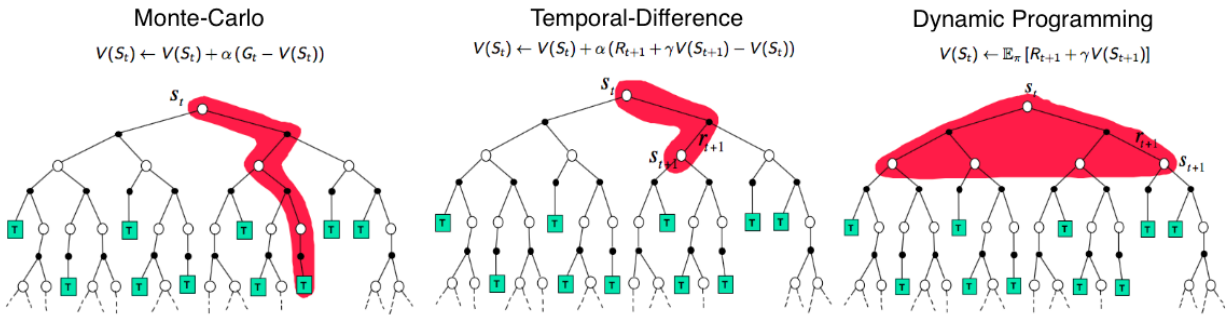


Figure 25- Monte Carlo, Temporal Difference Learning & Dynamic Programming (Ref: David Silver's RL course)

Temporal difference (TD) learning refers to a class of model-free reinforcement learning methods which learn by bootstrapping from the current estimate of the value function. These methods sample from the environment, like Monte Carlo methods, and perform updates based on current estimates, like dynamic programming methods.

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle.

Symbol	Meaning
$s \in \mathcal{S}$	States.
$a \in \mathcal{A}$	Actions.
$r \in \mathcal{R}$	Rewards.
S_t, A_t, R_t	State, action, and reward at time step t of one trajectory. I may occasionally use s_t, a_t, r_t as well.
γ	Discount factor; penalty to uncertainty of future rewards; $0 < \gamma \leq 1$.
G_t	Return; or discounted future reward; $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$.
$P(s', r \vert s, a)$	Transition probability of getting to the next state s' from the current state s with action a and reward r.
$\pi(a \vert s)$	Stochastic policy (agent behavior strategy); $\pi_\theta(.)$ is a policy parameterized by θ.
$\mu(s)$	Deterministic policy; we can also label this as $\pi(s)$, but using a different letter gives better distinction so that we can easily tell when the policy is stochastic or deterministic without further explanation. Either π or μ is what a reinforcement learning algorithm aims to learn.
$V(s)$	State-value function measures the expected return of state s; $V_w(.)$ is a value function parameterized by w.
$V^\pi(s)$	The value of state s when we follow a policy π; $V^\pi(s) = \mathbb{E}_{a \sim \pi}[G_t \vert S_t = s]$.
$Q(s, a)$	Action-value function is similar to $V(s)$, but it assesses the expected return of a pair of state and action (s, a); $Q_w(.)$ is a action value function parameterized by w.
$Q^\pi(s, a)$	Similar to $V^\pi(.)$, the value of (state, action) pair when we follow a policy π; $Q^\pi(s, a) = \mathbb{E}_{a \sim \pi}[G_t \vert S_t = s, A_t = a]$.
$A(s, a)$	Advantage function, $A(s, a) = Q(s, a) - V(s)$; it can be considered as another version of Q-value with lower variance by taking the state-value off as the baseline.

Figure 26- The Notation Used In Reinforcement Learning

The Problem w/ Rewards

Reinforcement learning assumes the existence of a reward function. Usually, this is either given, or it is hand-tuned offline and kept fixed over the course of learning. We say "usually" because there are exceptions, such as imitation learning or inverse RL, but most Reinforcement Learning approaches treat the reward as an oracle which keeps outputting values in every situation.

Shaped rewards give increasing rewards in states that are closer to the end goal. This is in contrast to sparse rewards, which give a reward at the goal state, and no reward anywhere else. Shaped rewards are often much easier to learn, because they provide positive feedback even when the policy hasn't figured out a full solution to the problem.

Unfortunately, shaped rewards can bias learning. This can lead to behaviors that don't match what you want. For example in a boat race with points for passing checkposts and for finishing. The agent figures

out that it can get more points by farming points by crossing checkposts repeatedly rather than finishing the race.

Also designing rewards is difficult. For example how do you design rewards for make an agent learn to walk? We might say reward = velocity of the agent. But that would hardly enable the agent to learn in any meaningful and efficient way.

So what do we do?

INVERSE REINFORCEMENT LEARNING: LEARNING THE REWARD FUNCTION

In a traditional RL setting, the goal is to learn a decision process to produce behavior that maximizes some predefined reward function. Inverse reinforcement learning (IRL), as described by Andrew Ng and Stuart Russell in 2000, flips the problem and instead attempts to extract the reward function from the observed behavior of an agent.

For example, consider the task of autonomous driving. A naive approach would be to create a reward function that captures the desired behavior of a driver, like stopping at red lights, staying off the sidewalk, avoiding pedestrians, and so on. Unfortunately, this would require an exhaustive list of every behavior we'd want to consider, as well as a list of weights describing how important each behavior is. (Imagine having to decide exactly how much more important pedestrians are than stop signs).

Instead, in the Inverse RL framework, the task is to take a set of human-generated driving data and extract an approximation of that human's reward function for the task. Of course, this approximation necessarily deals with a simplified model of driving. Still, much of the information necessary for solving a problem is captured within the approximation of the true reward function. **As Ng and Russell put it, "the reward function, rather than the policy, is the most succinct, robust, and transferable definition of the task,"** since it quantifies how good or bad certain actions are. **Once we have the right reward function, the problem is reduced to finding the right policy, and can be solved with standard reinforcement learning methods.**

APPRENTICESHIP LEARNING: LEARNING FROM A TEACHER

In addition to learning a reward function from an expert, we can also directly learn a policy to have comparable performance to the expert. This is particularly useful if we have an expert policy that is only approximately optimal. Here, an apprenticeship learning algorithm, formulated by Pieter Abbeel and Andrew Ng in 2004 gives us a solution. This algorithm takes an MDP and an approximately optimal "teacher" policy, and then learns a policy with comparable or better performance to the teacher's policy using minimal exploration.

This minimal exploration property turns out to be very useful in fragile tasks like autonomous helicopter flight. A traditional reinforcement learning algorithm might start out exploring at random, which would almost certainly lead to a helicopter crash in the first trial. Ideally, we'd be able to use expert data to start with a baseline policy that can be safely improved over time. This baseline policy should be significantly better than a randomly initialized policy, which speeds up convergence.

Apprenticeship Learning Algorithm

The main idea behind Abbeel and Ng's algorithm is to use trials from the expert policy to obtain information about the underlying MDP, then iteratively execute a best estimate of the optimal policy for the actual MDP. Executing a policy also gives us data about the transitions in the environment, which we can then use to improve the accuracy of the estimated MDP. Here's how the algorithm works in a discrete environment:

First, we use the expert policy to learn about the MDP:

1. Run a fixed amount of trials using the expert policy, recording every state-action trajectory.
2. Estimate the transition probabilities for every state-action pair using the recorded data via maximum likelihood estimation.
3. Estimate the value of the expert policy by averaging the total reward in each trial.
4. Then, we learn a new policy:
5. Learn an optimal policy for the estimated MDP using any standard reinforcement learning algorithm.
6. Test the learning policy on the actual environment.
7. If performance is not close enough to the value of the expert policy, add the state-action trajectories from this trial to the training set and repeat the procedure to learn a new policy.

The benefit of this method is that at each stage, the policy being tested is the best estimate for the optimal policy of the system. There is diminished exploration, but the core idea of apprenticeship learning is that we can assume the expert policy is already near-optimal.

GENETIC ALGORITHMS (GA)

Genetic Algorithms (GA) are a metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate "survival of the fittest" among individual of consecutive generation for solving a problem. Each generation consist of a population of individuals and each individual represents a point in search space and possible solution.

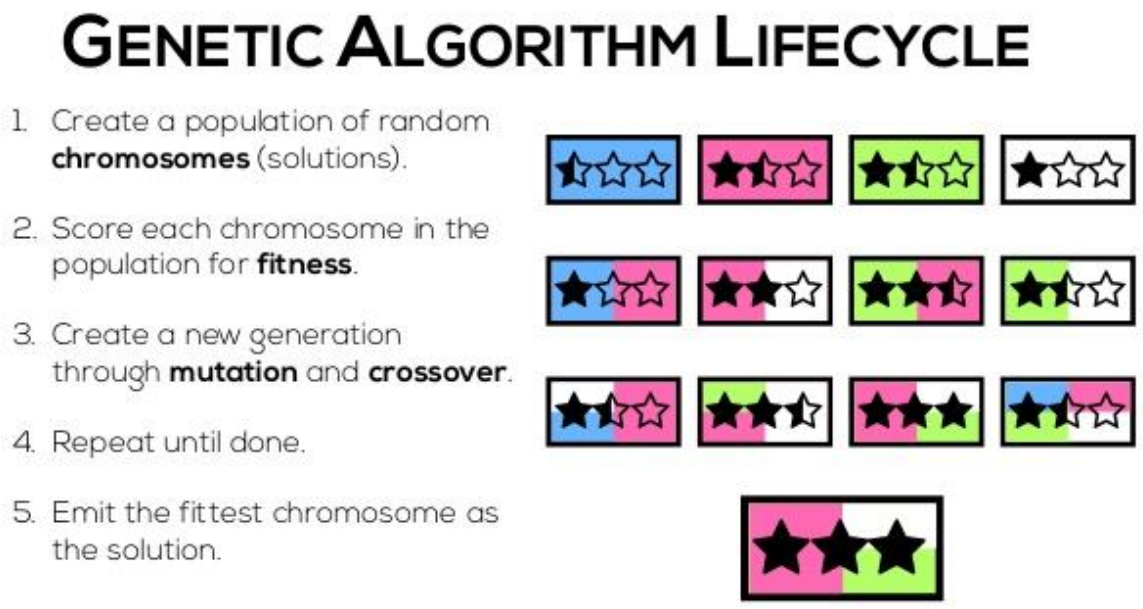


Figure 27- Genetic Algorithm Lifecycle

Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring).

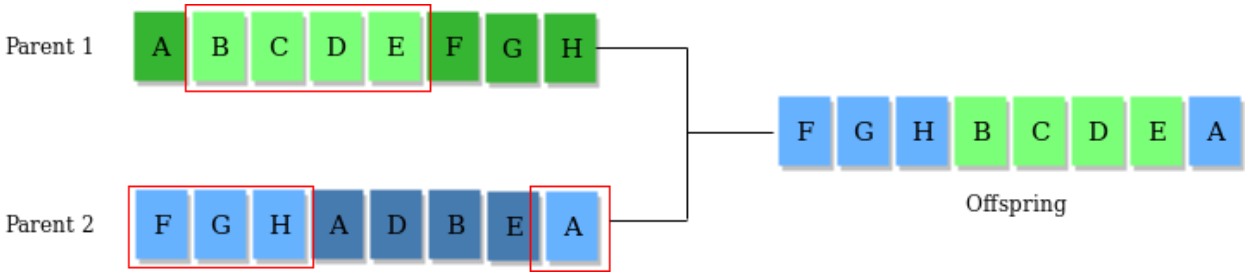


Figure 28- Crossover Operator

Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in population to avoid the premature convergence.

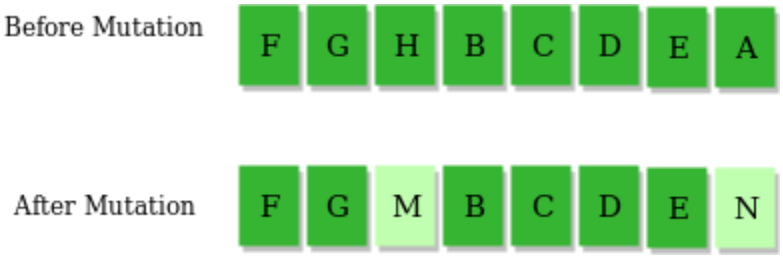


Figure 29- Mutation Operator

EVOLUTION STRATEGIES (ES)

Let's say we want to optimize a function f(x) and we are not able to compute gradients directly. But we still can evaluate f(x) given any x and the result is deterministic. Our belief in the probability distribution over x as a good solution to f(x) optimization is $p_\theta(x)$, parameterized by θ. The goal is to find an optimal configuration of θ.

Starting with an initial value of θ, we can continuously update θ by looping three steps as follows:

1. Generate a population of samples $D=\{(x_i,f(x_i)\}$ where $xi \sim p_\theta(x)$.
2. Evaluate the "fitness" of samples in D.
3. Select the best subset of individuals and use them to update θ, generally based on fitness or rank.

In Genetic Algorithms (GA), another popular subcategory of EA, x is a sequence of binary codes, $x \in \{0,1\}^n$. While in ES, x is just a vector of real numbers, $x \in R^n$.

Evolution Strategies VS. Genetic Algorithms

	ES	GA
Initial Population	Random mutations of the initial guess	Random or seeded
Evaluation	Objective Function	Fitness (Evaluation) Function
Selection	Truncation Selection	Different methods
Reproduction	Recombination + Mutation	Crossover + Mutation
Termination	Almost similar stop conditions	

Figure 30- Evolution Strategies VS Genetic Algorithms

OpenAI published a paper called **Evolution Strategies as a Scalable Alternative to Reinforcement Learning** where they showed that evolution strategies, while being less data efficient than RL, offer many benefits. The ability to abandon gradient calculation allows such algorithms to be evaluated more efficiently. It is also easy to distribute the computation for an ES algorithm to thousands of machines for parallel computation. By running the algorithm from scratch many times, they also showed that policies discovered using ES tend to be more diverse compared to policies discovered by RL algorithms.

ES enjoys multiple advantages over RL algorithms:

1. No need for backpropagation. ES only requires the forward pass of the policy and does not require backpropagation (or value function estimation), which makes the code shorter and between 2-3 times faster in practice. On memory-constrained systems, it is also not necessary to keep a record of the episodes for a later update. There is also no need to worry about exploding gradients in RNNs. Lastly, we can explore a much larger function class of policies, including networks that are not differentiable (such as in binary networks), or ones that include complex modules (e.g. pathfinding, or various optimization layers).

2. Highly parallelizable. ES only requires workers to communicate a few scalars between each other, while in RL it is necessary to synchronize entire parameter vectors (which can be millions of numbers). Intuitively, this is because we control the random seeds on each worker, so each worker can locally reconstruct the perturbations of the other workers. Thus, all that we need to communicate between workers is the reward of each perturbation. As a result, we observed linear speedups in our experiments as we added on the order of thousands of CPU cores to the optimization.
3. Higher robustness. Several hyperparameters that are difficult to set in RL implementations are side-stepped in ES. For example, RL is not "scale-free", so one can achieve very different learning outcomes (including a complete failure) with different settings of the frame-skip hyperparameter in Atari. As we show in our work, ES works about equally well with any frame-skip.
4. Structured exploration. Some RL algorithms (especially policy gradients) initialize with random policies, which often manifests as random jitter on spot for a long time. This effect is mitigated in Q-Learning due to epsilon-greedy policies, where the max operation can cause the agents to perform some consistent action for a while (e.g. holding down a left arrow). This is more likely to do something in a game than if the agent jitters on spot, as is the case with policy gradients. Similar to Q-learning, ES does not suffer from these problems because we can use deterministic policies and achieve consistent exploration.
5. Credit assignment over long time scales. By studying both ES and RL gradient estimators mathematically we can see that ES is an attractive choice especially when the number of time steps in an episode is long, where actions have longlasting effects, or if no good value function estimates are available.

NOVELTY SEARCH

Learning By Surprise

Psychologists have known for some time that if we experience a novel situation within a familiar context, we will more easily store this event in memory. But only recently have studies of the brain begun to explain how this process happens and to suggest new ways of teaching that could improve learning and memory.

(Our Brain Is A) Novelty Detector

One of the most important brain regions involved in discovering, processing and storing new sensory impressions is the hippocampus, located in the temporal lobe of the cerebral cortex. Novel stimuli tend to activate the hippocampus more than familiar stimuli do, which is why the hippocampus serves as the brain's "novelty detector."

"Abandoning objectives is often the only way to outperform the direct search for the objective." — Kenneth Stanley

---

Novelty search does not reward progress as defined by objectives or performance, rather it rewards being different. It is evolution without objectives. It believes complexity growth in natural evolution is not a byproduct of optimizing fitness, but rather a consequence of nature's open-ended propensity to

discover various ways to meet challenges of life. And this has lead to perpetual discovery of novelty, and increase in complexity.

Novelty search mitigates deception — in which seemingly promising fitness candidates fall into local maxima, and thus can never attain global maxima causing evolutionary algorithms to fail. Deception is unavoidable with objective-based search.

It is important to note that novelty is not a random walk kind of situation — novelty rewards diverging behaviors, thus creating a constant pressure to do something new. One might argue this is just replacing one objective function with another — one that maximizes novelty instead of fitness. But look closer, because they are conceptually different:

1. Fitness creates a gradient towards the objective — maximizing fitness is done with the intent of bringing the search towards a goal.
2. Novelty creates a gradient of behavioral differences — maximizing is done without any intent of search termination or direction.

DEEP NEUROEVOLUTION

	DQN	ES	A3C	RS	GA	GA
Frames	200M	1B	1B	1B	1B	6B
Time	~7-10d	~ 1h	~ 4d	~ 1h or 4h	~ 1h or 4h	~ 6h or 24h
Forward Passes	450M	250M	250M	250M	250M	1.5B
Backward Passes	400M	0	250M	0	0	0
Operations	1.25B U	250M U	1B U	250M U	250M U	1.5B U
amidar	**978**	112	264	143	263	377
assault	4,280	1,674	**5,475**	649	714	814
asterix	4,359	1,440	**22,140**	1,197	1,850	2,255
asteroids	1,365	1,562	**4,475**	1,307	1,661	2,700
atlantis	279,987	**1,267,410**	911,091	26,371	76,273	129,167
enduro	**729**	95	-82	36	60	80
frostbite	797	370	191	1,164	**4,536**	**6,220**
gravitar	473	**805**	304	431	476	764
kangaroo	7,259	**11,200**	94	1,099	3,790	**11,254**
seaquest	**5,861**	1,390	2,355	503	798	850
skiing	-13,062	-15,443	-10,911	-7,679	†**-6,502**	†**-5,541**
venture	163	760	23	488	**969**	†**1,422**
zaxxon	5,363	6,380	**24,622**	2,538	6,180	7,864

Figure 31- Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning

Deep artificial neural networks (DNNs) are typically trained via gradient-based learning algorithms, namely backpropagation. Evolution strategies (ES) can rival backprop-based algorithms such as Q-learning and policy gradients on challenging deep reinforcement learning (RL) problems. However, ES can be considered a gradient-based algorithm because it performs stochastic gradient descent via an operation similar to a finite-difference approximation of the gradient. That raises the question of whether non-gradient-based evolutionary algorithms can work at DNN scales.

Uber's AI Group demonstrates they can: we can evolve the weights of a DNN with a simple, gradient-free, population-based genetic algorithm (GA) and it performs well on hard deep RL problems, including Atari and humanoid locomotion. The Deep GA successfully evolves networks with over four million free parameters, the largest neural networks ever evolved with a traditional evolutionary algorithm.

These results

1. expand our sense of the scale at which GAs can operate,
2. suggest intriguingly that in some cases following the gradient is not the best choice for optimizing performance, and
3. make immediately available the multitude of neuroevolution techniques that improve performance.

By combining DNNs with novelty search, which encourages exploration on tasks with deceptive or sparse reward functions, we can solve a high-dimensional problem on which reward-maximizing algorithms (e.g. DQN, A3C, ES, and the GA) fail. Additionally, the Deep GA is faster than ES, A3C, and DQN (it can train Atari in ~4 hours on one desktop or ~1 hour distributed on 720 cores), and enables a state-of-the-art, up to 10,000-fold compact encoding technique.

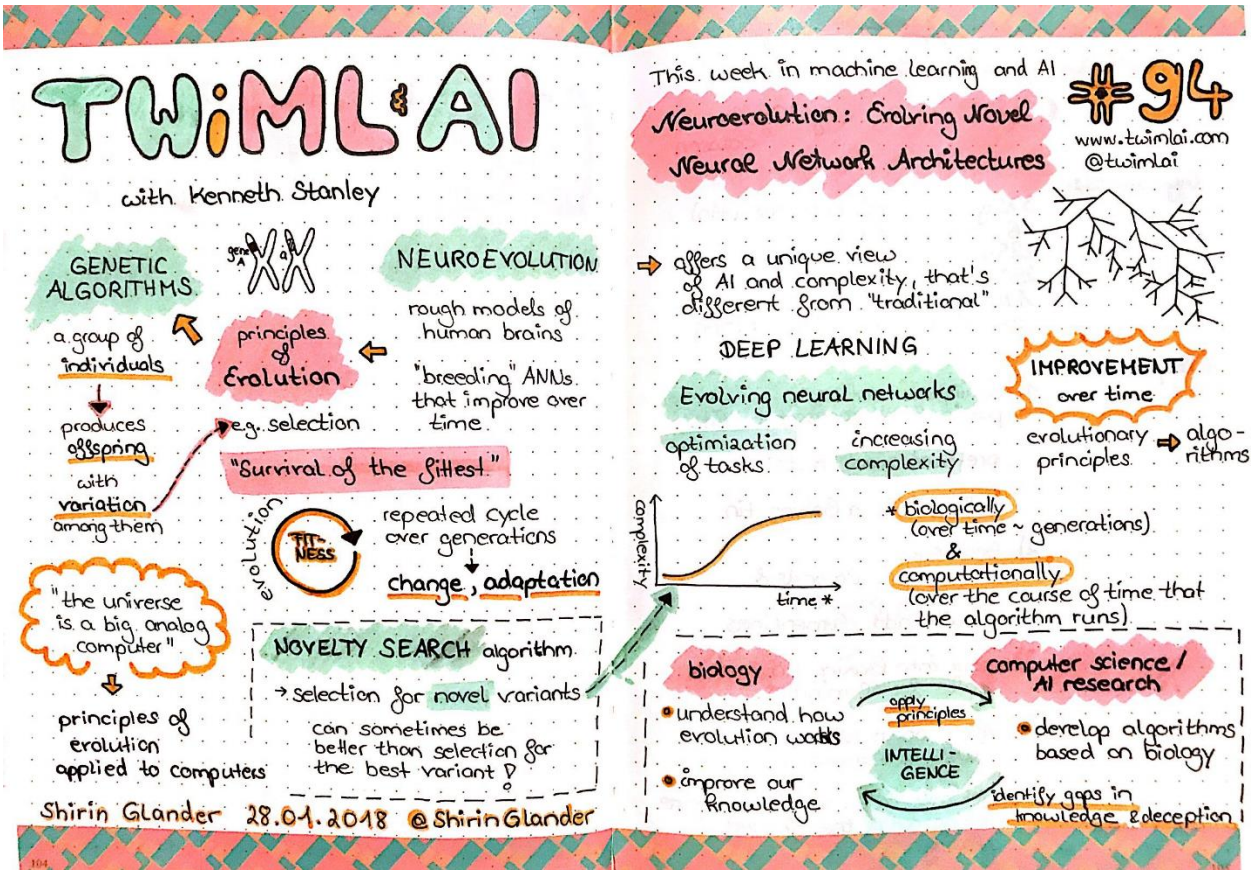


Figure 32- Neuroevolution (Ref: Kenneth Stanley)

Neuroevolution can also be used to evolve Neural Network Architectures. Which is the original definition of NeuroEvolution. Uber's view is limited to exploiting Deep GA's for Training Deep Reinforcement Learning Systems.

> Neuroevolution, is a form of artificial intelligence that uses evolutionary algorithms to generate artificial neural networks (ANN), parameters, topology and rules. It is most commonly applied in artificial life, general game playing and evolutionary robotics. (Ref: Wikipedia)

At a high-level, the idea is very simple. Instead of relying on a fixed structure for a neural network, why not allow it to evolve through a genetic algorithm? To me, this makes a lot of sense. Typically, when using a neural network, one selects a structure that may work based on empirical evidence. But is it the best structure that can be used? There's no way to know for sure.

A paper called Evolving Neural Networks through Augmenting Topologies discusses the algorithm NeuroEvolution of Augmenting Topologies, more commonly known simply as NEAT. The paper came out in 2002 and focused solely on evolving dense neural networks node by node and connection by connection.

NEAT and NEAT-like algorithms can be used to evolve neural net structure and then we can use back propagation and gradient descent to optimize these networks.

DEEPMIND MUZERO

For many years, researchers have sought methods that can both learn a model that explains their environment, and can then use that model to plan the best course of action.

Until now, most approaches have struggled to plan effectively in domains, such as Atari, where the rules or dynamics are typically unknown and complex.

MuZero, first introduced in a preliminary paper in 2019, solves this problem by learning a model that focuses only on the most important aspects of the environment for planning. By combining this model with AlphaZero's powerful lookahead tree search, MuZero set a new state of the art result on the Atari benchmark, while simultaneously matching the performance of AlphaZero in the classic planning challenges of Go, chess and shogi. In doing so, MuZero demonstrates a significant leap forward in the capabilities of reinforcement learning algorithms.

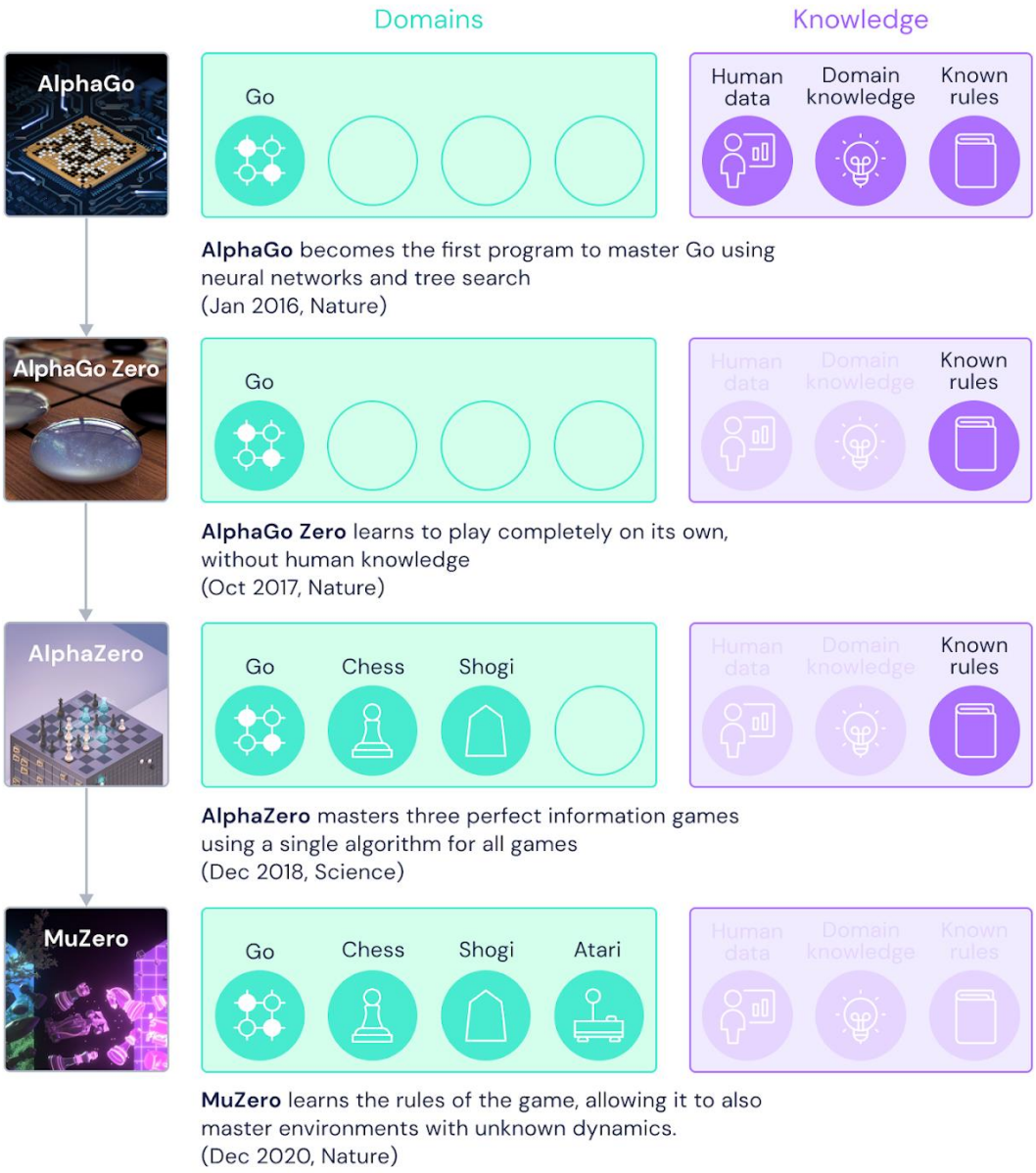


Figure 33- DeepMind MuZero

MuZero masters Go, chess, shogi and Atari without needing to be told the rules, thanks to its ability to plan winning strategies in unknown environments.

The ability to plan is an important part of human intelligence, allowing us to solve problems and make decisions about the future. For example, if we see dark clouds forming, we might predict it will rain and decide to take an umbrella with us before we venture out. Humans learn this ability quickly and can generalize to new scenarios, a trait we would also like our algorithms to have.

Researchers have tried to tackle this major challenge in AI by using two main approaches:

1. lookahead search or
2. model-based planning.

Systems that use lookahead search, such as AlphaZero, have achieved remarkable success in classic games such as checkers, chess and poker, but rely on being given knowledge of their environment's dynamics, such as the rules of the game or an accurate simulator. This makes it difficult to apply them to messy real world problems, which are typically complex and hard to distill into simple rules.

Model-based systems aim to address this issue by learning an accurate model of an environment's dynamics, and then using it to plan. However, the complexity of modelling every aspect of an environment has meant these algorithms are unable to compete in visually rich domains, such as Atari. Until now, the best results on Atari are from model-free systems, such as DQN, R2D2 and Agent57. As the name suggests, model-free algorithms do not use a learned model and instead estimate what is the best action to take next.

MuZero uses a different approach to overcome the limitations of previous approaches. Instead of trying to model the entire environment, MuZero just models aspects that are important to the agent's decision-making process. After all, knowing an umbrella will keep you dry is more useful to know than modelling the pattern of raindrops in the air.

Specifically, MuZero models three elements of the environment that are critical to planning:

1. The value: how good is the current position?
2. The policy: which action is the best to take?
3. The reward: how good was the last action?

Ref: (Paraphrased from the original announcement by DeepMind)

AUTOMATSKI NON-DETERMINISTIC TURING MACHINE BASED REINFORCEMENT LEARNING

Agent	Median	Mean	Env. Frames
Ape-X	434.1%	1695.6%	22.8B
R2D2	1920.6%	4024.9%	37.5B
MuZero	**2041.1%**	**4999.2%**	20.0B
IMPALA	191.8%	957.6%	200M
Rainbow	231.1%	-	200M
UNREAL	250%	880%	200M
LASER	431%	-	200M
MuZero Reanalyse	**731.1%**	**2168.9%**	200M

Figure 34- Reinforcement Learning In General Requires Gazillions Of Samples Otherwise (Ref: DeepMind)

Learning a Model and Planning is a problem Automatski has solved long time back. The first and foremost challenge in real world problems is to cut/reduce the sampling inefficiency. Automatski has been able to cut the needed sample counts from exponential to polynomial. This is equivalent to solving a NP-Hard Problem in Polynomial Time (Though Only Approximately)

To achieve this Automatski has deployed its Non-Deterministic Turing Machine. It's the first of its kind invention in the world. And in principle can be used to solve "Any" NP problem in polynomial time.

Automatski's Reinforcement Learning Algorithm can be used for Real World Problems with complexities, uncertainties and unknowns.

To understand the Principle behind this we say…

If you can figure out how to choose the best you don't have to worry about the rest; i.e. constraints, models, rules etc. And if you can do this in any situation then you have a pretty generic universal algorithm. And that's what Automatski's RL Algorithm does and it does so without any Exponential Data or Computational Needs. Without a detailed or complex model. (And yes it's an approximation algorithm and delivers an approximate, close to best, solution depending on the complexity of the problem statement.)

SIMULATION FRAMEWORKS FOR REINFORCEMENT LEARNING

For Real World solutions, model based Reinforcement Learning is probably the only way forward for Reinforcement Learning. To do that we would need to make complex simulation environments. In which the agents can train. Some such environments available today are GVGAI, Griddle, Pommerman, BabyAI, microRTS, and DMLab2D.

1 TRILLION PARAMETER NEURAL NETWORKS

Figure 35- Scaling To Trillion Parameter Models (Ref: Microsoft)

It is generally believed that the human brain has ~100 billion neurons give or take a few billion. Vendors are trying to scale Deep Neural Networks to a trillion parameters in the hope that it will achieve Human Like performance on cognitive tasks like vision, speech, motion, natural language etc.

1 Trillion Parameters is the Moon Shot, Deep Learning is hoping to make. It remains to be seen, that even if it is achieved, if it will fulfill the original objectives. Critics remain skeptic for now.

NEUROMORPHIC COMPUTING

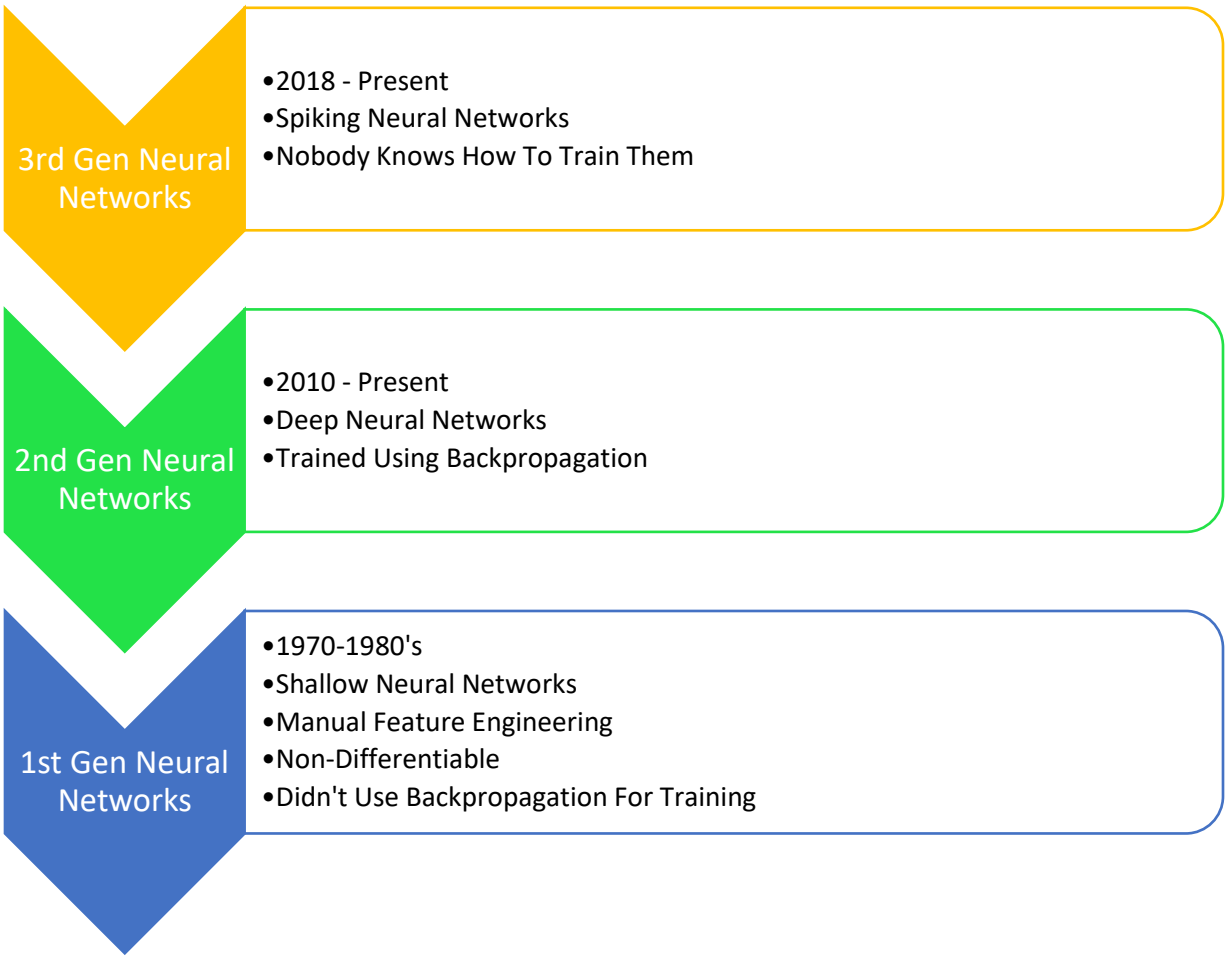


Figure 36- 3 Generations Of Neural Networks

Neuromorphic computing emulates the neural structure of the human brain using something called Spiking Neural Networks.

The Loihi research chip from Intel includes 130,000 neurons optimized for spiking neural networks.

IBM's TrueNorth is a manycore processor network on a chip design, with 4096 cores, each one having 256 programmable simulated neurons for a total of just over a million neurons. In turn, each neuron has 256 programmable " synapses " that convey the signals between them. Hence, the total number of programmable synapses is just over 268 million (2^{28}). Its basic transistor count is 5.4 billion. Since memory, computation, and communication are handled in each of the 4096 neurosynaptic cores, TrueNorth circumvents the von Neumann-architecture bottleneck and is very energy-efficient, with IBM claiming a power consumption of 70 milliwatts and a power density that is 1/10,000th of conventional microprocessors.

SpiNNaker 2 is the largest Neuromorphic Computing Platform in the world, besides BrainScaleS, and can simulate brain-size networks in real time. TU Dresden and the University of Manchester have developed it as a part of the Human Brain Project.

It consists of 10 Million ARM cores distributed across 70.000 Chips in 10 server racks. It combines high-throughput machine learning, sensor/actuator processing at millisecond latency, high-energy efficiency and strict real-time operations.

Its real-time performance for an integration time step of 1ms which “generally suffices” for applications in robotics and artificial neural networks, though a “time step of 0.1ms” is typical for neuroscience applications.

SpiNNaker 2 can support detailed biological models of the cortex – the outer layer of the brain that receives and processes information from the senses – delivering results very similar to those from an equivalent supercomputer software simulation.

RADIAL BASIS FUNCTION NETWORKS

A radial basis function network is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters. Radial basis function networks have many uses, including function approximation, time series prediction, classification, and system control.

A radial basis function (RBF) is a function that assigns a real value to each input from its domain (it is a real-value function), and the value produced by the RBF is always an absolute value; i.e. it is a measure of distance and cannot be negative. i.e. f(x) = f(||x||)

The straight-line distance between two points in Euclidean space aka Euclidean distance, is typically used.

Radial basis functions are used to approximate functions, much as neural networks act as function approximators.

The following sum:

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|),$$

represents a radial basis function network. The radial basis functions act as activation functions.

The approximant f(x) is differentiable with respect to the weights W, which are learnt using iterative updater methods common among neural networks.

Meshless Methods for PDE's

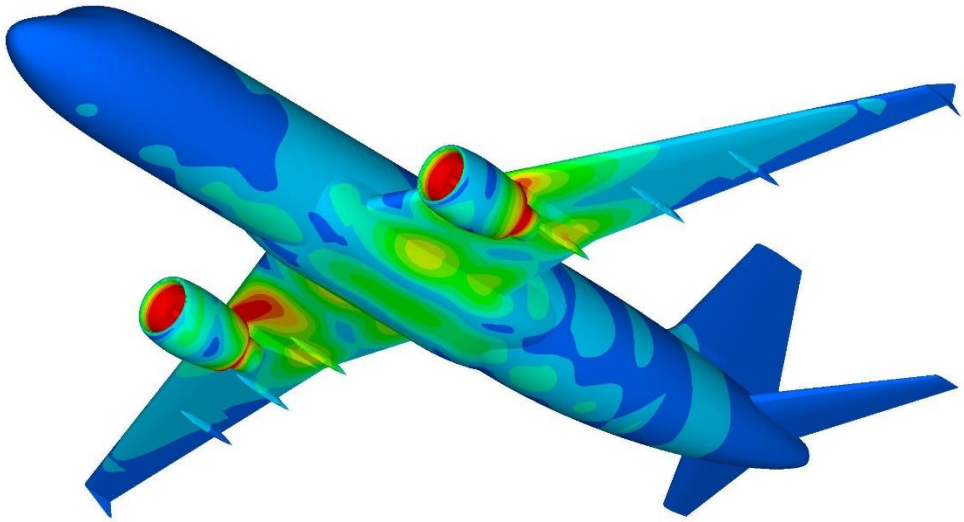


Figure 37 - Finite Element Method

Although many numerical and analytical schemes exist for solving engineering problems, the meshless method is a particularly attractive method that is receiving attention in the engineering and scientific modeling communities. Finite difference (FDM), finite volume (FVM), and finite element (FEM) methods have been historically used to model a wide variety of engineering problems in complex geometries that may require extensive meshing. The meshless method is simple, accurate, and requires no meshing.

The need to accurately simulate various physical processes in complex geometries is important, and has perplexed modelers utilizing conventional numerical schemes for many years. Today, advances in numerical schemes and enhanced hardware have lead to many commercial codes that can employ Herculean efforts to solve complex stress-strain, heat transfer, fluid flow, and other nearly intractable problems. Recently, advances in the development and application of meshless techniques show they can be strong competitors to the more classical finite difference/volume and finite element approaches.

Meshless methods are uniquely simple, yet provide solution accuracies for certain classes of equations that rival those of finite elements and boundary elements without requiring the need for mesh connectivity. Ease in programming, no domain or surface discretization, no numerical integration, and similar formulations for 2-D and 3-D make these methods very attractive.

There exist several types of meshless methods. The more common techniques include kernel methods, moving least square method, meshless Petrov- Galerkin, partition of unity methods, smooth-particle hydrodynamics, **and radial basis functions**. Each technique has particular traits and advantages for specific classes of problems. **Generally the simplest and easiest to implement is the radial basis function approach.**

Meshless methods utilizing RBFs create mesh-free algorithms that are significantly simpler to employ than FDM/FVM, FEM, and BEM approaches, and truly eliminate the need for meshes requiring connectivity and optimization.

SCIENTIFIC MACHINE LEARNING

Deep learning is playing an increasing role in mathematical areas, **like inverse problems and partial differential equations (PDEs)**. A factor behind this development is the idea of complementing physics-based models with data-driven components, the latter to introduce an adaptability to the data and make up for necessary modelling inaccuracies. Another benefit of deep learning is that once trained they are computationally fast to apply. Hence, this offers new ways to tackle computationally challenging problems, like the numerical solution of **high-dimensional and non-linear PDEs**. The potential for deep learning in enhancing these mathematical models has this far been supported by empirical evidence. Very little has been achieved in the context of mathematical theory that can support this empirical evidence. Yet it works.

- Physics-Informed Neural Networks for automated PDE solving
- Forward-Backwards Stochastic Differential Equation (FBSDE) methods for parabolic PDEs
- Deep-learning-based solvers for optimal stopping time and Kolmogorov backwards equations

Are some of the techniques applied to solving High Dimensional and Non-Linear - Partial Differential Equations, with Neural Networks.

NEURAL DIFFERENTIAL EQUATIONS

Instead of using regular layers in a neural network with neurons and activators. We can use differential equations as 'black box' layers.

Neural Ordinary Differential Equations (Neural ODEs) are a new and elegant type of mathematical model designed for machine learning. This model type was proposed in a 2018 paper and has caught noticeable attention ever since. The idea was mainly to unify two powerful modelling tools: Ordinary Differential Equations (ODEs) & Machine Learning.

Neural ODE's are differentiable and trainable using backpropagation. The ODE layers are treated as a black box and we use Adjoint Sensitivity Method to train the layers.

Neural ODE's have many applications in control, generative modeling and forecasting.

DEEP COMPLEX NETWORKS

The vast majority of building blocks, techniques, and architectures for deep learning today are based on real-valued operations and representations. However, recent work on recurrent neural networks and older fundamental theoretical analysis suggests that complex numbers could have a richer representational capacity and could also facilitate noise-robust memory retrieval mechanisms.

Many deterministic signals, such as seismic data or electrical signals, contain significant information in the phase of the signal. Also if we want to apply Deep Learning to scientific applications such as Physics, Chemistry, Materials, Nuclear Fusion, Biology or to Simulate Quantum Computing we need to use Complex Numbers.

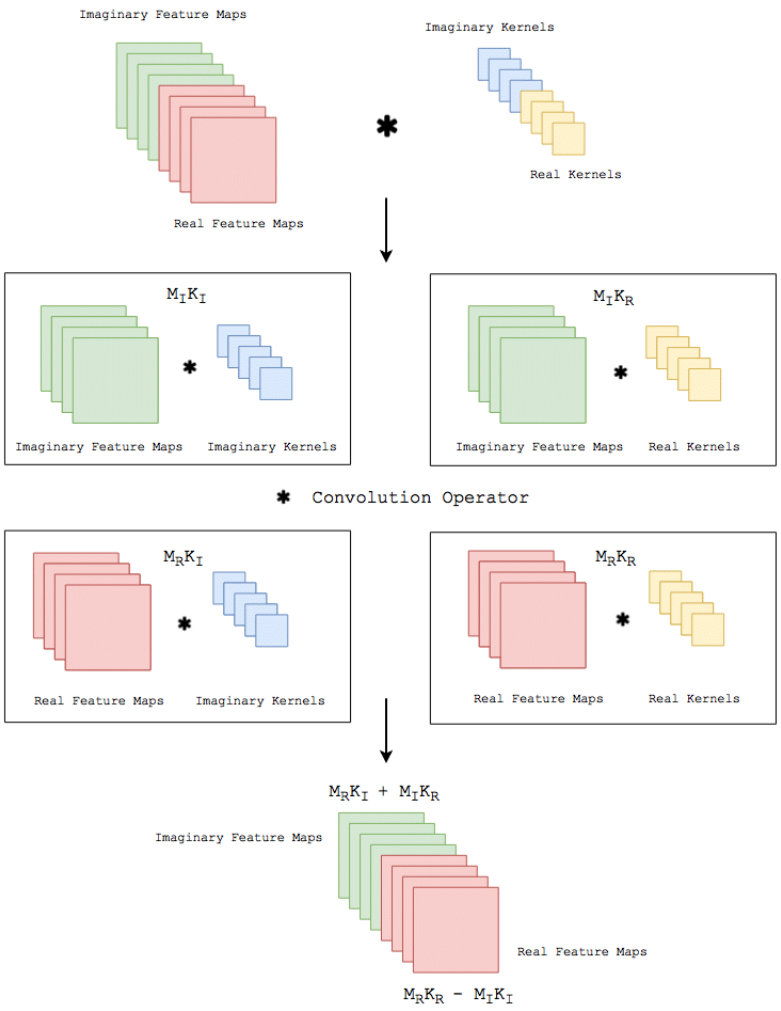


Figure 38- A Complex Convolution Neural Network

FERMIONIC NEURAL NETWORKS

Is an attempt to apply Deep Learning to Ab-Initio Solution of the Many-Electron Schroedinger Equation with applications in Chemistry. It can also be applied to material sciences and condensed matter physics and any physics involving Fermions.

Given access to accurate solutions of the many-electron Schrödinger equation, nearly all chemistry could be derived from first principles. Exact wave functions of interesting chemical systems are out of reach because they are NP-hard to compute in general, but approximations can be found using polynomially scaling algorithms. The key challenge for many of these algorithms is the choice of wave function approximation, or Ansatz, which must trade off between efficiency and accuracy. Neural networks have shown impressive power as accurate practical function approximators and promise as a compact wave-function Ansatz for spin systems, but problems in electronic structure require wave functions that obey Fermi-Dirac statistics. FermiNet is a novel deep learning architecture, (The Fermionic Neural Network), a powerful wave-function Ansatz for many-electron systems. The Fermionic neural network is able to

achieve accuracy beyond other variational quantum Monte Carlo Ansatz on a variety of atoms and small molecules.

THE PAULINET

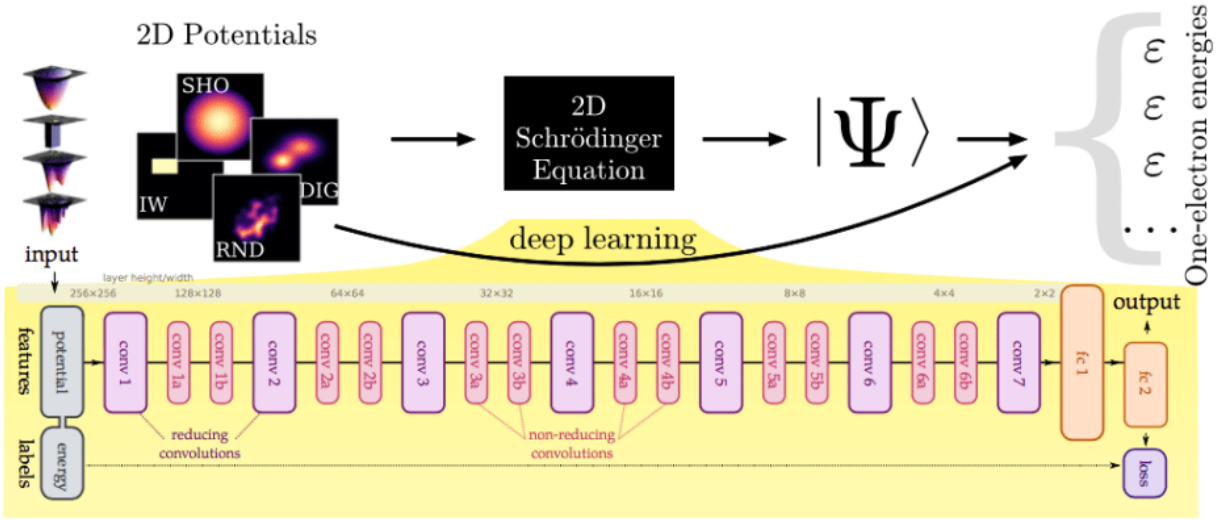


Figure 39- The PauliNet

The Schrödinger Equation is a crucial formalism at the center of quantum mechanics. It is used to work out how quantum systems are, and how they evolve. It is a computationally exponential challenge to solve precisely for a system made of more than anything but a few particles. Historically dozens of approximation methods have been used till now. Even when exploiting Supercomputers. Solving this problem is so hard.

A new approach, called PauliNet, which is a deep neural network that can get the exact solution for the equation for molecules with up to 30 electrons.

The deep neural network designed by Professor Noé's team at Freie Universität is a new way of representing the wave functions of electrons. Analysing energies of electron configurations of molecules is one of the primary requirements of Computational Chemistry. Bu tinstead of modelling a neural network capable of simple mathematical components and learning and building everything on top of it on its own from the given data. What Professor Noé's team has done is to create the Physics Primitives in Neural Networks that allow it to learn how the complex patterns of how electrons are located around the nuclei. There is an antisymmetry involved, when two electrons are exchanged the wafe function must change its sign. Such a property was built right into their neural networks for the rest of the physics to work. This feature, is known as 'Pauli's exclusion principle,' is why the team calls their method the 'PauliNet.'

Electronic wave functions also have other fundamental physics properties and the reason PauliNet is so successful is that PauliNet integrates such properties into the building blocks of its deep neural network. "Building the fundamental physics into the AI is essential for its ability to make meaningful predictions in the field," says Noé.

NEURAL NETWORK VERIFICATION

Till now there has been no way to verify the robustness of Neural Networks. Which have been prone to errors, biases, and adversarial attacks.

Neural network verification might become a powerful technology as it offers the promise of provable guarantees on neural networks satisfying desirable properties or specifications.

In these verification algorithms, often comes a trade-off between the computational efficiency of the verification algorithm and its "tightness", i.e. the gap between properties that are known to be true and properties that the verification method is able to verify to be true.

Currently there are two algorithms for neural network verification

1. Efficient non-convex reformulations
2. Memory-Efficient first-order semidefinite programming

ALGORITHMIC DIFFERENTIATION

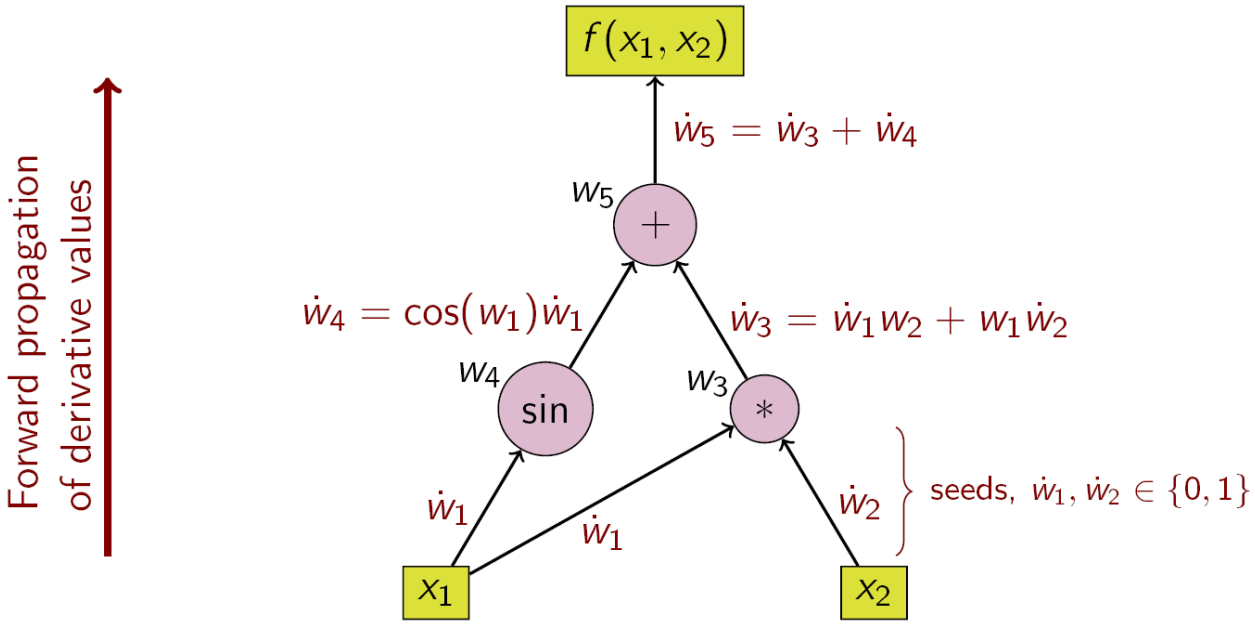


Figure 40- Algorithmic Differentiation

Gradients or Derivatives play an important role in Optimization Procedures and many Scientific Solutions. Training a Deep Learning Network is basically optimization (minimization) of the cost function.

Automatic differentiation (AD) is a conceptually well-established tool to calculate numerical gradients up to machine precision, by iteratively applying the chain rule and successively walking through the computational graph of numerically implemented functions. Therefore, AD facilitates the computation of exact derivatives of coded algorithms with respect to all its variables without having to implement any other expression explicitly. AD circumvents challenges arising in traditional approaches. For example, the evaluation of analytical expressions tends to be inefficient, and numerical gradients can be unstable due

to truncations or round-off errors. AD has been successfully implemented to various applications. For example, AD has been used for quantum control in open-quantum systems to find an optimal time-dependent field to obtain a specific state. Leung et. al used a parallelized machine learning tool, TensorFlow, to obtain the optimal parameters to minimize several kinds of cost-functions to tune the evolution of quantum states.

Using AD we can compute gradients of variational quantum circuits in a way that is compatible with classical techniques such as backpropagation. The automatic differentiation algorithms common in optimization and machine learning can be extended to include quantum and hybrid computations like optimization of variational quantum eigensolvers, quantum approximate optimization, quantum machine learning models etc.

BRAIN REPLAY

Researchers have created a new way of adapting a neuroscience concept called “brain replay” to the digital realm of artificial neural networks to enable **continuous learning**.

From a neuroscience perspective, the concept of brain replay is analogous to a streaming service that activates repeat showings from its vast archives of stored pre-recorded content. The brain can replay memories by reactivating the neural activity patterns that represent prior experiences, whether asleep or awake. This ability for memory replay starts in the hippocampus, then continues in the cortex.

In incremental learning, it seems logical to expect an algorithm that was initially trained to learn to classify cats and dogs, then cows and goats would be fully capable of differentiating goats from cats. However, artificial neural networks currently lack this ability, and often require costly retraining in order to learn new tasks, such as, in this example, distinguishing between a goats and a cat.

So how can the brain replay its past experiences?

One approach is using memory data storage where the neural network stores and retrieves data examples for an exact replay. But this is costly and time-consuming.

Another approach is to generate the data for replay. Generative replay system architecture normally consists of two parts: a main neural network model that acts like the cortex, and the generator neural network that acts like the hippocampus.

Brain Replay techniques enable artificial neural networks to learn incrementally from experience in a more scalable and efficient manner.

TRANSFORMERS

The Transformer is a deep learning model introduced in 2017, used primarily in the field of natural language processing (NLP).

Like recurrent neural networks (RNNs), Transformers are designed to handle sequential data, such as natural language, for tasks such as translation and text summarization. However, unlike RNNs, Transformers do not require that the sequential data be processed in order. For example, if the input data is a natural language sentence, the Transformer does not need to process the beginning of it before the

end. Due to this feature, the Transformer allows for much more parallelization than RNNs and therefore reduced training times.

Since their introduction, Transformers have become the model of choice for tackling many problems in NLP, replacing older recurrent neural network models such as the long short-term memory (LSTM). Since the Transformer model facilitates more parallelization during training, it has enabled training on larger datasets than was possible before it was introduced. This has led to the development of pretrained systems such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), which have been trained with huge general language datasets, and can be fine-tuned to specific language tasks.

Transformers are used for processing sequential data, such as natural language text, genome sequences, sound signals or time series data. Yet most applications of transformer neural networks are in the area of natural language processing.

The secret sauce in transformers is the attention mechanism. The attention mechanism represents how important other tokens in an input are for the encoding of a given token. For example, in a machine translation model, the attention mechanism allows the transformer to translate words like 'it' into a word of the correct gender in French or Spanish by considering all relevant words in the original sentence.

TRANSFER LEARNING

Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. From the practical standpoint, reusing or transferring information from previously learned tasks for the learning of new tasks has the potential to significantly improve the sample efficiency of a reinforcement learning agent. And it can be used to create a library of pre-trained deep learning models which can be used to inherit from and incorporated in a new deep learning model. By freezing the inherited parts and only fine tuning the new parts we can very quickly and economically train new specific models.

Transfer learning reuses any relevant training data, feature representations, neural-node architectures, hyperparameters and other properties of existing models.

It is an engineering tactic. And it is very commonly used.

DATA, MODEL AND PIPELINE PARALLEISM

Training models need to be accurate and yet at the same time need to be done faster. The natural thought is to use parallelism. Of which there are various types.

Data Parallelism

There are two standard ways to speed up moderate-size DNN models. The **data parallelism** approach employs more machines and splits the input data across them. Another way is to move the model to accelerators, such as GPUs or TPUs, which have special hardware to accelerate model training. However, accelerators have limited memory and limited communication bandwidth with the host machine.

Model Parallelism

Thus, model parallelism is needed for training a bigger DNN model on accelerators by dividing the model into partitions and assigning different partitions to different accelerators. But due to the sequential nature of DNNs, this naive strategy may result in only one accelerator being active during computation, significantly underutilizing accelerator compute capacity. On the other hand, a standard data parallelism approach allows concurrent training of the same model with different input data on multiple accelerators, but cannot increase the maximum model size an accelerator can support.

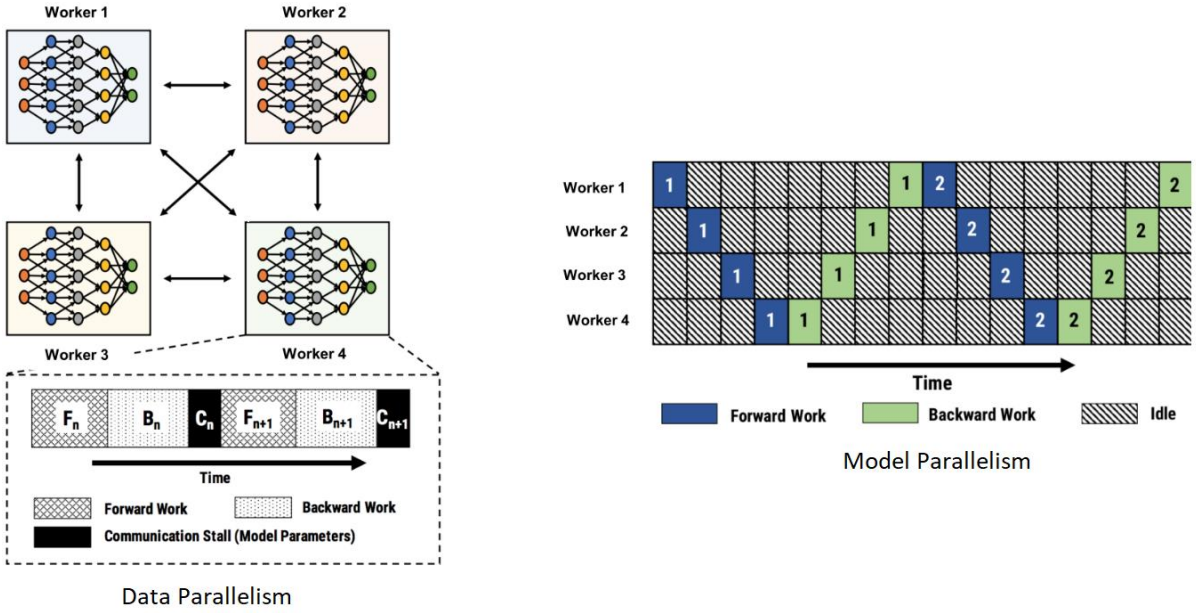


Figure 41- Data Parallelism & Model Parallelism

Pipeline Parallelism

To enable efficient training across multiple accelerators, we partition a model across different accelerators and automatically splits a mini-batch of training examples into smaller micro-batches. By pipelining the execution across micro-batches, accelerators can operate in parallel. In addition, gradients are consistently accumulated across micro-batches, so that the number of partitions does not affect the model quality.

Pipeline-parallel computation involves partitioning the layers of a DNN model into multiple stages, where each stage consists of a consecutive set of layers in the model. Each stage is mapped to a separate GPU that performs the forward pass (and backward pass) for all layers in that stage.

Given a specific deep neural network, we need to determine how to partition the operators of the DNN based on a short profiling run performed on a single GPU, balancing computational load among the different stages while minimizing communication for the target platform.

Pipeline parallelism is the newest kid in town that's painting the town red with its capabilities.

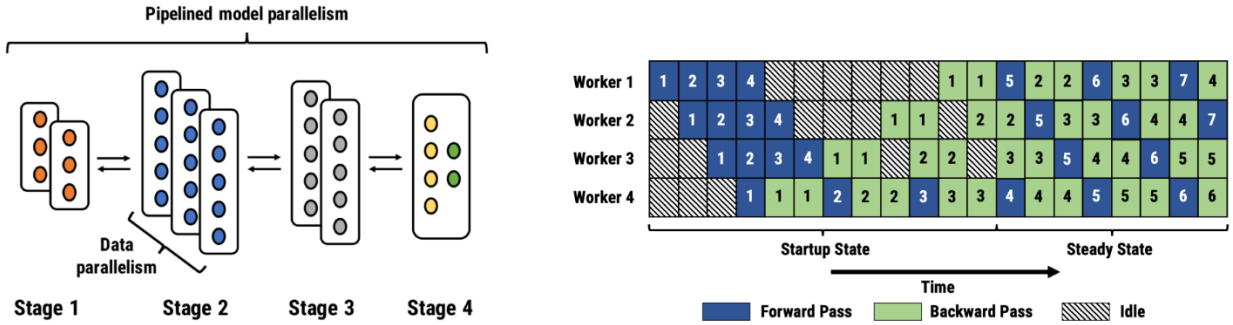


Figure 42- Pipeline Parallelism

DISTRIBUTED TRAINING

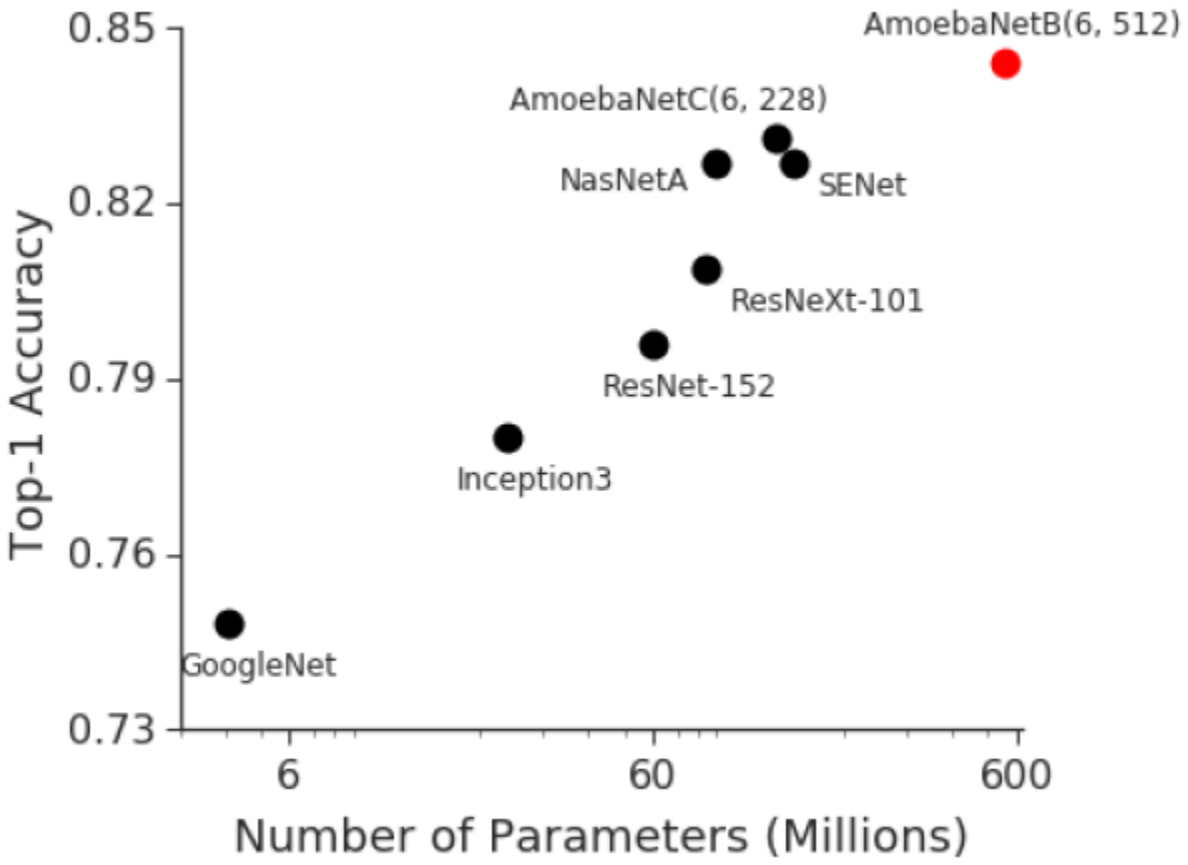


Figure 43- Correlation Between Accuracy & Model Size

There is a strong correlation between model accuracy and model size. The only problem is that the model size needs to grow exponentially for an incremental improvement in accuracy. And we are hitting the limits of computation. Yet we still need to train larger and larger models.

Until a while back the only way to scale training was by using Distributed TensorFlow with parameter servers.

Or, one could use DeepMind TF-Replicator, a framework that helps train TensorFlow models on GPUs and Cloud TPUs. Which uses data and model parallelism. Therefore, in that sense it is old school.

And, that was that until distributed training frameworks given below came along.

Uber Horovod

Horovod is a distributed deep learning training framework for TensorFlow, Keras, PyTorch, and Apache MXNet. Horovod can run on a single-GPU, multiple-GPUs, or even multiple hosts without any further code changes. Horovod core principles are based on MPI (Message Passing Interface) concepts such as size, rank, local rank, allreduce, allgather and, broadcast.

Microsoft PipeDream

PipeDream, a system developed as part of Microsoft Research's Project Fiddle, introduced pipeline parallelism, a new way to parallelize DNN training by combining traditional intra-batch parallelism (model and data parallelism) with inter-batch parallelism (pipelining).

Microsoft DeepSpeed

DeepSpeed delivers extreme-scale model training for everyone, from data scientists training on massive supercomputers to those training on low-end clusters or even on a single GPU:

1. Extreme scale: Using current generation of GPU clusters with hundreds of devices, 3D parallelism of DeepSpeed can efficiently train deep learning models with trillions of parameters.
2. Extremely memory efficient: With just a single GPU, ZeRO-Offload of DeepSpeed can train models with over 10B parameters, 10x bigger than the state of arts, democratizing multi-billion-parameter model training such that many deep learning scientists can explore bigger and better models.
3. Extremely long sequence length: Sparse attention of DeepSpeed powers an order-of-magnitude longer input sequence and obtains up to 6x faster execution comparing with dense transformers.
4. Extremely communication efficient: 3D parallelism improves communication efficiency allows users to train multi-billion-parameter models 2–7x faster on clusters with limited network bandwidth. 1-bit Adam reduces communication volume by up to 5x while achieving similar convergence efficiency to Adam, allowing for scaling to different types of GPU clusters and networks.

Early adopters of DeepSpeed have already produced a language model (LM) with over 17B parameters called Turing-NLG, establishing a new SOTA (State Of The Art) in the LM category.

Google GPipe

GPipe is a distributed machine learning library that uses synchronous stochastic gradient descent and pipeline parallelism for training, applicable to any DNN that consists of multiple sequential layers.

GPipe, was used to train an AmoebaNet-B with 557 million model parameters and input image size of 480 x 480 on Google Cloud TPUv2s. This model performed well on multiple popular datasets, including pushing the single-crop ImageNet accuracy to 84.3%, the CIFAR-10 accuracy to 99%, and the CIFAR-100 accuracy to 91.3%.

NVIDIA TENSORRT

TensorRT is an SDK provided by NVIDIA which focuses on running pre-trained networks quickly and efficiently for inferencing on NVIDIA GPUs and deep learning accelerators. It includes a deep learning inference optimizer and inference runtime that delivers low latency and high-throughput for deep learning inference applications. TensorRT-based applications perform up to 40x faster than CPU-only platforms during inference.

EDGE AI

The best definition of Edge is not that it is far away from the Data Center or Cloud and has a remote connection.

It can be best describe probably in terms of what constraints it operates under…

1. Edge Devices have very limited Energy/Power
2. They have very little Compute Capacity
3. And they have very little Network Capability

The idea that all edge devices will send all the information to a central cloud for processing has met with limited success. Even a comparatively small-scale deployment overwhelms the network infrastructure.

This led to Fog Computing where small cloud infrastructures were placed close to the Edge, which could semi-locally perform computations. And had high bandwidth and compute to leverage the central cloud too on the other side. This too met with limited success as we don't have federated models of computation, which can judiciously use the network and compute to perform globally optimum decision-making.

This led to the thought that we need to bring A.I. to the Edge Devices. Intel offered the Movidius Neural Compute Stick which could be plugged into Edge Devices. Amazon offered DeepLens an Edge Camera that had machine learning compute capability built in and could perform inference locally.

The Best way to do Edge AI is an open problem. The solution to which might involve Federated Compute Infrastructures, Local A.I. ASICs, Memristors, Neuromorphic Computing or even better, new breakthrough compute and energy efficient algorithms on the other extreme.

INTELLIGENT CAMERAS

AI systems perceive the world only after recording and transmitting visual information between sensors and processors. But many things that can be seen are often irrelevant for the task at hand, such as the detail of leaves on roadside trees as an autonomous car passes by. However, at the moment all this information is captured by sensors in meticulous detail and sent clogging the system with irrelevant data, consuming power and taking processing time. A different approach is necessary to enable efficient vision for intelligent machines.

"We can borrow inspiration from the way natural systems process the visual world — we do not perceive everything — our eyes and our brains work together to make

sense of the world and in some cases, the eyes themselves do processing to help the brain reduce what is not relevant."

By implementing Convolutional Neural Networks (CNNs) directly on the image plane. The CNNs the University of Manchester team has developed can classify frames at thousands of times per second, without ever having to record these images or send them down the processing pipeline.

Figure 44- SCAMP-5d vision system. Ref: The University of Manchester, 2020

The SCAMP is a camera-processor chip that the team describes as a Pixel Processor Array (PPA). A PPA has a processor embedded in each and every pixel which can communicate with each other to process in truly parallel form. This is ideal for CNNs and vision algorithms.

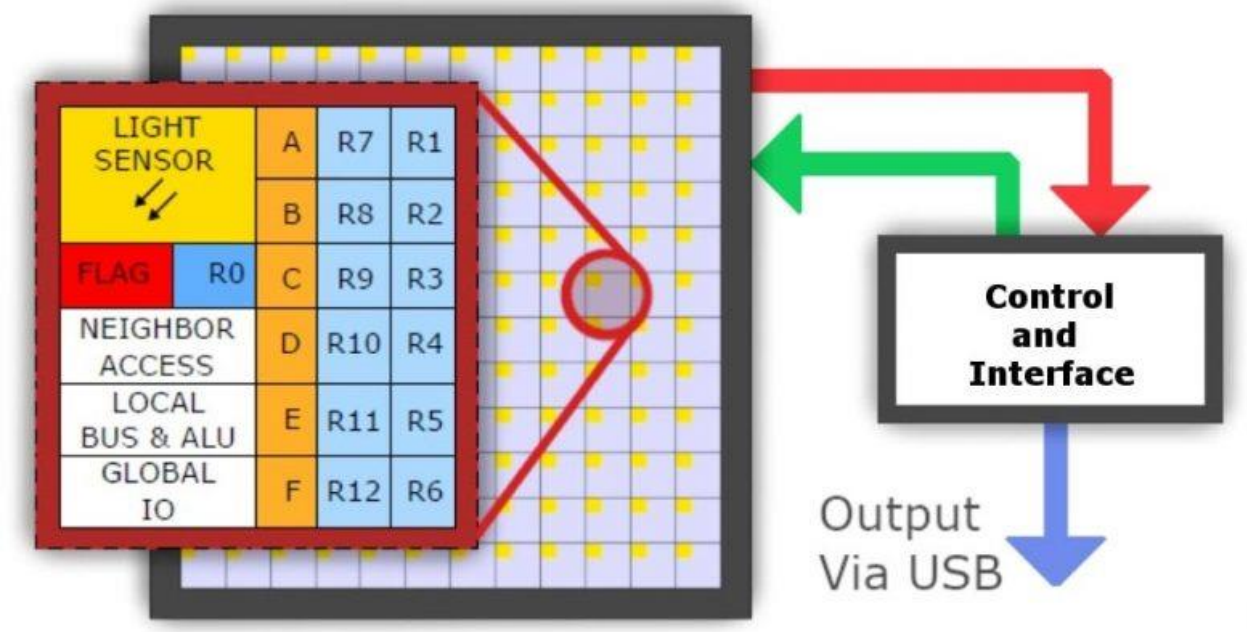


Figure 45- SCAMP-5d's hardware architecture. It incorporates a 256 x 256 PPA array of pixel-processors, each containing light sensor, local memory registers and other functional components. Ref: The University of Manchester, 2020

PROBABILISTIC COMPUTING

Natural data is noisy, unstructured, has omissions and even conflicting information. And we need to deal with uncertainty in the real world. Probabilistic Models can be self learnt in a self supervised fashion. And hence is one step closer to the vision of The Fathers of Deep Learning 'Self Supervised Learning'. Probabilistic Computing has been combined with General Purpose Programming and Deep Learning too.

Each variable is not a single value it is a probability distribution. This is non-intuitive to most engineers. 3D Graphical Plots help with visualization and come to the rescue. Probabilistic Computing computes probability distributions at its core. And hence its name Probabilistic Computing.

And it has proven to deliver more accurate and reliable results in real world scenarios beating a lot of benchmarks. Though yet it remains in the shadows of Deep Learning which is being pushed by large vendors and investors. It is quite unknown and unheard of in the developer community.

Probabilistic Computing improves our ability to engineer artificial intelligence and reverse-engineer natural intelligence. By integrating probabilistic inference, generative models, and monte carlo methods into the building blocks of software, hardware, and other computational systems.

GENERATIVE AI

Today's machine learning models mostly interpret and classify existing data: for instance, recognizing faces or identifying fraud. Generative AI is a fast-growing new field that focuses instead on building AI that can generate its own novel content. To put it simply, generative AI takes artificial intelligence beyond perceiving to creating.

Two key technologies are at the heart of generative AI: generative adversarial networks (GANs) and variational autoencoders (VAEs). Of which GANs are the preferred choice.

The magic sauce is an Adversarial Architecture which has two parts fighting with each other to produce better and better synthetic data (generated images, video's, text, music, designs, applications, art… are basically data).

Deepfakes was one of the first most prominent outcomes of generative A.I.

IMITATION LEARNING

If an agent instead of learning based on hit and trial exploration, rewards and punishments. Learnt by observing some demonstrations of some skill and reasoning about them in its mind. Its called learning by imitation or imitation learning.

Humans learn social skills by behavioral imitation. They learn to express emotions and to respond to emotions by imitation.

Imitation learning is useful when it is easier for an expert to demonstrate the desired behavior rather than to specify a reward function which would generate the same behavior or to directly learn the policy.

Figure 46- Imitation

Behavior Cloning is the simplest naïve method by which an agent takes a set of state and action pairs. And then the agent undergoes supervised learning to learn exactly those actions for those states. But behavior cloning doesn't generalize well beyond the given examples. As there is no inherent learning other than

just cloning. Yet it is great in some scenarios like setting up industrial robots to perform some fixed tasks on the assembly line.

The difference between behavior cloning and imitation learning is the ability to generalize to scenarios beyond the given set of examples.

Imitation Learning combined with Causal Reasoning can learn with a few examples. And learning by watching demonstrations by experts is safer than learning to drive a car by hit and trial methods.

FEDERATED LEARNING

Federated learning (also known as collaborative learning) is a machine learning technique that trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them. This approach stands in contrast to traditional centralized machine learning techniques where all the local datasets are uploaded to one server, as well as to more classical decentralized approaches which often assume that local data samples are identically distributed.

Federated learning enables multiple actors to build a common, robust machine learning model without sharing data, thus allowing to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data. Its applications are spread over a number of industries including defense, telecommunications, IoT, and pharmaceutics.

Federated learning makes it possible for AI algorithms to gain experience from a vast range of data located at different sites. The approach enables several organizations to collaborate on the development of models, but without needing to directly share sensitive data with each other. Over the course of several training iterations the shared models get exposed to a significantly wider range of data than what any single organization possesses in-house.

The secret ingredient to all this is the central server at the core of the Federated Learning Scheme. Its job is to store the master copy of the model and all the parameters (it's a parameter server). And it shares copies of this model and parameters to the participating nodes which perform their own incremental training in isolation and share the results back to the central server.

REPRESENTATION LEARNING

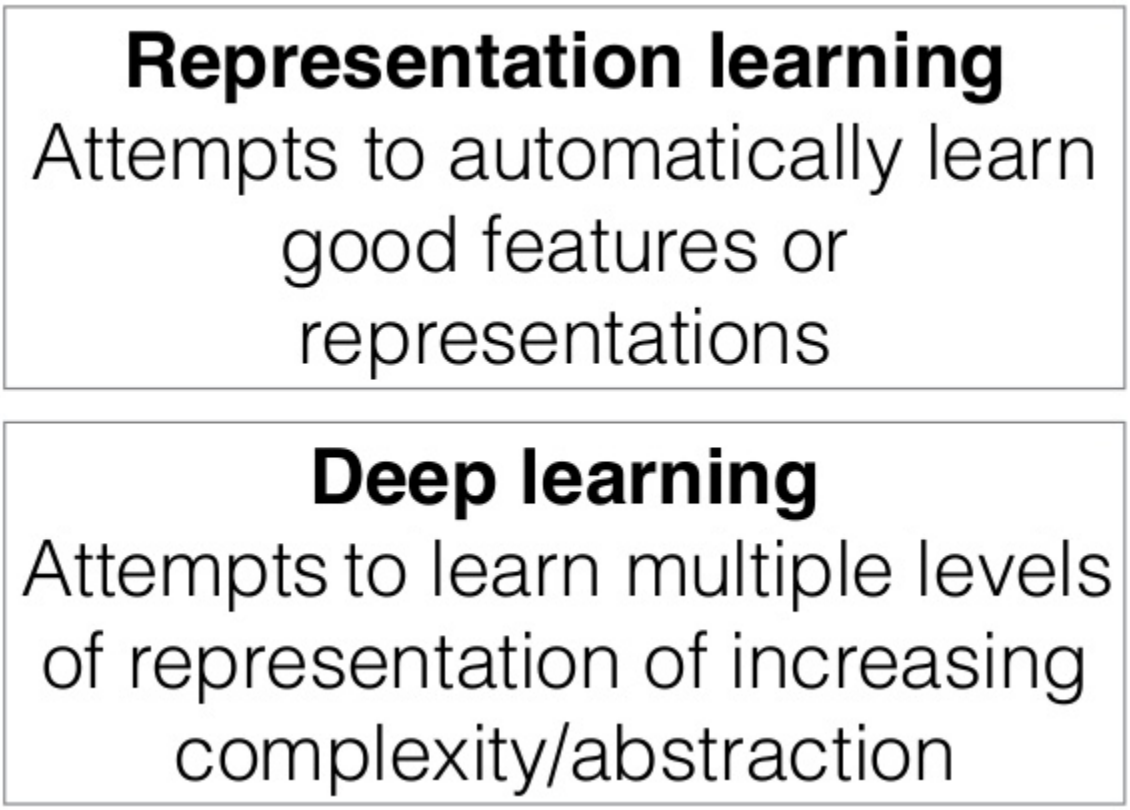


Figure 47- Representation Learning & Deep/Machine Learning

What are representations?

A representation is a formal system which "makes explicit certain entities and types of information," and which can be operated on by an algorithm in order to achieve some information processing goal. Representations differ in terms of what information they make explicit and in terms of what algorithms they support.

The input data needs a suitable representation, so that it fits well with our learning algorithm.

Representation learning is all about finding a representation of the data - the features, the distance function, the similarity function- that dictates how the predictive model will perform.

Also, we can say any Machine Learning or Deep Learning Algorithm boils down to just learning representations and hierarchies of representations.

MULTI-TASK LEARNING

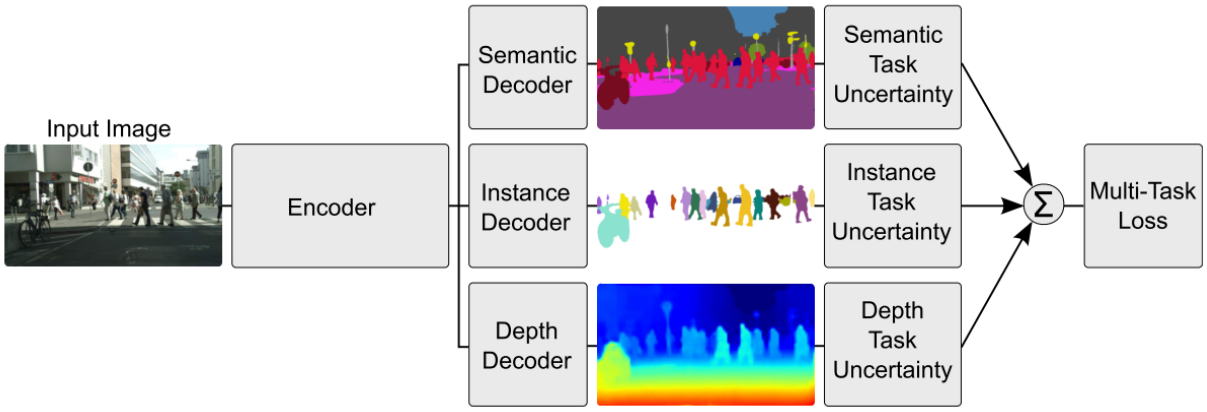


Figure 48- Multi Task Learning (Ref: ruder.io)

With Deep learning models, we usually aim to learn a good representation of the features or attributes of the input data to predict a specific value. Formally, we aim to optimize for a particular function by training a model and fine-tuning the hyper parameters until the performance cannot be increased further.

By using Multi-Task Learning, it might be possible to increase performance even further by forcing the model to learn a more generalized representation as it learns (updates its weights) not just for one specific task but a bunch of tasks.

Biologically, humans learn in the same way. We learn better if we learn multiple related tasks instead of focusing on one specific task for a long time.

So instead of training a neural network to play Go, we also train it to play chess, checkers etc. at the same time. So the neural network can learn representations of each game and also shared representations of all the games. This shared knowledge is more akin to intelligence than just learning to recognize cats or elephants by itself.

We hope to create Generalized Learning Algorithms and not one trick ponies.

META-LEARNING

Meta learning is a subfield of machine learning where automatic learning algorithms are applied to metadata about machine learning experiments. As of 2017 the term had not found a standard interpretation, however the main goal is to use such metadata to understand how automatic learning can become flexible in solving learning problems, hence to improve the performance of existing learning algorithms or to learn (induce) the learning algorithm itself, hence the alternative term **learning to learn**.

Flexibility is important because each learning algorithm is based on a set of assumptions about the data, its inductive bias. This means that it will only learn well if the bias matches the learning problem. A learning algorithm may perform very well in one domain, but not on the next. This poses strong restrictions on the use of machine learning or data mining techniques, since the relationship between the learning problem (often some kind of database) and the effectiveness of different learning algorithms is not yet understood.

By using different kinds of metadata, like properties of the learning problem, algorithm properties (like performance measures), or patterns previously derived from the data, it is possible to learn, select, alter or combine different learning algorithms to effectively solve a given learning problem. (Ref: Wikipedia)

Compared to machines which require a huge number of samples to learn from, humans can recognize a cat from a dog after just seeing a few examples. People who know how to ride a bike can learn to drive a car very easily. On the face of it it seems that humans are inherently capable of learning with very few examples. But under the hoods, humans are generalizing and learning to learn each time. If machines could learn to learn then they could too learn with very few examples, and learn new concepts and skills very fast. That is what learning to learn or meta-learning hopes to solve in machine learning.

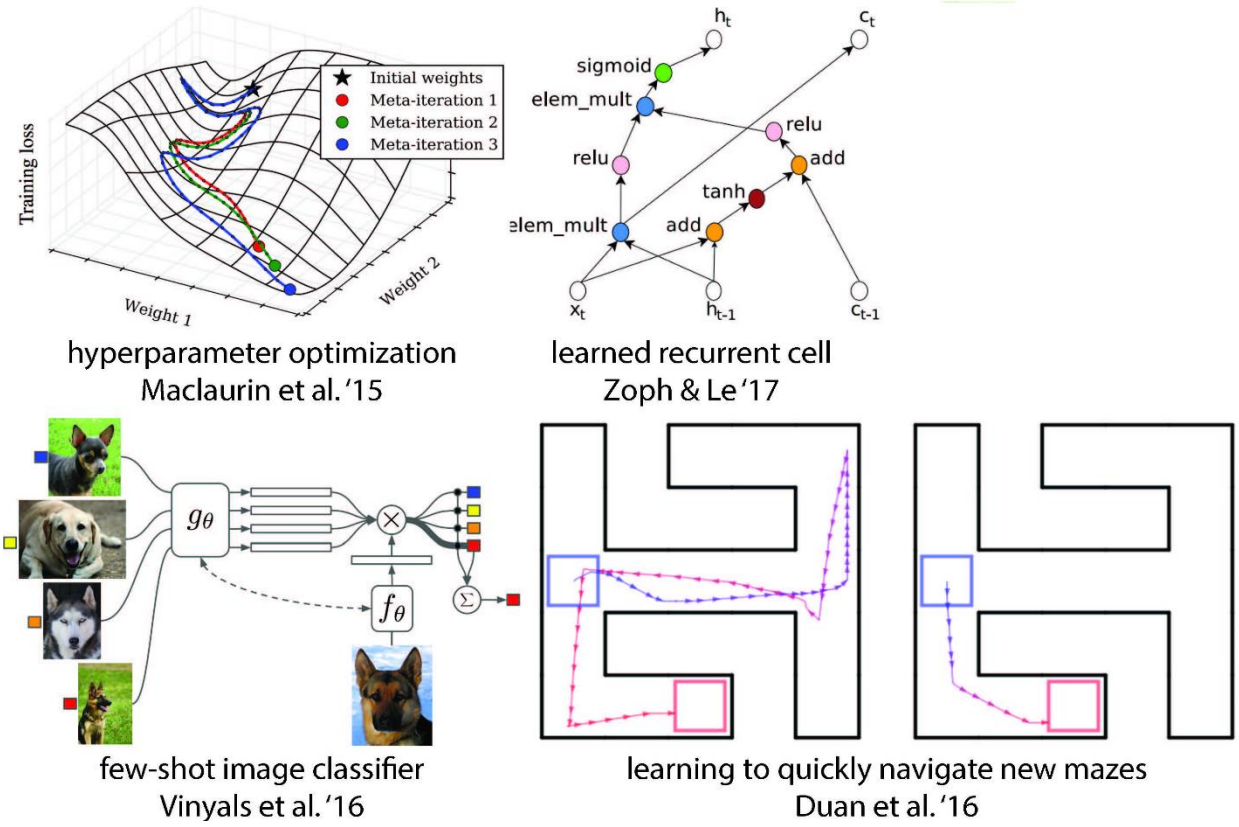


Figure 49- Various Current Meta Learning Approaches

NEURAL NETWORK COMPRESSION

Deep Neural Networks are pushing the limits of computation. They are both computation intensive and memory intensive. It is difficult to run inference on trained models not only at the Edge in limited resource scenarios. But even in Data Centers as Deep Learning Models approach Trillions of Neurons scale. A lot of cutting edge models even today have double digit billions of parameters.

Lets say researchers and engineers develop and train a model. And we can tolerate a further error of +/- 2%. What can we do to compress it?

The first technique used is pruning, which simply means removing nodes and connections, aka removing neurons and synapses, which don't have a large impact on the results or don't contribute significantly in comparison to the results.

The second technique is weight sharing. Neural networks are made up of parameters and each parameter is a 64 bit floating point number. If there are billions of such numbers we can save space by discretizing/quantization or sharing weights/parameters by using the most common 'N' nearest neighbors. Lets say the most common 100 parameter values. If that's within our error tolerance limit, Yay! Nearest neighbors is many times better than linear fixed interval quantization.

The third technique is using low precision fixed point numbers for example 16 bit floating point numbers instead of 64 bit ones.

Large networks like AlexNet have recently been reengineered into compact architectures like ShuffleNet and MobileNet using such techniques. These networks are less than half their original size and have improved their efficiency, for example, MobileNet gives a 7x speedup.

Though all this is not a simple process. One has to take care of saddle points, plateaus, regularization, re-initialization, re-training, breaking symmetry, distributed training, mixed precision, gradient accumulation, check pointing, and layer and tensor fusion.

ENERGY EFFICIENCY

Engineers rarely think in terms of energy. But in fact energy is a fundamental limiting factor of what's possible computationally. Energy Efficient Machine Learning and Deep Learning is a vast research area in itself. Which takes directly into account the energy resources of wearables, mobiles, embedded systems, edge devices that need to run such models. And uses such constraints to drive the design and engineering of neural network models.

EXTREME CLASSIFICATION

Extreme Classification is the problem of classification of data into classes where the number of classes is very very large, in millions. Extreme Classification is used for example in search engines.

Taking online shopping as an example here, one of the biggest challenges when it comes to online product search is the sheer number of products. There are millions of people shopping, using millions of words for online products which exceed the number of 100 million products online.

The best state of the art classifiers use Logarithmic Memory and O(K*log(K)) compute.

Merged-Averaged Classifiers via Hashing (MACH) are the state of the art and are used for K-classification with ultra-large values of K (The number of classes). Compared to traditional one-vs-all classifiers that require O(Kd) memory and inference cost, MACH only need O(d log K) (d is dimensionality) memory while only requiring O(K log K + d log K) operation for inference.

MACH is subtly a count-min sketch structure in disguise, which uses universal hashing to reduce classification with a large number of classes to few embarrassingly parallel and independent classification tasks with a small (constant) number of classes.

It provides theoretical quantification of discriminability-memory tradeoff.

With MACH we can train the ODP Dataset with 100,000 classes and 400,000 features on a single GPU, with the classification accuracy of 19.28%, which is the best-reported accuracy on this dataset. Before this work, the best performing baseline is a one-vs-all classifier that requires 40 billion parameters (160 GB model size) and achieves 9% accuracy. In contrast, MACH can achieve 9% accuracy with 480x reduction in the model size (of mere 0.3GB).

With MACH, we can completely train the fine-grained imagenet dataset (compressed size 104GB), with 22k classes, on a single GPU.

3D DEEP LEARNING (3DDL)

Deep learning is proven to be a powerful tool to build models for language (one-dimensional) and image (two-dimensional) understanding. Tremendous efforts have been devoted to these areas, however, it is still at the early stage to apply deep learning to 3D data, despite their great research values and broad real-world applications. In particular, existing methods poorly serve the three-dimensional data that drives a broad range of critical applications such as...

1. Robotics
2. Augmented Reality
3. Autonomous Driving
4. Medical Imaging & Diagnostics
5. Neuroscience
6. Computer Vision
7. Graphics, and
8. Scientific Simulations

Typical objectives in 3D Deep Learning are...

1. 3D Pose Estimation
2. Single Object Classification
3. Multiple Objects Detection
4. Scene/Object Semantic Segmentation
5. 3D Geometry Synthesis/Reconstruction
6. Texture/Material Analysis and Synthesis
7. Style Learning and Transfer
8. Scene Synthesis/Reconstruction
9. Scene Understanding
10. Operations in Noisy Crowded Scenarios

3D data possesses multiple popular representations, such as...

1. Point Cloud
2. Polygonal Mesh
3. Volumetric Field
4. Primitives Based
5. Multi-View Images and

6. Parametric Models

Each suitable for different application scenarios. Each type of data format has its own properties that pose challenges to neural network design while also provide the opportunity for novel and efficient solutions.

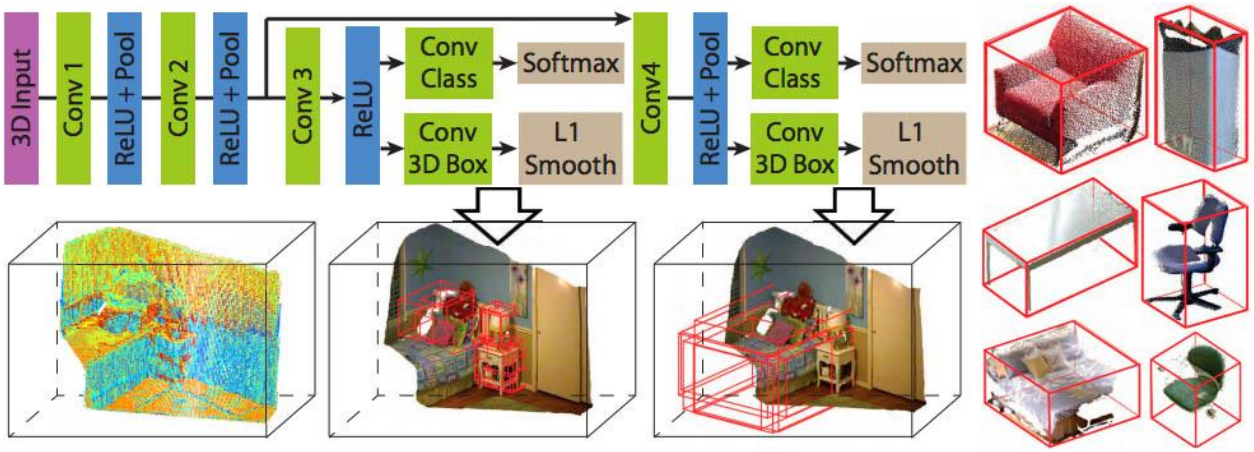


Figure 50- 3D Convolutional Neural Networks

3D CNN (convolutional neural networks) are an extension of regular CNN architectures applied to 3D Environments and Data. Most applications of neural network models to 3D data require extensive and manual pre-processing of 3D data.

Geospatial Deep Learning is a specific niche application of 3DDL used in extracting geographical features from satellite imagery and point cloud datasets. Such features are very small compared to most Vision problems.

Pytorch3D supports learning in 3D Environments.

DIFFERENTIABLE GRAPHICS LAYERS

A computer graphics pipeline requires 3D objects and their absolute positioning in the scene, a description of the material they are made of, lights and a camera. This scene description is then interpreted by a renderer to generate a synthetic rendering.

In comparison, a computer vision system would start from an image and try to infer the parameters of the scene. This allows the prediction of which objects are in the scene, what materials they are made of, and their three dimensional position and orientation.

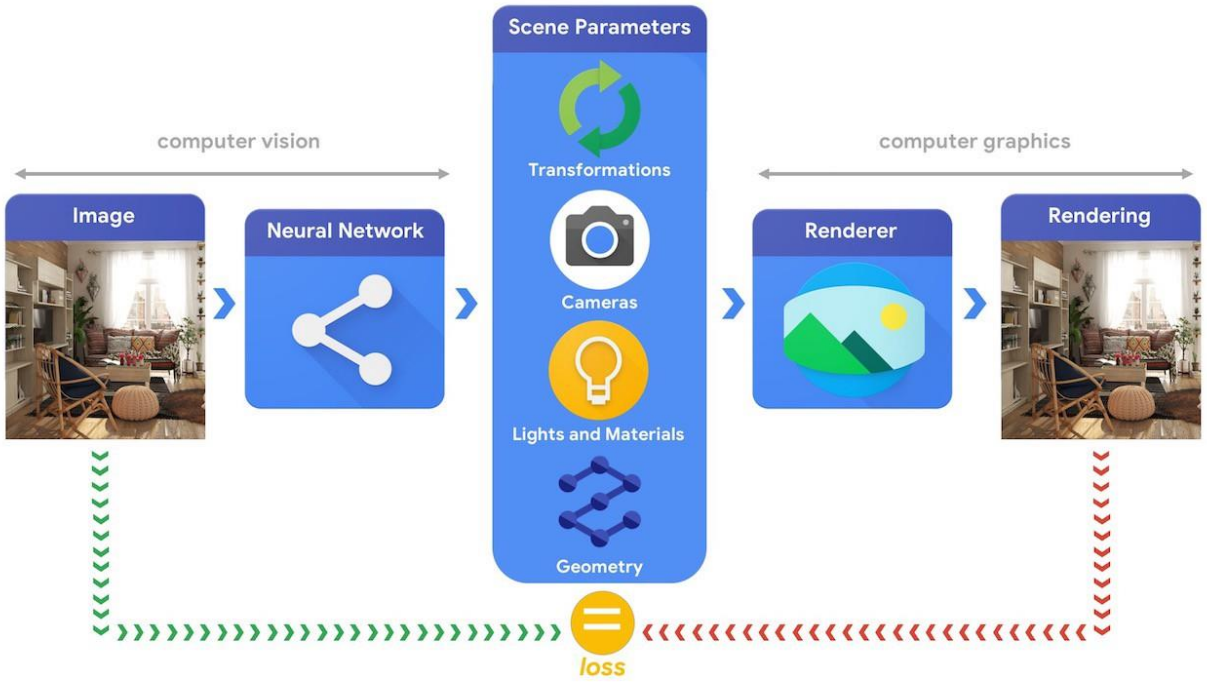


Figure 51- Combining Vision & Graphics In Deep Neural Networks

We can combine the two (Vision & Graphics) pipelines where the vision system extracts the scene parameters and the graphics system renders back an image based on them. If the rendering matches the original image, the vision system has accurately extracted the scene parameters. In this setup, computer vision and computer graphics go hand-in-hand, forming a single machine learning system similar to an autoencoder, which can be trained in a self-supervised manner.

The name Differentiable Graphics Layers comes from the fact that incorporating Graphics functionality in Deep Neural Networks as layers. Requires that the layer and all such functionality be mathematically differentiable. Because training only works if backpropagation can differentiate all the layers and hence estimate parameters in the neural network. (This is a limitation of current Deep Learning algorithms which has been mentioned by Geoffrey Hinton. And has been fixed by Automatski i.e. Automatski's Deep Learning Platform has eliminated Backtracking completely. Just like the fathers of deep learning said the future of Deep Learning has to be)

CAN A.I. DO SYMBOLIC MATH?

Guillaume Lample and François Charton, scientists working in Facebook's AI research group in Paris, unveiled a successful first approach to solving symbolic math problems with neural networks. Their method didn't involve number crunching or numerical approximations. Instead, they played to the strengths of neural nets, reframing the math problems in terms of a problem that's practically solved very well: language translation.

They could produce precise solutions to complicated integrals and differential equations — including some that stumped popular math software packages.

EQUATION	SOLUTION
$y' = \frac{16x^3 - 42x^2 + 2x}{(-16x^8 + 112x^7 - 204x^6 + 28x^5 - x^4 + 1)^{1/2}}$	$y = \sin^{-1}(4x^4 - 14x^3 + x^2)$
$3xy\cos(x) - \sqrt{9x^2\sin(x)^2 + 1y'} + 3y\sin(x) = 0$	$y = c\exp(\sinh^{-1}(3x\sin(x)))$
$4x^4yy'' - 8x^4y'^2 - 8x^3yy' - 3x^3y'' - 8x^2y^2 - 6x^2y' - 3x^2y'' - 9xy' - 3y = 0$	$y = \frac{c_1 + 3x + 3\log(x)}{x(c_2 + 4x)}$

Figure 52- Can A.I. Do Symbolic Math?

"When we see a large function, we can see that it's composed of smaller functions and have some intuition about what the solution can be," Lample said. "We think the model tries to find clues in the symbols about what the solution can be." He said this process parallels how people solve integrals — and really all math problems — by reducing them to recognizable sub-problems they've solved before.

REASONING

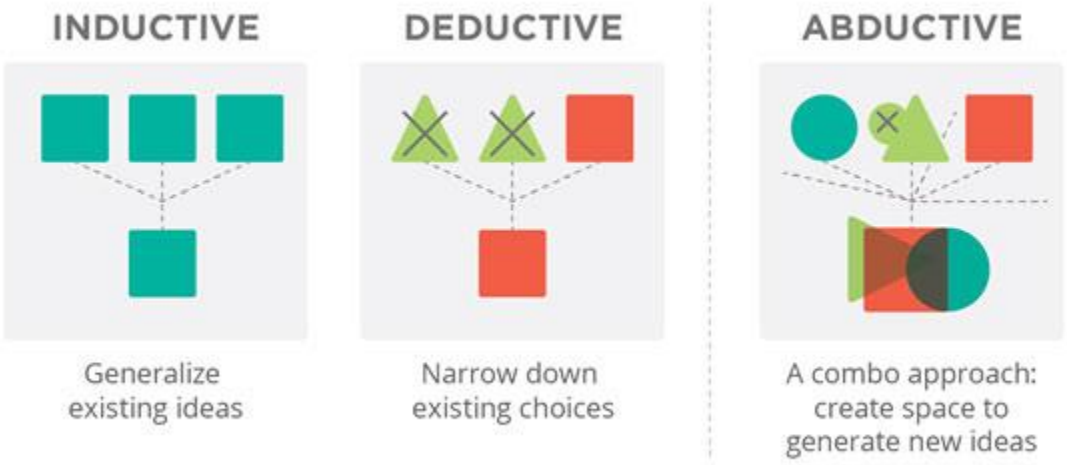


Figure 53- Reasoning

Reasoning is the mental process of deriving logical conclusion and making predictions from available knowledge, facts, and beliefs. Or, "Reasoning is a way to infer facts from existing data." It is a general process of thinking rationally, to find valid conclusions.

In artificial intelligence, the reasoning is essential so that the machine can also think rationally. Which also means that they will be useful otherwise they will just be relegated to amusement or a nuisance.

*** Automatski's Definition of A.I. is "Logically Perfect"

There are three types of reasoning in principle…

Abductive Reasoning

Incomplete Observations -> Best Prediction (Maybe True)

Inductive Reasoning

Specific Observation(s) -> General Conclusion (Maybe True)

Deductive Reasoning

General Rule -> Specific Conclusion (Always True)

A reasoning system is a software system that generates conclusions from available knowledge using logical techniques such as deduction and induction. Reasoning systems play an important role in the implementation of artificial intelligence, robotics and knowledge-based systems.

Reasoning systems have a wide field of application that includes scheduling, business rule processing, problem solving, complex event processing, intrusion detection, predictive analytics, robotics, computer vision, and natural language processing.

KNOWLEDGE REPRESENTATION AND REASONING

Knowledge representation and reasoning (KR^2, KR&R) is the field of artificial intelligence (AI) dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in a natural language. Knowledge representation incorporates findings from psychology about how humans solve problems and represent knowledge in order to design formalisms that will make complex systems easier to design and build. Knowledge representation and reasoning also incorporates findings from logic to automate various kinds of reasoning, such as the application of rules or the relations of sets and subsets.

Examples of knowledge representation formalisms include semantic nets, systems architecture, frames, rules, and ontologies. Examples of automated reasoning engines include inference engines, theorem provers, and classifiers.

DEEP REASONING

Reasoning is an NP-Hard Problem with Exponential Complexity to get a perfect solution. Hence many Deep Learning Approaches are being used to solve reasoning aka Deep Reasoning.

Basically Deep Reasoning combines logic and constraint reasoning with stochastic-gradient-based scheme of neural network optimization/learning.

For example Deep Reasoning can solve standard combinatorial problems -- 9-by-9 Sudoku puzzles and Boolean satisfiability problems (SAT)

An extension of Deep Reasoning is Deep Hierarchical Reasoning, where the system learns and reasons with a hierarchy of information, and plans.

Other extensions are

1. Symbolic Reasoning e.g. reasoning in algebra, mathematics, science or symbols
2. Visual Reasoning e.g. objects, relations, features, attributes, functionalities
3. Concept & Meta-Concept Reasoning e.g. figuring out the inside broader meaning and generalizing the learnt concept
4. Geometric Reasoning e.g. shapes, objects
5. Spatial Reasoning – physics, space, motion, obstacles, paths
6. Common Sense Reasoning
7. Natural Language Reasoning

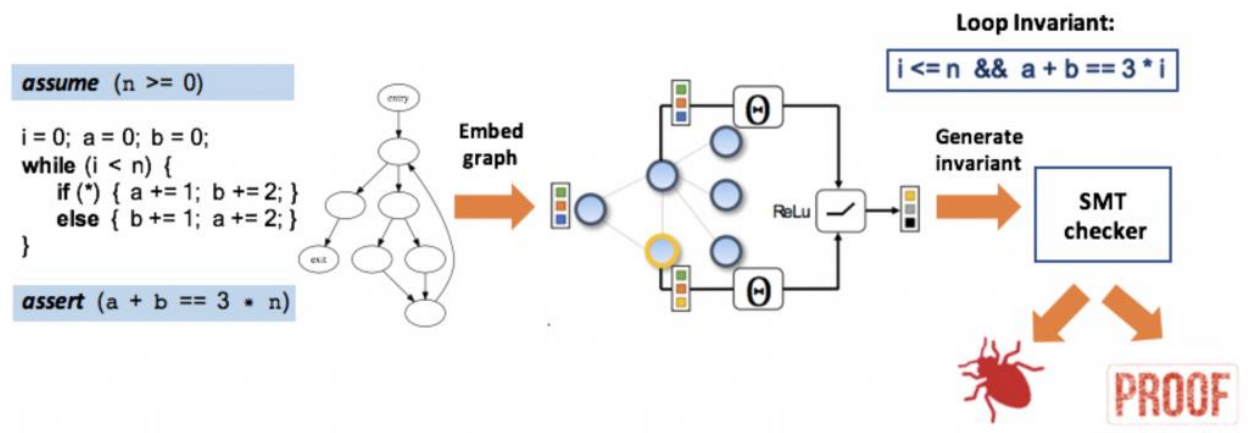


Figure 54- Deep Logical Reasoning

BENEVOLENT A.I.

Figure 55- Benevolent A.I.

If A.I. can do symbolic math for us. The next logical question to ask is what all can it do? And even if we make the most powerful A.I. possible, what is the best way to exploit it for the benefit of mankind.

The general idea is that if we can teach the most powerful A.I. we can create, everything we know. All our science and technologies. Can we ask it to solve problems for us? Which we cannot solve ourselves.

Assuming this A.I. is powerful and more computational infrastructure, time and resources will make it more powerful. And if it is benevolent. Then the answer seems to be yes!

Such an A.I. is called Benevolent A.I. We can ask that computer to invent all the things that humans invent or want to invent. All the problems that have stymied the human race for decades like curing cancer or inventing Fusion Power, making Space Travel cheap, reversing Climate Change, better Fertilizers, Solar Capture, Drug Discovery and Drug Design. Intelligent systems could help us solve such problems for us in hours or days.

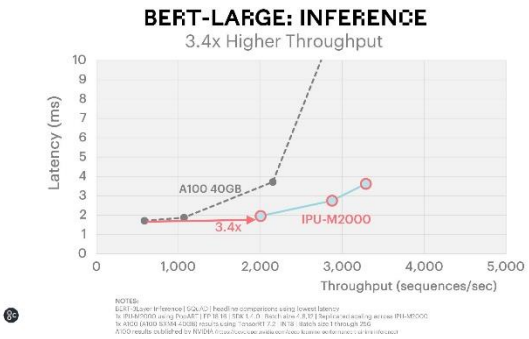
WHICH COMPUTATIONAL INFRASTRUCTURE TO INVEST IN?

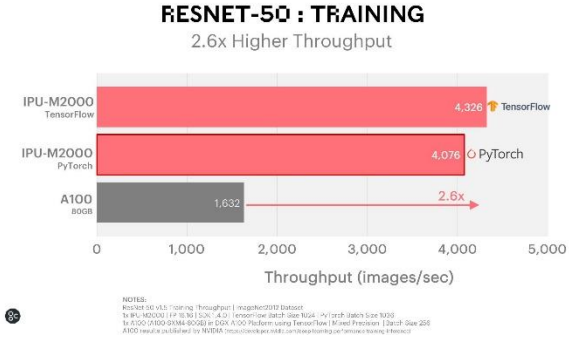
If someone just needs to get started with Deep Learning he should probably get himself a top of the line GPU like **NVIDIA RTX 3090** for his workstation.

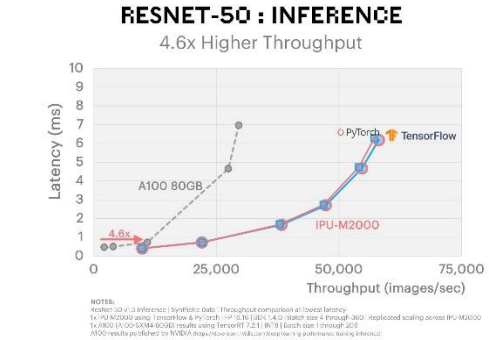
But the next choice is quite interesting. If one were to decide on a Server Rack Infrastructure to invest in. The natural choice to look at is **NVIDIA DGX A100-80GB**. But that's probably not the best choice to go with.

Recent results show its way better to adopt **Graphcore** hardware infrastructure and tooling's instead. Its far superior tech, and painless deployment and maintenance. They are a dropin replacement for GPU's and support all major Deep Learning frameworks out of the box.

The IPU-M2000 system from Graphcore scales to PetaFlops and significantly outperforms the Nvidia A100 DGX across the board, with orders of magnitude performance improvements for some models. It delivers faster time to result, model accuracy, better efficiency, lower TCO (Total Cost of Ownership) or the chance to make new breakthroughs in AI with the IPU.







If someone is doing something really specific and has specific specialized routines to accelerate, which are mature and don't change much. Then he is probably better off programming an FPGA.

A.I. ASICs

Purpose-built chips (aka A.I. ASICs) for running machine learning tasks in data centers are all the rage among the Super 7 (Amazon, Facebook, Google, Microsoft, Alibaba, Baidu, and Tencent). You need them for two tasks broadly 'training' and 'inference'. Over the lifetime of a solution these two tasks share costs in the ratio of 10% to 90%. The needs of both of these use cases are different and mostly warrant a separate chip each.

A.I. ASICs are more energy efficient than both GPU's and FPGA's etc. More and more startups and vendors are offering A.I. ASICs. And each ASIC usually supports Tensorflow and Pytorch frameworks out of the box. The first one to create an A.I. ASIC was Google which created TPU (Tensor Processing Unit), which delivered about 45 TFlops/TPU. A reasonable estimate to create an A.I. ASIC, board etc. is ~ $100 million.

Amazon also announced its chips Trainium and Inferentia, for training and inference respectively, available on the AWS Cloud. Which it claims surpasses all existing ASIC's and GPU based systems.

Proprietary Hardware Accelerators or A.I. ASICs which tie one to a vendors specific Cloud Platform are probably not worth the lockin sometimes. So one needs to decide on a case by case basis.

One of the other earliest startups was Nervana which was acquired by Intel. The Intel Nervana Neural Network Processors (NPP) — NNP-T1000 for training and NNP-I1000 for inference is available for purchase. The Intel Movidius Vision Processing Unit (VPU) for edge media, computer vision, and inference applications is also available now.

The neuASIC platform from another startup eSilicon was based on swappable Tiles for the specific A.I. functionality the customer required. eSilicon was more like an IP aggregator and integrator, and offered some in-memory computing features which not many others offered at that time. It is a great way to start with a customizable A.I. ASIC.

ARE WE HITTING THE LIMITS OF COMPUTATION?

Yes! It looks like we are maxed out. That's a huge problem nobody is talking about yet. And unless we invent new algorithms and make major breakthroughs in the field of A.I. it seems that we have hit the limits of what is possible computationally and economically. (Or in theoretical sense, how much better we can do with the limits of energy available to spend on A.I.)

Without major breakthroughs in Deep Learning Algorithms and Techniques; reducing the error rate from 11.5% to 1% in ImageNet Classification would require ~$100 billion. In the last 30 years there have been very few major breakthroughs, almost everything has been incremental achievements and exploitation of Big Data and Computational Resources. Real progress in machine learning and deep learning is virtually stagnant.

Benchmark	Error Rate	Computation Required (Gflops)	Environmental Cost (CO2)	Economic Cost ($)
Image Net	Today: 11.5%	10^{14}	10^{4}	10^{6}
	Target 1:5%	10^{19}	10^{10}	10^{11}
	Target 2: 1%	10^{21}	10^{20}	10^{20}
MS COCO	Today: 47%	10^{14}	10^{4}	10^{4}
	Target 1: 30%	10^{23}	10^{14}	10^{15}
	Target 2: 10%	10^{44}	10^{36}	10^{36}

Figure 56- We Have Hit The Limits In A.I. – Conclusion from State of AI Report

THE FUTURE

Artificial Empathy/Affective Computing

Figure 57- Affective Computing

Affective computing is the study and development of systems and devices that can recognize, interpret, process, and simulate human affects. It is an interdisciplinary field spanning computer science, psychology, and cognitive science. One of the motivations for the research is the ability to give machines emotional intelligence, including to simulate empathy. The machine should be able to interpret the emotional state of humans and adapt its behavior to them, giving an appropriate response to those emotions.

Self Supervised Learning

Till now in Deep Learning humans have to manually label the data which is then used to train a machine learning algorithm. This process can be painfully slow. Even if you crowdsource it you have to triple or quadruple verify the labels. This is the most limiting problem in Machine Learning. And progress in the last decade was primarily because of availability of huge labelled Big Data sets and huge computational power.

Self-supervised learning, also known as (Externally - Unsupervised Learning), is an emerging solution to such cases where data labeling is automated, and human interaction is eliminated. It is the solution to fixing Machine Learning proposed by Yan LeCun, one of the fathers of Deep Learning. In self-supervised learning, the learning model trains itself by leveraging one part of the data to predict the other part and generate labels accurately. In the end, this learning method converts an unsupervised learning problem into a 'internally' supervised one behind the scenes. In which the data gets automagically labelled and the

algorithm can supervise itself and learn. Pretty much like humans. The algorithm in a way figures out to make sense of the data and learn by itself.

Capsule Networks

A Capsule Neural Network is a machine learning system that is a type of artificial neural network that can be used to better model hierarchical relationships. The approach is an attempt to more closely mimic biological neural organization. It is Geoffrey Hintons way of fixing whats wrong with CNN's and a part of his efforts to fix Deep Learning by throwing it all away and starting all over again.

CapsuleNets can generalize very well with very less data. They can perform very well in crowded scenes and handle ambiguity very well. Detailed pose information (such as precise object position, rotation, thickness, skew, size, and so on) is preserved throughout the network.

CapsuleNets were first introduced in 2011 by Geoffrey Hinton, et al., in a paper called Transforming Autoencoders, but it was only in November 2017, that Sara Sabour, Nicholas Frosst, and Geoffrey Hinton published a paper called Dynamic Routing between Capsules, where they introduced a CapsuleNets architecture that reached state-of-the-art performance on MNIST (the famous data set of handwritten digit images), and got considerably better results than CNNs on MultiMNIST (a variant with overlapping pairs of different digits)

Energy Based Models

Energy-Based Models discover data dependencies by applying a measure of compatibility (scalar energy) to each configuration of the variables. For a model to make a prediction or decision (inference) it needs to set the value of observed variables to 1 and finding values of the remaining variables that minimize that "energy" level.

In the same way, machine learning consists of discovering an energy function that assigns low energies to the correct values of the target variables, and higher energies to the incorrect values. A so-called "loss functionals," that's minimized during training, is used to measure the quality of the energy functions. Within this framework, there are many energy functions and loss functionals available to design different probabilistic and non-probabilistic statistical models.

What's the Advantage of Energy-based Machine Learning?

Energy-based learning is a unified framework for all these probabilistic and non-probabilistic approaches. Especially for non-probabilistic training of graphical or other structured models. This learning approach is an alternative to probabilistic estimation for classification, prediction and decision-making.

Since proper normalization is not required, energy-based approaches don't have the issues arising from estimating the normalization constant in probabilistic models. Also, the absence of any normalization condition allows for much more flexibility in the design of learning machines. And they are capable of outputting multiple solutions i.e. function approximation with one input and multiple outputs.

Energy based models consist of four ingredients:

1. The energy function: to assign a scalar to each configuration of the variables.
2. Inference: given a set of observed variables, to find values of the remaining variables that minimizes the energy.
3. Learning: to find an energy function that assigns low energy values to correct configurations of the variables, and high energy values to incorrect configurations of the variables.
4. Loss function: minimized during training, which measures to quality of the energy function.

Inductive Bias

Yoshua Bengio wrote a paper titled "Inductive Biases for Deep Learning of Higher-Level Cognition". It asked the question if we have achieved perfect A.I. and just need to scale our models to derive more powerful A.I., or if we are missing something very fundamental which we need to do before successfully making A.I.?

Bengio's goal is to better understand the gap between current deep learning and human cognitive abilities so as to help answer these questions and suggest research directions for deep learning with the aim of bridging the gap towards human-level AI. His main hypothesis is that deep learning succeeded in part because of a set of inductive biases, but that additional ones should be added in order to go from good in-distribution generalization in highly supervised learning tasks (or where strong and dense rewards are available), such as object recognition in images, to strong out-of-distribution generalization and transfer learning to new tasks with low sample complexity.

Remember all the 3 fathers of deep learning have said current deep learning won't get us to A.I. and we have to throw everything and start all over again. And this "Inductive Bias" hypothesis is Bengio's way of fixing what is wrong with Deep Learning today. And as stated in the title of the paper the goal is to create "Higher Level Cognition"

Inductive Bias	Corresponding property
Distributed representations	Patterns of features
Convolution	group equivariance (usually over space)
Deep architectures	Complicated functions = composition of simpler ones
Graph Neural Networks	equivariance over entities and relations
Recurrent Nets	equivariance over time
Soft attention	equivariance over permutations

Figure 58- Examples Of Inductive Biases Currently Used In Deep Learning

This might seem confusing. So its best if we clearly define what Bengio means by Inductive Bias. Perhaps that will clear things up.

The inductive bias (also known as learning bias) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered.

Further, in machine learning, one aims to construct algorithms that are able to learn to predict a certain target output. To achieve this, the learning algorithm is presented some training examples that demonstrate the intended relation of input and output values. Then the learner is supposed to approximate the correct output, even for examples that have not been shown during training.

> **Note**: Without any additional assumptions, this problem cannot be solved since unseen situations might have an arbitrary output value. The kind of necessary assumptions about the nature of the target function are subsumed in the phrase inductive bias.

So basically when trying to predict output from a given previously 'unseen' input. The assumptions we make about the target function that could probably do this is what Bengio calls Inductive Bias. And he outlines examples of Inductive Biases currently used in Deep Learning today. And he goes on, in the paper, to describe a whole lot more.

Bengio's team notes that Deep Learning already incorporates several key inductive biases found in humans and non-human animals. ***They propose that augmenting these inductive biases — with a focus on those involving higher-level and sequential conscious processing*** — could advance Deep Learning from its current successes on in-distribution generalization in highly supervised learning tasks to stronger and more humanlike out-of-distribution generalization and transfer learning abilities.

Bengio et al. discuss inductive biases based on higher-level cognition, declarative knowledge of causal dependencies, and biological inspiration and characterization of higher-level cognition. They leverage the System 1 and System 2 dichotomy introduced by Daniel Kahneman in his book Thinking, Fast and Slow. In this classification, System 1 refers to what current deep learning is very good at — intuitive, fast, automatic, anchored in sensory perception; while System 2 represents rational, sequential, slow, logical, conscious, and expressible with language. The researchers say DL models that can perform System 2 tasks by taking advantage of the computational workhorse of System 1 abilities will be better equipped to deal with dynamic, changing conditions, i.e., will learn to think and behave more like humans.

Lets see where this leads us to. Critics think that the concept of Inductive Biases in humans and animals are used to come up with assumptions about the target function between input and output. But the entire problem of A.I. would then boil down to how will an agent ever comeup with such 'good' assumptions on its own depending on the unseen situations and data. We guess we will have to play wait and watch on this one.

Tensorflow Quantum

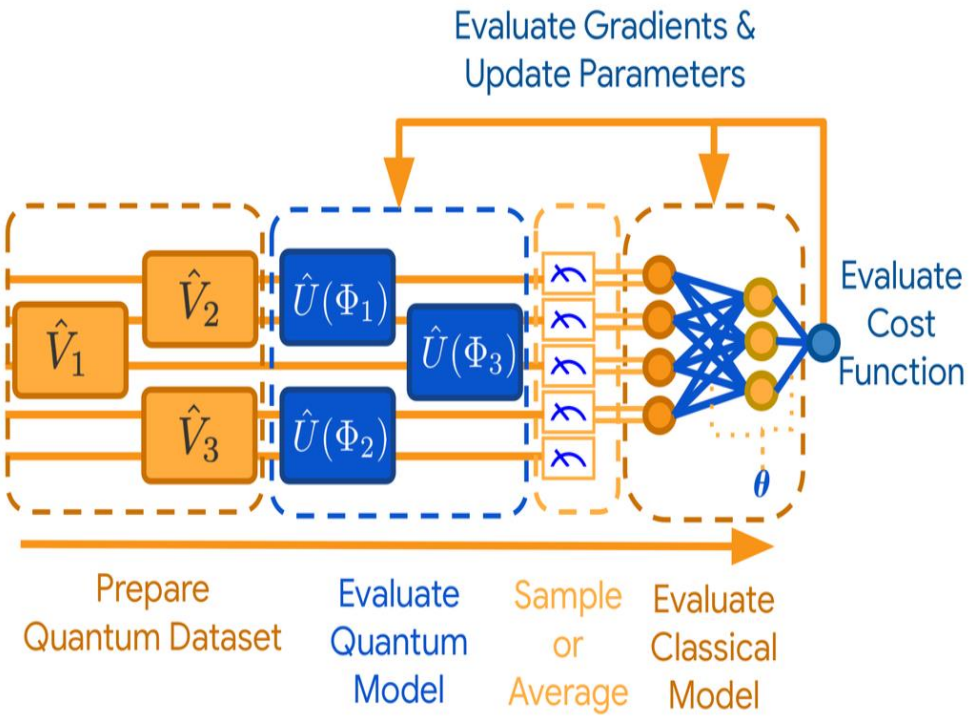


Figure 59- Hybrid Quantum + Deep Learning Algorithms

TensorFlow Quantum (TFQ) is a quantum machine learning library for rapid prototyping of hybrid quantum-classical ML models. It combines Cirq, an open source quantum circuit library, and the TensorFlow machine learning platform.

Today, TensorFlow Quantum is primarily geared towards executing quantum circuits on classical quantum circuit simulators. In the future, TFQ will be able to execute quantum circuits on actual quantum processors that are supported by Cirq, including Google's own processor Sycamore.

It integrates with QSim (Quantum Simulator) which supports executions of quantum circuits with 30 qubits on a laptop and 40 qubits on the cloud. It works great for a lot of machine learning scenarios, without requiring a physical hardware quantum computer.

The future might not be Tensorflow Quantum specifically but the general category of Quantum-Classical Machine Learning Models.

Neurosymbolic Computing

Neurosymbolic models combine neural network elements capable of identifying complex patterns in data with symbolic logic constructs that are able to represent higher level concepts. Reconciling the two dominating symbolic and connectionist paradigms of AI under a principled foundation.

Principled knowledge representation and reasoning mechanisms integrated with deep learning-based systems can provide sound and explainable models.

Neural-symbolic computing aims at integrating, two most fundamental cognitive abilities: the ability to learn from the environment, and the ability to reason from what has been learned.

Neural Networks and Symbolic Computing each by themselves have stagnated and have been proven to not lead to A.I. on their own. Neuro Symbolic Computing seems to be the only possible way forward towards A.I.

Simulated Environments, Incremental Learning & Representations

To succeed in the Real World Reinforcement Learning will have to learn in Simulated Environments. Even if we can do that RL will still be sample inefficient. It requires an exponential number of samples to learn. So we will have to devise new ways of incremental learning, where an Agent can learn both from past experience and new information. We will also need better ways of Representing an Agents Learning (at the core A.I. is mostly about learning Representations, which can be then used to drive decisions, actions or inferences)

Spiking Neural Networks

The 3rd generation of neural networks, spiking neural networks, aims to bridge the gap between neuroscience and machine learning, using biologically-realistic models of neurons to carry out computation. A spiking neural network (SNN) is fundamentally different from the artificial neural networks (ANN) that the machine learning community knows.

The idea behind Spiking Neurons is that information can be coded through the time of arrival of incoming impulses. The neuron fires only when the membrane potential reaches a certain value and then resets, after which it sets a recovery period in which it doesn't respond to inputs before reactivating.

Neurons in an ANN are characterized by a single, static, continuous-valued activation. Yet biological neurons use discrete spikes to compute and transmit information, and the spike times, in addition to the spike rates, matter. Spiking neural networks (SNNs) are thus more biologically realistic than ANNs, and arguably the only viable option if one wants to understand how the brain computes. SNNs are also more hardware friendly and energy-efficient than ANNs, and are thus appealing for technology, especially for portable devices. However, training deep SNNs remains a challenge. Spiking neurons' transfer function is usually non-differentiable, which prevents using backpropagation.

> Sidenote: Training Spiking Neural Networks.
>
> To our knowledge nobody uses Spiking Neural Networks (SNN's) much because nobody knows how to directly train them. The only method which is successful in training SNN's is supervised training of a regular ANN (Artificial Neural Network) and then converting the model into SNN.

*** At Automatski we have fixed this. We have 'completely' eliminated Backpropagation. So we can create A.I. which can use arbitrary functions and schemes. Which may or may not be differentiable. This is exponentially more powerful, scalable and efficient compared to anything we have today. And we can finally also train Spiking Neural Networks with this.

Swarm Intelligence

MIT says - Swarm intelligence is a form of artificial intelligence (AI) inspired by the insect kingdom. In nature, it describes how honeybees migrate, how ants form perfect trails, and how birds flock. In the world of AI, swarm systems draw input from individual people or machine sensors and then use algorithms to optimize the overall performance of the group or system in real time.

Swarm intelligence is now being used to predict everything from the outcome of the Super Bowl to fashion trends to major political events. Using swarm intelligence, investors can better predict market movements, and retailers can more accurately forecast sales.

Neural Turing Machines

A Neural Turing machine (NTMs) is a recurrent neural network model. The approach was published by Alex Graves et. al. in 2014. NTMs combine the fuzzy pattern matching capabilities of neural networks with the algorithmic power of programmable computers. An NTM has a neural network controller coupled to external memory resources, which it interacts with through attentional mechanisms. The memory interactions are differentiable end-to-end, making it possible to optimize them using gradient descent. An NTM with a long short-term memory (LSTM) network controller can infer simple algorithms such as copying, sorting, and associative recall from examples alone.

The authors of the original NTM paper did not publish their source code. The first stable open-source implementation was published in 2018 at the 27th International Conference on Artificial Neural Networks, receiving a best-paper award. Other open source implementations of NTMs exist but are not sufficiently stable for production use. The developers either report that the gradients of their implementation sometimes become NaN during training for unknown reasons and cause training to fail; report slow convergence; or do not report the speed of learning of their implementation.

Differentiable neural computers *are an outgrowth of Neural Turing machines, with attention mechanisms that control where the memory is active, and improve performance.*

Graph Neural Networks

Deep Learning has been the key to solving many machine learning problems in fields of image processing, natural language processing, and even in the video games industry. All this generated data is represented in spaces with a finite number of dimensions i.e. 2D or 3D spaces.

Yet, in most current applications, generated data is generated from non-Euclidean domains that represent data as graphs with relationships and mutual dependency between objects. This has led to a growing interest in deep learning research that focuses on the structure of graph data.

Some models are…

- Spectral-based GNN layers
- Spatial-based GNN layers
- Pooling Schemes for Graph-level Representation Learning
- Graph Neural Networks Based Encoder-Decoder models

Modeling physics system, learning molecular fingerprints, predicting protein interface, and classifying diseases require a model to learn from graph inputs. In other domains such as learning from non-structural data like texts and images, reasoning on extracted structures, like the dependency tree of sentences and the scene graph of images, also needs graph reasoning models.

Graph neural networks (GNNs) are connectionist models that capture the dependence of graphs via message passing between the nodes of graphs. Unlike standard neural networks, graph neural networks retain a state that can represent information from its neighborhood with arbitrary depth. Although the primitive GNNs have been found difficult to train for a fixed point, recent advances in network architectures, optimization techniques, and parallel computation have enabled successful learning with them. In recent years, systems based on variants of graph neural networks such as graph convolutional network (GCN), graph attention network (GAT), gated graph neural network (GGNN) have demonstrated ground-breaking performance on many tasks mentioned above.

Probabilistic Logic Networks

A probabilistic logic network is a conceptual, mathematical and computational approach to uncertain inference; inspired by logic programming, but using probabilities in place of crisp (true/false) truth values, and fractional uncertainty in place of crisp known/unknown values. In order to carry out effective reasoning in real-world circumstances, artificial intelligence software must robustly handle uncertainty. However, previous approaches to uncertain inference do not have the breadth of scope required to provide an integrated treatment of the disparate forms of cognitively critical uncertainty as they manifest themselves within the various forms of pragmatic inference. Going beyond prior probabilistic approaches to uncertain inference, a probabilistic logic network is able to encompass within uncertain logic such ideas as induction, abduction, analogy, fuzziness and speculation, and reasoning about time and causality.

Probabilistic Graphical Models

Probabilistic graphical models are a powerful framework for representing complex domains using probability distributions, with numerous applications in machine learning, computer vision, natural language processing and computational biology. Graphical models bring together graph theory and probability theory, and provide a flexible framework for modeling large collections of random variables with complex interactions.

Quantum Neural Networks

Quantum neural networks are computational neural network models which are based on the principles of quantum mechanics. The theory of quantum mind, posits that quantum effects play a role in cognitive function. However, typical research in quantum neural networks involves combining classical artificial neural network models (which are widely used in machine learning for the important task of pattern

recognition) with the advantages of quantum information in order to develop more efficient algorithms. One important motivation for these investigations is the difficulty in training classical neural networks, especially in big data applications. The hope is that features of quantum computing such as quantum parallelism or the effects of interference and entanglement can be used as resources.

Quantum Machine Learning

Quantum machine learning is the integration of quantum algorithms within machine learning programs. The most common use of the term refers to machine learning algorithms for classical data executed on a quantum computer, i.e. quantum-enhanced machine learning. While machine learning algorithms are used to compute immense quantities of data, quantum machine learning utilizes qubits and quantum operations or specialized quantum systems to improve computational speed and data storage done by algorithms in a program. This includes hybrid methods that involve both classical and quantum processing, where computationally difficult subroutines are outsourced to a quantum device. These routines can be more complex in nature and executed faster on a quantum computer. Furthermore, quantum algorithms can be used to analyze quantum states instead of classical data.

The primary goal of Quantum Machine Learning is to achieve a Quadratic or Exponential Acceleration over Classical Machine Learning. Though there could be other secondary advantages in Machine Learning with Quantum Computing due to attributes of Quantum Mechanics.

Few Shot Learning

Few-shot learning refers to the practice of feeding a learning model with a very small amount of training data, contrary to the normal practice of using a large amount of data. It is based on the concept that reliable algorithms can be created to make predictions from minimalist datasets. Other related techniques are Zero Shot Learning and One Shot Learning.

Augmented Analytics

Augmented Analytics is an approach of data analytics that employs the use of machine learning and natural language processing to automate analysis processes normally done by a specialist or data scientist.

LET THE PATENT WARS BEGIN

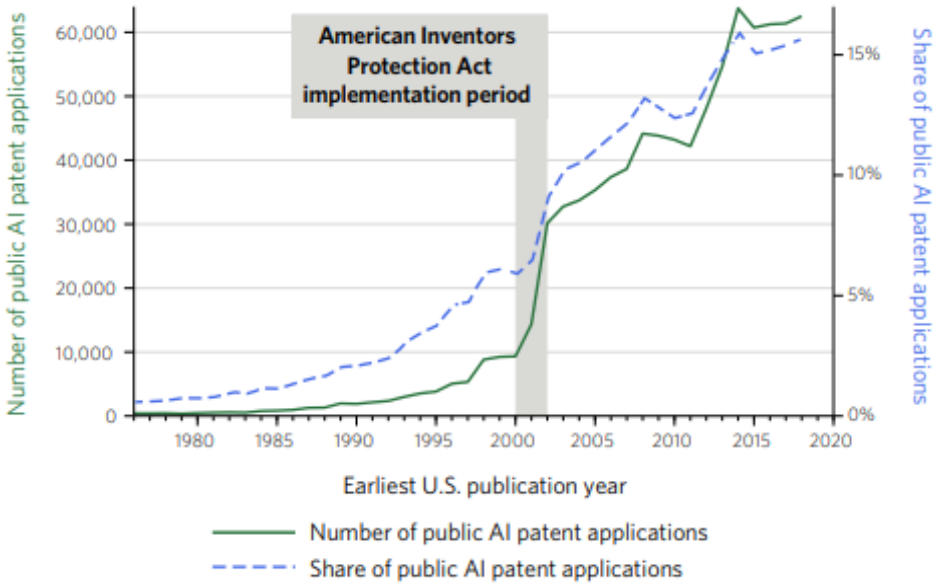


Figure 60 - The Volume & Share of Public AI Patents

As per USPTO…

- Artificial intelligence (AI) is increasingly important for invention, diffusing broadly across technologies, inventor-patentees, organizations, and geography.
- In the 16 years from 2002 to 2018, annual AI patent applications increased by more than 100%, rising from 30,000 to more than 60,000 annually. Over the same period, the share of all patent applications that contain AI grew from 9% to nearly 16%.
- Patents containing AI appeared in about 9% of all technology subclasses used by the USPTO in 1976 and spread to more than 42% by 2018.
- The percentage of inventor-patentees who are active in AI started at 1% in 1976 and increased to 25% by 2018. Growth in the percentage of organizations patenting in AI has been similar.
- Most of the top 30 AI companies are in the information and communications technology sector, with some notable exceptions such as Bank of America, Boeing, and General Electric.

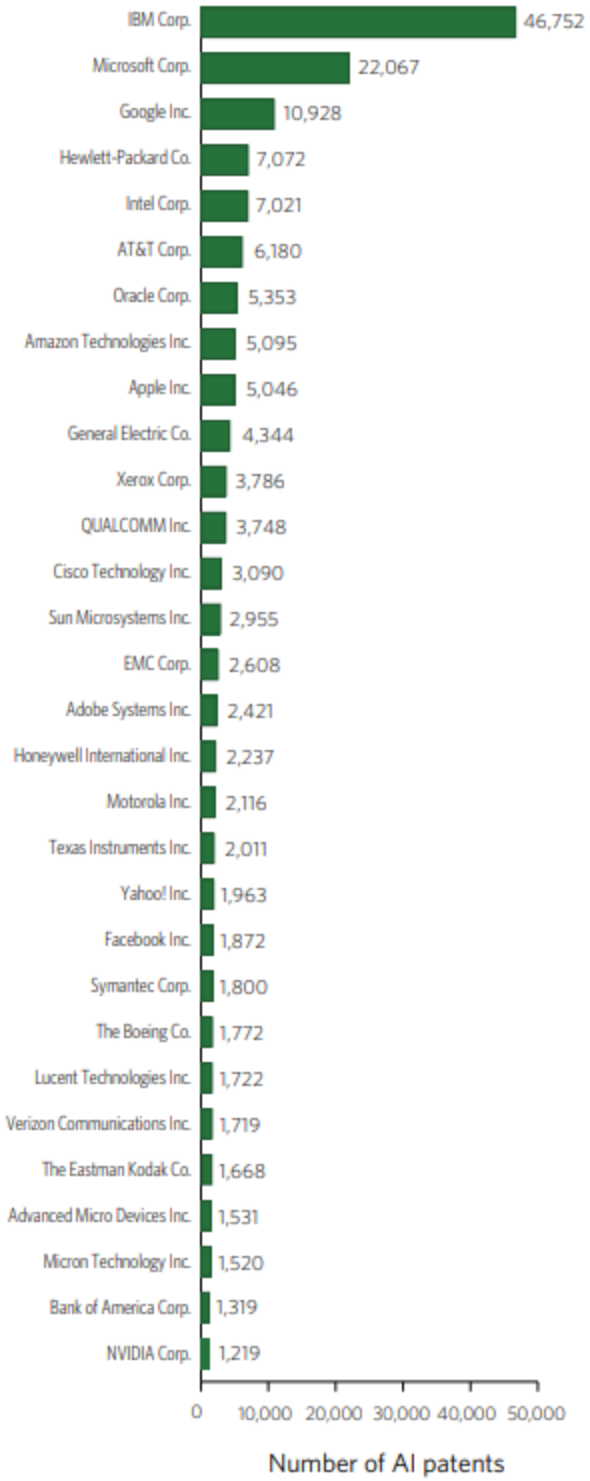


Figure 61 - Top 30 US A.I. Patent Owners at Grant

As per Center for Security & Emerging Technology (CSET)

1. Patent applications increased by 500 percent from 2009 to 2019 within the Chinese patent office—90 percent were domestic applications. The U.S. patent office has seen a 35 percent increase in applications during the same time, 48 percent of which were domestic.
2. Large companies—notably IBM, Microsoft, and Google—dominate AI patenting among U.S. organizations. Meanwhile, Chinese AI patenting is distributed much more broadly across companies (e.g., Ping An, Baidu, Tencent), government organizations (e.g. State Grid), and universities (e.g., Electronic Sci/Tech, Zhejiang, Xidian).
3. China focuses AI patenting on Computer Vision, Japan on Control Systems, and Korea on Speech Processing. The United States is more evenly distributed across research fields.

Lets draw an analogy to understand whats happening here. We all know what is Bureaucracy. While some people spend their lifetimes to solving problems, inventing things and building solutions. There are a whole lot of people in this world who spend their time excelling at creating or fighting Bureaucratic battles and red tape. Similarly some people invent and solve problems while others fight over patents and territorial rights.

The concept of Patents is quite controversial. Contrary to the original purpose behind it, that of protecting inventions in return for a full and unconditional disclosure. Does it really reward inventors? That's questionable because to file a patent you really don't need to invent anything. Nobody verifies anything physically. All you need is to describe an idea of something in detail and lay claims to its attributes, designs for a purpose by arguing the novelty you bring via the idea.

But if you don't need to invent anything to file patents and all you need is a description. Then why does everyone spend time and money on doing it? Simple, so that they can stop others from doing it commercially. That's called a block patent. And is the game of Patent Trolls.

And just because you successfully file a patent means literally nothing. You need to figure out who is violating it and how. And spend years and millions of dollars on patent litigation.

So which way you look at it. Patents might not really be promoting innovation. But rather be a technique for territorial warfare.

At Automatski we normally don't file patents. Primarily because we will never disclose our millennium inventions. And secondly, we demonstrate our inventions publicly and set it up as a prior art. So nobody else can patent it, ever. And everyone in the world is free to invent it or make in on their own and use it, or sell it. We believe that's what is meant by democratizing innovation. And that should be the way forward. But it's a free world. To each his own.

OPEN QUESTIONS

AutoML

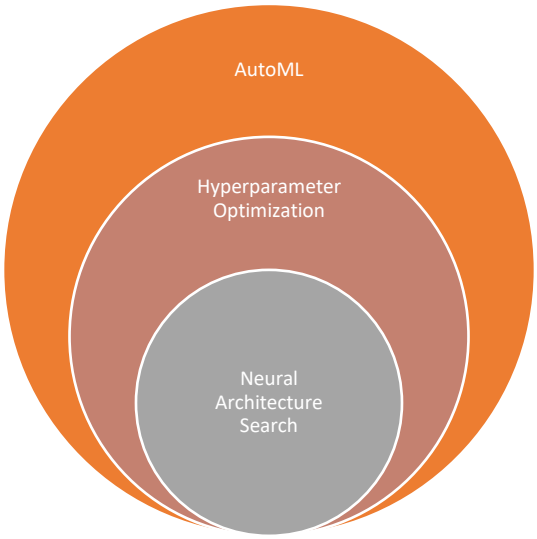


Figure 62- AutoML – Automatic Machine Learning

The first step to being smart about Machine Learning and Deep Learning Engineering is to start with Neural Architecture Search. Neural Architecture Search methods are search methods that seek to learn architectures for machine learning tasks, including the underlying build blocks. If we have a search space of all combinations of different parts of our models like layers, depth etc. And if we can change, add, remove these parts. The problem that NAS addresses is that of finding the best architecture based on these blocks which delivers the best performance for a given problem or situation.

Another step above that in smartness would be Hyperparameter Optimization. Hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters in the model are learned.

And the ultimate goal is to create Machine Learning and Deep Learning Models Automaticaly aka AutoML. AutoML, is the process of automating the time consuming, iterative tasks of machine learning model development and applying machine learning to real-world problems. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model in the end. The field of A.I. is so complicated that no one person understand everything, it is also quite difficult for small teams. AutoML under the hoods uses Machine Learning and other scientific algorithms to create and deploy models for us. Which we describe to the AutoML system in the format it requires. AutoML is basically the application of A.I. to create A.I. (cliché)

Explainability

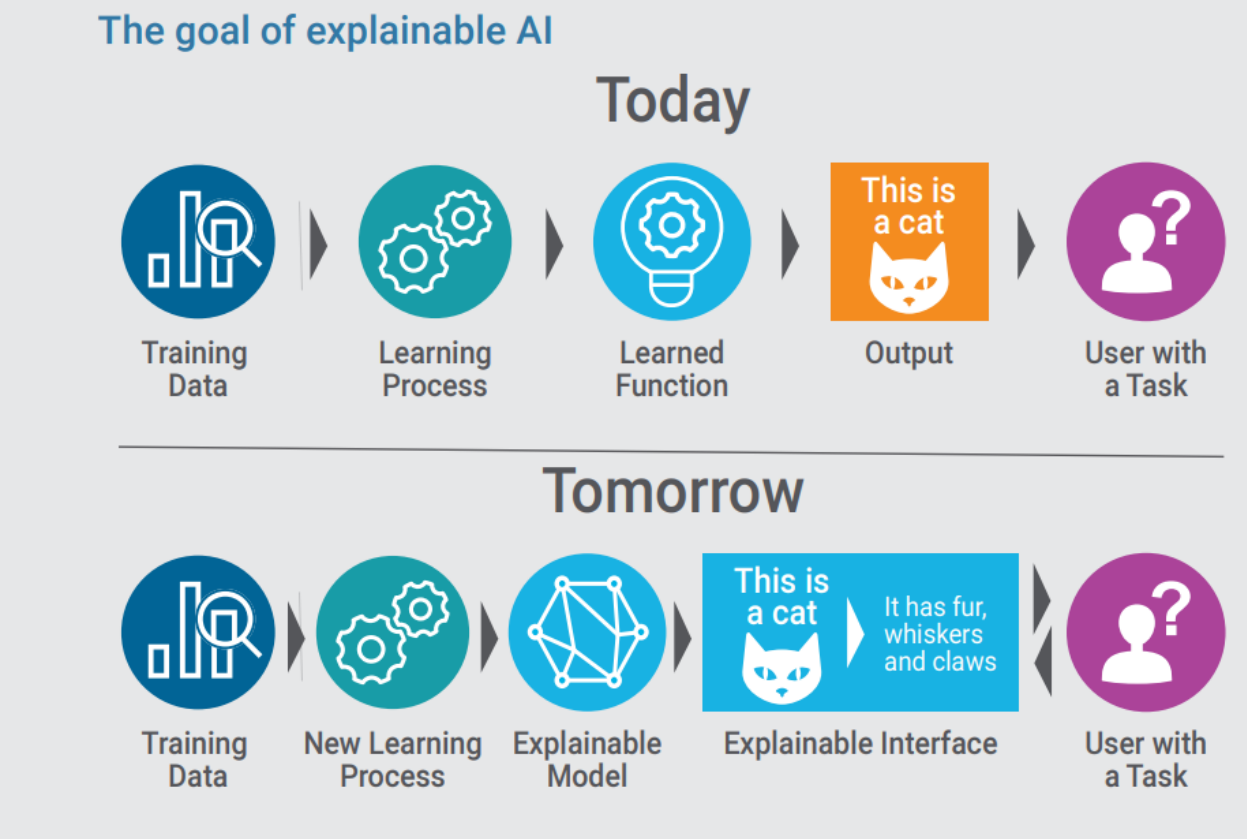


Figure 63- Opening Up Black Boxed w/ Explainable A.I. (Ref: Datanami)

We trust someone more If he can explain (in summary) how he arrived at some conclusion. The thing with Deep Learning and Machine Learning is that there is a mystery around how they work. Nobody really understand how they work. But they work extremely well at a lot of problems. They simply throw some answer back at us. And nobody knows if its correct or not. There have been many attempts to create Explainable A.I. Which can explain the result it gives us. But so far there have been no successes noteworthy or of worth talking about.

Regulation such as GDPR and the need for ML trustability, transparency and fairness have sparked a discussion around further needs of interpretability.

Explainable A.I. techniques either directly analyze model components, study sensitivity to input perturbations, or analyze local or global surrogate approximations of the ML model.

Such methods can be used to discover knowledge, to debug or justify the model and its predictions, and to control and improve the model.

Many model-agnostic explanation methods have been introduced in the last few years, which work for different types of ML models. But also model-specific explanation methods have been developed, for example, to interpret deep neural networks or tree ensembles.

Examples of aspects of explainability are sparsity, interaction strength, fidelity (how well an explanation approximates the ML model), sensitivity to perturbations, and a user's ability to run a model on a given input (simulatability).

Methods that study the sensitivity of an ML model are mostly model-agnostic and work by manipulating input data and analyzing the respective model predictions. These methods often treat the ML model as a closed system that receives feature values as an input and produces a prediction as output. We distinguish between local and global explanations.

1. Local methods explain individual predictions of ML models.
2. Global model-agnostic explanation methods are used to explain the expected model behavior, i.e., how the model behaves on average for a given dataset. A useful distinction of global explanations are feature importance and feature effect.

Surrogate models are interpretable models designed to "copy" the behavior of the ML model. The surrogate approach treats the ML model as a black-box and only requires the input and output data of the ML model (similar to sensitivity analysis) to train a surrogate ML model. However, the interpretation is based on analyzing components of the interpretable surrogate model.

Ideally, a model should reflect the true causal structure of its underlying phenomena, to enable causal interpretations. Arguably, causal interpretation is usually the goal of modeling if ML is used in science. But most statistical learning procedures reflect mere correlation structures between features and analyze the surface of the data generation process instead of its true inherent structure.

Such causal structures would also make models more robust against adversarial attacks, and more useful when used as a basis for decision making.

Feature dependence introduces problems with attribution and extrapolation. Attribution of importance and effects of features becomes difficult when features are, for example, correlated and therefore share information.

Also we do not know what the groundtruth explanation looks like and have no straightforward way to quantify how interpretable a model is or how correct an explanation is.

Even after throwing a lot of money and expertise at it we simply don't have anything that works, primarily because there are 100 million or 1 billion double precision variables affecting the results and each other. How do you even begin to explain them and what they are doing? This is NOT the same as explaining a rule generating model which is self explanatory to a large extent based on the rules it generates.

Debugging

And since we don't really know whats happening under the hoods in Machine Learning of Deep Learning. We really don't know how to fix things when things go wrong. The entire process of engineering such solutions is mostly based on experience, peer collaboration, a lot of hit and trial and lot of perseverence.

Assurance & Reliability

Once you have an answer from a Machine Learning or Deep Learning system. Unlike statistics which gives an assurance about the reliability of the answer. There is nothing analogous here. We have to take it at face value. Use it and see if it works. While Machine Learning and Deep Learning systems might give some

assurance based on testing on known datasets. When put in production its out in the wild. We can hope and pray nothing goes wrong. And so far not too many people have died and not too many catastrophes either. If that's any consolation.

Fairness and Bias

Machine Learning and Deep Learning systems are deployed in industrial environments, but also a lot in consumer, employee or citizen facing scenarios. Such systems learn a model or a representation from the data they have been trained on. And with any such data comes a lot of gottchas. Till date in scientific studies 100's of such models have shown biases and unfairness (we are not talking about outright errors) across various segments of customers, citizens and employees. While proponents claim that this is a sort of bias which can be fixed. Critics believe such issues are inherent to Machine Learning and Deep Learning by virtue of their definition i.e. function approximation or mappings from inputs to outputs. Which is the way these are created. And they cannot be fixed ever.

Robustness & Failures

And then there are outright failures and possiblity of sabotage using something called Adversarial Attacks. It is a surprising and publiclly unknown fact that 'All' Machine Learning and Deep Learning systems can be sabotaged e.g. you can walk invisible to a Vision System, in and out of a secure perimeter e.g. an airport or a factory. Anyone with the right techniques can game such systems. Such techniques have been demonstrated even on Reinforcement Learning Systems.

Generalization

These systems are trained on specific data. And the hope is that they would 'generalize' or work well in unknown scenarios. That's a tall order. If the unknown scenarious are relatively similar to the training data such systems manage to get by. But in other cases they completely tank.

Some vendors claim to have solved 50 year old problem of Protein Folding by training models on existing Protein Structures (of which we have a database of about 150,000 structures). The fallacy of such claims is that what we really need to solve gazillions of Protein Folding problems of which we have an antire universe out there. Of which we know nothing about their structures. What we really need are ab initio techniques which can start from scratch and predict a protein structure with cent percent accuracy.

Deep Learning models don't generalize very well or much.

Ethics

What is right? What is wrong? In scenarios which are not defined by the law of the land. That's ethics more or less. Ethics precedes law most of the times in history.

A.I. is basically unchartered territory. And the typical instantaneous response from actors is 'damage control'. Every actor is autonomous. Law doesn't even understand how to deal with all such scenarios. There is no 'generally accepted' concepts of right or wrong in A.I. Besides a few laws from Asimov. Which is better as a subject for movies rather than a set of guiding principles for A.I. systems which we have today. The other problem is that even if we all agree on such common ethical principles or guiding principles how do we even engineer them into the functionality of our A.I. systems? Under the hoods they are basically a hundred million or a billion software implementations of neurons firing.

Governance and Regulation

Assuming we all agree on whats right or wrong, whats ethical and unethical, whats legal and illegal in A.I. How do we govern and regulate millions of autonomous actors i.e. individuals, organizations or even rogue actors?

Do we need to simply extend what governance and regulations we have inside organization, states or across the world? or do we have to create them afresh aka a new world order??? A basic question to ask is if our law enforcement or management understands A.I. or do we even expect them to do so? Or do we need a new structure and scheme to govern and regulate A.I. across the world?

Audit

Given the authorization or a directive from an authority to audit an A.I. system. The problem we face as we mentioned earlier is that nobody understands how these systems work under the hood. There is no explanation. No assurance or reliability estimates. And we cannot debug such systems.

So what is anyone supposed to Audit in such A.I. systems? Someone would say if we cannot audit the system itself then we should audit the process that created the said system. But then that's just regular software development processes or even monkey coding in some scenarios. Nothing extraordinary there. So what are we supposed to Audit? And how?

Benchmark

Outside the few academic studies there are rarely any benchmarks of equivalent A.I. systems across vendors. (MLCommons is the governing body for the independent benchmarking organisation, MLPerf and is one such effort)

What we have is largely vendor speak and marketing jargon. Which hides the true objective understanding about the functionality and capabilities of such systems. A huge part of the reason is that vendors are mostly not ready to participate in such benchmarking studies. And their internal designs are a propreitary secret so no one can make reverse and cross cutting inferences in benchmarking studies. Eliminating the utility of such benchmarking studies. For example if the only thing benchmarking studies can show is a single table with single numerical values of accuracy on a test scenario. Then we won't have much use for such studies.

Do we need benchmarks? Yes! I think everyone will unanimously say we do.

Standards

Till the end of 2020 there are absolutely no standards in these fields. Even after decades of research and development. Things are in a continuous state of flux. New things or changes to models, techniques announced every week. A necessary criteria for standardisation is for things to be mature and stable enough so that we can cover the present and project into the future. And at the same time being comprehensive enough for that standard to be of utility.

When are we going to have any standards? Nobody knows. And that's not necessarily a priority for almost anyone right now. One can say we won't have any standards in the near future. We also need to ask if standards are even possible for something as complex as A.I.? and especially in the case of globally distributed autonomous actors? We might just need to create a new world order to achieve all this.

Causal Inference & Reasoning

Causal inference *is the process of drawing a conclusion about a causal connection based on the conditions of the occurrence of an effect. It analyzes the response of the effect variable when the cause is changed. Causal inference is an example of causal reasoning.*

When things go wrong, we try to perform what is called Causal Inference. Why did the system do this? Why did it say the person robbing the bank was John Doe? Why did the autonomous car turn a sharp left when the road was turning to the right? Why did it suddenly slam the brakes for apparently no reason?

As we have mentioned earlier. We cannot really debug Machine Learning, Deep Learning or Reinforcement Learning, or similar systems. And it is nearly impossible to make any specific causal inferences either.

Causal reasoning *is the process of identifying causality: the relationship between a cause and its effect.*

Causal Reasoning is missing from our Machine Learning and Deep Learning Models. If we can solve that problem then we would be able to create Explainable AI, implement Logical Reasoning and probably Common Sense Reasoning, Reliability, Assurance, work in Noisy Situations, and also probably fix Biases and Errors.

"The ultimate goal is to learn a Causal Model of the World"

It is the magic bullet which will sort out the majority of issues in A.I.

Commonsense Reasoning

Commonsense reasoning is one of the branches of artificial intelligence (AI) that is concerned with simulating the human ability to make presumptions about the type and essence of ordinary situations they encounter every day. These assumptions include judgments about the physical properties, purpose, intentions and behavior of people and objects, as well as possible outcomes of their actions and interactions. A device that exhibits commonsense reasoning will be capable of predicting results and drawing conclusions that are similar to humans' folk psychology (humans' innate ability to reason about people's behavior and intentions) and naive physics (humans' natural understanding of the physical world).

Endowing computers with common sense is one of the major long-term goals of Artificial Intelligence research. Commonsense knowledge and reasoning are relevant for many applications of current interest.

Examples include robot and human collaboration, transparent machine-learning systems that can explain their conclusions, social media and story understanding software, and dialogue systems.

Semantics-based representations for specific commonsense domains…

1. Time, change, action, causality
2. Commonsense physical and spatial reasoning
3. Legal, biological, medical, and other scientific reasoning incorporating elements of common sense
4. Mental states such as beliefs, intentions, and emotions
5. Social activities and relationships

Inference methods for commonsense reasoning…

1. Logic programming
2. Probabilistic, heuristic, and approximate reasoning
3. Nonmonotonic reasoning, belief revision and argumentation
4. Abductive and inductive reasoning
5. Textual Entailment

Sidenote: The Automatski Recommendation

One of the biggest issues with Common Sense Reasoning is that researchers treat it as some magical ingredient found in humans. Till now there haven't been too many even modest successes with Common Sense Reasoning in the field. The best way to understand and design for Common Sense Reasoning is to call it – Generally Prevalent Implicit Data & Logic in A.I.

Automatski's Non-Realtime A.G.I. uses the very same construct. And Automatski says that once we define it that way, we understand that the agent can build common sense reasoning capabilities on top of its existing logical reasoning and causality learning abilities by statistically learning from experience and extracting 'common sense' aka Generally Prevalent Implicit Data & Logic. Which it can then exploit in learning as well as reasoning across use cases and scenarios.

Automatski also suggests that instead of building common sense domain or knowledgebases into agents we should build this statistical learning capability into agents that it uses from scratch.

Law

The entire world is debating gently about the legal issues which will come along with deployment and usage of such systems. Who is responsible? To what extent? The Vendor? The Operator? The Manufacturer? The User?

Can someone simply bypass all responsiblility by making the other party sign a simple User Agreement? Who is accountable for all the errors, mishaps, accidents, damages, deaths or failures?

Surely we can't hold the machine responsible. So what is the formula, theory or scheme we need to create new laws. And based on which universally acceptable (more or less) fundamentals.

What if the governments or armies themselves are responsible for inflicting ill on others. Who will hold them accountable? And who are those guys accountable to? What about jurisdiction issues?

Privacy

I think it is clearly understood that any kind of Machine Learning, Deep Learning or A.I. system feeds on data. We need huge amounts of data to train them. And they execute on huge amounts of data to deliver their benefits for us. (Data is Voice, Vision, Images, Text, Emails, Sms, Logs …) And such systems need to be monetized, hence the most common way is to use all the data to send targeted advertising and marketing your way. And it seems logical for vendors to share this data with their advertising customers.

So A.I. Vendors integrate their A.I. systems with huge sources of data, anything they can find. More data means more accuracy for their A.I. Systems. Such systems are always on, always listening, always capturing data, always monitoring.

But the problem is that all of this massively invades our privacy. If we challenge the Vendors to fix this and come up with a solution they say “There is no human monitoring anything” as the only consolation they offer us.

Considering these A.I. systems are the lifeblood of these vendors and all major internet companies. And all of this means gazillions of dollars to them. If you take these A.I. systems away from them. They will have nothing left. It’s a tricky problem to get solved anytime soon.

> Sidenote: EU/GDPR
>
> The General Data Protection Regulation (EU) 2016/679 (GDPR) is a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA). It also addresses the transfer of personal data outside the EU and EEA areas. The GDPR's primary aim is to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU.

TOP GLOBAL A.I. COMPANIES

It is very difficult to shortlist companies in the A.I. space. Because 99% of the firms are wannabe’s or me too’s. We made our best effort to include genuine front runners. We hope we have not missed anyone of much consequence.

Amazon

Internally Amazon Ecommerce is rumoured to be using more than 50 A.I. models. Ecommerce is a wafer thin margin business and A.I. is Amazon’s secret sauce to surviving and flourishing so far in online Ecommerce.

Amazon is also one of the leaders in Cloud Computing. And offers various ready to consume A.I. cloud services, tools and frameworks. Recently Amazon has released propreitary ASIC’s for training and inference through its cloud.

Automatski

Why is Automatski mentioned here?

Because Automatski reinvented Machine Learning. Machine Learning at its core was about just 3 things – Regression, Classification and Clustering

Automatski created the worlds first Polynomial Time NP Algorithm for Clustering. It created the worlds first NP Algorithm for Global Optimization/Regression. And it also developed the worlds first Universal Model Classification Algorithms.

Also, Automatski literally (re) invented Deep Learning. Years ago it fixed most of the problems that The Fathers of Deep Learning were saying were blocking progress and we would need to start all over again from scratch. Some of which are…

1. Incremental Deep Learning
2. Completely Eliminating Backtracking
3. Exponentially Increasing The Efficiency of Deep Learning
4. Drastic Reduction in Energy Consumption
5. Complexity Reduction
6. Instead of End To End Training, we use models which are Hyper Local and yet Near Optimal in Aggregate, just like a brain
7. Combine Deep Learning with Production Grade Quantum Computers.

Automatski can train Spiking Neural Networks ab initio. And its Deep Learning & Reinforcement Learning Algorithms are Natively Distributed.

Its work in Reinforcement Learning involves Sample Efficient RL, Imitation Learning, Learning From Examples, Apprenticeship Learning, Inverse RL etc.

Besides Logic And Causality, Common Sense Reasoning, Creativity, Novelty Search, AutoML, AutoRL, Quantum A.I/ML/Neural Nets, Spiking Neural Networks, ML Data & Compute Privacy, Collaborative ML and Simulators.

Lastly, Automatski also has the worlds first Non-Realtime (Polynomial Time) Implementation of A.G.I. in its labs.

DeepMind

DeepMind has bet bigtime on A.I. Its primary focus is on Reinforcement Learning techniques. And has been responsible for beating World Champions at the game of GO, Chess. Mastering Atari Games etc.

DeepMind is rumoured to be now focussing on solving scientific problems with Reinforcement Learning. Which is tricky because Deep Learning and Reinforcement Learning are both function approximation methods and when it comes to scientific problems where the search space for one best most perfect solution is in gazillions. Deep Learning and Reinforcement Learning fall short. But this is an exciting space to watch. And everyone is waiting and watching with great interest.

Google

Google has bet big time on A.I. for its future. Though some of its claims have met with criticism e.g. its claim of reducing data center energy consumption by 30%, its claim of having solved Protein Folding etc.

Google has invested a lot of time and money into developing its Open Source Deep Learning Framework called Tensorflow. And also to develop propreitary A.I. ASIC's called TPU (Tensoir Processing Units) which it offers through its cloud.

IBM

IBM made headlines with its NLP Q&A Systems which beat the Jeopardy World Champion. It later created a Debating system. And then spent a lot of time and effort on applying its NLP systems to Medical Diagnostic Applications. Which met with lot of criticism from the Medical Fraternity. And then IBM went very silent except for its Cloud A.I. offerings. Today it is mostly in the news for its Quantum Computing and Supercomputing efforts on scientific endeavours.

Microsoft

The Deep Learning Group at Microsoft is focussed on applications to natural language processing, computer vision, multi-modal intelligence, and for making progress on conversational AI.

The Reinforcement Learning Group at Microsoft is focussed on Researching and Building 'Real World' Reinforcement Learning Algorithms and Applications.

Microsoft has the second highest number of patents on A.I. according to filings at USPTO.

It also offers consumable A.I. services on its Azure Cloud. Besides quite a few open source tools and frameworks for deep learning.

OpenAI

OpenAI started out with the aim of creating responsible A.I. as a Non-Profit. But its founder Elon Musk parted ways due to disagreements with the researchers. OpenAI transformed into a for profit and was in the news for raising billions of dollars from Microsoft in funding and contracts.

OpenAI is best known for its GPT-n series of models especially GPT-3 which is a text generator. They claimed it was so dangerous that it should be locked away. And yet to critics amusement they packaged it as a commercial cloud service and released it to the world for consumption. Today there are thousands of early adopters which have used GPT-3 to generate computer programs, fake news, speeches, articles etc.

Tesla

Tesla's inital Autonomous Car efforts was based on technology supplied by Mobileye, an Israeli startup since acquired by Intel. Tesla and Mobileye have since then parted ways. Tesla uses cameras instead of lidars for its AutoPilot. It has remained in controversy for its Fully Autonomous Driving Capability Claims since many years which has not materialized till now. Elon Musk the founder of Tesla is the second richest man in the world today. And Tesla stock has increased more than 700% this year. So it has a lot of believers.

Uber

At its core Uber is a Taxi Service, and maybe a Delivery Service and a few other things. But surprisingly it spends billions of dollars on research every year. Outside A.I. Ubers engineering tools and platforms are used across the world. Uber is a strong believer in Open Sourcing, and it open sources almost everything it develops. Maybe except for its most core systems.

In Ubers own words - Uber AI powers applications in computer vision, natural language processing, deep learning, advanced optimization methods, and intelligent location and sensor processing across the company, as well as advancing fundamental research and engaging with the broader AI community through publications and open-source projects.

Vicarious

Vicarious is developing machine learning software based on the computational principles of the human brain. One such software is a vision system known as the Recursive Cortical Network (RCN), it is a generative graphical visual perception system that interprets the contents of photographs and videos in a manner similar to humans. The system is powered by a balanced approach that takes sensory data, mathematics, and biological plausibility into consideration. On October 22, 2013, beating CAPTCHA, Vicarious announced its model was reliably able to solve modern CAPTCHAs, with character recognition rates of 90% or better when trained on one style.

The Recursive Cortical Network uses a Markov Random Field to segment images (similar to the visual cortex) and process it through a hierarchy consisting of lateral connections. Thanks to the architecture, the model has extreme data efficiency (almost 900,000x more data efficient than traditional deep neural networks). The RCN is part of a model that incorporates a **causal model and concept induction system** recently created by Vicarious.

Vicarious hopes to use it's A.I. models to revolutionize Robotics and make it common place.

*** We have left out companies and startups which are using Deep Learning or Machine Learning for solving specific niche applications or problems. We have tried restricting our list only to companies which are deeply vested in the progress of A.I. as a technology. Which may or may not include applications or engineering.

KEY DRIVERS, USE CASES & ISSUES

The Key Drivers for A.I. adoption across enterprises are…

1. The Early Adopters want a Competitive Advantage
2. The Early Majority wants to catchup with their Competitors
3. To Solve Unsolvable Problems
4. Create New Products & Services
5. Cost Reduction
6. Better ROI

The Business Use Cases are…

1. Impossible Problems

2. Previously Uncomputable or Intractable Problems (Using Known Methods)
3. Problems Unsolvable in any known way (Method Not Known)
4. Problems that are too Complex
5. Problems that require too many Resources
6. Problems that took too long to solve
7. Problems that were Economically Unviable to solve earlier

*** All these Use Cases had been abandoned earlier, sometimes after long and expensive efforts.

The consololation was that it was ok because no one else could do them either. There is a long backlog of Use Cases, someone has to just (rediscover and list the problems and) match the pending problems with new techniques. And get started. It is critical to the business to solve these problems which otherwise remain unsolved.

Issues

Almost everyone acknowledges that Artificial Intelligence will disrupt entire industries.

But the immediate problem almost everyone is facing is the huge skill shortages. And what about existing teams? Can't hire and build an entire new organization for both existing and new products and services. We have to figure out ways to enhance, train and use a majority of the existing teams.

The biggest confusion is – What is the best way for our organizations to 'consume' the innovations in A.I.? Can we create a couple of new 'Processes'? Are there any gottcha's we should know of? Any tips and tricks of the trade? Any checklists or best practices?

WHO TO HIRE?

Basically people who can – "Create Algorithms & Do Tensor Computations On Distributed Computational Graphs"

	Job Description	Time Horizon	Qualifications
Researchers	Discover and/or invent new methods and algorithms.	~10 years	M.S. / Ph.D. / Post Doctoral Fellows
Engineers	Apply those methods to solve many types of problems.	~3 Years	M.S. / Ph.D. / Post Doctoral Fellows
Application Developers	Develop the applications and wiring around the core solutions.	~1 Year	B.S.

MARKET FORECAST

The global artificial intelligence (AI) market size was valued at USD 27.23 billion in 2019 and is projected to reach USD 266.92 billion by 2027, exhibiting a CAGR of 33.2% during the forecast period.

We have not done our own market forecasting studies and due to copyright reasons we cannot elaborate further than this. This is from the Public Information issued online by Analysts earlier.

LOW HANGING FRUIT(S)

For adopters of A.I. aka Deep Learning probably the best use case to start with are in Image Recognition, Vision based solutions. Things like Face Recognition, Object Recognition, Activity Recognition, Scene Analysis, Image/Face/Art Generation, Copying Motion etc.

These can be developed using Convolution Neural Networks and GAN's. One can use NVIDIA RTX-3090 GPU based Workstations for Development to start with. And if you are doing something very complex, you can probably get one of NVIDIA's DGX A100 systems. Or its best to get Graphcore's systems.

One can go with the standard Pytorch (simpler) or Tensorflow based platforms. But for complex Use Cases it is highly recommended to look at Solutions from Automatski which fix known issues with All Deep Learning Implementations in the world today. Some of which are mentioned by The Fathers Of Deep Learning but are necessary to create reliable and efficient production grade solutions.

The other Use Cases that are easy to start with and which will deliver reasonably good results are…

- Industrial Diagnostics & Preventive Maintenance
- Fraud Detection
- Network Intrusion Detection & Security
- Personalization
- Customer Analytics
- Marketing
- MultiModal Search
- Translation & Transcription
- Generative Models (Generating Text, Art, Audio, Speech, Images, Video, Games, UI, Apps,…)

THE FUNDAMENTAL DILEMMA WITH CURRENT STATE OF A.I.

aka Question 0 (Zero)

"Should we deploy this A.I. (???) system into production?"

Corollary: What all can go wrong?, If you leave aside hyperbole and jargons does this even work?

THE FUTURE OF MANKIND

Is…

A.I. & Quantum Computing & The Combination Of The Two

These two things and their combination will define the next 100-1000 years of mankind. They will overshadow almost any and all other technology achievements and innovations in all probability.

Figure 64- The Future Is Coming Faster Than You Think

Both Quantum Computing & A.I. will help the human race become an inter-galactic race.

But there is a difference…

1. Quantum Computing will have a $200-$1000 trillion impact on mankind
2. Deep Learning will only have a sub $1 Trillion impact on mankind at best
3. 'Realtime' A.I. if invented will have a $200-$1000 trillion impact on mankind

*** (3) Because we will literally invent a new species

*** We assume Robotics is subsumed by A.I.

Currently there is almost $2 billion in venture capital chasing Quantum Computing every year. And about $20 billion chasing Deep Learning every year. The A.I. spring and its hype is slowing down. While Quantum

Computing is picking up. Even though outside Automatski, Production Grade Quantum Computers are probably decades or even centuries away. Both investors and customers place more value in Quantum Computing as we know a lot about its power and applications, and have more than 120 years of research behind it. Deep Learning only has 30+ years behind it and has a lot of uncertainty in terms of its applications and success. And there aren't many takers for the possibility of A.I. being invented anytime soon. (Note: A.I. is artificial intelligence, it has not been invented yet. It is different from Deep Learning)

HOW TO CITE ?

The BibTex Format Reference is

```
@report{automatski_millennium_firm_2020_89257478,
        author      = {The Automatski Team},
        title       = {The Executives Guide To Artificial Intelligence v2020},
        institution = {Automatski},
        publisher   = {Automatski},
        month       = Dec,
        year        = 2020,
        url         = {https://bit.ly/3r7exeY}
}
```

About Automatski

Automatski is a millennium firm which deals almost exclusively with solving problems which cannot be solved in a millennia (1000 years) given the current state of human technology and capability. Automatski has over a 100+ millennium inventions including but not limited to the worlds first Billion Qubit Infinite Precision Quantum Computer developed in 2014 CE.

DISCLAIMER

The information contained herein has been obtained from sources believed to be reliable. Automatski Solutions Private Limited disclaims all warranties as to the accuracy, completeness, or adequacy of such information. Automatski Solutions Private Limited shall have no liability for errors, omissions, or inadequacies in the information contained herein or for the interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice.

Automatski conducts business primarily in US and EU/UK. It doesn't have a business presence in India. And India is not its target market.

Investors

List of Automatski ventures http://automatski.com/investors.html

Email: investment@automatski.com

Business & Partnerships

Email: info@automatski.com

Technology & Research

Email: teagan@automatski.com

COVER IMAGE: ENVATO.COM