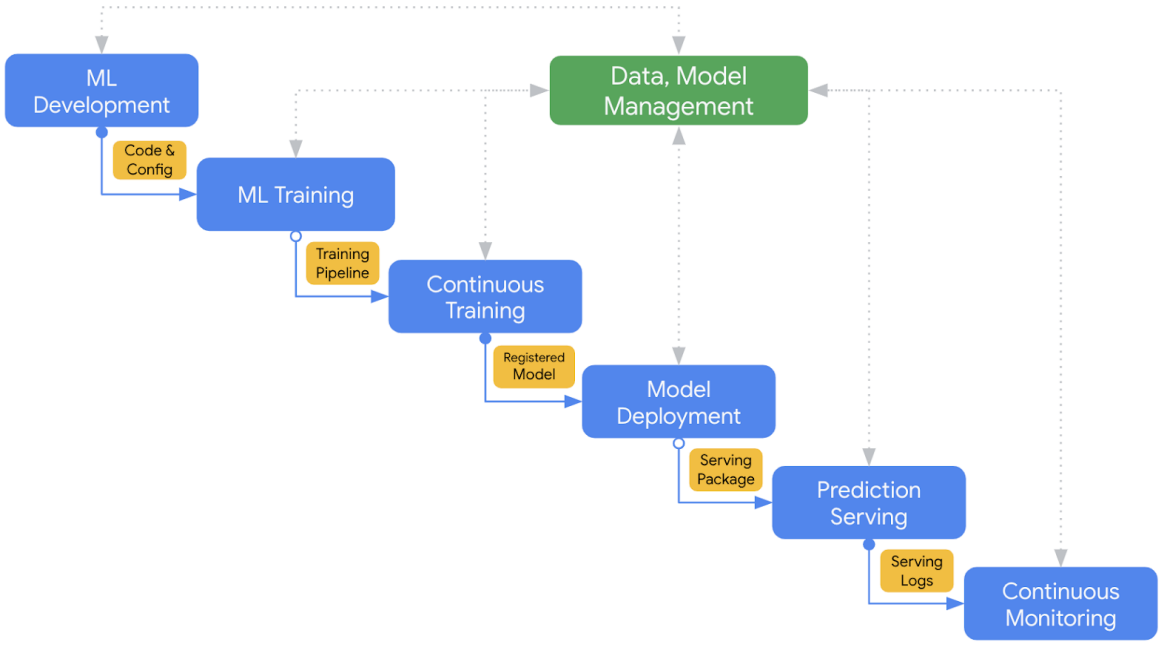
Summary of What You'll Learn

About This Reading

In this course, we are going to introduce you to MLOps from a TinyML perspective. Lots of ideas relevant to MLOps apply to both larger-scale applications as well as small-scale applications.

We refer to the traditional larger-scale applications as "BigML" in contrast to the TinyML that we have already familiar with in the context of embedded computing systems.

MLOps has become an integral part of machine learning in industry and performs analogous functions to DevOps in software development, except it focuses on both code *and* data.

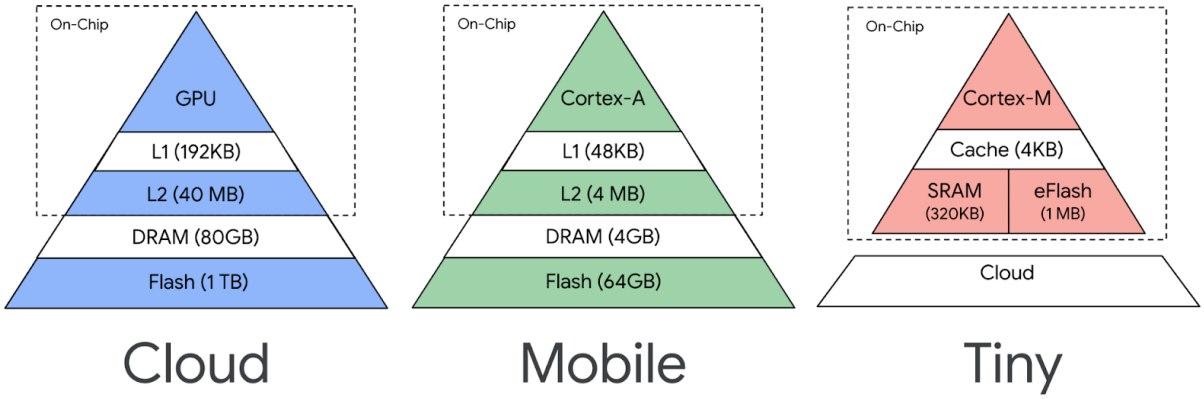


Overview

At a high level, the differences between BigML and TinyML are threefold: hardware capabilities, software capabilities, and network capabilities.

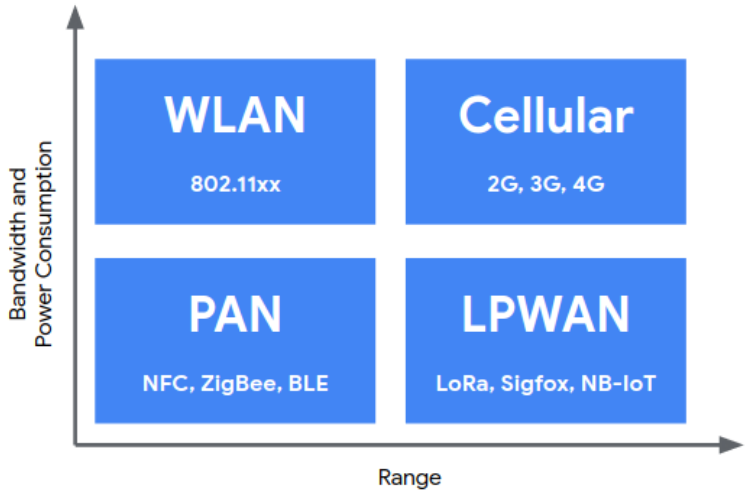
- **Hardware**: TinyML works with resource-constrained devices that have limited hardware resources to perform functions. This means that we have to be very efficient with our implementations and minimize resource usage in terms of memory, compute, and communication. In the cloud, we have access to very large amounts of computing resources and data storage, with large caches available on-chip for performing computations. In contrast, mobile devices using processors such as the Cortex-A are

more constrained than the cloud (as shown in the figure below), but they still contain relatively large amounts of data storage along with a memory management unit (MMU) for storing memory off-chip. TinyML is even more constrained than mobile ML, having all the memory (SRAM, cache, and eFlash) on-chip, no MMU, and a very small amount of resources available. This presents additional challenges for realizing MLOps.



- **Software**: A corollary of the hardware constraints is the software constraints - the minimal resources available within TinyML means that we are unable to run any software other than things like extremely lightweight clients. This means that traditional software used for MLOps within the realm of BigML cannot feasibly be used, like Docker, Ansible, Kubernetes, Git, and TensorFlow. This brings about constraints that suggest we will probably need a whole new set of tools to perform MLOps in TinyML. TinyML systems are also more heterogeneous than their BigML counterparts, leading to more heterogeneous software operating environments.

- **Network**: Because communication between devices requires a lot of power, the network capabilities of TinyML devices are also more constrained than in BigML and must be treated more conservatively or use more power-efficient wireless technologies for communication. Wireless technologies such as NB-IoT and LoRa are more preferable to the standards used in mobile devices such as 3G and 4G.



The table below summarizes the differences between BigML and TinyML in the context of (1) compute, (2) software, and (3) network capabilities.

BigML	**TinyML**
Big compute hardware and memory resource availability	**Small compute** hardware and memory resource availability
Homogenous software operating environments	**Heterogeneous** software operating environments
Stable network and communication capabilities	**Unstable network** and communication capabilities

The Ops corresponding to each of these paradigms (BigML vs. TinyML) is correspondingly different in their characteristics, although the overall procedure remains largely unchanged. TinyMLOps is associated with resource-constrained edge devices, small models, power consumption, and fewer capabilities than BigML systems, such as logging and debugging. Conversely, traditional MLOps is associated with powerful devices, large models, and a large number of capabilities. The differences are summarized in the table below.

MLOps	TinyMLOps
Models are trained and deployed on **powerful** devices	Models are deployed on **resource-constrained** edge devices
Large models with ***multiple*** supported hardwares and Ops	**Small** models with ***few*** supported hardwares and Ops
Accuracy and **availability** are the important metrics that matter	**Latency**, **throughput**, **power** consumption are more important
Containerization, CI/CD, logging, and monitoring is **required**	No containers; logging and monitoring is **difficult** to do
Performance checks and model updates are done **regularly**	Performance checks and model updates are **difficult** to do
Robustness and **autoscaling** to workload traffic	Scaling is **difficult after deployment**; Robustness is important

What You Will Learn

In this course, we will build upon the topics discussed above with the aim of answering the following questions:

- What is (Tiny) MLOps?
- Why is (Tiny) MLOps necessary?
- How do we automate (Tiny) MLOps?
- What are (Tiny) MLOps platform features?
- What are some examples of (Tiny) MLOps platforms?

By the end of this course, you will understand what TinyMLOps is, why it is necessary, how it works, examples, and the stages involved in developing a TinyMLOps workflow. Let's get started!