Artificial Intelligence and Human Ignorance

By William Cutler

Magnus Carlsen, world chess champion, kicks off the live-streamed game, and the opponent takes the first move--knight to f3. Carlsen responds with a pawn to d6, a standard move. Though impossible to tell from this board position alone, Carlsen has no chance of winning. And he knew that going in.

You see, Carlsen is not playing an ordinary opponent, but the latest AI from Chess.com, rated 3200 to his 2881. For every move he makes, the AI gives what appears to be the optimal response. From the very beginning, the AI earns a slow but consistent positional advantage. In the middlegame, it gives up a pawn to preserve this position as knights are traded for bishops, and vice versa. Then the queens trade, and it's almost the endgame. Now the pieces are bouncing around with more freedom to explore the board: attacking and defending, feinting and retreating. Each move affects the status of multiple other pieces as both sides develop contingencies for their contingencies. It's like watching two professional musicians trade solos or two fluent speakers converse. There's a rhythmic give and take, an ebb and flow. Carlsen is putting up a good fight, but each time he pauses for twenty seconds to make a move, his opponent responds instantly. Eventually, after a great performance, Carlsen is boxed into a situation where his best option is to trade his rook for a less-valuable bishop. The computer analysis shows it was, indeed, the best he could do, but it's a winning advantage for the AI. Carlsen resigned shortly thereafter.

* * * * * *

Throughout the game, what was the chess AI thinking? Was it "thinking" at all? These are very difficult questions. It can't just explain its thought process to us like Carlsen could. If the chess bot were simple and could be expressed as a series of statements like "if the board looks like X do Y," then we would have no problem understanding it. But such a program would require nearly as many "if" statements as there are atoms on Earth (10^{46} board positions vs 10^{50} atoms, roughly speaking). We also might not consider such a "brute force" program to be all that intelligent. So, if the program wasn't written with an impossible amount of code to account for every situation, how is it so good? Somehow, by playing millions of times against itself or other agents, it picks up move sequences, tactics, and checkmating patterns that improve each time. It's like how humans learn to play—through pattern recognition over many games but sped up to the n^{th} degree. Whether an AI that dominates at chess ought to be considered "intelligent" depends heavily on definitions and is for a different story. What's more intriguing is that we humans have managed to build something that is (A) better than we are and (B) acting in ways we don't really understand. The researchers at Chess.com who made the AI could tell you in excruciating detail how it works, but only from an outside perspective: the number of moves it looks ahead at, the heuristics it employs, and the training procedure it underwent in preparation for this game. Indeed, we could trace through every matrix multiplication and math operation that goes into a single move, but we still wouldn't really know what it is thinking. The same goes

for the AI programs making bigger waves lately, and if we want to have a chance at understanding and harnessing AI for good, we need to first understand how it is constructed.

The latest and greatest in AI that's been in the public spotlight, like ChatGPT and Sydney, are what's known as a generative deep learning model, or in particular a large language model (LLM). More generally, a *model* is a function that takes an input and gives an output to effectively approximate something complex. If the model is derived from principles and can be represented as a simple formula, it is an explicit mathematical model, like the Ideal Gas Law, which approximates the behavior of gasses in non-extreme cases. If the model is achieved through a training procedure, where it is shown many inputs and their expected outputs, it will learn a function that aligns well with the examples it has seen. This is known as "Machine Learning," or ML. ChatGPT is a *generative* model, meaning that it creates content of some kind—images, videos, text, etc. A model that predicts housing prices based on various factors is not generative, but one that designs a house given a user's text prompts is. Whether it's programmed explicitly or developed through a complex learning process, every model is just a function converting inputs to outputs. Some, like deep learning models, are just better at it than others. A lot better.

In creating an ML model, one critical choice is its *architecture*, or structure. This will affect how difficult the model is to train, how much data it would need, and how it performs. Neural networks (NNs) are the foundation of many architectures used today and have huge variability in their size and cost. A basic NN designed to classify images of handwritten digits can perform

quite well with only a few thousand parameters and could be trained on your laptop. The models making headlines today have billions of parameters, specialized NN architectures, and cost over $1 million in raw computing power just to train. One particularly important aspect of deep learning driving the latest progress is *scaling*, which quantifies how well the model improves just by giving it more data, parameters, or computing power, without ingenious changes to its structure. And it has been known for quite some time that deep learning scales very well. While there is more complexity behind the scenes than simply making a large neural network, understanding the basics of an NN will tell us about the structure of these deep learning models and highlight just how little we know about what they are thinking.

Put simply, a neural network is a parameterized function. It converts inputs to outputs, but how it accomplishes this depends on its parameters. If your toaster oven is a function to convert bread into toast, the temperature you set is a parameter that affects precisely how it turns bread into toast. For a neural network, the inputs may range from the pixel values of an image to a text prompt or scientific data collected in a particle accelerator; if it can be stored on a computer, it can be given to a neural network. The structure of a neural network is made of two types of components: neurons and weights separated into layers that are loosely inspired by neurons and their connections in the brain. The diagram below depicts an input layer, multiple hidden layers (where the magic happens), and an output layer. It's the many hidden layers that make this "deep" learning. Computation proceeds from left to right, earning it the name "feed-forward neural network." During training, if the network gives a bad output, its parameters are adjusted

in reverse order through a process called "back-propagation." The neuron simply stores a single number, called a *feature*, that is either an *input feature* or a *hidden feature* depending on the layer the neuron is in. Neurons in adjacent layers are connected by *weights*, which are just another number (parameter) that gets tuned as the model learns the function. The weights that go into a neuron govern which features in the previous layer it focuses on. The weights that come out of a neuron indicate the features in the next layer it will affect the most. For example, in a facial recognition network, one neuron might focus on pixels around the pupil, assigning a higher weight to them, but ignore pixels on the person's cheek. Unfortunately, this is purely an illustrative example—in reality, we have little idea what the deep, hidden features represent. Much like with chess AI, researchers know the ins and outs of how their neural network is structured and trained and could inspect each of the billions of computations occurring each second. But knowing the number stored in a neuron doesn't give any indication as to what that hidden feature means.

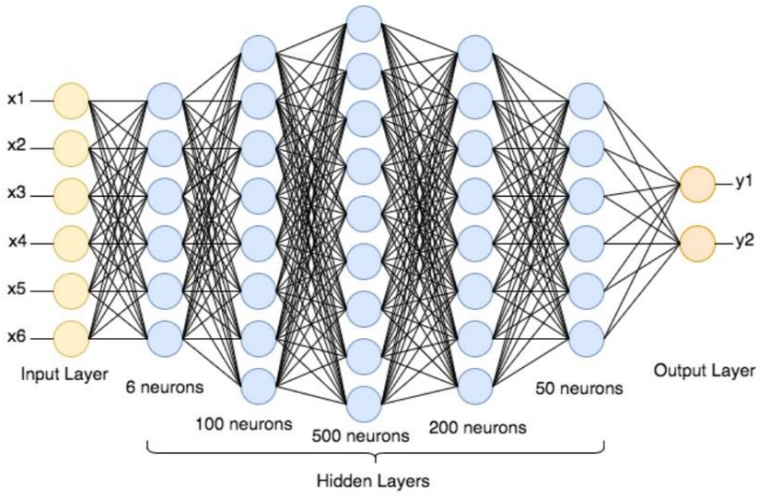

Figure 1 - Example diagram of a neural network. Each circle is a neuron, and the black lines represent a dense connection between adjacent layers, where every neuron in layer n can affect every neuron in layer (n+1)

So, what exactly do we know about how these models work and why they have suddenly become much more “intelligent”? Concerningly little. Even when the models were much smaller, we were having trouble understanding them. Dr. Dean Pomerleau, a senior researcher at Carnegie Mellon University, became familiar with this problem long ago as a graduate student. He was working on a very early autonomous vehicle system that used neural networks to process inputs from mounted cameras and make decisions about how to drive the car. During one of his tests, everything went well until the car approached a bridge and swerved suddenly, almost crashing the vehicle. If the model weren’t an undecipherable “black box,” Pomerleau could have inspected its internal logic and determined which factors it considered relevant for its dangerous decision. Instead, he had to test extensively from the outside to find out the perception model was too reliant on the road being lined with grass. Compared to the billions of parameters, dozens of layers, and complicated recurrent behavior of today’s deep learning models, this network only had a few thousand weights and one hidden layer in a simple feed-forward architecture.

If it was important to know why Pomerleau’s model almost crashed the car when approaching a bridge, it’s critical now to understand the AI models being deployed today as they take on increasingly important tasks. Whenever the latest models threaten users or state falsehoods with confidence, researchers are largely unable to pinpoint what part of the model or training procedure failed. Fortunately, the field of explainable AI (XAI) and related concerns about AI safety are increasingly taking hold. The goal of explainable AI is in the name, where researchers

attempt to develop tools to test and understand AI models or develop AI models that can explain their own thinking. As an IBM article on the topic points out, the benefits of XAI are many—speedier development, increased (well-deserved) trust in the systems, and lower costs trying to patch away perpetuated stereotypes. It's certainly an investment, requiring more upfront development cost, but it pays dividends when diagnosing an AI's mistakes or bad behavior is straightforward. Otherwise, it's like trying to teach someone who doesn't speak the same language how to play chess. Despite the benefits of explainability, other researchers and business leaders are worried it isn't sufficient to justify diverting resources away from increasing the capability of these models. Indeed, companies and countries are racing to stay on top of progress, beginning a so-called AI arms race. Some top leaders in AI have gotten so worried as to call for a pause on AI development altogether until our safety mechanisms catch up. They are worried that an AI with access to computing power to improve itself and communications to manipulate people could suddenly become too intelligent to control, leaving us humans entirely at its mercy. It's difficult to assess whether concerns about the potentially catastrophic consequences of unregulated AI development are realistic or overblown, but two things are certain: (1) it's hard to stop what you don't understand, and (2) we don't understand AI.