The Role of Sensors in TinyML Applications

Sensors are important for machine learning applications as they act as a vehicle between algorithmic resources and characteristics of the physical environment. Today there is a continued push towards low-cost and high-performance sensors, especially for applications such as air quality monitoring and predictive maintenance. There are two main types of sensing: *in situ* and remote sensing.

In situ sensors come in numerous forms and each have a particular sensing mechanism. For example, the sensing mechanism may involve chemical reactions, light, or measuring voltages from material changes caused by stress. *In situ* sensors are often characterized as physical sensors and are constrained to measure local attributes at the location of the sensing element. Remote sensing methods work at a range, and are almost always optical techniques. For example, temperature can be measured at a distance using an infrared thermometer, and gas concentrations in the atmosphere can be measured by columnar measurements made by satellites in low Earth orbit. *In situ* sensors can use optical mechanisms, but are often distinguished because they only measure local attributes.

In TinyML, we are typically interested in sensor data in the form of an image or time series, enabling us to perform spatial or time series analysis using machine learning models. The utility of sensors for tiny machine learning allows real-time feedback and monitoring of the local environment, either at the point of the sensing element or at a distance using remote sensing methods. For example, an array of *in situ* sensors each with gas sensing elements can be used to monitor pollution levels across a university campus, or surrounding an industrial facility. Similarly, an example of a remote sensing method might be a camera used to monitor local weather conditions based on the appearance of overlying clouds - perhaps useful in locations with limited internet access.

For a sensor outputting a time series, the sensor measurement can be used as a single feature in a machine learning model. A segment of the time series might also be used for classification or anomaly detection purposes. For an image-based method, the bitmap can be flattened and each pixel passed as a feature in a neural network.

Sensors can be used individually or can be combined to produce synergistic effects. This provides even more powerful capabilities, especially when combined with online machine learning. Perhaps the most utilized example of this for TinyML currently is the combination of an accelerometer and gyroscope. These devices are often present in smartphones and other embedded devices, and can be used together to differentiate between complex 6-axis movements. These allow your smartphone to know when it has been picked up, or to know when the holder has fallen over. They can also be used in more complex devices such as drones to estimate the local wind speed and direction via how it perturbs from normal operation. Other sensors apart from accelerometers and gyroscopes are also common. Vibration sensors are becoming increasingly common in industrial applications for predictive maintenance. These

sensors are used with the aim to improve machinery uptime by detecting anomalous vibrations and repairing them before a catastrophic failure occurs.

Using sensors for embedded machine learning presents several additional challenges. Firstly, sensors have a finite working life, and the designer should bear this in mind during development. For example, we might not want a complex sensing system to become unusable if a single sensor becomes defective, and we might want it to default to a specific action in the event of a failure. Sensors also tend to exhibit noise and drift, which is often infeasible to remove or model statistically. TinyML systems should be robust enough that they are not adversely impacted by this noise or drift during regular operations. In a similar vein, sensors that are not properly isolated electrically can have the potential to distort other measurements. Finally, the more sensors that are present in a given system, the more complex and power-hungry it becomes. To be an adept user of TinyML, it is not enough to know about machine learning and embedded systems, but also mindful of the constraints imposed by electrical and mechanical engineering challenges.

To overcome the above challenges, the following rules of thumb are recommended:

1 - Collect rich data. In the machine learning world, data is power. A sufficiently large amount of data should be collected to ensure that our TinyML models are able to effectively approximate the distribution of the data. More data never hurts!

2 - Don't over-engineer your system. While it may seem attractive to add fifteen sensors to a system, we should follow Occam's razor when engineering our system that "*entities should not be multiplied without necessity*". Being intelligent and yet parsimonious with our sensor selection will minimize the possibility of running into hardware complications.

3 - Plan to cover all possible sources of variation. The statistical power of our model will always be limited by the data at hand. Consequently, it is helpful to cover as much ground as possible without conflicting with #2. This can involve collecting data that covers edge cases as well as using additional sensors that are expected to provide some additional explanatory power to the model.

4 - Use the highest workable sampling rate. Although it is highly expensive to transmit and store high-frequency data, we limit these issues when we are using TinyML systems since inferences are made on-device and streamlined to minimize storage. It is much easier to downsample data when there is too much than to upsample it when there is not enough. Thus, when collecting data to use for model training, it is recommended to obtain the highest resolution data possible to provide flexibility in downstream data engineering processes.

Regardless of whether you are using sounds, images, vibrations, electrical signals, or other sensor data, these signals can be combined and used to train machine learning models to help model, classify, or predict events at the edge in real-time using inexpensive microcontrollers, which is a very powerful tool indeed.