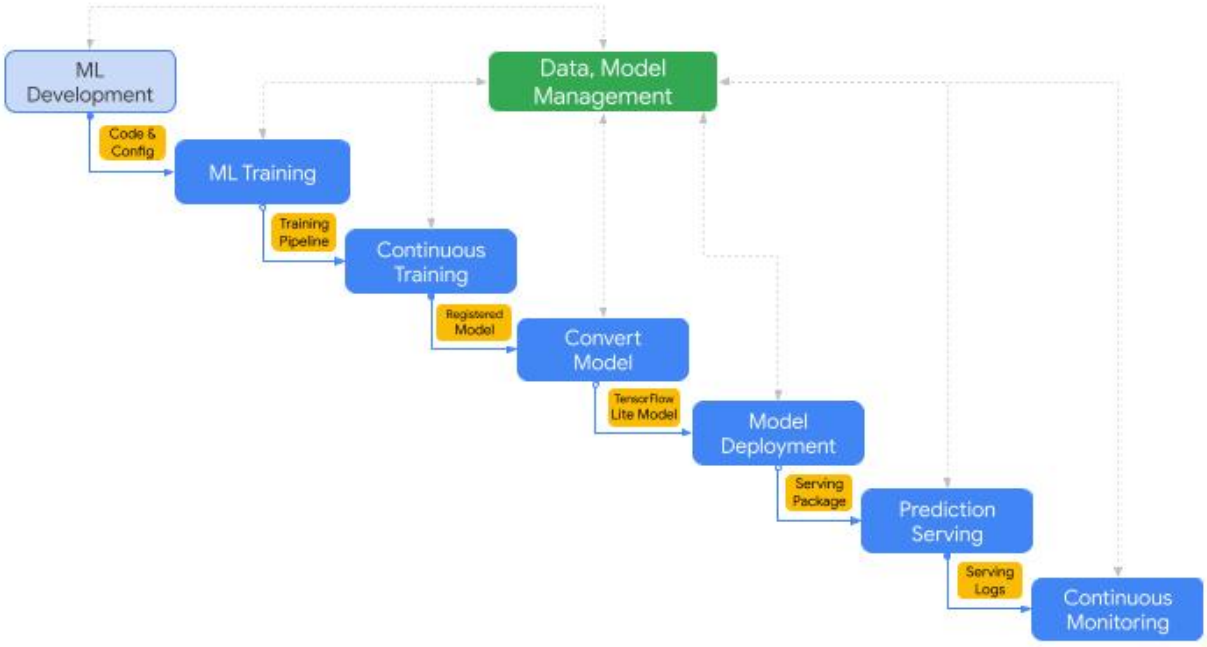
ML Development



Objectives

The first stage of the MLOps pipeline is ML development. This stage should be somewhat familiar to learners who have taken TinyML Course 1--3 (since we have been doing this repeatedly since Course 1)! But don't worry if you are taking Course 4 alone, you aren't missing out. ML development involves the development of a model that is able to meet the performance criteria of a well-defined ML use case. This requires answering several questions such as:

(1) What is the task?
   - E.g., keyword spotting, wake word detection, object recognition

(2) How can we measure business impact?
   - Will our model increase sales, reduce inefficiencies in business processes, increase security?

(3) What is the evaluation metric? And are these metrics measurable before deployment?
   - Sales might require measuring performance before and after deployment, while image classification can be readily evaluated through a model's test accuracy

(4) What data is needed? And does this data already exist?
   - Some features may be fundamental for developing an effective model, and in some cases, we may have no or minimal data and may need to procure data or simulate it in some cases.

(5) What are the training and serving requirements?
   - Some models might need to be updated daily, weekly, or monthly

    - Some models may require high inference rates (e.g., 10 inferences per second)
    - For TinyML, a key serving requirement we have is that the model must be small enough in memory to fit on our resource-constrained device.

In this section, we will look at these questions all structured in a systematic way, organized as follows: Problem Definition → Data Selection → Data Exploration → Feature Engineering → Model Prototyping → Model Validation → Model Evaluation → (repeat)

Experimentation Is Key

A key aspect of ML development is **experimentation**. Experimentation is split up into the following tasks: data discovery, data preparation, and model prototyping. From the model perspective, this process involves trying different machine learning models (not just neural networks!), different model architectures, hyperparameters. From the data perspective, this involves deciding which features are most relevant, performing feature engineering, and determining whether the marginal gain in performance of including a specific feature outweighs the costs to onboard memory and compute operation.

Keeping track of the impact of all of these items can be overwhelming, so often experiment tracking is used to simplify the process. This involves keeping track of all of the features, hyperparameters, and models being used for a given task and comparing the relevant performance metrics. Experiment tracking is very useful during the prototyping phase and determining which model should be selected for the final project that will be sent for model validation and potential deployment.

Where Do MLOps Fit In?

MLOps can help with the model development process in various ways. For experiment tracking, we can save our models and all of their relevant information in a model registry so that we can easily reference and compare them. If a model needs to be updated or retrained on a regular basis, this can be set up using a continuous integration/continuous delivery pipeline. This procedure is commonly done in the software development world. For example, TensorFlow runs nightly builds every evening (tf-nightly) that include all-new feature additions and ensure that the key functionality of the package remains intact using a cron-like scheduler. This feature requires the model pipeline to be integrated with a version control system such as GitHub but allows multiple individuals to work on and update the model infrastructure simultaneously, making the development process more efficient and collaborative.

We will delve into the ML development process in more detail for the rest of this chapter and we will use Keyword Spotting as the TinyML application use case to present the MLOps details.