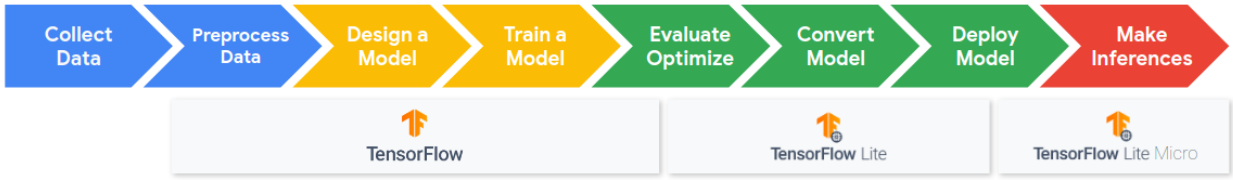
Summary

In this section you explored the TinyML flow and data engineering in the context of anomaly detection applications, focusing on some unique challenges presented by unsupervised learning. Anomaly detection represents a common TinyML use case of classifying some time-series data as ‘normal’ or ‘abnormal’. Anomaly detection can be applied to data coming from a wide variety of sensors in a number of different application domains.



Anomaly Detection in Industry

You explored an example of anomaly detection in an industrial setting where you learned about the primary constraints of tinyML anomaly detection. First, anomaly detection systems must be fast enough to keep up with the rate at which sensors produce data. Often the sensors produce too much data to be sent over a wireless network therefore computation must be done locally. To accomplish this, the anomaly detection model must fit within the tight memory constraints of a microcontroller. Finally, an anomaly detection system must mitigate false negative and false positives, which, depending on the application, can have dire consequences.

Data and Datasets

Anomaly detection is unique in that it is impractical to collect real examples of anomalous data. Collecting data on a failure is often too expensive since it usually involves intentionally breaking something. Additionally, anomalies are inherently rare and can take many different forms therefore it is difficult to train anomaly detection models in a normal supervised manner. That all said, you explored the MIMII and ToyADMOS datasets as well as the concept of generating synthetic data to understand some of the processes used to try to collect an anomaly detection dataset.

Unsupervised Learning: K-Means and Autoencoders

To avoid some of the issues with supervised learning you then explored a classical and neural network based approach to unsupervised learning. For the classical approach, you explored anomaly detection through the lens of K-means clustering and learned about its advantages (ease of implementation and use) and disadvantages (scalability). You then learned about autoencoders and how they can be applied to many domains, among which is anomaly detection. Autoencoders are trained to reconstruct normal data after compressing it into an embedding. By comparing the error between the input and the output of the autoencoder we are able to determine how similar the input is to our normal training data. We perform anomaly detection by selecting a threshold above which we classify the input as an anomaly. You also learned about how the model architecture impacts the autoencoder and how metrics like ROC and AUC help us evaluate our performance. Critically, you learned that anomaly detection models often have low transferability and therefore have to be trained for their specific application.