WEEK 1 – Foundations & IMU Gesture Recognition (Part 1)

1. Day 1 – Python Foundations for Edge AI

1.1. Overview

Day 1 establishes the programming foundation required for modern Edge AI systems. Python is central to data processing, model training, and embedded AI workflows. Participants gain confidence writing scripts, organizing project files, handling sensor logs, and building processing utilities that will be used throughout the gesture and keyword spotting modules.

1.2. Key Learning Areas

1.2.1.Python Essentials

Participants are introduced to fundamental Python constructs used throughout Edge AI development:

- **Variables & Data Types**
  How Python manages numbers, text, booleans, and floating-point IMU readings. Understanding type behavior enables efficient manipulation of sensor streams.
- **Lists, Dictionaries & Tuples**
  These structures form the backbone of dataset organization:
  • *Lists* store sequences of IMU samples
  • *Dictionaries* map labels to gesture categories
  • *Tuples* hold fixed sensor triads (ax, ay, az)

This foundation is essential for working with time-series data, activity labels, and extracted features.

1.2.2.Writing Reusable Functions

Participants learn to structure logic into functions to reduce repetition and improve clarity. Examples include windowing functions, feature extractors, file loaders, and transformation utilities.
This modular practice follows principles recognized in software engineering research (Lämmel et al., Empirical Software Engineering, 2006).

1.2.3.Working with Files

Participants explore how Python reads and writes CSV/TXT logs, allowing IMU data to move through cleaning, annotation, and batching stages. They learn how to:

- Systematically parse large datasets
- Save structured feature files
- Maintain consistent folder organization

1.2.4.Navigating the File System with Python's OS Module

Efficient dataset management includes iterating through session folders, merging recordings, and preparing training/validation splits.

1.3. What Participants Achieve by the End of Day 1

- Ability to write and structure Python programs
- Confidence in managing IMU sensor logs and project files
- Capability to construct reusable data-processing utilities
- Solid foundation for all gesture and audio modules

2. Day 2 – Data Operations with NumPy (IMU Processing)

2.1. Overview

NumPy enables fast numerical computation for IMU sensors. Participants learn array manipulation, vectorized computation, and statistical operations essential for gesture recognition pipelines.

2.2. Key Learning Areas

2.2.1.Understanding NumPy Arrays

Participants understand the array as a high-performance structure for storing and manipulating IMU signals. They learn:

- Array creation from lists or files
- Manipulating multi-axis IMU samples
- Reshaping data into windowed gestures
- Indexing and slicing gesture segments
  This forms the basis of efficient pre-processing.

2.2.2.Vector and Scalar Operations

Participants explore vectorized mathematical operations over entire IMU windows:
• magnitude = $\sqrt{(ax^2 + ay^2 + az^2)}$
• derivative = diff(array)
• normalization and scaling

These operations are fundamental for gesture patterns.

2.2.3.Analytical APIs

Participants learn statistical methods including:

- min, max, peak detection
- mean, variance, RMS
- argmax for identifying peak gesture points

2.3. What Participants Achieve by the End of Day 2

- Ability to compute high-speed IMU transformations
- Understanding of features derived directly from arrays
- Foundation for feature extraction and deep learning pipelines

3. Day 3 – Data Wrangling with Pandas (Gesture Dataset Construction)

3.1. Overview

Participants learn to prepare structured, labeled datasets from raw IMU logs using Pandas. This day focuses on building clean gesture datasets ready for modeling.

3.2. Key Learning Areas

3.2.1.Importing and Organizing Data

Participants load multi-session gesture recordings into DataFrames and understand:

- Timestamp alignment
- Sensor column structure
- Grouping and tagging motion intervals

3.2.2.Accessing and Filtering Data

Participants learn to isolate:

- Specific gesture segments
- Individual sensor axes
- Noise or non-gesture periods

This is essential for trimming and cleaning session recordings.

3.2.3.Aggregation & Summary Operations

Participants compute descriptive statistics for each gesture, supporting both exploratory data analysis and model validation.

3.3. What Participants Achieve by the End of Day 3

- Ability to clean and prepare high-quality gesture datasets

- Structured organization of IMU logs for modeling
- Competence in Pandas-based time-series manipulation

4. Day 4 – Data Visualization with Matplotlib (Gesture Waveform Analysis)

4.1. Overview

Visualization reveals gesture structure, consistency, and signal quality. Participants learn to interpret IMU waveforms and overlay features.

4.2. Key Learning Areas

4.2.1.Plotting Time-Series Data

Participants visualize multi-axis accelerometer and gyroscope gestures to understand:

- Typical waveform shapes
- Gesture amplitude differences
- Temporal alignment

4.2.2.Scatter & Histogram Visualization

Participants interpret statistical distribution and variability in gesture execution.

4.2.3.Overlaying Features

Graphs with combined raw and derived features help confirm whether selected features are discriminative.

4.3. What Participants Achieve by the End of Day 4

- Ability to visualize gesture dynamics
- Insight into motion characteristics
- Capability to debug dataset quality

5. Day 5 – IMU Fundamentals, Hardware Setup & Gesture Data Logging

5.1. Overview

Participants learn IMU physics, practical hardware considerations, and real gesture data collection.

5.2. Key Learning Areas

5.2.1.TinyML Concepts & Applications

Real examples:

- Wrist gestures
- Swipe detection
- Wearable control systems

5.2.2.IMU Sensor Fundamentals

Participants understand:

- Sampling rate selection (25–200 Hz)
- Sensor noise & drift
- Gravity vs motion components
- Axes orientation

5.2.3.Hardware Setup & Data Logging

Participants set up Arduino Nano 33 BLE Sense or ESP32:

- Configure sampling
- Stream IMU sensor data
- Capture labeled gesture datasets

5.3. What Participants Achieve by the End of Day 5

- Clear understanding of IMU operation
- Ability to log real gestures
- Well-structured dataset for Week 2 modeling

WEEK 2 – IMU Gesture Recognition (Part 2: Feature Engineering to Deployment)

6. Day 6 – Feature Extraction for IMU Gesture Recognition

6.1. Overview

Day 6 focuses on transforming raw IMU signals into high-quality features that improve model accuracy.

6.2. Key Learning Areas

6.2.1.Importance of Feature Engineering

Participants analyze why handcrafted features often outperform raw signals in microcontroller contexts.

6.2.2.Statistical Features

Participants compute:

- Mean
- variance
- std

6.2.3.Sliding Window Techniques

Window-based segmentation increases temporal resolution and supports continuous gesture recognition.

6.2.4.Feature Visualization

Scatter and box plots help evaluate feature separability.

6.3. What Participants Achieve by the End of Day 6

- Full feature extraction pipeline
- Understanding of feature importance
- Well-prepared feature datasets

7. Day 7 – ML Model Training with TensorFlow/Keras

7.1. Overview

Participants train neural networks for gesture classification.

7.2. Key Learning Areas

7.2.1.Neural Network Foundations

Overview of suitable architectures:

- MLP for features
- CNN for temporal patterns
- LSTM for dynamic sequences

7.2.2.Model Training Workflow

Participants learn:

- Train/validation/test splitting
- Batch processing
- Loss curves & overfitting prevention

7.2.3.Evaluation Metrics

Participants interpret confusion matrices, accuracy, recall, and per-class performance.

7.3. What Participants Achieve by the End of Day 7

- Fully trained gesture classifier
- Understanding model strengths and weaknesses

8. Day 8 – Model Conversion & Deployment Preparation

8.1. Overview

Participants convert their gesture model to a microcontroller-compatible TFLite version.

8.2. Key Learning Areas

8.2.1.TensorFlow Lite Conversion

Steps for reducing model complexity:

- Float → TFLite
- TFLite → int8 quantized

8.2.2.Quantization Techniques

Reduction in size, memory footprint, and inference latency.

8.2.3.Comparing Model Footprints

Participants understand trade-offs between accuracy and efficiency.

8.3. What Participants Achieve by the End of Day 8

- Fully optimized gesture model
- Ready for deployment on constrained hardware

9. Day 9 – Running Gesture Inference on Arduino

9.1. Overview

Participants learn to execute gesture classification live on microcontrollers.

9.2. Key Learning Areas

9.2.1.Flashing TFLite Models

Integrating the model into C++ firmware.

9.2.2.Real-Time Inference Pipeline

Participants execute:

- IMU streaming
- Window buffering
- Feature extraction
- Model prediction

9.2.3.Feedback Integration

Actions triggered by gestures include:

- LEDs
- Serial output
- Motor triggers

9.3. What Participants Achieve by the End of Day 9

- Working gesture recognition system
- Understanding of the full inference loop

10. Day 10 – Gesture Recognition Final Project

10.1.Overview

Participants deliver a complete IMU gesture system.

10.2.Key Learning Areas

10.2.1.Optimizing for Speed & Power

Microcontroller-specific tuning.

10.2.2.Advanced Gesture Applications

- Gesture sequences
- Personalized gesture training
- Multi-gesture prediction

10.2.3.Final Project Implementation

Participants combine all techniques into a functional real-time system.

10.3.What Participants Achieve by the End of Day 10

- End-to-end gesture recognition project
- Deployment-ready TinyML solution

WEEK 3 – Keyword Spotting Fundamentals

11. Day 11 – Introduction to Speech & Audio Processing

11.1.Overview

Understanding speech signals as time-series data for keyword spotting applications.

11.2.Key Learning Areas

- Anatomy of audio waveforms
- Sampling rate, frames, windowing
- Challenges in real-time speech detection

11.3.What Participants Achieve

- Understanding of fundamental audio concepts
- Ability to evaluate speech datasets

12. Day 12 – Spectrograms & Mel-Frequency Features

12.1.Overview

Participants convert raw audio into model-ready representations.

12.2.Key Learning Areas

- STFT and spectral analysis
- Mel filterbanks
- Log-Mel spectrograms
- Parameter selection for TinyML (window, hop, FFT size)

12.3.What Participants Achieve

- Ability to generate spectrogram features for KWS models

13. Day 13 – Keyword Spotting Dataset Creation

13.1.Overview

Participants build a complete dataset with multiple keywords.

13.2.Key Learning Areas

- Recording keywords

- Labeling audio samples
- Adding background noise, silence, and augmentation

13.3.What Participants Achieve

- A clean, balanced multi-class audio dataset

14. Day 14 – Training CNN Models for KWS

14.1.Overview

Participants train small-footprint CNN architectures optimized for microcontrollers.

14.2.Key Learning Areas

- DS-CNN architecture
- 1D vs 2D CNNs for audio
- Model training workflow

14.3.What Participants Achieve

- A trained KWS classifier

15. Day 15 – TFLite Conversion & Optimization (Audio)

15.1.Overview

Participants prepare their audio model for embedded deployment.

15.2.Key Learning Areas

- Full integer quantization
- Reducing model size
- Balancing speed vs accuracy

15.3.What Participants Achieve

- Deployment-ready KWS model

WEEK 4 – Keyword Spotting Deployment & Final Project

16. Day 16 – Streaming Audio & On-Device Feature Extraction

16.1.Overview

Participants implement spectrogram extraction on embedded hardware.

16.2.Key Learning Areas

- Audio pipeline logic
- Memory-efficient buffering

16.3.What Participants Achieve

- Fully functional on-device audio frontend

17. Day 17 – Real-Time Keyword Detection Pipeline

17.1.Overview

Participants integrate audio features with ML inference.

17.2.Key Learning Areas

- Prediction smoothing
- Trigger thresholds
- Handling continuous recognition

17.3.What Participants Achieve

- Real-time KWS system

18. Day 18 – Optimization for Embedded Speech AI

18.1.Overview

Participants tune their system for reliability and efficiency.

18.2.Key Learning Areas

- Latency reduction
- Power optimization
- Reducing false triggers

18.3.What Participants Achieve

- A polished, optimized KWS pipeline

19. Day 19 – Voice-Activated Edge AI System Integration

19.1.Overview

Participants build full voice-controlled embedded workflows.

19.2.Key Learning Areas

- speech systems
- Real-world noise handling

19.3.What Participants Achieve

- Robust voice interaction system

20. Day 20 – Keyword Spotting Final Project

20.1.Overview

Participants design, implement, test, and present a keyword-based voice application.

20.2.Key Learning Areas

- Multi-keyword system design
- Interface actions (LEDs, motors, wireless triggers)
- Demo-ready deployment

20.3.

20.4.What Participants Achieve

- A complete voice-controlled TinyML project
- Mastery across audio processing, model training, and deployment