Beyond programming:
The quest for machine intelligence

Inaugural lecture, Universiteit Leiden, 2017/10/27

Holger H. Hoos

Mijnheer de Rector Magnificus, beste collega's,
dear colleagues and friends,
liebe Freunde und Familienmitglieder,

as we sit in this magnificent hall, I invite you to lean back and look up. Send your gaze towards the ceiling, then through and beyond, past the rooms and attics above, through the roof of this venerable building, through layers of clouds, straight to the stars. Now, as you contemplate the constellations, let your mind drift back in time.

The sky as we see it now is nearly the same as that observed in October 1620 by one Christopher Jones, master of a ship carrying 102 passengers across the Atlantic ocean [28]. This ship was the Mayflower, and about half of her passengers were members of a Puritan congregation from this very city of Leiden, about to settle on the New England shore and to play a pivotal role in the early history of the United States of America [4].

The stars by which the officers of the Mayflower navigated, as many ships before and after, have fascinated humankind from its early days. Astronomical observations were undoubtedly among the first scientific endeavours. There is, in fact, much evidence that the regularity of celestial mechanics and the desire to predict astronomical events, such as lunar eclipses, inspired some of the earliest instances of computational thinking, as well as computational devices, such as the astrolabe [26], the Antikythera mechanism [11] (both 200 BCE) and al-Jazari's castle clock (1206 CE) [1].

1 The age of computation

The notion of computation underlying these early examples evolved and solidified through the centuries. It connects 13th-century Majorcan writer and philosopher Raymundus Lul-

lius, who designed and realised a device for calculating answers to philosophical questions using logical combinatorics, to Gottfried Wilhelm Leibniz, who, in the 17th century, not only worked on the binary number system underlying modern digital computers, but also devised several highly sophisticated and versatile mechanical calculators.[1] Still, it would take until the middle of the 19th century for the idea of what we now know as a universal computer to emerge – a device that can be freely programmed to perform arbitrary sequences of logical and arithmetic operations.

It was Charles Babbage, an eminent English scientist, philosopher and engineer, who originated the concept of a general-purpose, digital computer – a complex and ingenious mechanical device he called the analytical engine [24]. In Babbage's time, the term 'computer' was in common use, but it referred to a person performing mathematical calculations manually (and had been used in this sense since the early 17th century, when Leibniz designed his mechanical calculators). Inspired by the mechanisation of industrial processes that disrupted society at the time, Babbage wanted to automate the work of these human computers, in order to perform complex calculations faster and more accurately.

The concept of computation pursued by Babbage and Leibniz before him, foreshadowed by Lullius and the originators of the Antikythera mechanism, was that of *precise instructions, flawlessly executed*. This, along with the specifications of inputs (*i.e.*, data given at the beginning of the sequence of instructions) and outputs (*i.e.*, data produced by the computation), and with the requirement that the sequence of instructions eventually terminates, characterises the key concept of an algorithm, which lies at the core of computation (see, *e.g.*, [21]).

The word algorithm, derived from the name of 9th-century Persian mathematician, astronomer and geographer al-Khwarizmi, can be found in English literature dating back to the 14th century.[2] Intriguingly, there is no single, universally accepted mathematical definition of an algorithm; instead, it has been shown that a large range of different definitions are equivalent, in that any algorithm formalised according to one can be faithfully translated into all others.[3]

Algorithms are neither restricted to processing numbers, nor to being executed by a machine. The human computers whose work was instrumental to a broad range of military and scientific applications, as well as to early human space flight in the 1960s, executed algorithms manually. At the same time, the series of punched cards that determined weav-

[1] In fact, Leibniz also believed that, to a large extent, human reasoning could be reduced to computation and worked towards a formalism, the *calculus ratiocinator*, to describe such computations. In Leibniz's words: *Quando orientur controversiae, non magis disputatione opus erit inter duos philosophus, quam inter duos computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus.* – If controversies were to arise, there would be no more need of disputation between two philosophers than between two calculators. For it would suffice for them to take their pencils in their hands and to sit down at the abacus, and say to each other (and if they so wish also to a friend called to help): Let us calculate.

[2] Geoffrey Chaucer uses the early form *augrym* in his Canterbury Tales (The Miller's Tale, Line 3210).

[3] In theoretical computing science, this has lead to an important hypothesis known as the Church-Turing thesis [20].

ing patterns for the automated loom invented by Joseph Marie Jacquard in 1804 are algorithms, as are the precise sequences of instructions for synthesising chemical compounds and, in a slightly relaxed sense, musical scores and cooking recipes.

When Charles Babbage elaborated the plans for the analytical engine, his general-purpose mechanical computer, his mind was firmly set on using it to perform complex numerical calculations. It fell to his collaborator and friend, Ada, Countess of Lovelace, to realise that the machine and the algorithms running on it could work on data other than numbers – a vision that foreshadowed developments that would only start in earnest 100 years later, in the middle of the 20th century [24, Note A].[4]

Babbage never completed his analytical engine, mostly because he could not secure the requisite funding. However, it is now generally believed that the machine could have been built using the technology and materials available at the time, and that it would have worked as intended. Many of the principles underlying the design and use of the analytical engine were to provide a solid foundation for the development of the first general-purpose computers in the 1940s, and indeed, for the field of computing science.

This field has two distinct roots, one in engineering and one in mathematics. The engineering aspect of computing science is aimed at designing and building practical computing devices; the mathematical aspect is concerned with abstract models of computation. The work of Charles Babbage was, for the most part, a complex and challenging engineering endeavour – likely the most ambitious such endeavour of his century. Ada's vision to use such devices for computing on data other than numbers, on the other hand, is much more closely aligned to what was to become the mathematical foundation of computing, laid by several mathematicians during the first half of the 20th century, most notably, Alan Turing. Later, as computing machinery and the algorithms running on it approached levels of complexity otherwise seen only in living organisms, a third foundation was laid in empirical science.

At this point, let me briefly comment on the nature of my field. It is commonly thought that computer science is, indeed, a science of computers – that is, of computing devices. This is a serious misperception; in the words of Edsger Dijkstra, arguably the most influential Dutch computer scientist and an alumnus of this university:

> *Computer science is no more about computers than astronomy is about telescopes.* [30]

So what then is it that computer scientists like me study? It is primarily computation

[4] To appreciate this fully, it is useful to contemplate the following statement from Ada's Note A: "The operating mechanism can even be thrown into action independently of any object to operate upon (although of course no result could then be developed). Again, it might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."

– systems and processes that operate, in clearly and precisely defined ways, on data, on information. This is why, in most European languages other than English, the name of our field does not reference computers, but rather information – as in the Dutch and Italian *informatica*, German *Informatik* and French *informatique*. It is merely a small step in the right direction to use the term computing science for this field that is focussed on computation rather than computing devices.

Computation not only takes place in the engineered devices we call computers, but also in practically all biological systems, from cells to complex vertebrate brains. As mentioned earlier, deliberate, manual computations are also performed by humans, individually and in groups. Computation occurs wherever information is being processed by means of algorithms, and computing science is the study of these processes. Designing and analysing such processes requires a mind- and skill-set often referred to as computational thinking, which involves formalisation, abstraction and modelling, along with a large dose of general problem-solving skills. That said, of course, it is the availability and use of computing machinery that truly gives wings to computational thinking, and thus brings about one of the most profound changes in human history.

Over the last 40 years, as computers became ever cheaper, smaller and faster, algorithms have begun to affect almost all aspects of our lives, industry, society and culture. Our phones, cars, TV sets and washing machines are all running increasingly sophisticated algorithms. Banks, insurance companies and stock markets completely rely on algorithms. Artistic production and scientific discovery are increasingly enabled by algorithms. We have entered an age of computation, and – in the words of an excellent article on the subject published just a few months ago in the Economist – "algorithms are everywhere" [7].

2 The power of heuristic search

As we think about algorithms – clear and precise instructions that, flawlessly executed, solve a specific problem – it is not hard to see that most problems can be solved by many different algorithms. Consider, for example, the task of finding a name in a long list of names. This could be done by checking every entry in the list, starting from the first and working forward towards the end. Or we could start at the end and work backwards. Or we could split our list in the middle and search both parts, concurrently or one after the other. One problem, many algorithms for solving it – but which of those should we use?

150 years ago, Ada Lovelace was keenly aware of this question when she wrote:

> *One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.* [24, Note D]

This desire to solve given problems as fast as possible lies at the heart of much work in

computing science and its applications. It is often thought that the major improvements in our ability to solve large-scale, challenging problems computationally are primarily due to advances in computer hardware, which have been following an exponential trajectory for the last 50 years, with the complexity of the integrated circuits that make up computers and the performance of microprocessors doubling roughly every 2 years. Since 1996, when I started my Ph.D. research, computer performance thus increased by a factor of 1 400, and since 1985, when I first began programming computers, by a factor of over 65 000. This means that a task that took 1 year of computing time in 1985 could be completed in 8 days in 1996, and in 8 minutes today. [5]

However, it is important to realise that far greater performance gains are achieved by improving not computer hardware, but the algorithms running on it (see, *e.g.*, [27], p. 71). This is especially true for a broad range of particularly challenging problems known as NP-hard problems. How fast these problems can be solved is one of the biggest open problems in computing science, but most experts strongly believe that the time required for solving them increases exponentially with problem size. As an example, consider the problem of finding the shortest (or fastest) round trip visiting a given number of places – say, wall poems in the city of Leiden. (Interestingly, the shortest route between two places can be found much more efficiently, using a clever algorithm due to the previously mentioned Edsger Dijkstra.) NP-hard problems arise prominently in many computing applications, from logistics and transportation to conservation biology, from drug discovery to theoretical physics. Solving these problems essentially requires searching for feasible or optimal solutions within astronomically large spaces.

While it is generally believed that the computationally costly search process cannot be avoided, it can be greatly accelerated using so-called heuristics – "rules of thumb" that guide the search towards the desired solutions to a given problem. Unfortunately, good heuristics are difficult to find, and their efficacy can usually be only established empirically, using computational experiments and statistics.

Heuristics can be amazingly efficient. There are over 5.7×10^{169} different round-trips visiting all 107 wall poems of Leiden – considerably more than the 10^{80} elementary particles estimated to make up our universe. Even if every one of the 10^{80} particles were in fact a computer running as fast as theoretically possible, it would take billions of years to check the length of all these round-trips in order to find the shortest. Yet, using state-of-the-art, extremely powerful heuristics, the shortest round-trip can be determined in about a second on an ordinary laptop computer – such is the power of heuristics.[6]

It is important to realise that most challenging computational problems are fundamentally

[5] The doubling rate for circuit complexity is due to Gordon E. Moore [25]. While the rate of performance doubling is often cited as 18 months, likely in reference to a statement attributed to Intel executive David House, the true hardware speed-up between 1980 and 2010 has been around 50 000-fold, on an exponential curve with a doubling rate of 24 months [23, Figure 2.2]. In my illustrative example, I assume near-perfect parallelisation speed-ups, as can be achieved quite easily, *e.g.*, for state-of-the-art stochastic local search algorithms.

[6] In fact, if we were content with a slightly sub-optimal solution to our shortest round-trip problem, the computation would be even faster, as is the case for many NP-hard optimisation problems.

search problems, whose practically efficient solution critically hinges on heuristics. This is particularly the case for many problems in artificial intelligence – the area of computing science dedicated to automating tasks that traditionally require human intelligence, such as planning and reasoning.

I started studying heuristic search methods for solving such problems as efficiently as possible when I was a Master student in the group of Prof. Wolfgang Bibel in Darmstadt, Germany [3, 14]. I was fortunate to be able to make a series of contributions that substantially improved the state of the art in solving several prominent NP-hard problems, including the so-called propositional satisfiability (short: SAT) problem – an intriguingly easy-looking logical reasoning problem that not only lies at the heart of computational complexity theory, but also has important applications in ensuring the correct operation of computer hard- and software (see, *e.g.*, [17]).

Without going into detail, my contributions in this area were made possible by two key elements: Highly stochastic search techniques (*i.e.*, techniques that heavily use randomised heuristic decisions) and advanced empirical methodology for studying algorithms whose behaviour is inaccessible to traditional mathematical analysis. I became quickly convinced that effective, heuristic algorithms for solving challenging computational problems could and should be studied using the scientific method, and specifically, by means of carefully designed experiments and statistical analysis of the data obtained from these. I thus embraced this third pillar of our field, and dedicated myself to helping build the empirical foundation of computing science [16].

3 The machine learning revolution

Programming computers is hard, mainly because of the difficulty of designing correct and effective algorithms. Take, for example, the task of determining whether a patient has breast cancer based on visual characteristics of the cells from a FNA biopsy. This classification task is challenging even for human experts, and manually constructing an algorithm whose predictions are as accurate would be impossible for most (if not all) computing scientists. Intriguingly, it is possible to construct such an algorithm automatically, essentially by searching within a large space of algorithms one that produces a minimal number of misclassifications on a given set of cases for which correct diagnoses are available.

The automatic construction of algorithms that perform well on given data conceptually resembles the way in which humans develop a broad range of skills: by means of learning. The idea to program computers through a learning process can be traced back to a seminal article written by Alan Turing in 1950 [32], and has since given rise to one of the richest and most impactful research areas in computing science: the area of machine learning. The key idea behind machine learning is to program computers such that they can essentially program themselves – to design algorithms that do not merely solve a given problem, but rather produce a good algorithm for the problem at hand (such as

detecting cancer from an image of a cell sample). This idea is not only intriguing, but also very powerful; its broad adoption, currently well underway, fundamentally transforms the way we program and use computers.

Technically, most forms of machine learning are based on concepts from statistics and optimisation. As a simple example, consider a technique known as decision tree learning (see, *e.g.*, [5, 29]). Imagine we have a number of characteristics that describe a cell sample, such as the uniformity of cell sizes and shapes. A decision tree essentially is a sequence of rules that examine one such feature at a time and then, based on the observation of this feature, selects the next rule to be applied. Each rule is a simple yes/no question, such as: "Does the distribution of cell sizes have standard deviation larger than 2?" The last rule in the sequence decides whether the sample is classified as cancer. This corresponds to a hierarchical decision procedure that can be drawn in the shape of a tree, with the first rule to be applied at the trunk (or root) of the tree, and each subsequent rule corresponding to smaller and smaller branches, all the way to the leaves, which are labelled with the final cancer diagnosis, 'yes' or 'no'.

Good decision trees can be constructed automatically. While the details are somewhat involved, the key idea is to grow the tree from the root, one branch (or rule) at a time. At any stage of this process, the training cases (*i.e.*, cell sample characteristics with confirmed diagnosis) are assigned to the current leaves of the tree, strictly according to the rules in the tree. Thus, our first tree has no rules and all training cases in its single leaf. This means that regardless of whether we label this leaf 'yes' (cancer) or 'no' (cancer-free), many cases are misclassified. Now, in each step we add the rule that essentially gives the maximum reduction in misclassifications. The decision trees (*i.e.*, systems of rules) thus obtained can solve our problem reasonably well; however, the procedure can be much improved by constructing multiple trees, each on a randomly sampled subset of the training cases and grown using a randomised and restricted rule selection mechanism. This way, we obtain a set of trees – a so-called random forest.

Random forests and other state-of-the-art machine learning techniques have achieved astounding success in many applications, ranging from diagnosis of diseases, such as cancer and Parkinson's, to drug design; from detecting credit card fraud to recommending books and movies. Over the last five years, an approach known as deep learning, which uses neural network models directly inspired by the physiology of our brain, has rapidly gained prominence and achieved impressive performance on complex learning tasks involving large data sets (see, *e.g.*, [2]). State-of-the-art applications in computer vision, such as face recognition, and natural language processing, such as automatic translation, summarisation and categorisation of text, are based on deep learning, and self-driving car technology, currently under development by several companies, heavily relies on it.

4 Automated machine learning

Machine learning provides powerful tools and techniques for extracting information from data, and more importantly, for automatically constructing algorithms, such as decision trees, random forests or neural networks, that can solve challenging classification, prediction and modelling tasks. However, effective use of those techniques demands considerable human expertise, and typically, many non-obvious choices must be made in order to achieve the performance levels required in challenging real-world applications. The reason for this is the same as discussed earlier in the context of solving computationally challenging, NP-hard problems: the need to rely on expert-designed and -calibrated, empirically optimised heuristics.

As data science transforms the way we analyse and leverage large amounts of data, the adoption and use of machine learning outpaces the availability of the expertise required for the effective use of these techniques. In addition, there is evidence that human experts are usually unable to make the truly best design choices (see, *e.g.*, [18, 22]). This should not be too surprising, since optimising over many design choices that interact in complex ways essentially requires searching within a high-dimensional, vast space – precisely the kind of challenge encountered when solving NP-hard problems, for which we know that human experts are no match for good heuristic algorithms running on blindingly fast computers.

This observation gives rise to the idea of automating the selection and calibration of machine learning algorithms, by means of powerful, heuristic search and optimisation techniques. The pursuit of this idea defines an area of machine learning known as automated machine learning (short: AutoML), and my group at the University of British Columbia – in close collaboration with my colleague, Prof. Kevin Leyton-Brown – was one of the first to work on the topic [31]. While other work on AutoML focussed on specialised techniques for the challenging problem of selecting and configuring machine learning procedures, we were interested in a general-purpose approach that could make performance-optimising design choices not just for machine learning procedures, but for arbitrary algorithms.

The key idea behind our approach is that of sequential model-based optimisation. It is based on the observation that evaluating a specific combination of design choices is usually quite expensive in terms of computing time, since it requires, in the case of AutoML, training a machine learning procedure, such as a random forest classifier, and then evaluating it on test data. To save some of this work, we use machine learning to predict the performance obtained for any combination of design choices, and we then make the choices that are predicted to be best. The problem with this approach is, of course, how to obtain sufficiently accurate predictions. We overcome this by starting with a cheaply obtained, usually inaccurate prediction model. We then alternate between using the model for making design choices, evaluating the resulting machine learning procedure and improving the model based on the observed performance [19]. This really is learning to search effectively, and since in this case, the aim of the search is to find a good machine

learning procedure, we effectively learn how to learn.

Of course, much remains to be done in the automation of machine learning. With colleagues in Eindhoven, we have started working on automating the construction of entire machine learning workflows. Here in Leiden, my group is working on automated semi-supervised learning, an approach that can leverage large amounts of unlabelled data. Meanwhile, researchers at Google have begun a major effort to apply automated machine learning to deep neural networks.

5 Programming by optimisation

This much broader view of machine learning leads us to what I consider one of the most exciting ideas I have been working on: the concept of programming by optimisation (PbO) [15]. PbO is a rather radical departure from the traditional way we think about programming computers, which requires clear and precise instructions – instructions that can be unambiguously executed with precisely predictable results. In PbO, we deliberately leave open design decisions that we cannot make in a compellingly justified way during the design of an algorithm, and the result of this design process is not an algorithm or program, but a space of programs. Within this potentially vast space, we then find specific programs that perform well on the kind of data characteristic for a given application situation, using powerful search, optimisation and machine learning techniques, such as the previously outlined sequential model-based optimisation process.

My group and I have leveraged programming by optimisation to build better solvers for a broad range of widely studied problems, from propositional satisfiability and its important applications in hard- and software verification to AI planning; from protein structure prediction to supervised machine learning and wildlife conservation (see, *e.g.*, [15]). With partners from industry, we have also worked on real-world applications in forestry resource management, decision support for the oil and gas industry, and effective use of clean energy.

Once we adopt the PbO paradigm, software design changes quite radically. Rather than locking in choices during the design phase, based on limited data and ad-hoc experimentation, algorithm designers and software developers can now focus on the creative task of devising a range of design options, of proactively seeking design alternatives – and let the machine figure out what works best under various circumstances. Thus, PbO leverages expert intuition and algorithmic efficiency, human creativity and computational power to build better software, to find better solutions for a broad range of challenging problems. At the core of PbO lies a fundamental departure from traditional thinking about algorithm design, a shift in paradigm made possible by a generalised notion of machine learning that boldly moves us beyond programming.

6 The future of computation

In 1973, science-fiction author Arthur C. Clarke famously wrote:

> *Any sufficiently advanced technology is indistinguishable from magic.* [6, p. 21]

Indeed, many of the capabilities information technology gives us these days would have appeared magical only a few decades ago. Smart home technology lets us control lights, turn up the heat and unlock doors with voice commands. Advanced computer graphics produces believable appearances of actors long dead. And computer programs can trick at least some people for some time into believing they are chatting with another person in on-line fora.

Which brings me to the second part of my title: The quest for machine intelligence. Whether and how we can construct machines that are as intelligent as we are is arguably one of the great questions of humanity, along with questions about life (or intelligence) beyond Earth, the origins and destiny of the universe, and the origin and creation of life. In his article titled "Computing machinery and intelligence", published in 1950, Alan Turing provided a visionary and enlightened view on the subject [32]. In it, he discusses the question "Can machines think?" and proposes what amounts to an operational definition of intelligence. This definition is based on a concept now known as the Turing test, in which a person communicates with a test subject by exchanging free-form text messages on arbitrary topics, with the purpose of finding out whether the test subject is another person or an algorithm running on a machine. When this decision cannot be made with a reasonably low margin of error (*i.e.*, error probability significantly below 50%), the machine should be considered intelligent.

Although Turing's article is now almost 70 years old, it is very much worth reading, as it contains some timely thoughts on artificial intelligence (AI). Notably, investigating how human-level AI could be achieved, he draws a strong analogy to the way human intelligence develops in children and lays the foundation for the field that is now known as machine learning. For Turing, machine learning is "a departure from the completely disciplined behaviour involved in computation" [32, p. 459], and he notes that "processes that are learnt do not produce a hundred per cent. certainty of result; if they did they could not be unlearnt" [32, p. 459].

Like Turing and most contemporary AI researchers, I believe that human-level AI can be achieved, and that machine learning is an essential ingredient. It is important to note that machine learning alone is insufficient; indeed, in most current demonstrations of human-level performance in games like chess, go, poker, Jeopardy, and in AI applications such as cancer detection, machine learning techniques work in concert with other computational methods, including heuristic search, natural language analysis and image processing (see, *e.g.*, [8]).

Of course, even just in the area of machine learning, there are many open challenges. These include the design of more effective methods for generalising to data that qualitatively differs from given training examples; the development of frugal techniques that can learn from less data, on less powerful hardware, with less complicated algorithms; and work on methods that are more understandable and free from unfair bias. In all this, leveraging my research on automated machine learning and programming by optimisation, I aim to make major contributions. More generally, my group and I will strive towards predictable, robust and performant techniques in machine learning and artificial intelligence – techniques that permit us to build trustworthy, high-performance systems that make responsible use of potentially sensitive data.

As awareness of artificial intelligence has broadened – fuelled by impressive successes in limited domains – of late, there has been increasing debate on the risks associated with human-level AI. I note that this debate is useful and necessary, but not new. In a lecture in 1951, Alan Turing already noted:

> *[...] it seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers. [...] they would be able to converse with each other to sharpen their wits. At some stage therefore, we should have to expect the machines to take control [...]* [33, p. 10][7]

Does this sound disturbing to you? It most definitely should. Within the AI research community, there are radically different opinions on the issue. I decisively side with Turing (and many of my colleagues) by believing that human-level AI, once achieved, will necessarily evolve into super-human AI, since it is not subject to the limited resources of our biological hardware. This crucial step from human-level to super-human intelligence will likely occur very quickly. Like renowned physicist, Steven Hawking, I believe that, once this happens,

> *Every aspect of our lives will be transformed. In short, success in creating AI could be the biggest event in the history of our civilisation.* [12]

Many of my colleagues are starting to think very seriously about the risks and implications of creating human-level AI. This timely and important conversation should not be limited to AI researchers, but involve experts from other disciplines, politicians and the general public. In my view, AI researchers have a responsibility for engaging in this conversation. It is beyond the scope of this lecture to discuss this topic in any depth, but I will briefly touch on three points.

[7] From the context in which he made this statement, it is unclear how serious Turing took the concern he raised; he did foresee opposition against the realisation of human-level general AI, but dismissed it without deeper discussion. Interestingly, earlier in the same lecture, he covered machine learning (to be precise: reinforcement learning) and the need for randomisation, in order to achieve non-deterministic and unpredictable behaviour.

First, it may well be that, even assuming a committed and sustained push for it, general human-level AI will take at least 50 years to achieve, and this leads some to believe that it is far too early to worry about it. I believe this is somewhat naïve, since crucial advances towards this goal are impossible to predict, and we may well have far less time than 50 years to think through the ramifications of this transformative event.

Second, there are likely no easy solutions to the question of how to deal with the risks and opportunities of AI. Take, for example, the seemingly simple solution of making sure that any intelligent system has an off-switch – a way to cut off its power, or to disconnect it from its communication channels, sensors and actuators. Aside from the ethical question whether and under which circumstances it is justifiable to take this drastic action on an intelligent, self-aware entity capable of fear and suffering (as any general, human-level AI must be by definition), the off-switch would likely be useless: As argued previously, general-human level AI, once achieved, should be expected to very quickly exceed human intellectual capabilities. This type of intelligence would likely be able to influence and manipulate us in subtle ways – and thus make sure that we would never want to switch it off.

Third, an argument can be made that we may well depend on machine intelligence to help us cope with the consequences of our severely limited ability to responsibly manage our environment and the crucial resources it provides to us. Evolution has us well equipped for living in a world predominantly governed by local, short-term phenomena and interactions, where the consequences of our actions are limited in their reach and scope. But over just a few centuries, we have brought about profound change. Our world is now densely interconnected, and much of what we do as societies has long-term, global effects. Evolution is far too slow to help us adapt to this new situation; therefore, we must utilise technology to help us overcome our cognitive limitations.

However, I strongly believe that general, human-level (or super-human) AI is neither required nor best suited to meet this need. Of course, we should use advanced computational methods to enhance our ability to see and manage the long-term, long-range consequences of our actions, but for this, we do not need to *replicate* human intelligence – we need to *augment* it, to help us compensate for our biases and shortcomings. Hence, our quest for machine intelligence should have this goal: to *augment*, not to *replace* human intelligence.

7 Coda

Let us briefly return to the group of Puritan separatists who left Europe for the New World in the autumn of 1620. Those so-called pilgrims, whose colony and culture was to gain central importance to US American identity up to the present day, came from Leiden, where they had led a quiet and modestly comfortable life. In fact, most of them lived in small houses a mere stone's throw from this building. They had come to Leiden (via Amsterdam) to escape religious oppression and persecution in their native England. In

the words of their chronicler, William Bradford:

> *For these & other reasons they removed to Leyden, a fair & bewtifull citie, and of a sweete situation, but made more famous by ye universitie wherwith it is adorned, in which of late had been so many learned man.* [4, p. 17]

And yet, despite this much improved situation, they decided to move once again – as documented by Bradford:

> *So they lefte [that] goodly & pleasante citie, which had been ther resting place, nere 12 years; but they knew they were pilgrimes, & looked not much on these things; but lift up their eyes to ye heavens, their dearest cuntrie, and quieted their spirits.* [4, p. 57]

So why did they embark on what must have been an uncertain and perilous journey? In part, they were looking for a chance for economic betterment; however, there appears to have been another reason. Back then (as is still the case today), the Dutch were well-known for their tolerance. As much as the pilgrims benefitted from this, they did not share the attitude, but were quite concerned about the 'libertine' morals of the Dutch and about dilution of their own culture [4, p. 24].

This rings strangely, and perhaps disturbingly, familiar. It is worth recalling that the motto of this university is *libertatis praesidium* (bastion of freedom). The stained glass windows in the wall behind me serve as another powerful reminder how precious and important that concept is and always will be. Things get truly complicated when liberties collide. It is then, when neither logic nor learning can provide easy solutions, that human bias and short-sightedness become truly problematic. But it is also then that humility, compassion, measured tolerance and understanding shine. It is my view and aim that advanced computation and machine intelligence can (and should) help us recognise and overcome shortsightedness and bias, and I am optimistic that this goal can be achieved, if we commit to it and pursue it vigorously. And what better place to do it than in

> *[...] a fair & bewtifull citie, and of a sweete situation, but made more famous by ye universitie wherwith it is adorned, in which of late had been so many learned man.* [4, p. 17]

an institution committed and known to uphold *libertatis praesidium*.

Acknowledgements

This brings me to the final part of this lecture. Precisely 97 years ago (and 300 years after the Mayflower approached the American coast), on the 27th of October 1920, Albert

Einstein gave a lecture titled "Ether and the theory of relativity" [10] – his inaugural lecture here, at *Universiteit Leiden*.[8] Back then, in many ways, physics was what computing science is now: a field that shapes the world and our thinking, a discipline that excites and inspires, unifies and connects. I consider myself very fortunate to be working in this field, and especially in the area of artificial intelligence.

Standing before you today, I am humbled and deeply honoured to become part of a succession of scholars reaching back 442 years to the beginning of this venerable institution. While I cannot hope to match the contributions made by many of them, I will certainly do my best to fulfil and exceed the expectations associated with my position here.

I thank the Executive Board (*College van Bestuur*), the Board of the Faculty of Science, and all others who have contributed to my appointment as Professor of Machine Learning for the trust they have placed in me.

I also thank my colleagues at LIACS, who have welcomed me with open arms and open minds, notably Jaap van den Herik, Joost Kok, Wessel Kraaij, Grzegorz Rozenberg and Harry Wijshoff. Furthermore, I am deeply grateful to our exceptional staff, notably Vianney, Marloes, Annemart, Tjitske, Mariska and Abdel: Without your help and guidance, I would have been lost and stranded. My thanks also go to Bert van Polen and his staff at the ISSC, whose support in setting up my computational infrastructure I greatly value.

Hooggeleerde Plaat, beste Aske, since I met you barely two years ago, you have become a role model and an inspiration to me. I am certain that under your exemplary leadership, LIACS will reach new heights, and I am very excited to have the opportunity to work with you and your management team towards this goal.

Zeergeleerde van Duijn, beste Max, our conversations and nascent projects truly broaden my horizon, and I look forward to see our vision take shape.

Hooggeleerde Verbeek, beste Fons, our twinned gowns have created an unexpected and delightful connection that I hope will last for many years.

Hooggeleerde Bäck, lieber Thomas, I am fortunate to count you among my colleagues and my friends. Without your vision and support, I may have never discovered the joys of working at this university and at LIACS.

Dear Chuan, Marie and Jesper, as one of my favourite poets wrote: *"Und jedem Anfang wohnt ein Zauber inne"* – something magic dwells in every beginning [13, Band 2, p. 257]. I am very happy to have you as my companions, as we discover the magic in the beginning of our ADA research group.

I would also like to acknowledge my students at UBC – Chris Fawcett, Sam Bayless, Julieta Martinez, Chris Cameron and Yasha Pushak, as well as my former students and postdocs, interacting with whom has been and is a joy and a privilege.

[8] The date in the printed version, 5 May 1920, is incorrect, as Einstein's inaugural lecture had to be postponed for rather interesting reasons [9, p. 61].

I am deeply grateful to my UBC colleagues, Anne Condon and Alan Mackworth, whose mentorship has made me a better scientist and a better person. Furthermore, my heartfelt thanks go to my collaborators at UBC, notably Kevin Leyton-Brown, with whom I share a long and very productive history of intellectually stimulating exchange.

Reaching further back in time: Dear Bart, your research excellence and enthusiasm has inspired me for over 20 years, and without your kind and generous support, I would not be here today.

Lieber Wolfgang, I am truly delighted that you are here today. As my Ph.D. advisor, role model and mentor, your impact on my intellectual development and academic career could hardly be overstated.

Lieber Professor Walter, I thank you for illuminating my path at a critical junction, and for giving me unique opportunities and freedoms during my time in Darmstadt.

Dear friends, thank you all for being here. Tobias, Meret, Thomas, Jürgen, Morten, Florissa, Claire and Steve, I cherish the many moments and memories we share, the levity and hilarity, the conviviality and companionship that connects us.

Liebe Eltern, liebe Geschwister, it means much to me to share this moment with you. We can choose our friends, but not our family; still, given the choice, I would gladly choose you. To a large extent, my being here today is a result of all that you gave me at the beginning of my journey, throughout the years and at every step of my way.

Liebe Heike, lieber Robin, lieber Till, in many ways it was you who brought me here. You enrich my life in ways I could not have imagined, and that which we share completes me.

You all have been, and – I hope – continue to be part of my journey, a journey that has taken me there and back again: From my native Germany to Canada, a place that has profoundly changed me, personally and in terms of my academic endeavours, then back to my European roots. It was during my 20 years in Canada that I learned what I now know about taming the dragon of computational complexity and that I discovered what I consider my academic calling. I cannot know what lies ahead on this next leg of my journey, but I am certain to embark on it with an open mind and in excellent company.

Ik heb gezegd.

Additional acknowledgements. I gratefully acknowledge useful comments on earlier drafts of this document by Thomas Bäck, Aske Plaat, Simon Portegies Zwart, Grzegorz Rozenberg and Harry Wijshoff.

References

[1] Ibn al-Razzāz al Jazarī. *The Book of Knowledge of Ingenious Mechanical Devices.* Translated and annotated by Donald R. Hill. D. Reidel Publishing Company, Dordrecht, The Netherlands, 1974.

[2] Yoshua Bengio, Yann LeCun, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[3] Antje Beringer, Gerd Aschemann, Holger Hoos, Michael Metzger, and Andreas Weiß. GSAT versus Simulated Annealing. In *Proceedings of the 11th European Conference on Artificial Intelligence*, pages 130–134.

[4] William Bradford. *History of Plymouth Plantation*. Boston: Privately printed, 1856. https://archive.org/details/historyplymouth00bradgoog.

[5] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Wadsworth International Group, Belmont (CA), USA, 1984.

[6] Arthur C. Clarke. *Profiles of the Future: An Inquiry into the Limits of the Possible (revised edition)*. Harper & Row, New York (NY), USA, 1973.

[7] Tim Cross. The Economist explains: What are algorithms? The Economist, 30 August 2017, https://www.economist.com/blogs/economist-explains/2017/08/economist-explains-24. Last visited 1 October 2017.

[8] Jennifer Chu-Carroll James Fan David Gondek Aditya A. Kalyanpur Adam Lally J. William Murdock Eric Nyberg John Prager Nico Schlaefer David Ferrucci, Eric Brown and Chris Welty. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

[9] Dirk van Delft. Albert Einstein in Leiden. *Physics Today*, 59(4):57–62, 2006. http://physicstoday.scitation.org/doi/10.1063/1.2207039.

[10] Albert Einstein. *Äther und Relativitätstheory*. Julius Springer, Berlin, Germany, 1920. http://alberteinstein.info/vufind1/images/einstein/ear01/view/3/CP7Doc38_pp305-309_321_000016788.pdf.

[11] Tony Freeth, Y. Bitsakis, X. Moussas, J.H. Seiradakis, A. Tselikas, E. Magkou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsay, A. Crawley, P. Hockley, T. Malzbender, D. Gelb, W. Ambrisco, and M.G. Edmunds. Decoding the ancient Greek astronomical calculator known as the Antikythera mechanism. *Nature*, 444(7119):587–591, 2006.

[12] Alex Hern. Stephen Hawking: AI will be 'either best or worst thing' for humanity. The Guardian, 19 October 2016, https://www.theguardian.com/science/2016/oct/19/stephen-hawking-ai-best-or-worst-thing-for-humanity-cambridge. Last visited 1 October 2017.

[13] Hermann Hesse. *Das Glasperlenspiel*. Fretz & Wasmuth Verlag, Zürich, Switzerland, 1943.

[14] Holger H. Hoos. Aussagenlogische SAT Verfahren und ihre Anwendung bei der Lösung des HC-Problems in gerichteten Graphen. Diplomarbeit, Fachbereich Informatik, Technische Universität Darmstadt, Germany, 1996.

[15] Holger H. Hoos. Programming by optimization. *Communications of the ACM*, 55:70–80, February 2012.

[16] Holger H. Hoos. *Empirical Algorithmics*. Cambridge University Press, Cambridge, UK, in preparation.

[17] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search–Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco (CA), USA, 2005.

[18] Frank Hutter, Domagoj Babić, Holger H. Hoos, and Alan J. Hu. Boosting verification by automatic tuning of decision procedures. In *Proceedings of the 7th International Conference on Formal Methods in Computer-Aided Design (FMCAD'07)*, pages 27–34, 2007.

[19] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION 5)*, volume 6683 of *LNCS*, pages 507–523. Springer-Verlag, Berlin/Heidelberg, Germany, 2011.

[20] Stephen C. Kleene. *Introduction to Metamathematics*. North-Holland Publishing Company, Amsterdam, The Netherlands, 1952.

[21] Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd edition)*. Addison-Wesley, Boston (MA), USA, 1997.

[22] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*. Addison-Wesley, Boston (MA), USA, 2015.

[23] Paul E. McKenney, editor. *Is Parallel Programming Hard, And, If So, What Can You Do About It?* Freely available on-line, 2017. https://www.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.2017.01.02a.pdf.

[24] Luigi Federico Menabrea and Augusta Ada King, Countess of Lovelace. Sketch of the analytical engine invented by Charles Babbage, Esq., with notes by the translator. *Scientific Memoirs, Selected from the Transactions of Foreign Academies of Science and Learned Societies and from Foreign Journals*, 3:666–731, 1843. http://www.fourmilab.ch/babbage/sketch.html.

[25] Gordon E. Moore. Progress in digital integrated electronics. In *Technical Digest, 1975 International Electron Devices Meeting*, pages 11–13, 1975.

[26] John D. North. The astrolabe. *Scientific American*, 230(1):96–107, 1974.

[27] President's Council of Advisors on Science and Technology (PCAST). Designing a digital future: Federally funded research and development in networking and information technology. https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/pcast-nitrd-report-2010.pdf. Last visited 1 October 2017.

[28] James F. Deetz Patricia Scott Deetz. Passengers on the Mayflower: Ages & occupations, origins & connections. The Plymouth Colony Archive Project, http://www.histarch.illinois.edu/plymouth/Maysource.html. Last visited 1 October 2017.

[29] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[30] Hamilton Richards. Edsger Wybe Dijkstra, 2012. A.M. Turing Award Winners, http://amturing.acm.org/award_winners/dijkstra_1053701.cfm. Last visited 1 October 2017.

[31] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-13)*, pages 847–855, 2013.

[32] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. https://academic.oup.com/mind/article-lookup/doi/10.1093/mind/LIX.236.433.

[33] Alan M. Turing. Intelligent machinery, a heretical theory, 1951. The Turing Digital Archive, AMT/B/4, http://www.turingarchive.org/browse.php/B/4.