Edge Impulse: Practical TinyML

Building Real-World Solutions

1 The Power of Local Intelligence - A Self-Driving Car Scenario

Imagine a self-driving car speeding down a highway. Suddenly, an obstacle appears in its path. The car's cameras capture images and sensors collect data, but what happens next?

In a traditional Internet of Things (IoT) system, this data would be sent to a remote cloud server for processing. The cloud would analyze the images, identify the obstacle (a log of wood, a child, or a small animal), and send instructions back to the car: "Slow down!", "Stop!", or "Cross over!".

However, internet connections aren't always reliable, and even a slight delay can be catastrophic. By the time the cloud makes a decision and sends the signal, it might be too late.

This is where **TinyML** comes in. With TinyML, the car itself is intelligent enough to analyze the sensor data and make decisions in real-time. The on-board microcontroller or microprocessor, trained with machine learning, can identify the obstacle and react instantly, without relying on a distant cloud. This dramatically improves safety and responsiveness. That is the power of TinyML in real-world deployment.

In this book, we'll explore how to build such intelligent systems using Edge Impulse, a powerful online platform that makes TinyML accessible to everyone.

2 What is TinyML and Why is it Important?

TinyML is a field of machine learning that focuses on deploying machine learning models on resource-constrained devices. This means running complex algorithms on microcontrollers and other embedded systems with limited processing power, memory, and energy.

Why is TinyML so important?

- **Real-time Decision Making:** As illustrated in the self-driving car example, TinyML enables devices to make decisions instantly without relying on cloud connectivity.
- **Reduced Latency:** Eliminating the need to send data to the cloud significantly reduces latency, which is crucial in applications requiring immediate responses.
- **Privacy and Security:** Processing data locally ensures sensitive information remains on the device, enhancing privacy and security.
- **Offline Functionality:** TinyML-powered devices can operate even without an internet connection, making them suitable for remote or unreliable environments.
- **Energy Efficiency:** TinyML models are designed to be lightweight and energy-efficient, allowing for long-lasting battery life in embedded systems.

3 Introducing Edge Impulse

Edge Impulse is a development platform designed for creating TinyML solutions. It provides an intuitive, low-code environment for building, training, and deploying machine learning models directly onto edge devices.

What makes Edge Impulse special?

- **Simplicity:** Edge Impulse is easy to use, even for those with limited coding experience.
- **End-to-End Deployment:** It supports the entire TinyML pipeline, from data collection to model deployment.
- **Device Compatibility:** Edge Impulse is compatible with a wide range of microcontrollers and edge devices.
- **Real-World Applications:** It empowers the development of practical solutions to real-time problems.

Benefits of Using Edge Impulse:

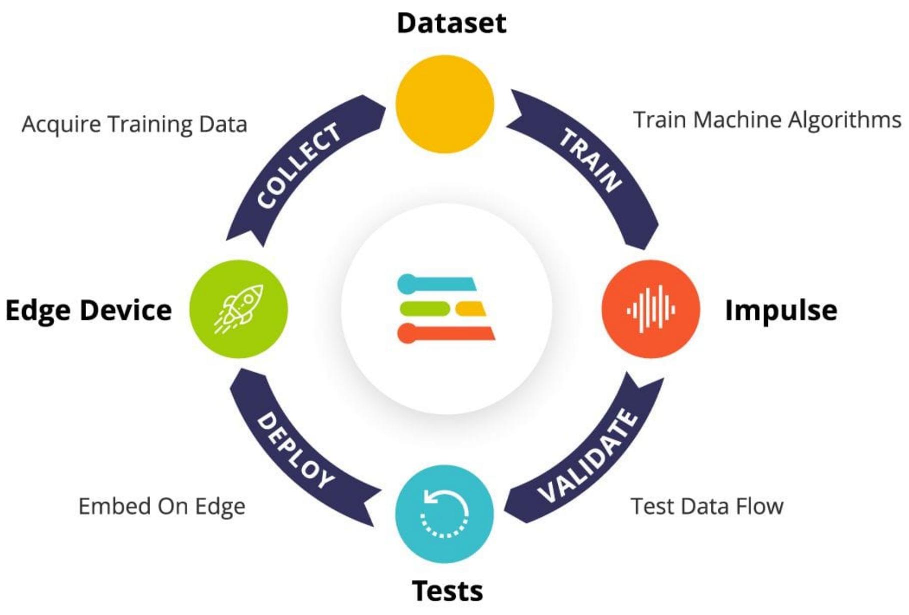

Figure 1: Edge impulse cycle

- **Real-time Data:** Works with real-time sensor data for training.
- **Broad Compatibility:** Supports various microcontrollers like Arduino, ESP32, and Raspberry Pi.
- **Optimized Performance:** Creates lightweight, fast, and functional offline models.
- **Reduced Model Size:** Achieves up to 80% reduction in model size.
- **Low-Code System:** Requires minimal coding skills.

4 Understanding the Edge Impulse Workflow

Edge Impulse follows a structured workflow to guide you through the TinyML development process:

1. **Data Collection:** This is the most crucial step. The quality of your data directly impacts the performance of your model.
2. **Feature Extraction:** This involves extracting relevant features from the collected data using signal processing techniques.
3. **Model Training:** This stage involves training a machine learning model using the extracted features.
4. **Deployment:** Finally, the trained model is deployed to the target edge device.

5 Building an Elderly Fall Detection System: Project Setup

Now, let's put our knowledge into practice and build a real-world TinyML application: an elderly fall detection system. This system will use the sensors in a smartphone to detect when an elderly person has fallen and potentially alert emergency services.

Project Goal: To create a system that can differentiate between a normal movement and a fall using a smartphone's sensors.

Step-by-Step Instructions:

1. **Create an Edge Impulse Account:** Navigate to the Edge Impulse Studio website (`https://studio.edgeimpulse.com/`) and create a free account.

2. **Create a New Project:** Once logged in, click on the "Create new project" button.

3. **Name Your Project:** Give your project a descriptive name, such as "Elderly Fall Detection". Choose a personal or public project depending on your needs. Personal projects have certain limitations.

4. **Connect Your Device (Smartphone):** In the project dashboard, click on "Collect new data". A QR code will be displayed. Scan this QR code with your smartphone's camera or a QR code reader app. This will open a link to connect your phone to Edge Impulse. Follow the instructions on your phone to establish the connection.

   - **Troubleshooting:** If your phone is not detected, ensure it is connected to the internet and that you have granted the necessary permissions to access the phone's sensors.

5. **Verify Device Connection:** Once connected, your phone should appear as a device in the Edge Impulse studio.

6 Data Acquisition: Capturing the Movement

Now that your phone is connected, it's time to collect the data that will train our fall detection model.

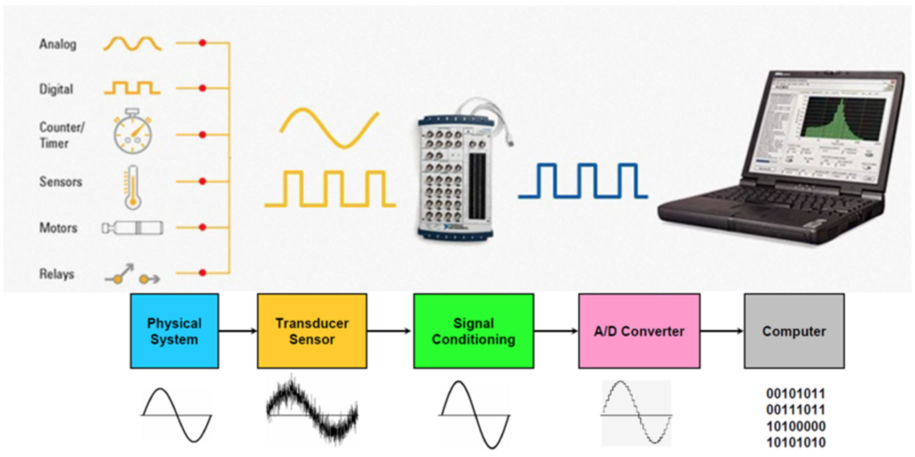

Figure 2: Data Pipeline

1. **Choose Your Sensor:** In the Edge Impulse studio, select "position" as the sensor. This is because the sensor is specifically designed for human movement.

   Remember, accelerometer measures motion in terms of 3-axis. Accelerometer combined with gyroscope is called IMU. The positional sensor uses the 3-axis gyroscope + accelerometer, for determining the human movement.

2. **Create Labels:** We need to create two labels: "safe" and "fall". "Safe" will represent normal movements, while "fall" will represent a fall event. Enter "safe" in the Label field.

3. **Set Sample Length:** Set the sample length to 10000 ms (10 seconds). This means each data sample will be 10 seconds long.

4. **Collect "Safe" Data:** While in the safe label, now simulate normal movements with your phone, such as walking, sitting, standing, and lying down. Keep the phone in your hand as you would normally. While moving like that, click "Start sampling" to begin recording data. Collect at least 20 samples.

5. **Collect "Fall" Data:** Enter "fall" in the Label field. Simulate a fall by moving the phone in a way that represents a fall. This might involve sudden drops and impacts. Collect at least 20 samples.

- It might be best to move your phone above a bed to not incur damage.

6. **Review Data:** Once you've collected enough data, review the samples to ensure they are representative of the desired movements.

7 Data Split: Testing vs. Training

For testing the model, we need to set aside test data. Ideally the data split should be 80% training data and 20% test data.

Click on the select multiple items. For each of the safe and fall data, select at least 4 data rows and click on the move to test data button.

8 Feature Extraction: Making Sense of the Data

With the data collected, it's time to extract meaningful features that the machine learning model can use to distinguish between "safe" and "fall" events.

1. **Create Impulse:** Click on the "Experiments" tab in the Edge Impulse studio. The button "Create Impulse" appears.

2. **Set Window Size:** Set the window size to 2000 ms (2 seconds). This means the data will be analyzed in 2-second segments. The original 10-second interval that we set will be broken into several 2-second windows, within which the model determines whether a fall has occurred.

3. **Add Processing Block:** Now, click on "Add a processing block". The processing block will filter noises from the data so the classification models work with relatively clearer data. Select "Spectral Analysis". Spectral analysis extracts frequency and power characteristics of a signal over time.

4. **Add Learning Block:** Click on "Add a learning block" and select "Classification (Keras)". A classification model learns patterns from data and applies these to new data. In our case the classification model will classify input sensor data as either a "fall" or "safe" action.

5. **Save Impulse:** Click on "Save Impulse" to create the impulse pipeline.

In the next part of this book, we will see how we go through with the remaining steps.