# Recapping (Tiny) ML and Its Data-centric Computing Role

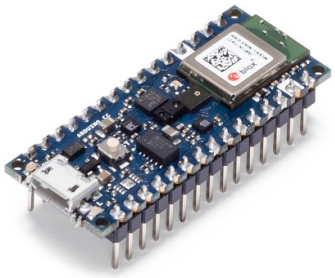
In Course 1, we have covered many of the fundamental topics of applied machine learning. We also looked at the broader computing ecosystem through the lens of an embedded platform where TinyML will be deployed. We start with these basics because they lay the groundwork for building TinyML applications.

Hopefully, the power of supervised machine learning is now apparent: supervised machine learning provides us with the ability to develop **data-driven function approximations**. These approximations can be applied to essentially any data structure: images, sounds, sensor outputs, etc. As a primer for TinyML, it is helpful to understand some of the historical context of machine learning.

## How Today's Computers Compare to the Past?

Machine learning has been around for decades. In fact, most of the algorithms in use today were developed in the 1980s and 1990s. Why were these algorithms not in widespread use at these times? The main bottleneck was a lack of available data and computing power. During these periods, floppy disks with a capacity of 1 MB were commonplace, and a state-of-the-art personal computer might have had the processing capacity of an Arduino. Given the space- and time-complexity of these algorithms, it was largely infeasible to run them on anything but small sets of data.

Over the course of time, improvements in processing and storage technologies, as well as rapid reductions in the cost of memory, have significantly reduced the barriers for performing machine learning. In the 2010s, it has become feasible to run machine learning models on local machines using one or more cores within the central processing unit (CPU). The demonstrable power of these techniques on larger datasets became apparent following the superior performance of AlexNet in the ImageNet Large Scale Visual Recognition Challenge in 2012.



An Arduino with a ARM Cortex-M4 processor in 2020 is 150,000x faster than 1946 ENIAC supercomputer, 60x faster than a 1990's laptop and 4x faster than a 2007 iPhone.

**Typical compute specifications**

Processor speed: 1 MHz ~ 400 MHz

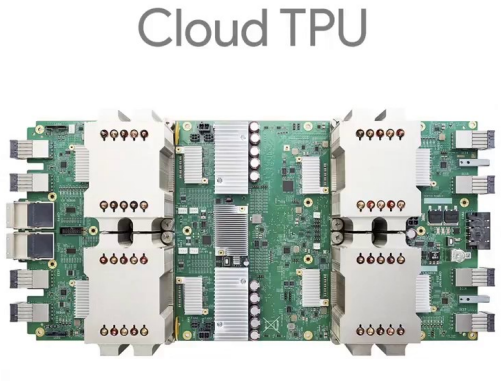
Main Memory: 2 KB ~ 512 KB

Flash Storage: 32 KB ~ 2 MB

Power: 150 uWatts ~ 23.5 mWatts

Shortly after, computation using graphics processing units (GPUs) became necessary to handle larger datasets and became more readily available due to introduction of cloud-based services such as SaaS platforms (e.g., Google Colaboratory) and IaaS (e.g., Amazon EC2 Instances). At this time, algorithms could still be run on single machines. Cloud-based services provide a convenient repository to localize data storage close to vast computing resources, a paradigm commonly referred to as the compute-centric paradigm.


GPUs unleashed deep learning.



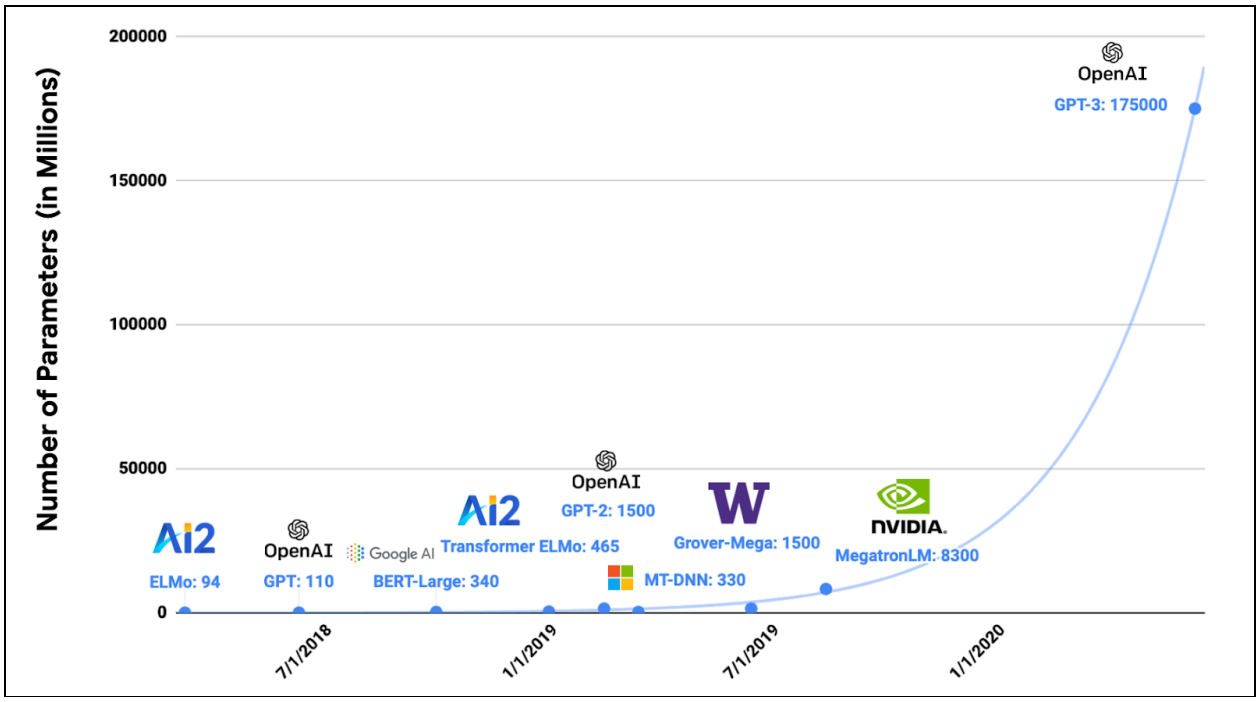

Companies like Google are building custom machine learning hardware to meet user demands.

More recently, we have seen the development of specialized application-specific integrated circuits (ASICs) and tensor processing units (TPUs), which can pack the power of ~8 GPUs and are readily available via cloud-based services. These devices have been augmented with the ability to distribute learning across multiple systems in an attempt to grow larger and larger models.

## Evolution of the Machine Learning Algorithms

During 2020, we have seen the release of Turing-NLG and GPT-3 natural language models, the two largest machine learning models ever created. Natural language processing (NLP), or the ML task that focuses on teaching machines to handle human language is remarkably demanding. In March 2020, Turing-NLG was released, boasting a model size of 17 billion parameters, 10x larger than GPT-2 (at the time the largest model in existence). The insatiable demand for NLP model complexity is driven by the desire to enable a wide variety of applications such as question and answering, summarizing text, improving user experience with personal assistants such as Alexa, generating text to aid in sentence completion and so forth.

The record set by Turing-NLG was short-lived. In May 2020, GPT-3 was released, a model with a whopping 175 billion parameters. Some estimates claim that GPT-3 cost around $10 million dollars to train and used approximately 3 GWh of electricity (approximately the output of three nuclear power plants for an hour). Nowadays, training a single machine learning model can emit as much carbon as five cars. The GPT-3 model size is 45 TB, far beyond the capabilities of any single machine.




While the achievements of GPT-3 and Turing-NLG are laudable, naturally, this has led some in the industry to criticize the increasingly large carbon footprint of the AI industry. The unrelenting and exponential increase in model size would seem to suggest that models in the future will only get larger.

## The Role of TinyML

So where does TinyML come into the equation? The key to this answer is in the data, so to speak. The data used in machine learning applications are derived from a data source. This is often a camera, microphone, or some form of sensor tasked with capturing information about the physical world and transducing it into a format digestible by computers. For the past few years, there has been a surge of such devices connected to the cloud, colloquially known as the internet of things (IoT).

These IoT devices periodically send data over the internet to the cloud and are often stored in a data warehouse which can easily be interfaced with via a cloud instance. Such warehousing of data from heterogeneously distributed devices requires a large amount of data to be transmitted through networking protocols from the IoT devices to the cloud. Some individuals have raised concerns with this infrastructure, namely related to (1) privacy, (2) latency, (3) storage, and (4) energy efficiency.

- **Privacy.** Transmitting data creates potential for privacy violations. Such data could be intercepted by a malicious actor and becomes inherently less secure when warehoused in a singular location (such as a data warehouse).
- **Latency.** For standard IoT devices, such as Amazon Alexa, these devices transmit data to the cloud for processing and then return a response based on the algorithm's output. In this sense, the device is just a convenient gateway to a cloud model, like a carrier pigeon between yourself and Amazon's servers. The device is pretty "dumb" and fully dependent on the speed of the

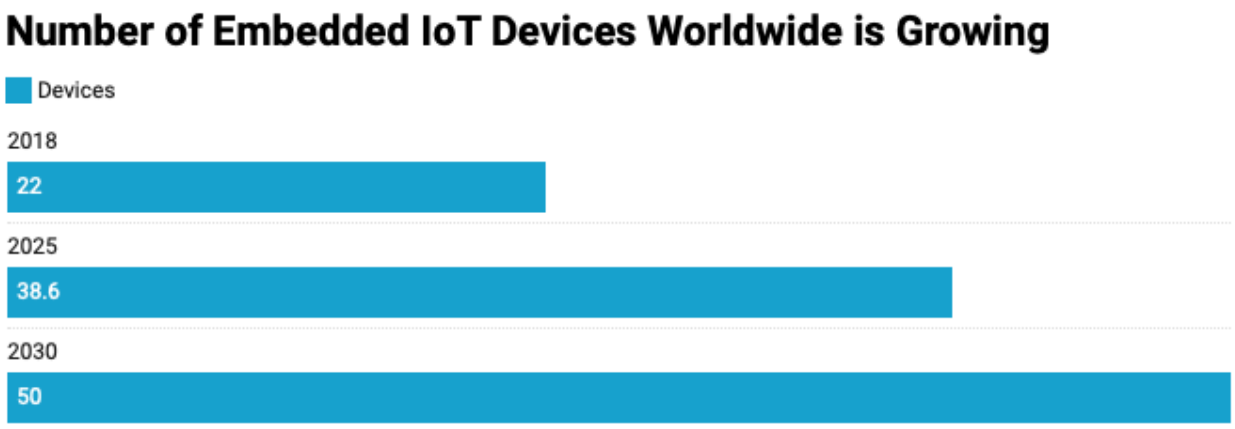
internet to produce a result. If you have a slow internet connection, Amazon Alexa will also become slow.

- **Storage.** For many IoT devices, the data they are obtaining and transmitting is of no merit. Imagine a security camera recording the entrance to a building for 24 hours a day. For a large portion of the day, the camera footage is of no utility, because nothing is happening.
- **Energy Efficiency.** Transmitting data (via wires or wirelessly) is energy-intensive, around an order of magnitude more so than onboard computations (specifically, multiply-accumulate units).

By empowering embedded systems with the ability to perform on-device machine learning, these problems are largely solved. IoT systems that can perform their own data processing require no data transmission, and are thus highly energy-efficient. Such devices would have minimal latency because there is reduced (if any) dependence on external communications. By keeping data primarily on the device and minimizing communications, security and privacy is improved. Additionally, for an intelligent system that only activates when necessary, lower storage capacity and fewer external communications are required. The industry term for such an intelligent IoT device is an **edge device** (devices at the "edge" of the cloud).

## The Future of TinyML

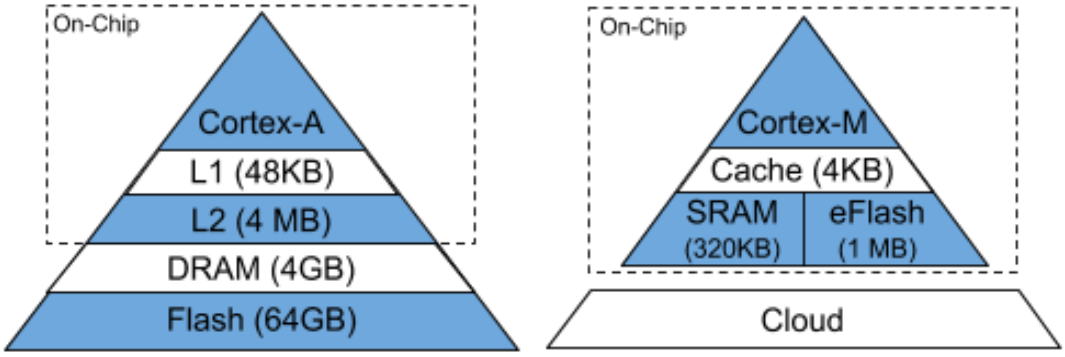
As IoT devices become increasingly ubiquitous, the above-mentioned concerns are exacerbated. Billions of IoT devices are already in use, ranging from smart fridges and clocks, to thermostats, alarm systems, locks, and even children's toys. At the start of 2020 there were roughly 8 billion active IoT devices. By 2030, it is anticipated that we will have nearly 50 billion IoT devices.




Too many devices communicating over our networks simultaneously can quickly saturate the bandwidth, and present many attack vectors for hackers and malicious actors. These are some of the factors that have stimulated interest in an alternative computing paradigm for edge devices, the **data-centric paradigm**. This paradigm flips the compute-centric paradigm on its head, taking the computing power to the data, instead of the data to the computing power.

This brings us to a similar situation with that of the algorithmic developers of the 1980s and 1990s - we have our algorithms but are constrained by the hardware characteristics of microcontroller platforms, which rarely have flash memory or static RAM above 1 MB. The hierarchy of the computing system is shown below and there are vast differences (that we will learn a lot more about in the next courses).

However, this situation is not exactly the same. Nowadays, we have access to our large scale models, they are just too big!




Traditional computing systems versus embedded machine learning systems.

The models used on edge devices need not necessarily be trained on edge devices - they can be trained in the cloud and subsequently ported to the edge device. The edge device need only perform model inference, which is significantly cheaper computationally. In many ways, running machine learning on an embedded device is similar to any production-ready machine learning model.

However, this is by no means a simple feat to achieve. Most microcontrollers do not have an operating system, and thus bare-metal and lightweight inference libraries (e.g., TFLite Micro, TVM, CMSIS-NN) are required, with essentially all unnecessary functionality (including debugging) removed. Similarly, model weights and computational graphs must be simplified to reduce their memory and computational overhead. This involves techniques such as model distillation, pruning, and quantization. This is the essence of TinyML.

Myriads of machine learning applications exist for edge devices, which are often located in remote areas which have sparse access to power and minimal networking capabilities. Edge devices constantly communicating via wireless transmitters consume approximately 10mW, whereas devices running TinyML models and only communicating when something of interest happens can function on less than 1mW, allowing them to run on a small coin battery continuously for more than a year.

Combined with their low-cost, this presents countless opportunities for industrial applications. Sensor networks to monitor air quality throughout cities could be developed at relatively low cost and provide widespread coverage, focusing communication only in conditions of poor air quality. Similarly, industrial processes could be monitored for preventative maintenance purposes, reducing costs and mitigating hazards. Sensors could be placed along long-distance oil and gas pipelines to monitor for leaks or potential blockages. Intelligent wearables could allow end users to gain a better understanding of their own body, or to mitigate or solve medical problems such as neural communications or muscle atrophy.

We are at an interesting crossroads where machine learning is bifurcating between the two computing paradigms of compute-centric and data-centric computing. Although we appear to be quickly moving towards a ceiling in the compute-centric paradigm, it is clear that work in the data-centric paradigm has only just begun.