www.pipelinepub.com
Volume 18, Issue 8

What is High-Efficiency AI?

By: Francisco Webber

Despite their popularity, steam-powered cars had a very short life. Why? Because they needed up to 30 minutes to start. They were rapidly replaced by automobiles with internal combustion engines and electric starters. Automobilists abandoned their beloved steam cars because they wanted efficiency. The shift toward electric cars we are experiencing in the first decades of the 21st century is motivated by their higher energy efficiency compared to conventional cars—60 percent versus 20 percent. The pattern is the same: it is all about efficiency. Efficiency is a key driver of innovation.

Astonishingly, there is a whole industry segment that is developing in the exact opposite direction: information and communications technology (ICT) in general, artificial intelligence (AI) in particular. While all major industries strive to reduce their carbon dioxide emissions and become more energy-efficient, the energy consumption of computing devices keeps growing. It is already equal to global air transport (4 percent) and is expected to reach that of global automobile transportation (8 percent) by 2030. This trend seems irrational but needs to be considered within context.

Living in the zettabyte age

1,000,000,000,000,000,000,000. 1 with 21 zeros. This is what a zettabyte is, a number impossible to grasp for our brains. However, this is the order of magnitude of data produced nowadays—sensor data, simulations and measurements produced by machines, but also text generated by humans, like emails, articles, reports, social media posts, and more. While numbers are easy to process because they do not give room for interpretation, human language makes jokes, expresses opinions, and utilizes style elements like metaphors and allegories. Besides the sheer quantity of content, this poses an overwhelming problem for computer systems.

Language can be seen as an open system, in which new words constantly come and enrich the existing vocabulary. According to the Oxford English Dictionary, the English language has 171,146 words—and several thousands are added every year. Some terms are frequent—*the*, *are*, *big*—while others are extremely rare—*biblioklept*, *acnestis*, *meldrop*. According to Stuart Webb, professor of applied linguistics at the University of Western Ontario, if you learn only 800 of the most frequently used word families in English, then you'll be able to understand 75 percent of the language as it is spoken in normal life. But if you want to read a novel or a newspaper, then you should learn 8,000 to 9,000 word families.

This is not very different with computer systems: the more specific the text, the more vocabulary they need to learn. In other words: the larger the training models must be. There's just one major difference: while humans are able to infer the meaning of a new term from its context, computers are unable to understand vocabulary they have never seen. Translated to business applications, it means that statistical systems essentially ignore new terms, with devastating impact on the quality of results.

Cortical.io has conducted a real-world experiment to find out how many examples a model would need to be able to cover 100 percent of the vocabulary found in 5,000 business emails. Results show that even extensive annotation does not guarantee full vocabulary coverage: with 20,000 examples, only 70 percent of the vocabulary was covered. Knowing that enterprise data sets rarely contain more than a few hundred examples, the limitations of current approaches are obvious. Statistics cannot fully describe language. However, current AI models are still fed with tons of statistics in the false hope that the more gigantic the models, the better the quality.

Brute force is not the answer

Language variety, variability and ambiguity are real challenges for computer systems, which can only do one thing: crunch numbers. It has long been thought that feeding AI systems with numbers derived from language—the so-called statistical approaches—would compensate for their lack of understanding the actual meaning. This has led to monster models like GPT-3 or BERT whose inflationary sizes—from billions at the beginning to hundreds of billions now—have begun raising concerns about their sustainability. These approaches have led to what could be described as a "million model universe": models are extensively trained to solve a very specific problem in a specific context and in a given language. In other words, these models are local. Each new problem requires another model, leading to a highly fragmented environment: the million model universe. No network effects can be generated in such an environment. Their efficiency in terms of replicability is zero.

However, this is not what the *MIT Technology Review* has in mind when describing these models as the "exhilarating, dangerous world of language AI." Rather, experts are referring to their gargantuan demand for computing power and their very high carbon footprint. They also point to the negative impacts at a society level: biases in the language models lead to discrimination in the way bank loans are granted or jobs attributed; the difficulty in accessing true information in an ocean of data facilitates fake news; the proliferation of data about consumers, which are collected greedily to perfect models is an invitation to populists and demagogues to misuse them.

A desperately congested highway

Millions of ants trying to enter their anthill altogether through a single hole—this is the kind of bottleneck the exponentially growing quantity of data generates with existing computing architectures. Known as the Von Neumann bottleneck, the problem comes from the architecture design that forces program memory and data memory to share the same bus and limits throughput and processing speed on large amounts of data. Next to using caches, parallel computing is one of the workarounds, but it comes at extra costs in terms of execution time and power consumption.

It looks like our computing infrastructure must be completely rethought to cope with the data tsunami. Quantum computing is one of the promising paths for the future. Leveraging the proprieties of quantum theory like the superposition of particles, quantum computers are expected to perform the order of magnitude parallel calculations necessary to survive in the zettabyte age. At least, this is what the theory and lab experiments promise. We'll have to arm ourselves with patience before seeing quantum computing hardware installed in real-world environments. The most optimistic experts estimate this won't happen until 10 years from now.

This is why other, more prosaic approaches are needed. Hardware acceleration is booming, with a plethora of AI-optimized chips designed to process large blocks of data in parallel. All major tech companies like Microsoft, Google, Amazon, Apple, and Tesla are working on their own AI processors—see Google's TPU (Tensor Processing Unit), developed specifically to speed neural network machine learning, in particular their own TensorFlow software. All in all, these are important steps to help us tame the data flood. But improving the hardware is like resolving only half of an equation with two unknowns. One needs to look at optimizing the software, too.

Taking lessons from nature

Why does the human brain use a mere 20 watts to reason, analyze, deduct, and predict, while an AI system like IBM Watson needs 1,000 times more power to perform complex operations? What is it that makes our brains so much more efficient in processing information? Despite all advancements in neurosciences, we still don't know. We can replicate the structure of the brain in a processor—the so-called neuromorphic chips—but we can't emulate how it works.

One theory developed by Jeff Hawkins, the founder of Numenta (as well as Palm Computing and Handspring, where he was one of the inventors of the PalmPilot and Treo), advocates that the brain uses a single representation format to process any kind of information, be it sound, image or language. He calls this representation a Sparse Distributed Representation (SDR) and describes its advantages in terms of efficiency and resiliency in his book *On Intelligence*.

Semantic Folding is a method for natural language understanding based on SDRs, focusing on the representation instead of statistics. In this method, text is converted in semantic fingerprints—binary 2D vectors sparsely filled with active bits that are distributed in such a way that those representing similar meaning are placed close to each other. Rendered as an image, a semantic fingerprint looks similar to brain imaging pictures. When exposing people to concepts, one can

see in fMRI that the same areas in their neocortex are activated. In other words, people have similar representations of similar concepts in their brains.

In Semantic Folding, analogies can be easily calculated by using simple Boolean operations on text. By comparing the overlap of their semantic fingerprints, terms can be immediately disambiguated. The beauty of this approach is that it requires ten times less training material to build a use-case specific language model than when training a custom model on top of BERT or GPT-3. This is because the text analysis happens at the semantic fingerprint level and leverages analogy. In other words, the accuracy of a Semantic Folding-based system is not affected by "unknown" words (words that are not contained in the training set), because the system infers the meaning of a document based on its similarity with other documents.

The fact that semantic fingerprints are sparse vectors (compared to the dense vectors with floating points used in Transformer models) results in immediate efficiency gains in terms of computing power—you need one to two hours on a laptop to compute a model. This is essential for applied AI in a business context. And it's essential if we want to reduce the carbon footprint of AI.

There is no way around high-efficiency AI. Reinventing computing architectures is a first step. Replacing brute force and billion-data-AI models with more efficient software is next.