# Can Machines think?

Mujtaba Hassan

January 8, 2024

The question "can machines think?" was famously asked by Alan Turing in his 1950 paper [18]. The question has since been an active subject of philosophical debate since. With the insurgence AI technologies in the recent years the question of artificial intelligence vs human intelligence is boggling the minds of more and more people. In this paper I explore this very question. Due to the vagueness of the original question, I will reformulate the question into something more arguable. To be precise, in this paper I argue that a computing machine cannot think on the same capacity that a Human can. The composition of this paper is as follows. I first reformulate the original question, and argue why reformulated question is subject of interest to us and a sufficient replacement for the original question in light of the recent AI debates. I then give some preliminary definitions and notions which the arguments that follows uses. I will then critique the validity of the Turing test as a sufficient answer to our question. From there I will argue why the thinking capacities of the Human mind surpasses those of a computing machine.

Like many others before me I too shall concede that the original question of "can machines think?" by itself is unanswerable. The notion of "machines" and "thinking" is too broad and vague to sufficiently answer the question. Can cats and dogs think? And is that thinking comparable to humans? What does one even consider a machine? An argument can be made that humans are just very complex biological machines, likewise an argument can be made that human cognition surpasses the idea of machines and no machine no matter how complex cannot match human cognition. To avoid these situations we first fix what we mean by a machine. The Turing machine was first introduced by Alan Turing in 1936 [16]. Turing originally introduced the notion to solve the entscheidungsproblem proposed by David Hilbert and Wilhelm Ackermann [8]. It is has since been used as the standard model of computation. Formally the Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_{\text{start}}, q_{\text{accept}}, q_{\text{reject}})$, where $Q$ is the possible states the machine can be in, $\Sigma$ is the set of tape alphabets these are the symbols the machine can write on its tape, $\Gamma$ is the set of input alphabets these are the possible symbols the machine may get as input on its tape, the transition function $\delta : Q \times \Gamma \rightarrow Q \times \Sigma \times \{L, R\}$ which dictates the action of the machine and take the machine from one configuration to another, and $q_{\text{start}}$, $q_{\text{accept}}$, $q_{\text{reject}}$ is the start state, the accept state and the reject state respectively[15]. Intuitively a Turing machine consist of an infinite tape, a read/write head which reads symbols from the tape and writes symbols on the tape and may more left or right along the tape, and finally a state control which dictates the actions of the machine whenever a particular symbol is read [15][4]. Alphabets are just a set of symbols, a string is a sequence of alphabets and a language is just a set of strings [15]. For example from alphabets $\{a, c, t\}$, some possible strings are $ac$, $cat$, $tac$, $act$, and a possible language is $\{cat,\ act\}$ [15]. A computational problem just refers to a language and deciding if a given string is part of the language or not [15]. So a "computational problem" and "deciding a language" are synonyms. It was then shown that any other sufficiently complex and physically realizable models of computation is equivalent to the Turing machine [17] [2] [12]. This is known as the Church-Turing Thesis [15]. So any modern computer would be at most as powerful as the Turing machine. So a computing machine just refers to a Turing machine. This notion is synonymous to the notion of algorithm [1]. So saying that an algorithm exits for a problem $P$, is the same as saying there exists a Turing machine $M$ that decides the language $P$ [15]. As as any modern computer would be at most as capable as the turing machine, and furthermore by Church-Turing thesis any physically realizable model of computation is at most as strong as the Turing machine it suffices to fix our definitions of machine as the Turing machine. All AI that runs on a digital computer would run on a Turing machine, so considering a model beyond this is unnecessary for our discussion. This is also helpful as these are abstract machines unrestricted by physical constraints. So we avoid arguments such as "a machine can think we just need a more powerful computer" or that "a machine cannot think as we cannot have enough RAM or strong enough processor". Now the question on thinking, as argued above the question on "can machines think?" is an unreasonable one to ask, maybe by some definition of thinking the machine maybe thinking while another definition of thinking would show the contrary. What is of interest to us is the capacity that the machine can "think", for the debate of artificial intelligence vs human intelligence the more pressing question is "can a machine think on the capacity that a human can?". To sum this up we now replace the question of "can machine think" with "can a Turing machine think on the same capacity that a human can?" This question is now more answerable as we now are dealing with a comparative question, human thinking and capabilities of a Turing machine can be quantified, and thus this new question is more answerable.

Turing in his 1950 paper when exploring the question of "can machine think?" argued that the question is too vague to answer, and introduced the idea of the Turing test, which Turing called the imitation game [18]. The game is played with three players, $A$ who is a machine, $B$ who is a human, and $C$ who is the interrogator. $C$ is seated a room apart from $A$ and $B$, and is not aware who the machine is and who the human is. $C$ then takes turns asking question to $A$ and $B$. After some turns $C$ needs to figure our who the machine is and who the human is. If $C$ cannot successfully conclude who the machine is then the machine has passed the turing test [18]. Turing argued that humans convince each other that they are thinking and therefore by passing the Turing test the machine has convinced the human that its thinking and therefore its a sufficient criteria to conclude a machine is thinking [18]. There are several argument made against the sufficiency of the Turing test as a criterion for machine thinking. Turing himself gives counter argument to several of these [18]. One big argument made against the Turing test is the monkey on the keyboard argument, Turing suggests decreasing the statistical significance of such occurrence by taking a large sample size [18]. I argue that even if a machine can pass the Turing test with a large success rate over a large sample size, it still cannot be concluded that the machine is thinking. The simple matter of fact is that syntax is not the same as semantics, specially in context of logic [19]. A machine can pass the Turing test by simple symbol manipulation of the input. Such a machine has no sense of semantics and works purely syntactically, the state control of such machine may hold no information regarding the semantics of its language and might work simply my arbitrary symbol manipulation. Can such a machine be said of as thinking? This is the famous chinese room argument by John Searle [14]. Suppose you don't know any chinese and you are seated inside a room with table containing a set of rules that dictates what chinese symbols to write when certain chinese symbols are provided to you. Now there is a window in that room through which chinese speaking and understanding people write you some text in chinese, you use your table to write down an output corresponding to the given symbols and return it to the person who gave you the input. Now suppose the Imitation game is played in chinese with you as player $A$. Suppose you successfully pass the test with your table. But you still don't understand any chinese. You passed the imitation game with simple syntactical manipulation, and understand no semantics of chinese. Searle argues that a Turing machine passing the Turing test is analogous to the chinese room [14]. The machine doesn't "understand" anything about its own language and thus cannot be said as thinking. Modern AI technologies utilizing machine learning algorithms which are trained over some data set are one such example.

Now I will construct my main argument that a computing machine cannot think on the capacity that a human can. We consider the system of first order predicate logic. It has been proven that predicate logic is complete that is that every true statement that can be formulated in the system can be proven [19]. Furthermore it has also been shown that predicate logic is undecidable that means there doesn't exist an algorithm which when given a statement can decide whether there exists a proof of that statement [9]. This gives us a limitation on the capabilities of a Turing machine. Many such limitation are known, we build our argument by going over few of these limitations. A problem or language $P$ is called undecidable if a Turing machine cannot effectively solve it that is there does not exists a Turing machine such that given an arbitrary string it always decides in a finite number of steps whether the string is part of the language or not [15]. The most famous of such problems is attributed to Turing himself and if known as the Halting problem. The Halting problems asks the following; "Given a Turing machine $M$ and a string $w$ will $M$ halt (produce an output in a finite number of steps) on input $w$?". The problem can equivalently be described as whether a given algorithm terminates on every input. Turing showed that the Halting problem is undecidable [16][15]. This is the most famous limitation on the thinking capabilities of a Turing machine. I argue that is it possible for a human to solve the halting problem. The thinking capabilities of the brain provides a solution to the problem. The claim that a human can solve the Halting problem can be empirically tested. The exercise of determining if a given algorithm terminates for all inputs is a standard exercise for introductory programming courses. The students usually are not taught methods to solve the problem but use their intuition to solve this problem. Which supports the claim that human intelligence is not completely algorithmic. For each problem the brain creates some heuristic which cannot be expressed as an algorithmic process but more like an oracle that the rest of the algorithm queries. Yes a Turing machine equipped with such oracle can be able to solve the problem like the human mind does but the construction of such oracle itself would not be algorithmic. As if such an a oracle can be constructed with an algorithm then this contradicts the undecidability of Halting problem. One argument against this can be made that a Turing machine can still recognize the Halting problem and there solve it for some instances and as we cannot test for all infinite instances of the problem maybe Humans can also only recognize the problem and not decide it. In the next paragraph I will address this argument. The argument made here is an instance of a Godelian argument. Godel's incompleteness theorem states that in any system of logic sufficient enough to deal with some basic arithmetic we can construct statements which we call Godel statements such that they can neither be proven nor disproven within the system [7]. John Lucas most famous for his Godelian argument argues that Godel's incompleteness theorem shows that a machine cannot think on the level a human can. Lucas argues that a machine is bounded by the system its created in, in every such system which powerful enough to deal with basic arithmetics (which in context of our argument a Turing machine can deal with) we can construct a Godelian statement which cannot be proven or disproven within the system. So the machine can neither prove or disprove the statement, however a human can observe the truth of such statements [10]. Such argument was later famously also made my Roger Penrose [11]. I further this argument with the following claims. The human mind is also able to devise algorithms for a given problem $P$. Such a feat is done by semantic analysis of the language $P$ and devising an algorithm with respect to that. The formal decision problem corresponding to this would be; "Given a language $L$ and a turing machine $M$, does $M$ decide $L$?". From rice theorem is can be inferred that this problem is also undecidable [13][15, ,241 ]. That means that there doesn't exist an algorithm which can construct an algorithm for any given problem. This provides another limitation on the capabilities Turing machine. It can again be argued that human brains are more than capable of accomplishing such feat. Human not only devise algorithms for given problems but also have proven that a given problem cannot have an algorithm such as the Halting problem. With the understanding of the semantics of a language humans can construct algorithms for them. That means that the process of devising an algorithm for a given problem isn't algorithmic itself. One counter argument to these claims is that we are basing our argument on a finite set of empirical evidence while the limitation of the machine's capabilities are proven for the infinite number of input instances, who is to say that these limitation do not apply to humans when considering infinite input instances. We assume that the empirical evidence is sufficient to show that the human beings truly are capable of accomplishing these tasks and the limitations of Turing machines do not apply on human beings. In next paragraph we provide an argument to why this isn't an unreasonable assumption.

The limitation proven on the capabilities of Turing machines comes from the fact that these machines are bounded in some system of logic. We construct these machines withing some systems and these machines follow that system. The contradiction showed by Turing in [16] to show the undecidability of the Halting problem also replies on this. For humans we do not have such system. Lets take Paul and Patricia Churchland's argument that human are indeed some sort of computers, it may not be a Turing machine but some sort of computing machinery that maps input strings to output strings like a Turing machine or a recursive function does [3]. Now whatever system the human brain is bounded on is a superset of the system our constructed Turing machine is bounded by. As Peter Dennings argues that thought is something that occurs before the formulation of language and computers programed in such language cannot think (or cannot think outside the language) [5]. Our human mind generates the language on which we build the system in which we construct our machines, so human mind need not be bounded by those limitations that the machine is bounded by as the system that generated the language which bounds the machine at least belongs to a language which is a superset of the constructed language. So claiming that humans are not subject to the same limitation as a Turing machine is not unreasonable. The empirical evidence that human mind can solve the halting problem in a larger capacity compared to a Turing machine is also supported by the way the human mind approaches the problem. Human mind in a way works by guessing a way to see if the algorithm halts or not. The guess comes from some heuristic that the human mind holds as discussed above which itself is constructed seemingly randomly the guesses themselves and the processing of checking if an algorithm halts or the process of devising an algorithm for a problem itself is quite random. The human mind seems to have a random number generator which guides many such decision. However a Turing machine cannot have a true random number generator. A model of computation with such a devise to generate random numbers can be considered, but then the question is if such a devise can even physically exist? I will further this argument I presented here which is based on the way the Human mind solves these problems by discussing **NP-hard** problems. We don't know if the problems in class **NP-hard** can be solved in polynomial time by a deterministic Turing machine and it is widely believed by mathematicians and computer scientists that the answer if they can be solved efficiently by a Turing machine is false. Lets take the graph coloring problem for example. The problem is to check if a given graph $G$ can be colored with some $k$ colors. Now the problem is known to be **NP-complete**, and no polynomial time solution exists for graphs in general unless $\mathsf{NP} = \mathsf{P}$. For a computer there exits many algorithms to solve this problem, but all exact solutions run in exponential time, they check exponential (to the size of the graph) amount of different coloring and check if graph can be colored with $k$ colors. The way the human brain approaches that problem is quite different. The human mind doesn't necessarily checks all possible colors as the current algorithms does but rather assigns colors to vertices seemingly randomly with some heuristic. The goal is to reach a possible $k$ coloring my taking some insight from the observed graph structure. Like some might start by assigning colors to denser regions first or might assign colors with cliques and so on. These insights let humans solve the problem without having to explore all color assignments.

Similarly for deciding if a $k$ coloring is not possible the human mind reaches the conclusion by observing the overall graph structure and concluding that the coloring is not possible again by some internal heuristics. This way human mind can solve the problem much more efficiently. Which is different on how the machine approaches the problem given the current heuristics. If $\mathsf{NP} \neq \mathsf{P}$ then we can see that the human mind solves these problems more efficiently than a machine.

We approach the question of if a machine can think in light of the recent artificial intelligence vs human intelligence debate. We formulated the general question into a more answerable one, and relevant to our discussion that "can a Turing machine think on the same level as a human can?". We went over several arguments to support our claim that a Turing machine indeed cannot think on the level a human can. It is to note that the arguments we provided doesn't close the question itself, but aims to provide evidence for one side of the debate. As argued by Clark Glymour and Kevin Kelly in their response to Penrose that the question if a machine can think is not even empirically provable [6]. The human mind is very complicated and as Paul and Patricia Churchland claims, its the most complicated and sophisticated thing on the planet [3]. So in order to completely close the question we need a better understanding of the human mind than we currently possess.

## References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, USA, 1st edition, 2009.

[2] Alonzo Church. An unsolvable problem of elementary number theory. *Journal of Symbolic Logic*, 1(2):73–74, 1936.

[3] Paul M. Churchland and Patricia S. Churchland. Could a machine think? *Scientific American*, 262(1):32–37, 1990.

[4] Computerphile. Turing machines explained - computerphile, Aug. 2014.

[5] Peter J. Denning. The science of computing: Is thinking computable? *American Scientist*, 78(2):100–102, 1990.

[6] Clark Glymour and Kevin Kelly. Why you'll never know whether roger penrose is a computer. *Behavioral and Brain Sciences*, 13(4):666–667, 1990.

[7] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[8] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik (Principles of Mathematical Logic).* Springer-Verlag, 1928.

[9] Timm Lampert and Anderson Nakano. Explaining the undecidability of first-order logic. *Humboldt-Universit ̈at zu Berlin Computer- und Medienservic*, 2021.

[10] John Lucas. Minds, machines and gödel. *Philosophy*, 36(137):112–127, 1961.

[11] R. Penrose. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics.* Oxford landmark science. Oxford University Press, 2016.

[12] Rózsa Péter and Stephen Cole Kleene. $\lambda$-definability and recursiveness. *Journal of Symbolic Logic*, 2:38, 1937.

[13] H. Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74:358–366, 1953.

[14] John R. Searle. Is the brain?s mind a computer program? *Scientific American*, 262(1):26–31, 1990.

[15] Michael Sipser. *Introduction to the Theory of Computation.* Course Technology, Boston, MA, third edition, 2013.

[16] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265, 1936.

[17] Alan Turing. Computability and $\lambda$-definability. *Journal of Symbolic Logic*, 2(4):153–163, 1937.

[18] Alan Turing. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[19] Anita Wasilewska. *Logics for Computer Science: Classical and Non-Classical.* Springer International Publishing, 2018.