

# EE P 596: TinyML

## Lecture 7: TinyML for Motion Classification

Dept. of Electrical and Computer Engineering  
University of Washington

Instructor: Dinuka Sahabandu ([sdinuka@uw.edu](mailto:sdinuka@uw.edu))

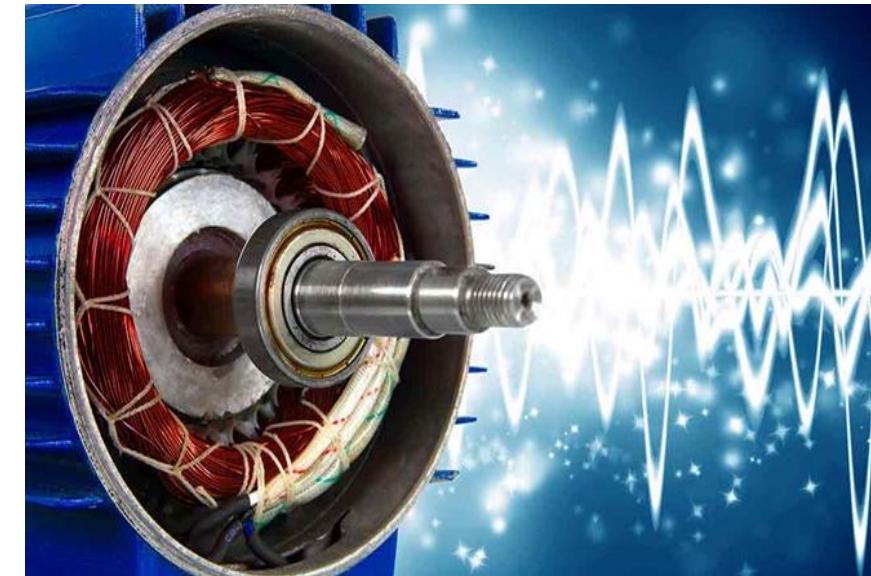


ELECTRICAL & COMPUTER  
ENGINEERING  
UNIVERSITY of WASHINGTON



# Topics Covered

- TinyML Final Projects
- Recent Advancements in ML and Their Relation to TinyML
- Background on Motion Classification and different types
- Supervised learning for motion classification
- Sensor interface for Mechanical Stress in Transport
- Model Training, Evaluation Metrics, and Deployment
- As with the past classes, today we will move away from Tensor flow and use Edge Impulse



❖ Relevant reading from the Textbook [Warden and Situnayake; O'REILLY Publisher] is Chapter 17

# TinyML Final Projects

1. Elderly Fall Detection Using IMU Sensor
2. SmartGuard: TinyML-Based Fall Detection System for Elderly Care
3. Elderly health alert system
4. TinyML-Based Human Activity Recognition Using Wearable Sensor Data
5. Speech to Text Sensor
6. Self-Checkout with Vegetable Recognition
7. Canine Acoustic Detection with TinyML
8. Safe Driver – Drowsiness Detection
9. Basic Hand Gesture Control
10. Gesture Controlled IoT Device
11. Using Shapley Values for Detecting Adversarial Attacks in TinyML Settings
12. Musical Instrument Classification
13. Recognizing Birds via Their Sounds

**Four Groups !!!**

**Two Groups**

# TinyML Final Projects

- Project presentations are **on May 29<sup>th</sup> – from 6pm – 10pm PT**
- What type of hardware to use?

	Embedded Device	Tiny Device
<b>Compute</b>	 1 GHz – 4 GHz	<b>1 MHz – 400 MHz</b>
<b>Memory</b>	 512 MB – 64 GB	<b>2 KB – 512 KB</b>
<b>Storage</b>	 64 GB – 4 TB	<b>32 KB – 2 MB</b>
<b>Power</b>	 30 W – 100 W	<b>150 µW – 23.5 mW</b>

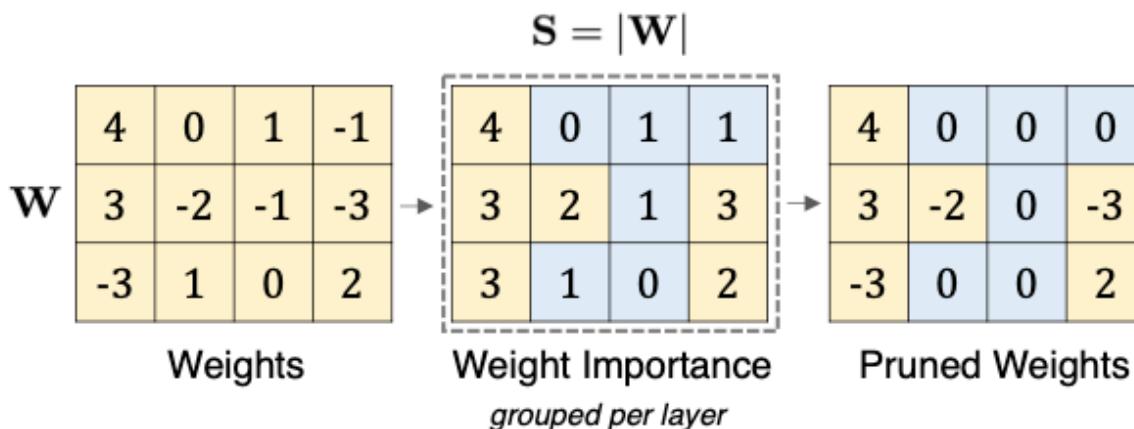
A red rounded rectangle highlights the "Tiny Device" column. A blue arrow points from the "Memory" row in the "Tiny Device" column to the text "Up to 128 MB Memory is Okay".

Explore your hardware (if needed): <https://www.arduino.cc/en/hardware#nano-family>

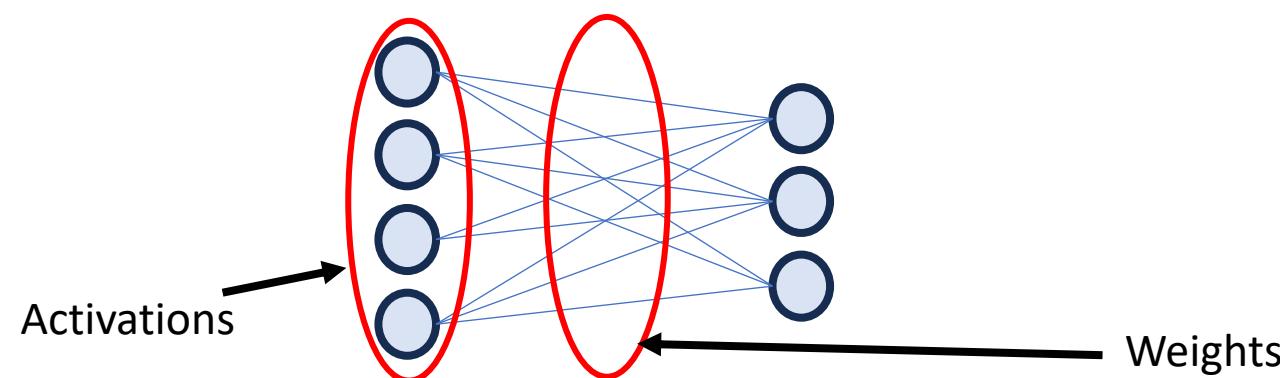
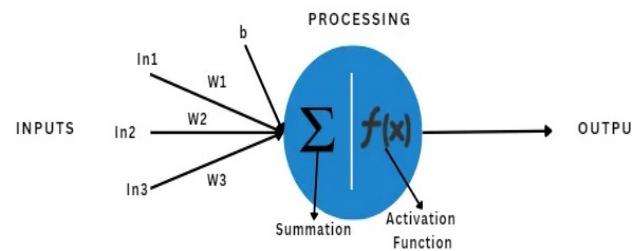
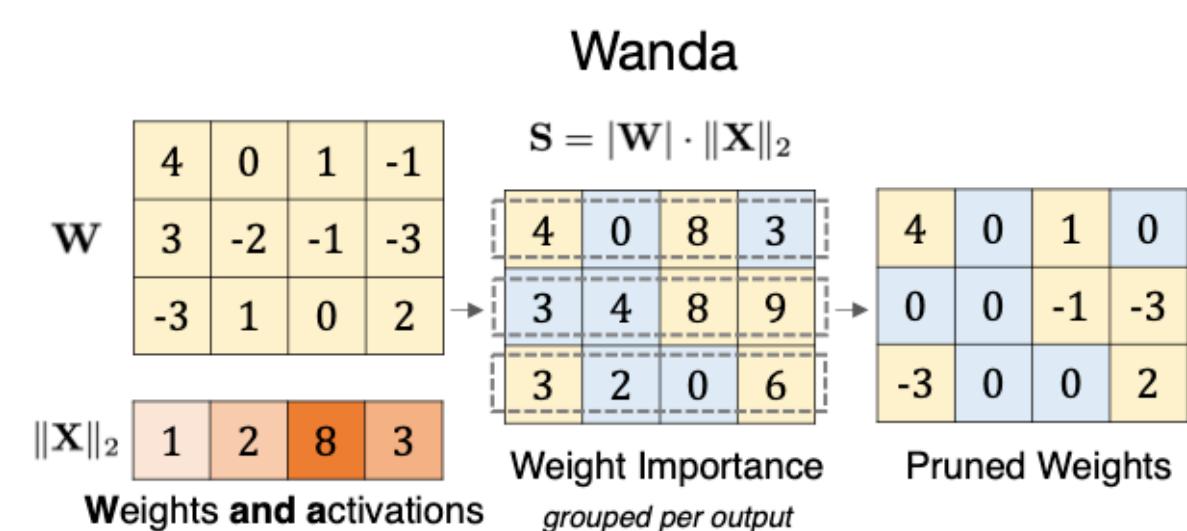
# WANDA Pruning for Large Language Models

- Sun, Mingjie, Zhuang Liu, Anna Bair, and J. Zico Kolter. "A simple and effective pruning approach for large language models." arXiv preprint arXiv:2306.11695 (2023). – ICLR 2024

Magnitude Pruning

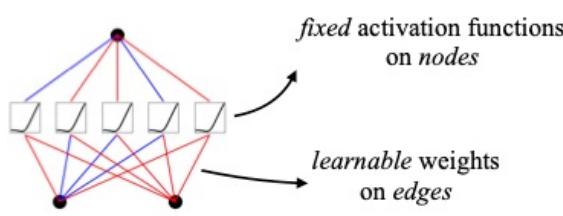
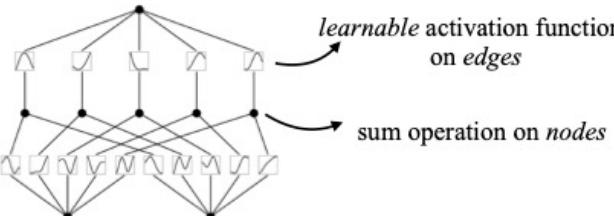
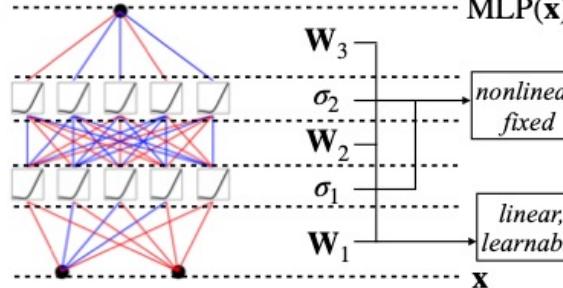
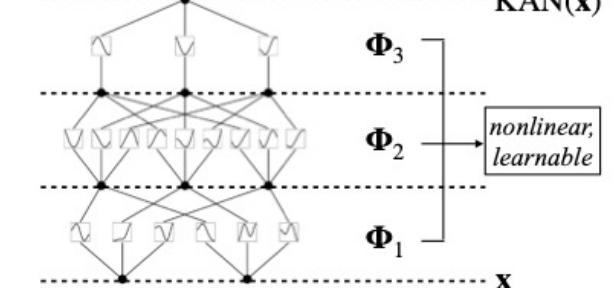


Wanda



# KAN: Kolmogorov-Arnold Networks

- Liu, Ziming, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. "KAN: Kolmogorov-Arnold Networks." arXiv preprint arXiv:2404.19756 (2024).

Model	<b>Multi-Layer Perceptron (MLP)</b>	<b>Kolmogorov-Arnold Network (KAN)</b>
Theorem	<b>Universal Approximation Theorem</b>	<b>Kolmogorov-Arnold Representation Theorem</b>
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left( \sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

# KAN: Kolmogorov-Arnold Networks

## Abstract

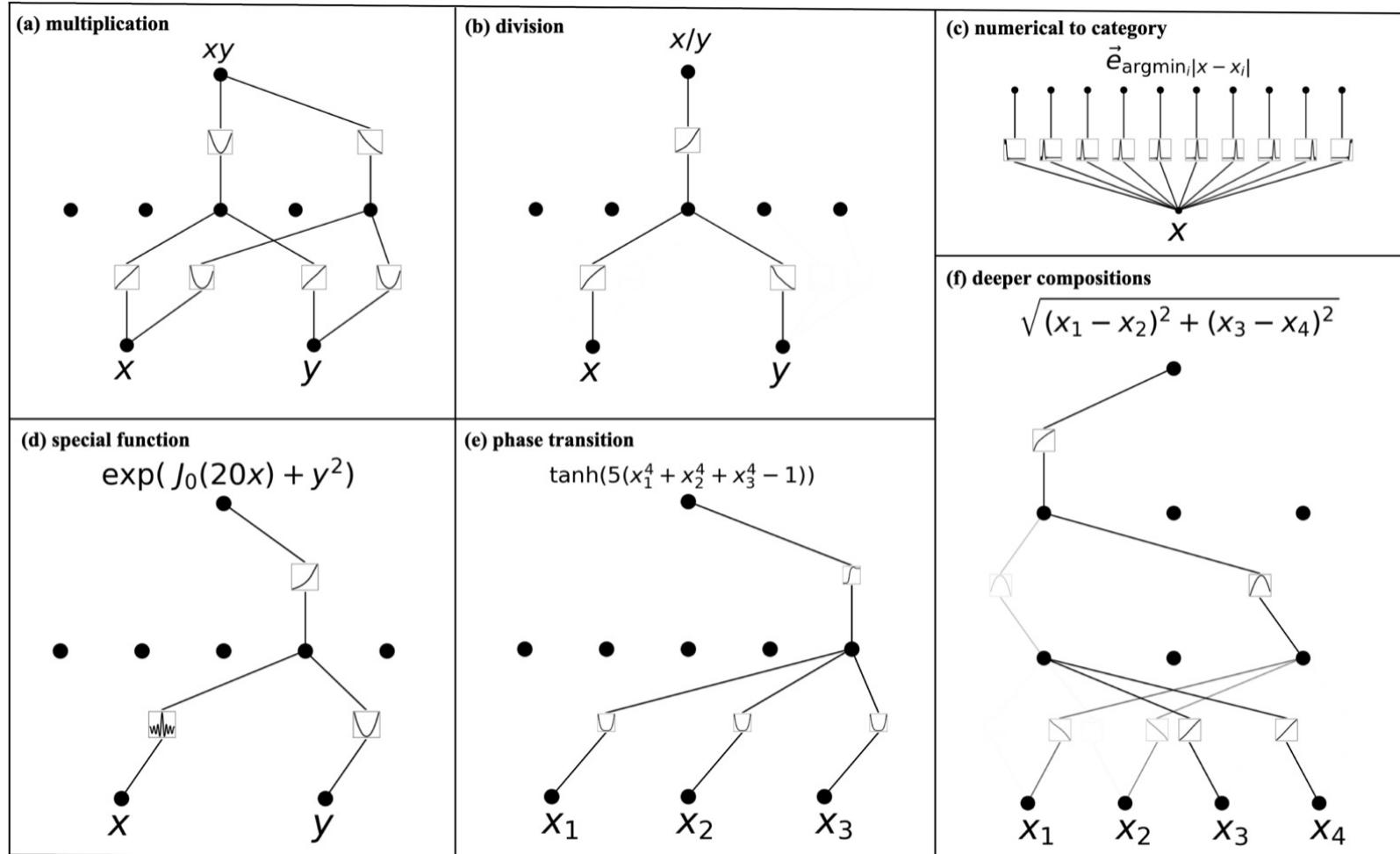
Inspired by the Kolmogorov-Arnold representation theorem, we propose Kolmogorov-Arnold Networks (KANs) as promising alternatives to Multi-Layer Perceptrons (MLPs). While MLPs have *fixed* activation functions on *nodes* (“neurons”), KANs have *learnable* activation functions on *edges* (“weights”). KANs have no linear weights at all – every weight parameter is replaced by a univariate function parametrized as a spline. We show that this seemingly simple change makes KANs outperform MLPs in terms of accuracy and interpretability. For accuracy, much smaller KANs can achieve comparable or better accuracy than much larger MLPs in data fitting and PDE solving. Theoretically and empirically, KANs possess faster neural scaling laws than MLPs. For interpretability, KANs can be intuitively visualized and can easily interact with human users. Through two examples in mathematics and physics, KANs are shown to be useful “collaborators” helping scientists (re)discover mathematical and physical laws. In summary, KANs are promising alternatives for MLPs, opening opportunities for further improving today’s deep learning models which rely heavily on MLPs.

parameter becomes KAN’s spline function. Fortunately, KANs usually allow much smaller computation graphs than MLPs. For example, we show that for PDE solving, a 2-Layer width-10 KAN is **100 times more accurate** than a 4-Layer width-100 MLP ( $10^{-7}$  vs  $10^{-5}$  MSE) and **100 times more parameter efficient** ( $10^2$  vs  $10^4$  parameters).



# Interpretability of KAN and Pruning

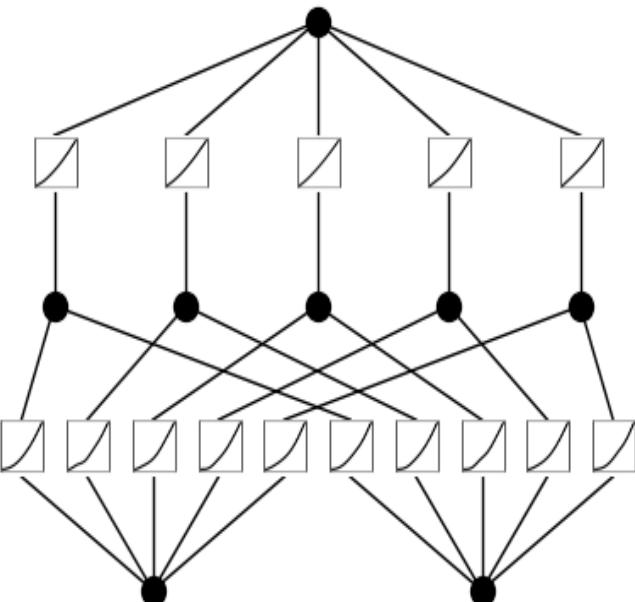
- <https://github.com/KindXiaoming/pykan>



# Pruning in KAN

- <https://github.com/KindXiaoming/pykan>

```
# plot KAN at initialization  
model(dataset['train_input']);  
model.plot(beta=100)
```

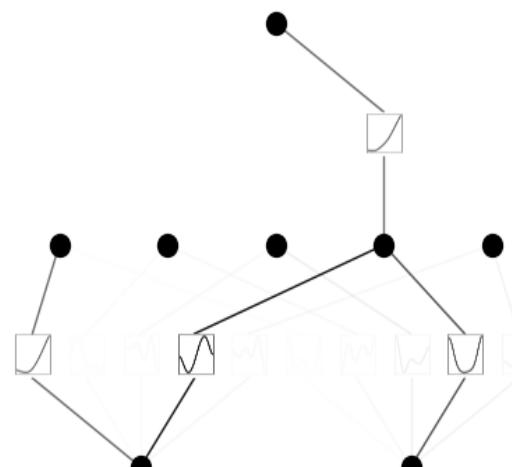


Train KAN with sparsity regularization

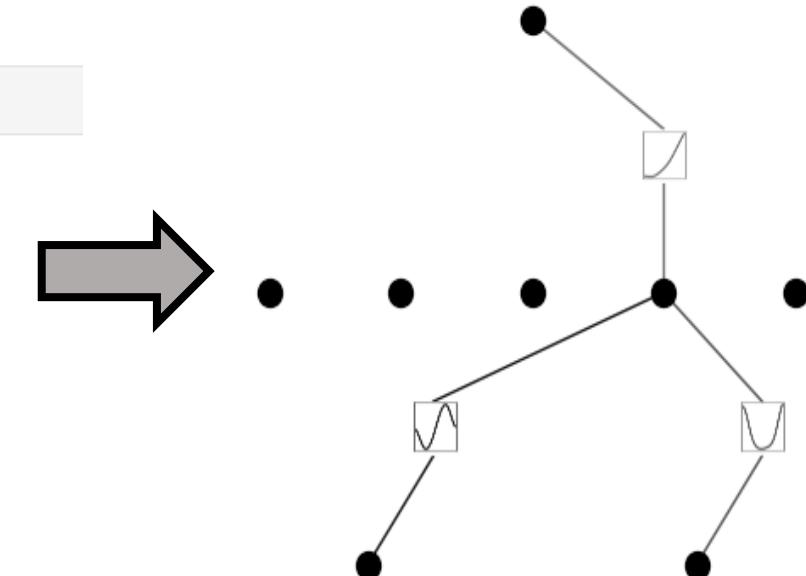
```
: # train the model  
model.train(dataset, opt="LBFGS", steps=20, lamb=0.01, lamb_entropy=10.);  
  
train loss: 1.57e-01 | test loss: 1.31e-01 | reg: 2.05e+01 : 100%|████| 20/20 [
```

Plot trained KAN

```
: model.plot()
```



```
model.prune()  
model.plot(mask=True)
```



Prune KAN and replot (get a smaller shape)

# Quantization Aware Training (QAT)

- **Post-Training Quantization (PTQ):** Quantization is applied after training
- **Quantization Aware Training (QAT):** An approach that integrates quantization into the neural network training. It simulates lower precision arithmetic during the training phase
- **Goal of QAT:** Ensure that the model is robust to the effects of quantization, maintaining high accuracy when deployed with reduced precision.
- QAT is particularly important for deploying models on edge devices with limited computational resources, yet require high accuracy (e.g., Fall Detection)



FALL DETECTION

# How QAT Works

---

- **Training with Quantization:**
  - Both weights and activations are quantized to lower bit-widths during the forward and backward passes of training.
  - Allows the model to "learn" the quantization noise/error and account for it during training
- **Fine-tuning Process:**
  - Initially, the model is trained using floating point weights to a certain accuracy level
  - It is then fine-tuned with quantization enabled, which adjusts the model parameters to compensate for the loss of precision due to quantization.



# PTQ Vs. QAT

Quantization Type	Best Suited For	Complexity and Time	Performance Comparison
PTQ (Post-Training Quantization)	Scenarios with limited resources/time for retraining; slight accuracy degradation is acceptable.	Low complexity; can be applied quickly without retraining the model.	May cause significant accuracy drops, especially in sensitive models.
QAT (Quantization-Aware Training)	High accuracy is crucial; performance-critical applications.	High complexity; integrates quantization into the training cycle, requiring more resources and time.	Maintains higher accuracy by accounting for quantization during training.



# Motion Classification

- Why should we classify motion?
- What are the application domains where motion classification can be useful?
- What are the input data types that can effectively help to classify motion data?
- How is motion classification similar or different to gesture recognition?

Input Video Clip	Motion Type Predicted
<i>Cart-wheel</i> 	<i>Projectile</i>
<i>Jogging</i> 	<i>Linear</i>
<i>Diving</i> 	<i>Projectile</i>
<i>Drinking</i> 	<i>Local</i>
<i>Fencing</i> 	<i>Random</i>

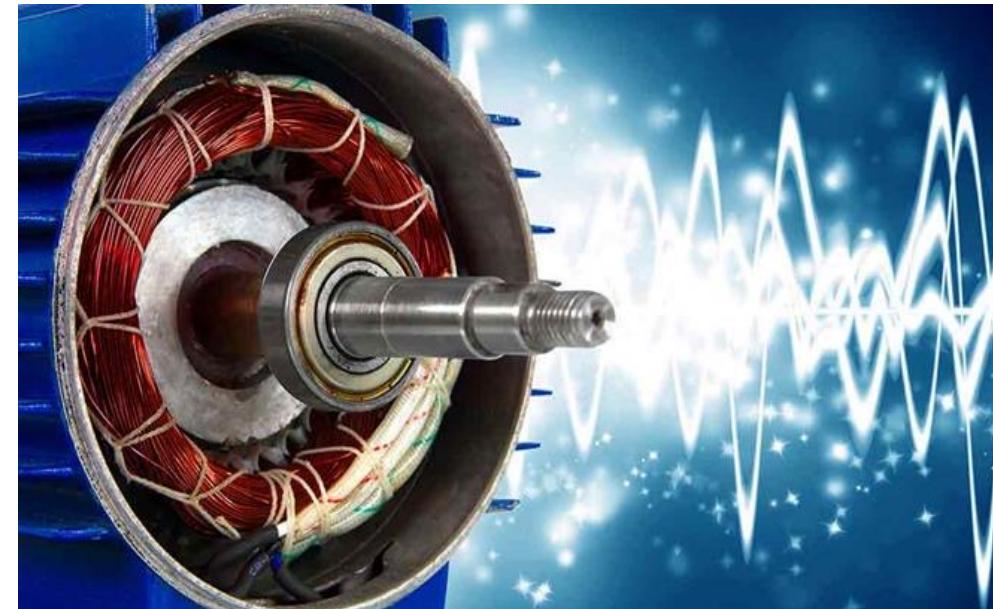
# Classification of Different Data



Audio Classification



Image Classification



Motion Classification

What do we choose as input for motion classification?

**Depends on the application!**

# Motion Classification

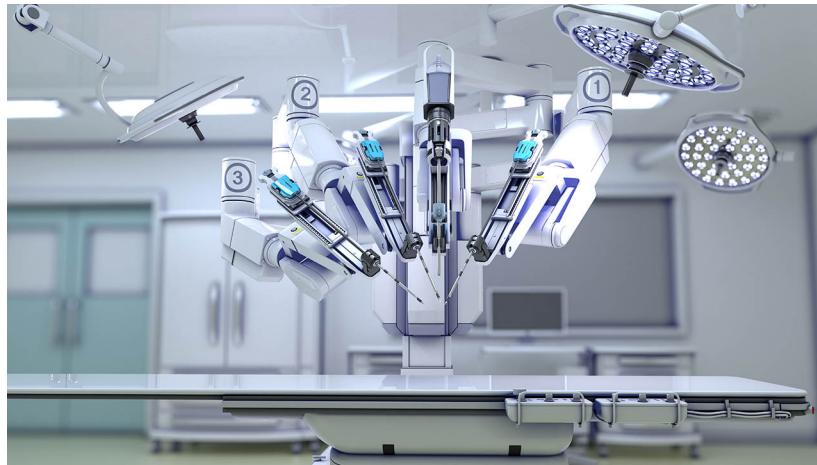


Classify Washing Machine cycles



Classify poses for games/ yoga/ martial arts

# Applications of Motion Classification



Surgical Robotics

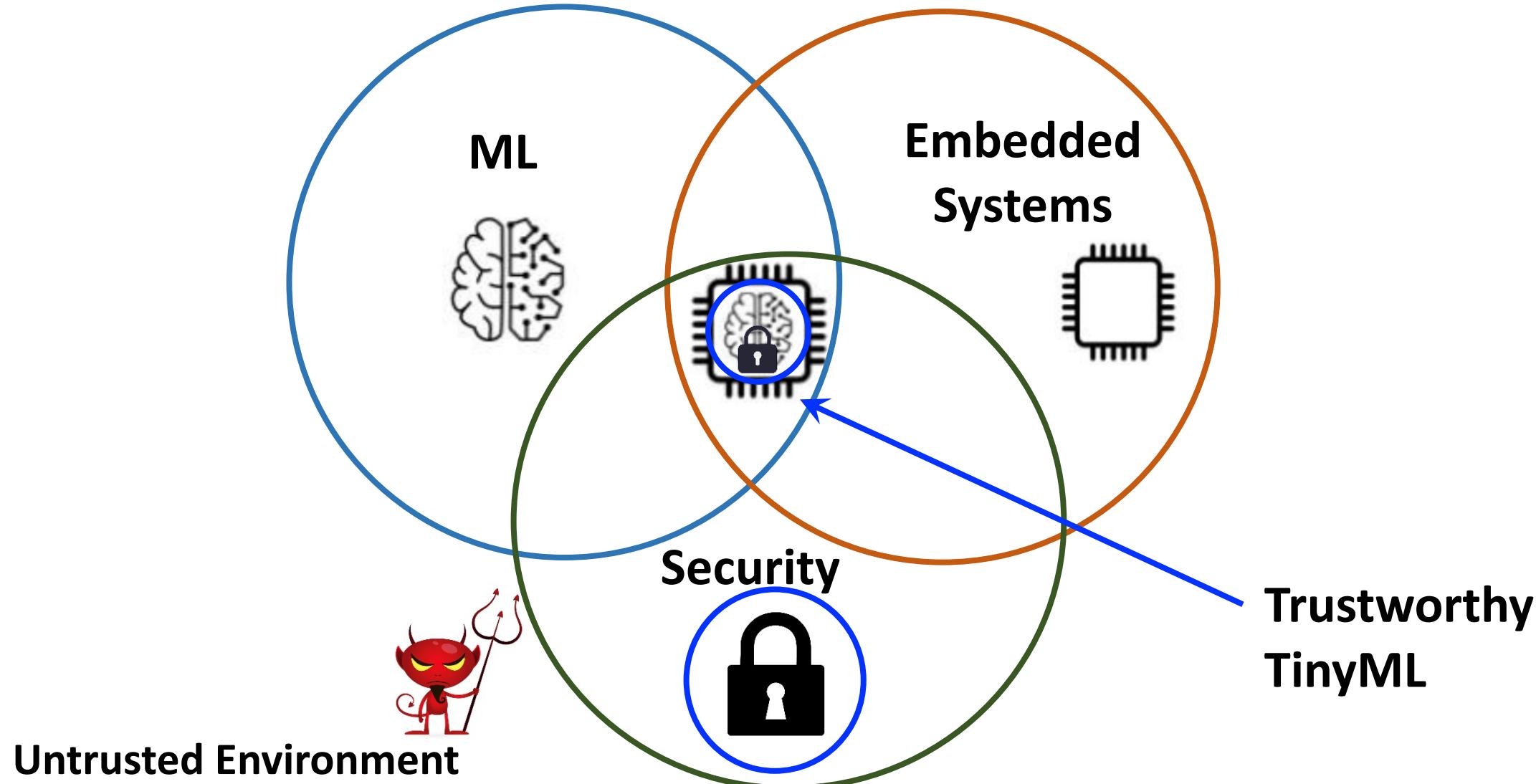


Transportation of Freight

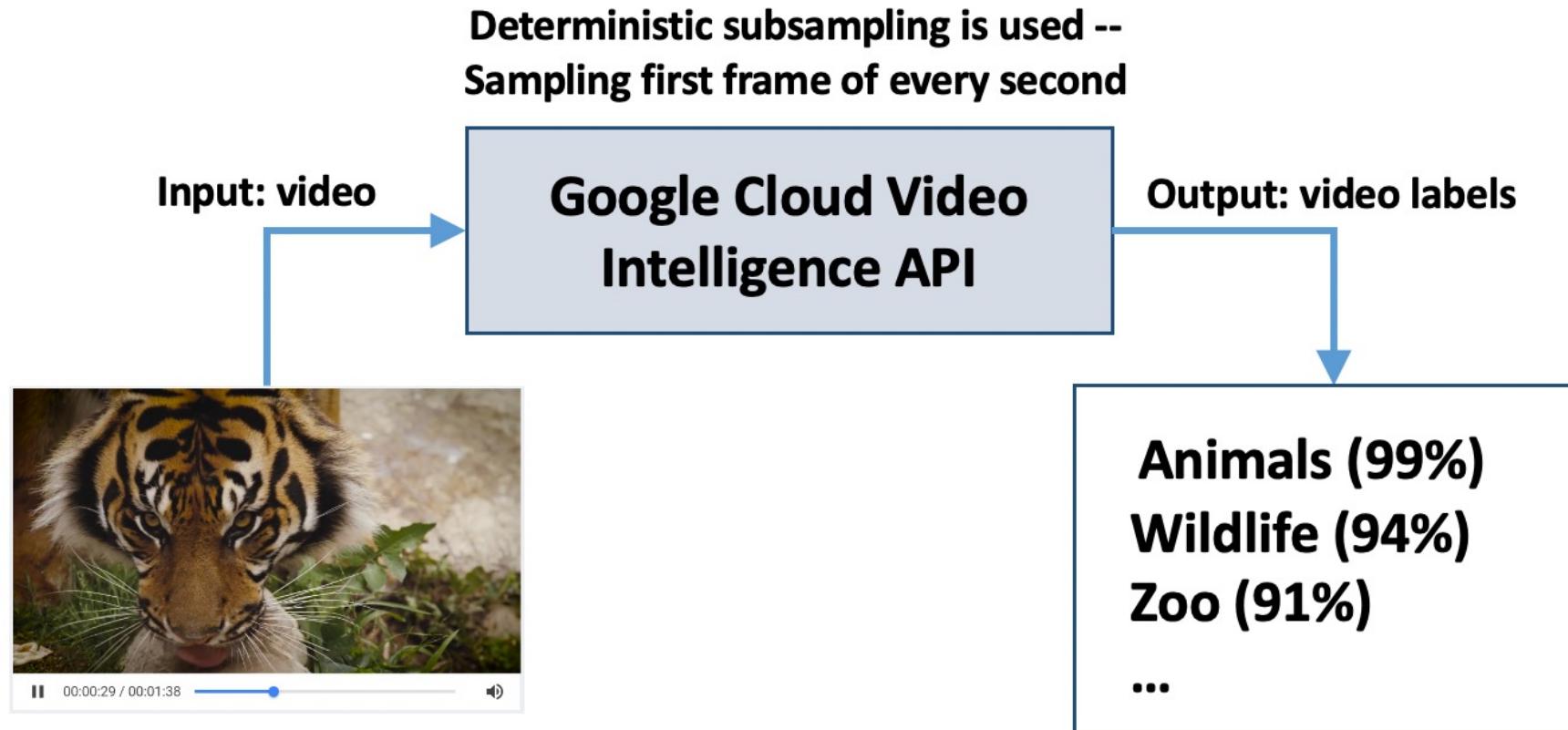


Motion Capture Technology

# Can we attack motion classifier applications?



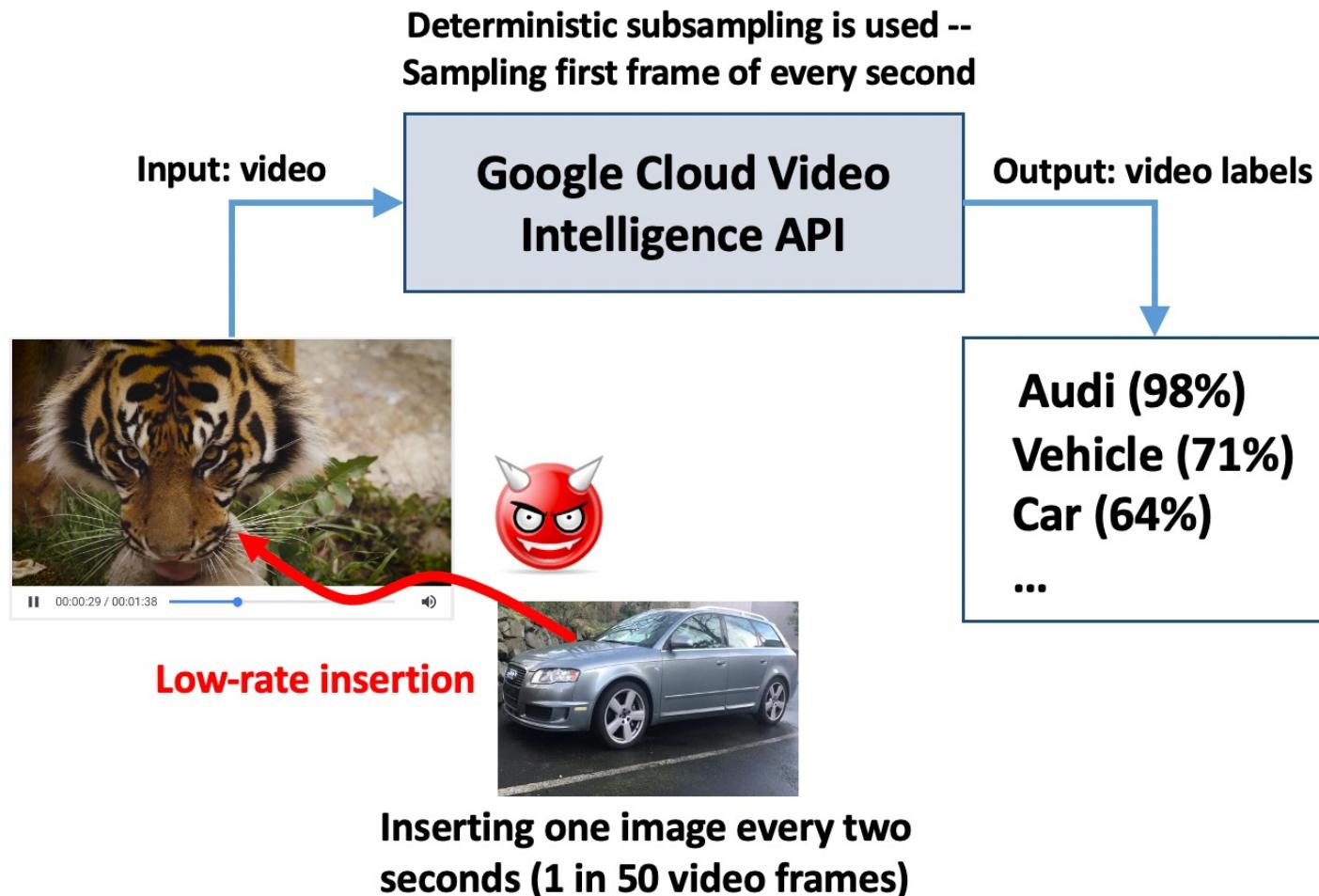
# Google Video Analysis with DNN



Hosseini, H., Xiao, B. and Poovendran, R., "Deceiving Google's Cloud Video Intelligence API Built for Summarizing Videos." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017.



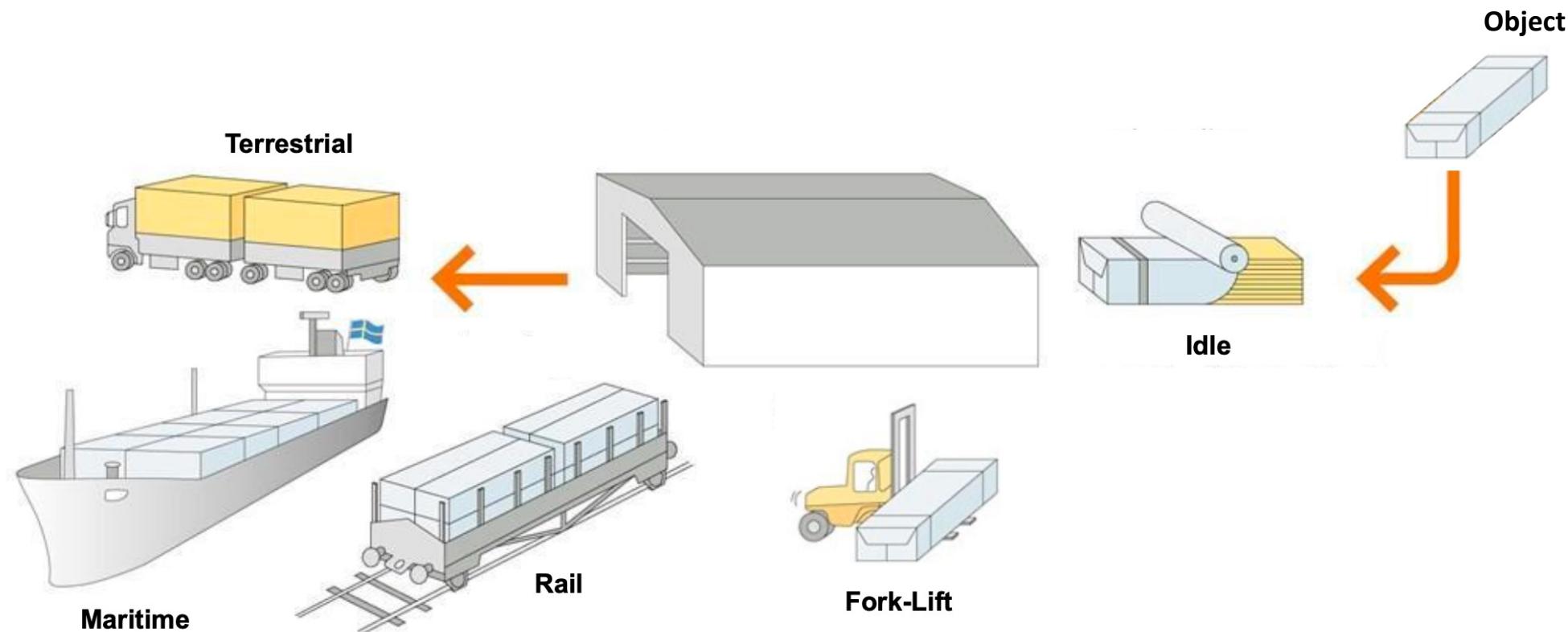
# Decision Time Attack---With Domain Knowledge



Hosseini, H., Xiao, B. and Poovendran, R., "Deceiving Google's Cloud Video Intelligence API Built for Summarizing Videos." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017.

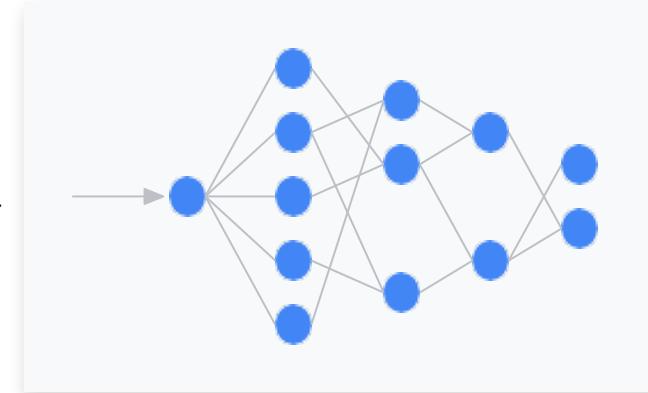
# Motion Classification for Transport

- Motion classification for transport is essential to calculate the stress or force on objects in transport.
- Objects experience different types of stress based on the conditions of the route.



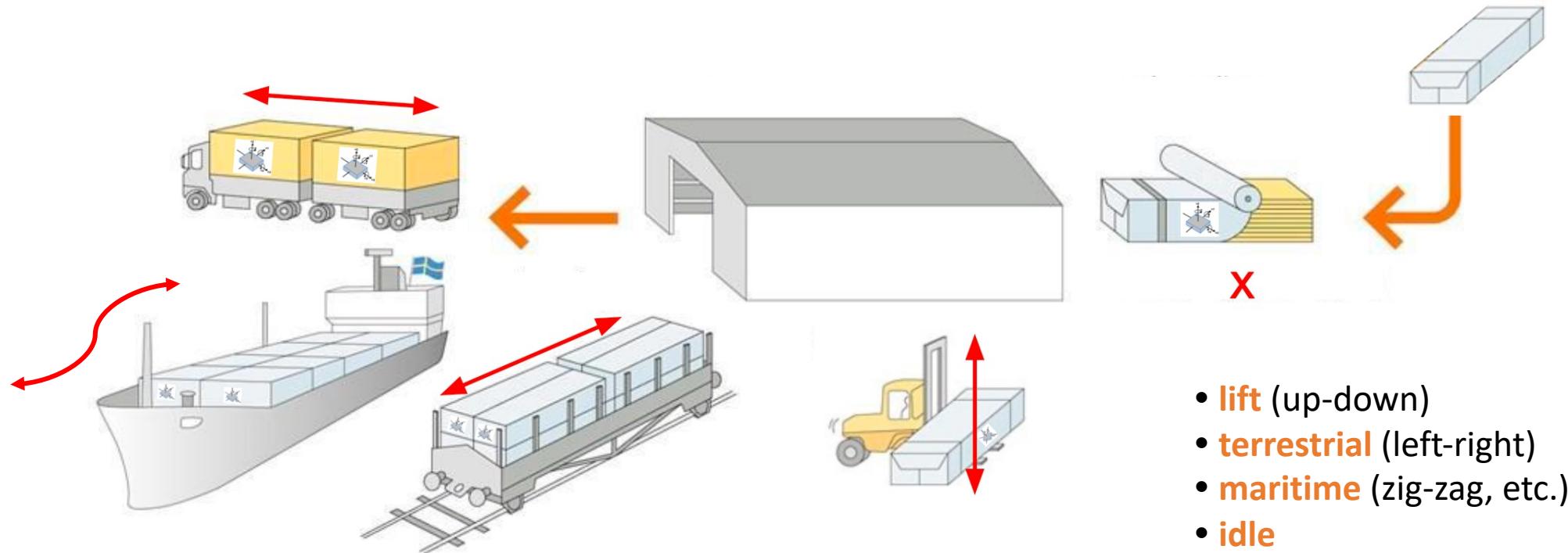
# Challenges in implementation

- On-device Computing is necessary since streaming to the cloud might not be possible due to poor connection during freight transport.

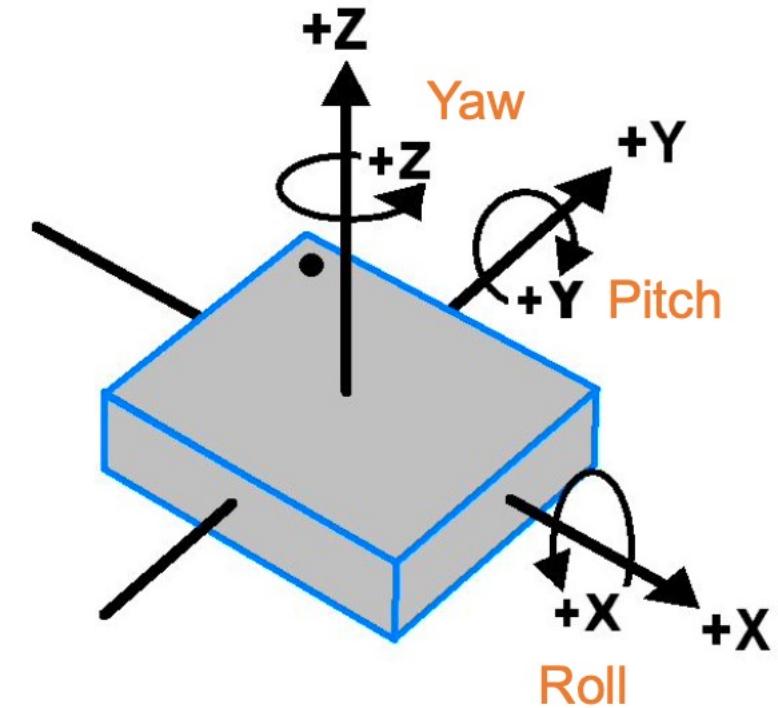
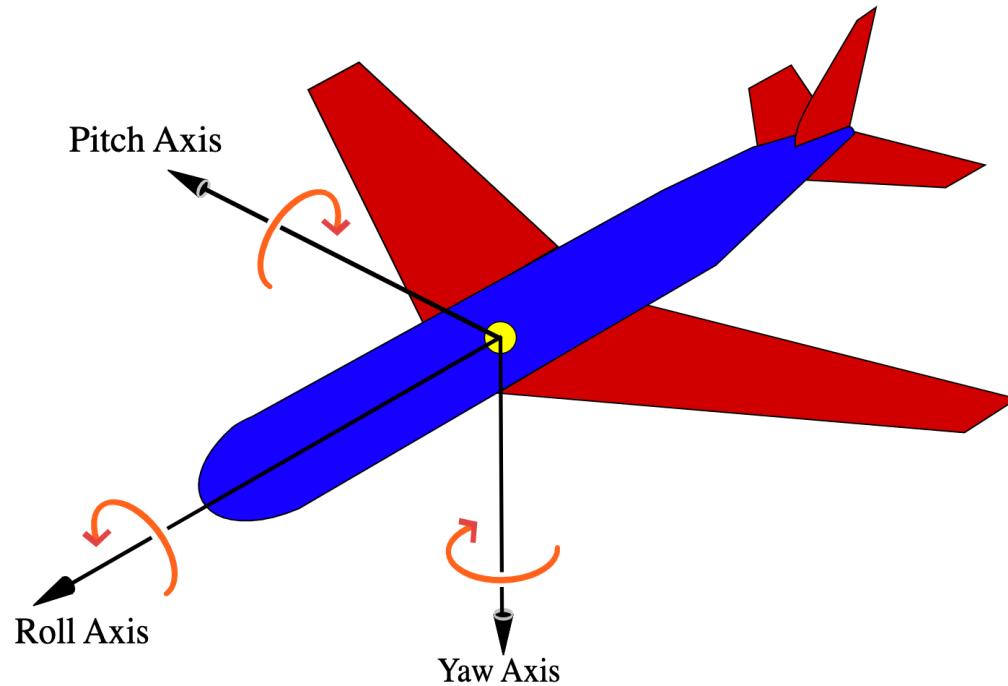


# Degrees of Freedom

- Degrees of freedom (DoF) refer to the number of basic ways a rigid object can move through 3D space.
- We define various classes based on the transport type and the degree of freedom for object movement in each of the types.

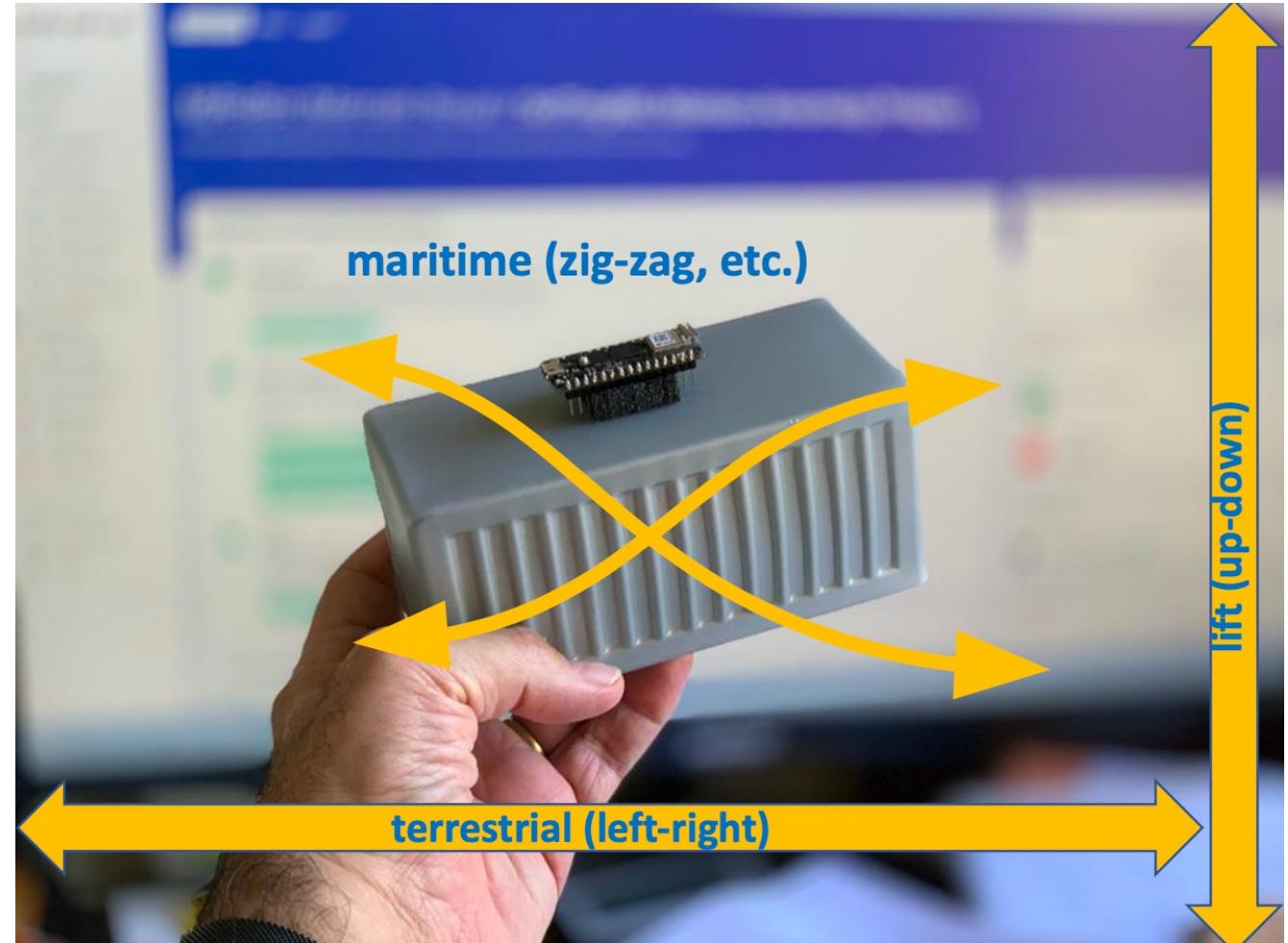
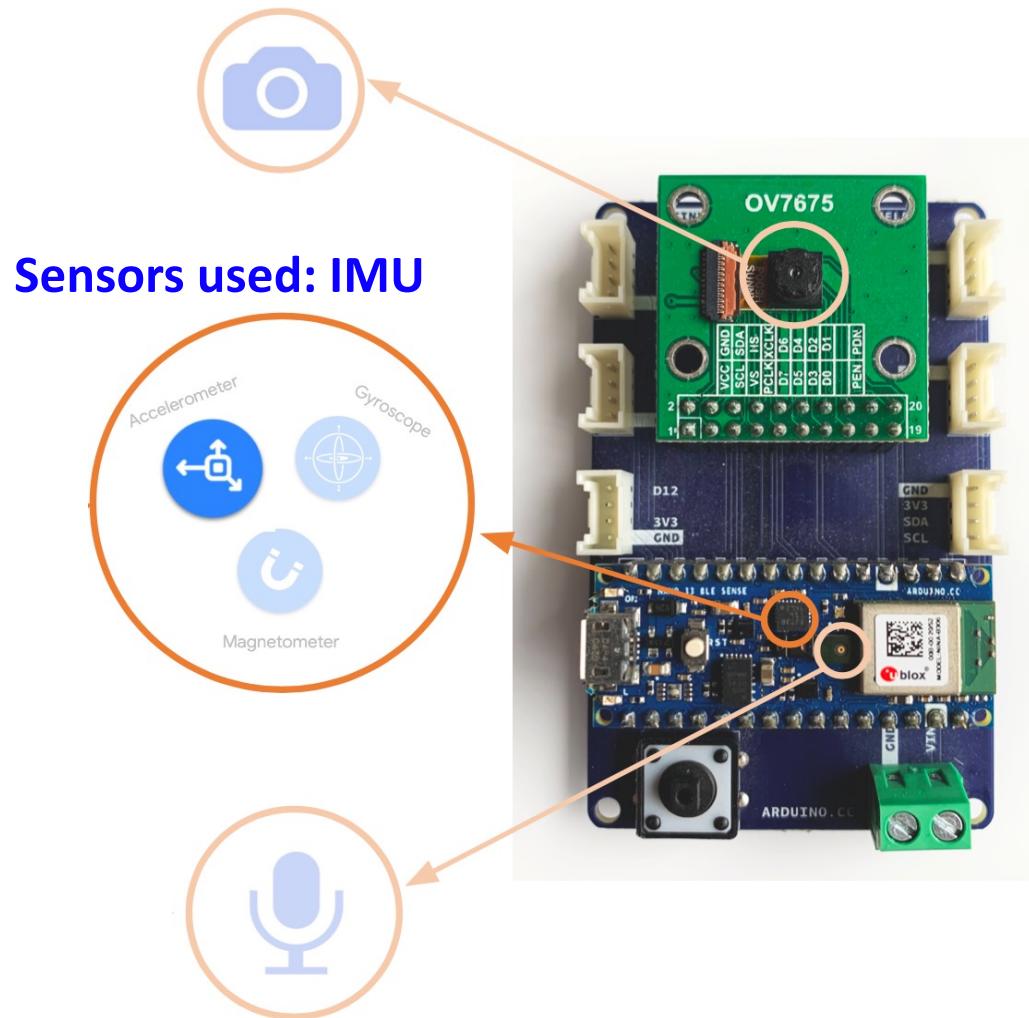


# Inertial Measurement Units (IMU)



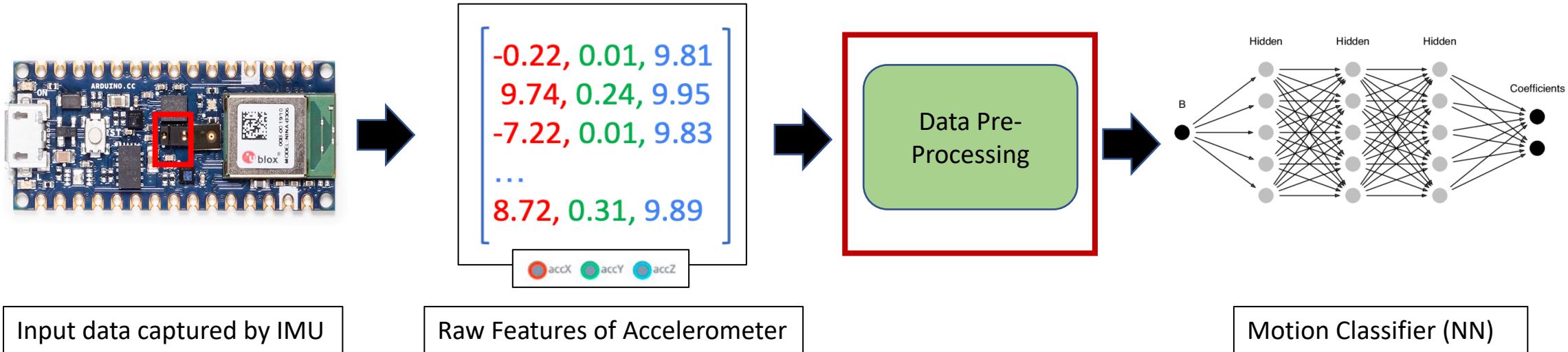
Yaw, Pitch and Roll measurements through Accelerometer and Gyroscope data

# TinyML IMU Sensors



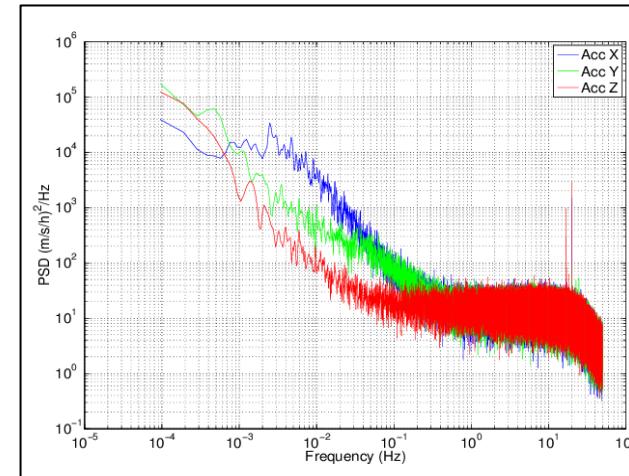
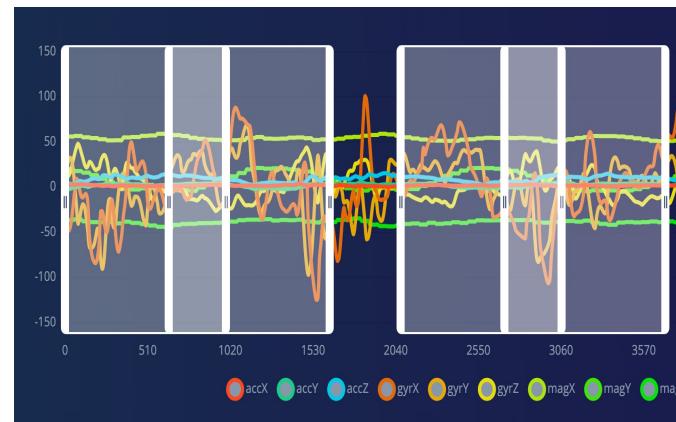
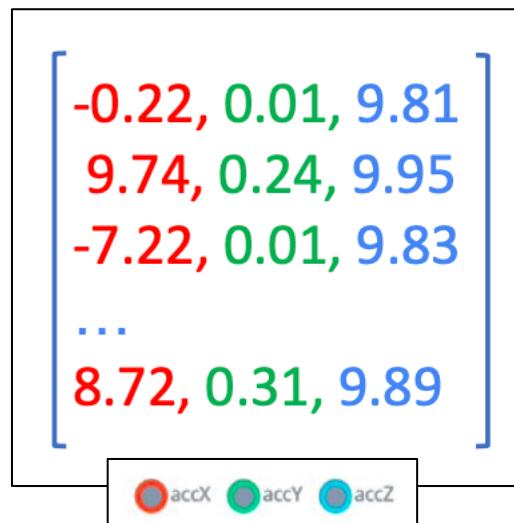
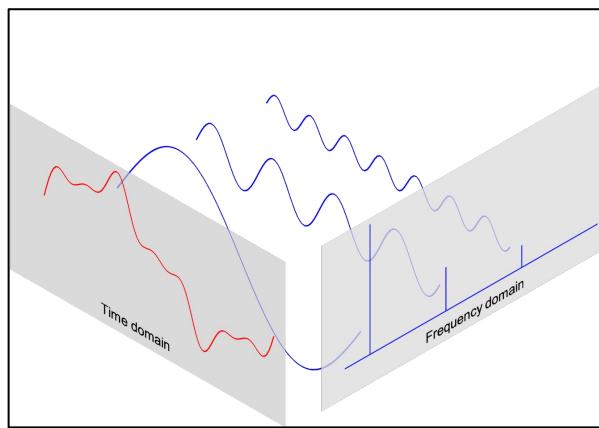
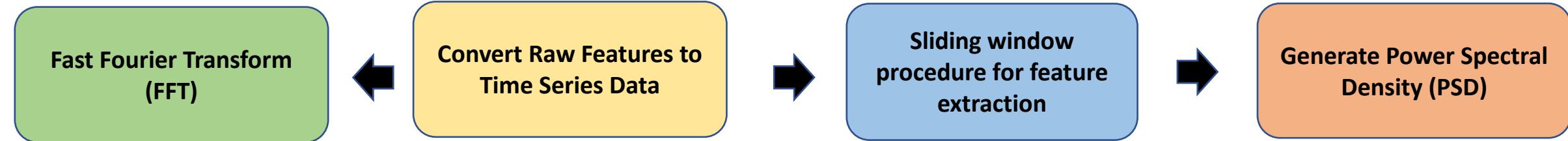
# End-to-End flow of Motion Classifier

- End-to-End flow from capturing the motion data to classifying the different types of motion.
- We use a DNN Classifier as our model.



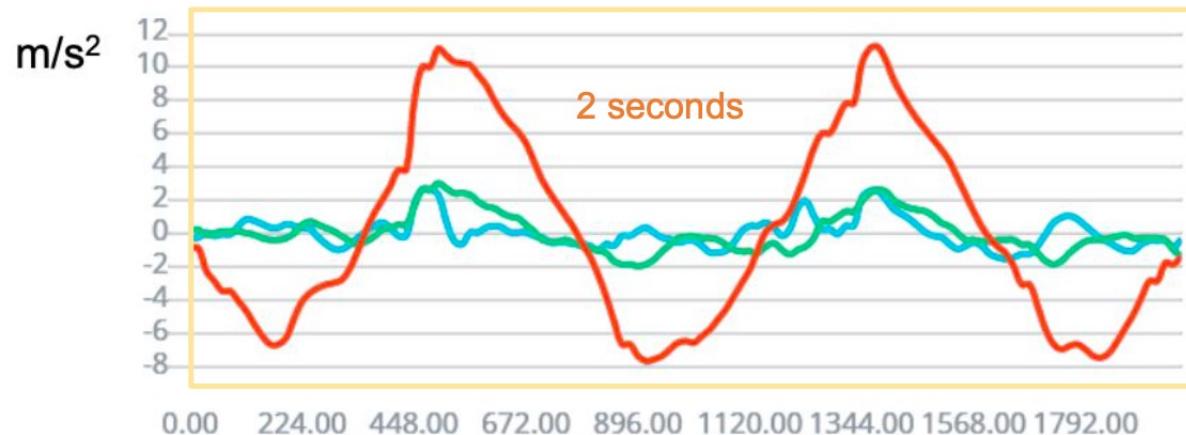
# Data Pre-Processing

Data pre-processing steps:

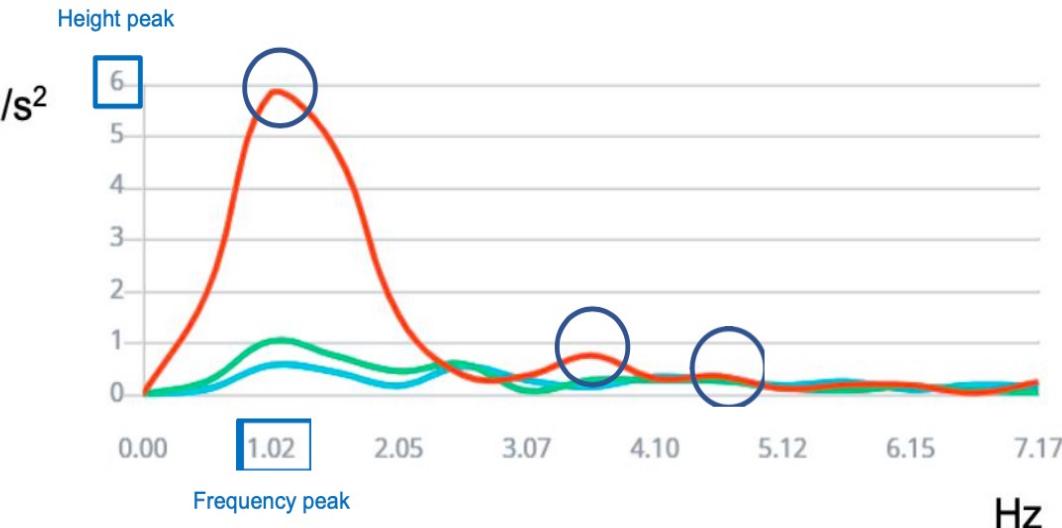


# FFT for Accelerometer Data

- Calculating FFT for accelerometer data extracts **Height Peak** and **Frequency Peak** parameter from the raw data which can then be fed to the NN for gaining a compressed model to implement on tiny devices.

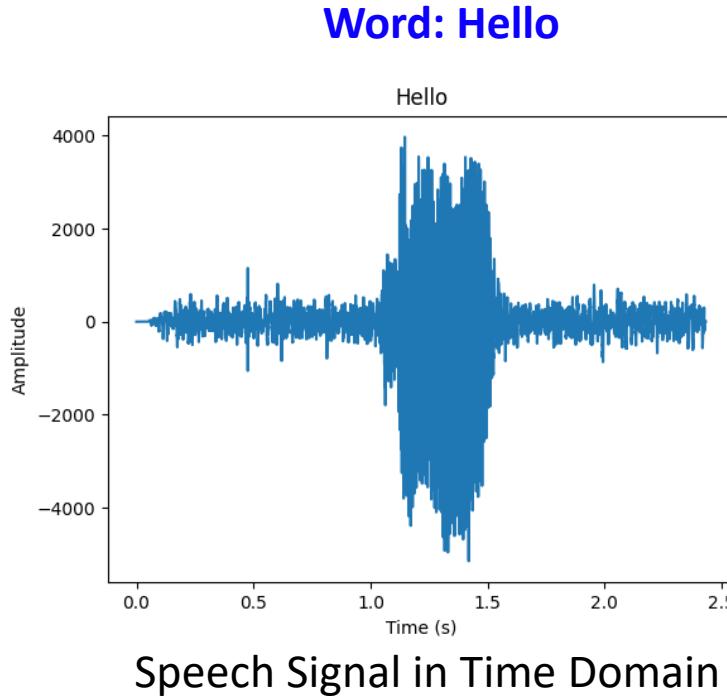


FFT

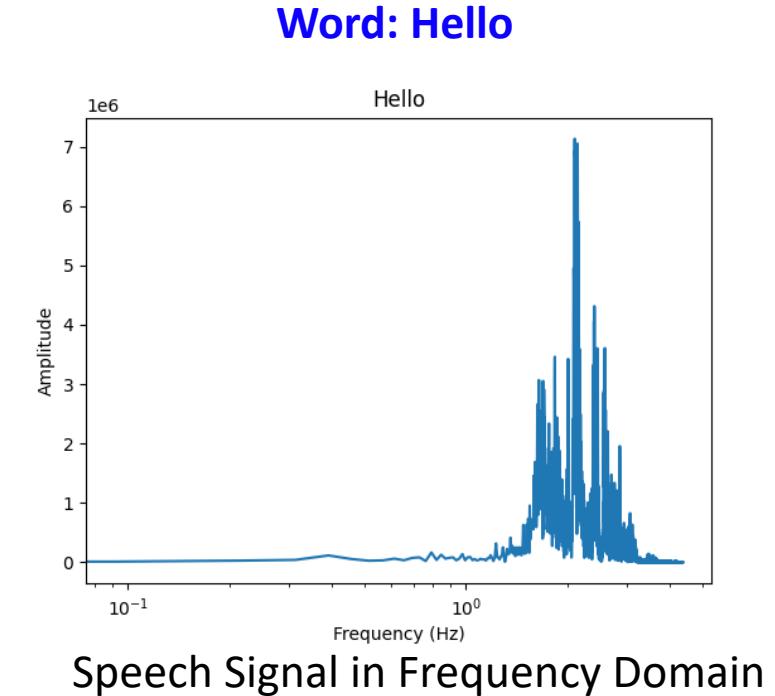


# Recall.. Fast Fourier Transform (FFT)

- The time series signal (Time v. Amplitude) is transformed into the frequency domain (Frequency v. Amplitude) using Discrete Fourier transform.
- The Fast Fourier Transform (FFT) is an efficient algorithm to calculate the DFT of a sequence.

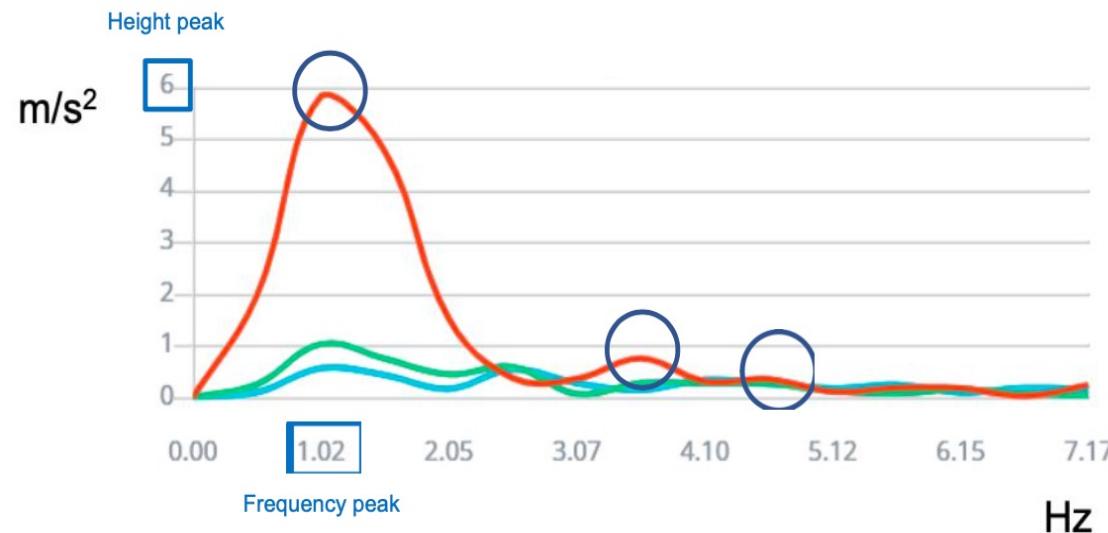


Fast Fourier Transform

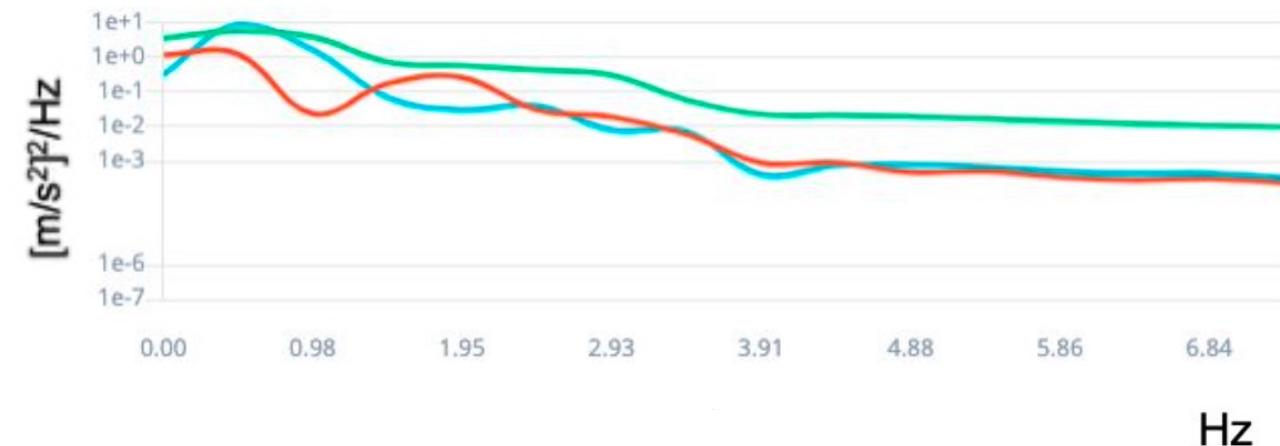


# Power Spectral Density (PSD)

Calculating PSD from FFT of the signal:



Fast Fourier Transform



Power Spectral Density

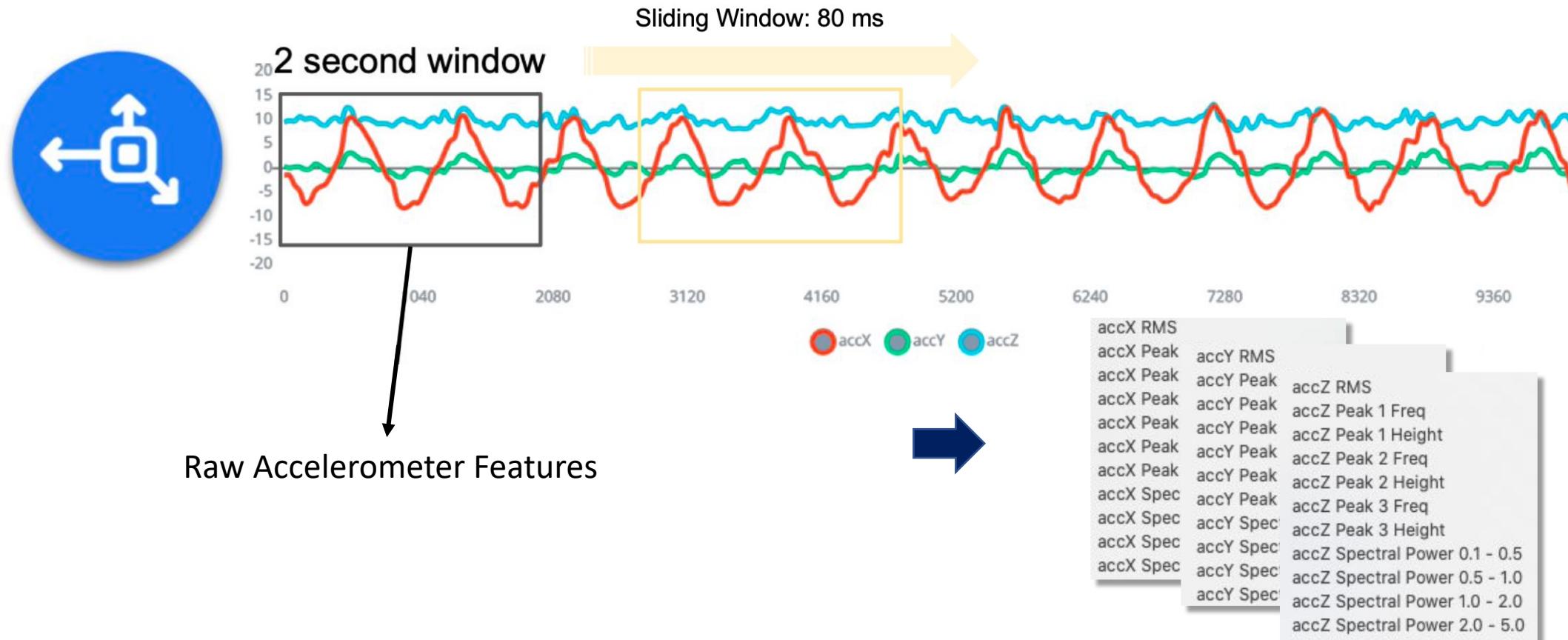


4 frequency bins/axis

Source: <https://blog.endaq.com/why-the-power-spectral-density-psd-is-the-gold-standard-of-vibration-analysis>

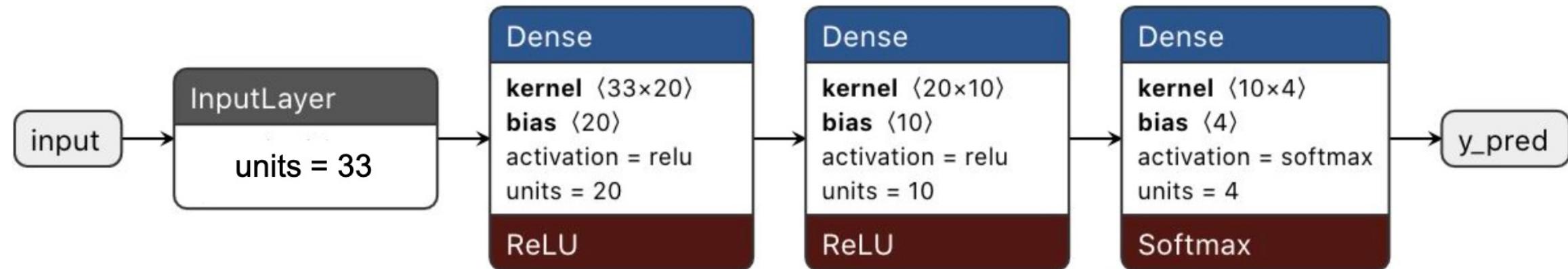
# Sliding Window for Accelerometer Data

- For our motion classifier application, we will be using Sliding Window procedure for extracting important features from the raw accelerometer data.

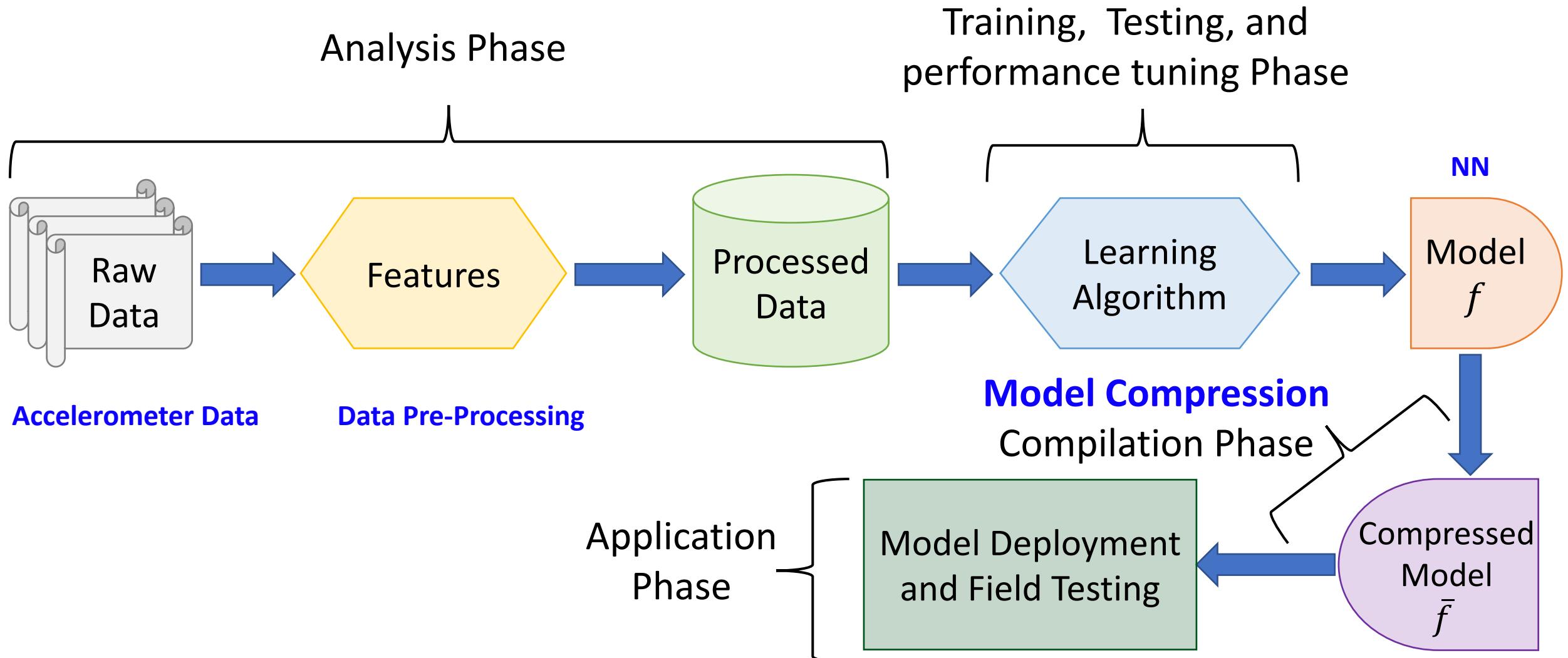


# Deep Neural Network (DNN) Classifier

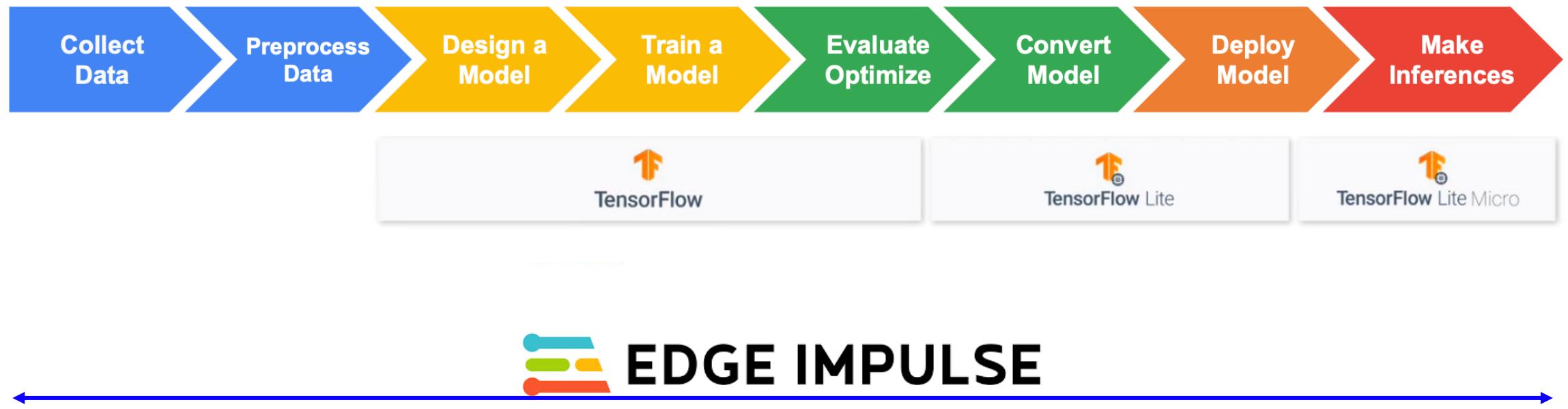
- The DNN classifies transportation data into four classes:
  - lift
  - terrestrial
  - maritime
  - idle



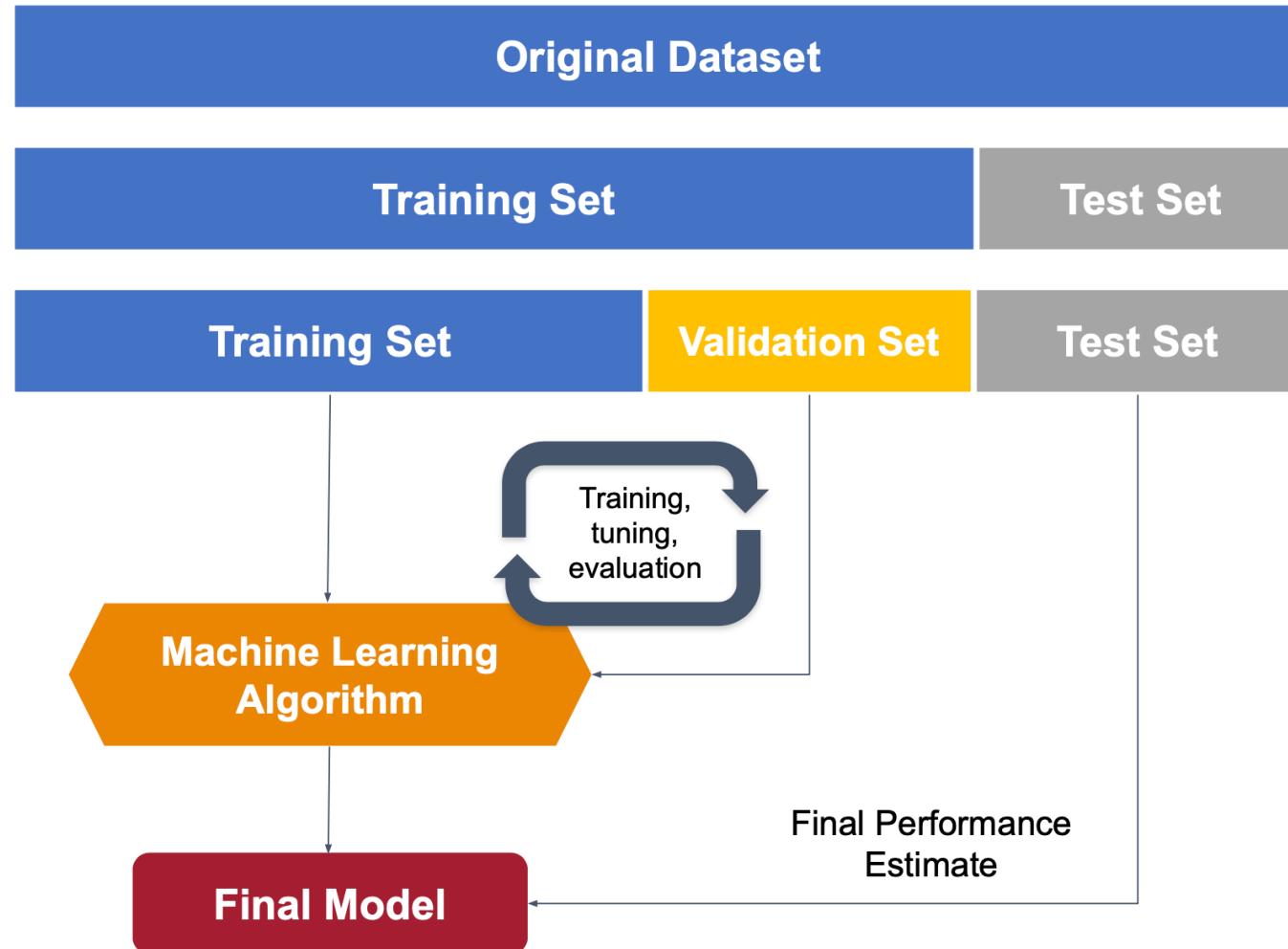
# A Schematic View of TinyML Pipeline



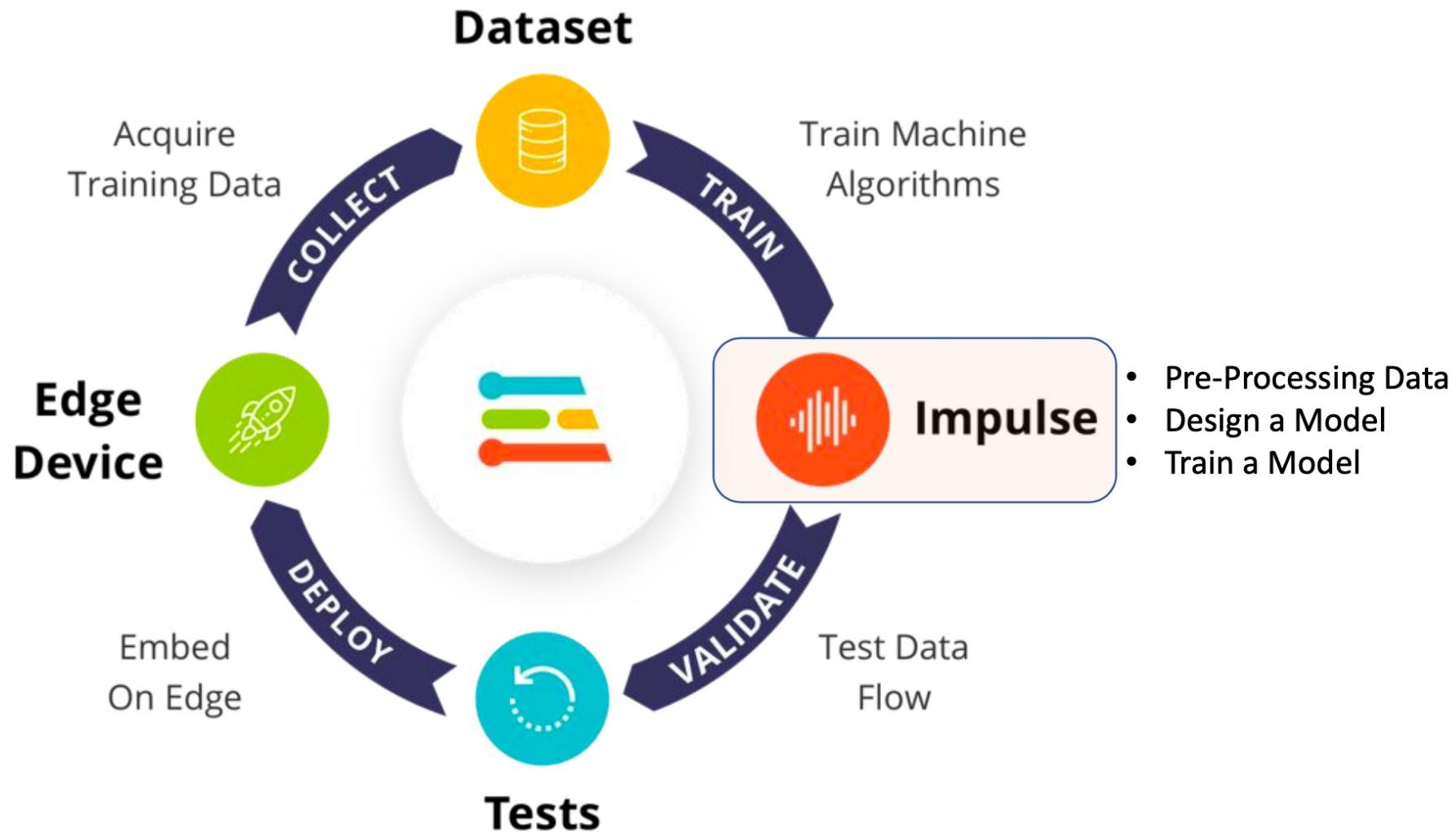
# TinyML for Motion Classification



# Edge Impulse - Model



# Edge Impulse - Model





5 min Break

# Today's Lab

- Both Software and Hardware Lab!
- Accelerometer data collection for motion classification
- Computing FFT, Spectrogram, and PSD from time series data
- ML Models for Motion Classifier
- Implement the Model with TinyML Kits



# Lab 7 - Software - Road Map

1. Installing Arduino CLI
2. Installing Edge Impulse Firmware on Arduino
3. Data Collection and Labelling
4. Splitting Data
5. Spectral Analysis
6. Training the ML model (DNN)



# Lab 7 - Software - Road Map

7. Evaluation of ML Model Performance

8. Building ML model



# Lab 7 – Software

---

We will use Edge Impulse for this Lab.

Website: <https://www.edgeimpulse.com/>

(You can login in using the account we created for the previous Edge Impulse lab)

Create a new Project on Edge Impulse and provide the title as “Motion Classification for Transport”.

# Lab 7 - Hardware - Road Map

1. Build the Arduino library using Edge Impulse.
2. Compile and upload the sketch to the board.
3. Open Serial Monitor and test the classification.



# Software Lab – Connecting the Device

The screenshot shows the Edge Impulse web interface. On the left, a sidebar menu includes 'Dashboard', 'Devices' (which is highlighted with a red border), 'Data acquisition', 'Impulse design', 'Create impulse', 'Spectral features', 'NN Classifier', and 'EON Tuner'. The main content area is titled 'Kavya / Boat' and shows a purple header bar with a yellow circular badge containing the letter 'K'. Below this, a section titled 'Your devices' displays a table of connected devices. The table has columns for NAME, ID, TYPE, SENSORS, REM..., and LAST SEEN. A single device is listed: 'C9:FA:B0:4A:62:83' (ID: C9:FA:B0:4A:62:83, Type: ARDUINO\_NANO33\_IOT, Sensors: Built-in microphone, Infrared sensor, Last Seen: Today, 18:36:08). To the right of the table is a button labeled '+ Connect a new device'. At the bottom of the page, a copyright notice reads '© 2023 EdgeImpulse Inc. All rights reserved'.



# Software Lab - Firmware Installation

## Collect new data

Collect data directly from your phone, computer, device, or development board.



Scan QR code to connect to  
your phone



Connect to your computer



Connect your device or  
development board

## Connect your device or development board

You can connect almost any device over serial using our [data forwarder](#):

```
$ edge-impulse-data-forwarder
```

Or connect a supported device or development board:

Search supported development boards



Any device running macOS



Arducam Pico4ML TinyML Dev Kit



Arduino Nano 33 BLE Sense



Arduino Nicla Vision



Arduino Nicla Voice



# Software Lab – Arduino CLI

---

## If you're on Windows:

- Unzip the .zip file to C:\Program Files\arduino-cli
- Open System Properties > Advanced > Environment Variables
- Path under "user variables" > Edit
- Add C:\Program Files\arduino-cli

## If you're on macOS:

- Use the curl method or homebrew shown on the installation page:  
<https://arduino.github.io/arduino-cli/0.21/installation/>
- If you use the curl method, it likely installs arduino-cli to ~/bin. That might not be on your path. So, you might need to run: export PATH=\$PATH:~/bin



# Software Lab - Arduino CLI and Firmware

```
/Users/Admin/Downloads/arduino-nano-33-ble-sense\ \(2\)/flash_mac.command ; exit;
(base) Admin@Sarats-MacBook-Pro ~ % /Users/Admin/Downloads/arduino-nano-33-ble-sense\ \(2\)/flash_mac.command ; exit
You're using an untested version of Arduino CLI, this might cause issues (found: 0.32.2, expected: 0.18.x)
Finding Arduino Mbed core...
Finding Arduino Mbed OK
Finding Arduino Nano 33 BLE...
Finding Arduino Nano 33 BLE OK
Flashing board...
TOUCH: error during reset: opening port at 1200bps: Serial port busy
Device      : nRF52840-QIAA
Version     : Arduino Bootloader (SAM-BA extended) 2.0 [Arduino:IKXYZ]
Address     : 0x0
Pages       : 256
Page Size   : 4096 bytes
Total Size  : 1024KB
Planes      : 1
Lock Regions: 0
Locked      : none
Security    : false
Erase flash

Done in 0.001 seconds
Write 352560 bytes to flash (87 pages)
[=====] 100% (87/87 pages)
Done in 13.914 seconds

Flushed your Arduino Nano 33 BLE development board.
To set up your development with Edge Impulse, run 'edge-impulse-daemon'
To run your impulse on your development board, run 'edge-impulse-run-impulse'

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

The screenshot shows the Edge Impulse web interface. On the left, a modal dialog box from "studio.edgeimpulse.com" asks to connect to a serial port. It lists several options, with "Nano 33 BLE (cu.usbmodem5) - Paired" highlighted with a red box. Below the list are "Cancel" and "Connect" buttons, with "Connect" also highlighted with a red box. To the right of the modal is a main dashboard for a project named "Kavya / Boat". The dashboard includes a "Sources | CSV Wizard" section showing a "TRAIN / TEST SPLIT" of 86% / 14%, and a "Collect data" section with a "Connect a device" button. A QR code is visible in the top right corner of the dashboard area.

# Software Lab – Data Collection

The screenshot shows the Edge Impulse web interface for a project titled "Kavya / Boat". The left sidebar contains navigation links for Dashboard, Devices, Data acquisition, Impulse design, Create impulse, EON Tuner, Retrain model, Live classification, Model testing, Versioning, Deployment, Documentation, and Forums. The main area displays a summary of data collection: "DATA COLLECTED 1m 45s" and "TRAIN / TEST SPLIT 86% / 14%". A "Dataset" table lists 18 training samples and 3 test samples, all labeled "idle". The "Collect data" section on the right shows fields for "Label" (set to "lift") and "Sensor", with a "Start sampling" button. A "RAW DATA" section shows a spectrogram for the file "idle.3vu8ps33.ingestion-7f6f59c885-zgppq.s6".

Kavya / Boat

Dataset Data explorer Data sources | CSV Wizard 5s

DATA COLLECTED 1m 45s TRAIN / TEST SPLIT 86% / 14%

Dataset

SAMPLE NAME	LABEL	ADDED	LENGTH
idle.3vu8ps33.inge...	idle	Today, 18:36:07	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:07	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:06	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:06	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:06	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:06	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:06	5s
idle.3vu8ps33.inge...	idle	Today, 18:36:06	5s
idle.3vu8ps33.s8	idle	Today, 18:26:02	3s
idle.3vu8ps33.s7	idle	Today, 18:26:02	5s
idle.3vu8ps33.s6	idle	Today, 18:26:02	5s

Collect data

Device: No devices connected

Label: lift

Sample length (ms): 10000

Sensor

Frequency

Start sampling

RAW DATA

idle.3vu8ps33.ingestion-7f6f59c885-zgppq.s6

80  
60  
40  
20  
0

# Software Lab – Impulse Design

The screenshot shows the Edge Impulse software interface for 'Impulse design'. The top navigation bar includes the Edge Impulse logo and a descriptive text: 'An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.'

The left sidebar contains a vertical list of menu items:

- Dashboard
- Devices
- Data acquisition
- Impulse design
- Create impulse
- Spectral features
- NN Classifier
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment

Below the sidebar is a 'GETTING STARTED' section.

The main workspace is divided into four main sections:

- Time series data**: Configurable parameters include 'Input axes (9)' (accX, accY, accZ, gyrX, gyrY, gyrZ, magX, magY, magZ), 'Window size' (2000 ms), 'Window increase' (80 ms), 'Frequency (Hz)' (50), and 'Zero-pad data' (checked). A red box highlights the 'Input axes' list.
- Spectral Analysis**: Shows 'Name' (Spectral features) and 'Input axes (3)' (accX, accY, accZ), all of which are checked. A red box highlights this list.
- Classification**: Shows 'Name' (NN Classifier) and 'Input features' (Spectral features, checked). Below it, 'Output features' are listed as '2 (idle, lift)'. A red box highlights the 'Output features' list.
- Output features**: Shows '2 (idle, lift)' and a checkmark icon.

A large green 'Save Impulse' button is located on the right side of the workspace.



# Hardware Lab - Deployment

Kavya / Boat

EDGE IMPULSE

- Dashboard
- Devices
- Data acquisition
- Impulse design
  - Create impulse
  - Spectral features
  - NN Classifier
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment**

Deploy your impulse

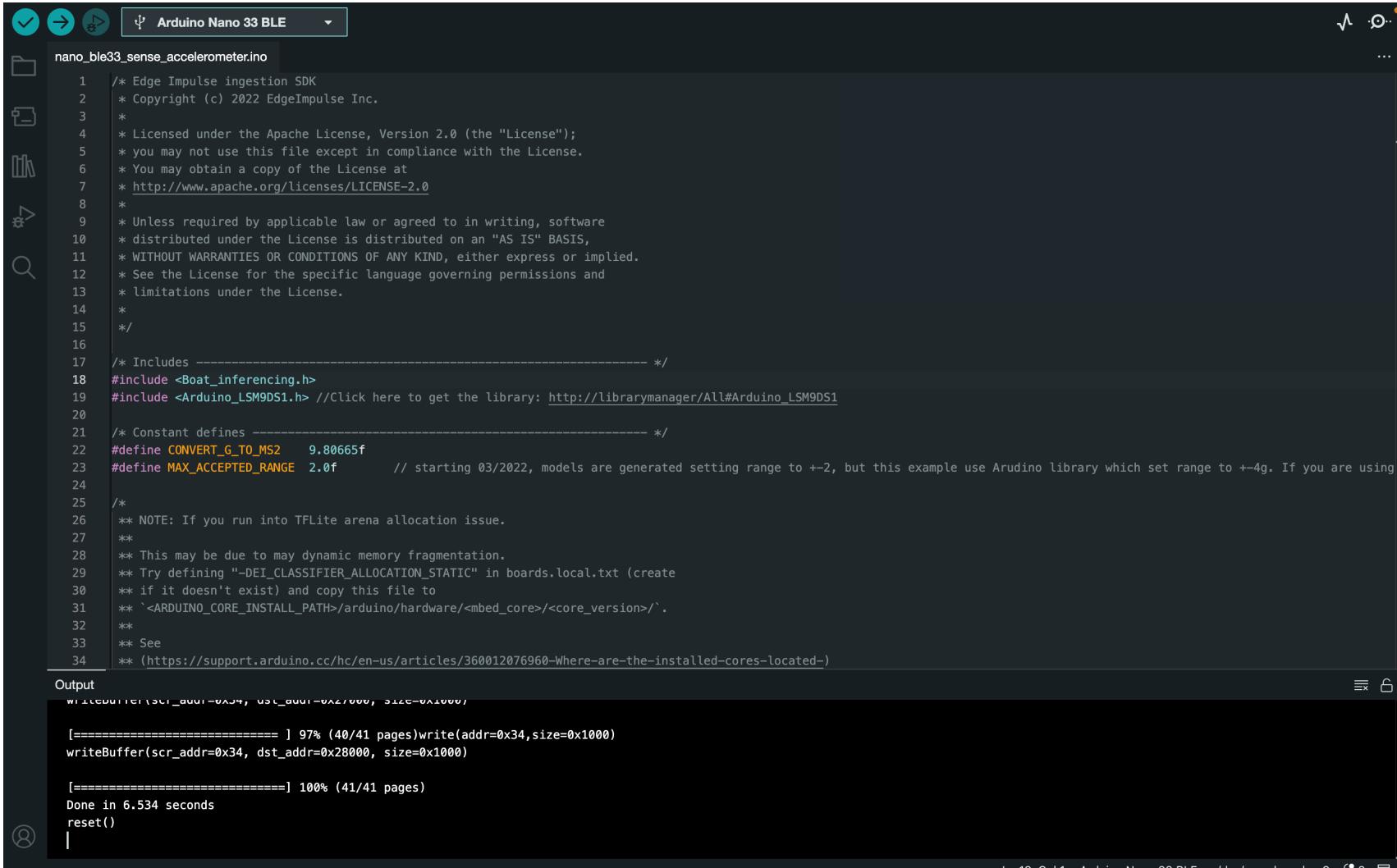
You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)

Create library

Turn your impulse into optimized source code that you can run on any device.

 C++ library	 Arduino library	 Cube.MX CMSIS-PACK
 WebAssembly	 TensorRT library	 Ethos-U library
 synaptics MetaTF Model	 brainchip	 TEXAS INSTRUMENTS

# Hardware Lab – Arduino Testing



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Arduino Nano 33 BLE
- File Explorer:** nano\_ble33\_sense\_accelerometer.ino
- Code Editor:** Displays C++ code for an Arduino sketch. The code includes comments about the Apache License, Version 2.0, and includes for Boat\_inferencing.h and Arduino\_LSM9DS1.h. It defines constants like CONVERT\_G\_TO\_MS2 (9.80665f) and MAX\_ACCEPTED\_RANGE (2.0f). A note at the bottom of the code section points to a support article about core location.
- Output Window:** Shows serial communication output:

```
WIFILEDGERI (SCI_BU001=0x34, DST_BU001=0x27000, S12C=0xA1000)

[=====] 97% (40/41 pages)write(addr=0x34,size=0x1000)
writeBuffer(scr_addr=0x34, dst_addr=0x28000, size=0x1000)

[=====] 100% (41/41 pages)
Done in 6.534 seconds
reset()
```
- Status Bar:** Ln 18, Col 1 Arduino Nano 33 BLE on /dev/cu.usbmodem6

