The smart physiotherapist

Authors: **Alberto Marrone and Etienne Marie Rault**

Academic Year: 2024-25

Abstract

This report details the development of an innovative and non-invasive system for recognizing specific human motor activities, with a primary focus on remote patient monitoring and assistance within rehabilitation contexts. Leveraging Ultra-Wideband (UWB) radar sensors, the system acquires motion data, which is subsequently pre-processed into a visual format and classified through advanced neural network model optimized for efficient inference on resource-constrained edge devices. The core objective is to achieve high classification accuracy while adhering to stringent hardware limitations concerning memory footprint and processing time. We present a comprehensive methodology encompassing data acquisition from two distinct 3-antenna radar platforms, the SR250Mate and an Infineon sensor, along with a detailed explanation of the Python-based data pre-processing pipeline that transforms raw radar signals Python numpy-like files into suitable image inputs. The report further discusses the training and optimization of these models using the Edge Impulse platform. A significant contribution of this work is a comparative analysis of the two radar systems, specifically highlighting the challenges and comparatively lower performance encountered with the SR250Mate unit, providing justifications rooted in signal characteristics and hardware capabilities. Performance metrics, including classification accuracy, inference latency, and memory consumption, are thoroughly evaluated, particularly considering the target deployment on high-end Microcontroller Units (MCUs) such as the STM32 series or the Arduino Nano 33 BLE Sense, given the substantial size of our dataset and the computational demands.

Contents

Introduction

The precise and continuous monitoring of human movement is a cornerstone in a multitude of contemporary applications, spanning from security surveillance and advanced virtual reality environments to human-machine interfaces. Amongst these, the application of assisting and supervising patients within rehabilitative frameworks has garnered significant attention, representing a critical frontier in personalized healthcare. The capacity to objectively track, analyze, and evaluate the quality and correctness of specific motor movements can furnish invaluable, real-time feedback to both patients undergoing therapy and the clinicians overseeing their recovery. Such objective insights are instrumental in enhancing the efficacy of rehabilitation programs, accelerating patient progress, and fostering greater independence and adherence to prescribed exercises.

Traditional motion monitoring systems, typically reliant on optical systems (e.g., cameras) or wearable inertial sensors, frequently encounter practical limitations. These drawbacks include inherent privacy concerns stemming from visual data capture, susceptibility to environmental variables such as lighting conditions, and the practical inconvenience or discomfort associated with continuous physical attachment of sensors to the subject. In response to these challenges, Ultra-Wideband (UWB) radar sensors have emerged as a highly promising and innovative alternative. UWB radar technology offers distinct advantages, including non-invasive detection capabilities, inherent robustness to diverse ambient conditions (such as varying light levels, smoke, or even operation through non-metallic obstacles like clothing or thin walls), and a unique ability to detect subtle micro-movements that might be imperceptible to other sensing modalities. Their independence from visual line-of-sight and illumination, coupled with their privacy-preserving nature, renders them particularly well-suited for continuous healthcare monitoring applications where patient comfort, dignity, and autonomy are paramount.

This project embarks on the development of a novel radar-based human activity recognition (HAR) system, meticulously designed for the precise monitoring and classification of upper and lower limb movements within a rehabilitative context. The overarching objective is two-fold: firstly, to achieve exceptionally high classification accuracy for a predefined set of crucial rehabilitative movements, encompassing "left arm up", "right arm up", "left leg up", "right leg up", and a "still position" as a baseline. Secondly, and equally critically, the project aims to ensure that the resulting neural network is sufficiently compact, computationally efficient, and robust to facilitate the deployment on resource-constrained microcontrollers. While initial considerations for Edge AI projects often gravitate towards platforms like the Arduino Nano 33 BLE Sense due to their accessibility and low power consumption, the substantial size and inherent complexity of our collected dataset, coupled with stringent requirements for inference time and memory footprint, strongly indicate the necessity for more powerful, higher-end Microcontroller Units (MCUs), such as those within the STM32 series, to ensure optimal and effective real-world deployment. The project's success hinges on balancing model sophistication with the practical realities of edge computing.

A distinctive feature of this research involves the comparative utilization and analysis of two distinct UWB radar platforms: the SR250Mate and an Infineon radar sensor. Both of these state-of-the-art radar units are equipped with an array of three dedicated receiving antennas (RX0, RX1, RX2). This deliberate choice of employing multiple radar platforms is driven by a comprehensive desire to rigorously analyze and understand the intrinsic differences in the characteristics of the raw radar data acquired from sensors exhibiting varying underlying architectures, signal processing capabilities, and overall performance specifications. Crucially, this comparative approach allows for a direct evaluation of how these inherent hardware and signal distinctions directly influence the efficacy of the machine learning model training process and, ultimately, the final classification performance. A significant segment of this report will be dedicated to meticulously outlining the specific challenges, limitations, and comparatively lower performance consistently encountered when employing the SR250Mate radar unit in our experimental setup. We will provide detailed justifications for these observed discrepancies, grounding our explanations in the fundamental characteristics of the acquired radar signals, potential noise profiles, and the inherent capabilities of the SR250Mate sensor hardware itself.

The foundation of our model development rests upon an organized dataset. This dataset was collected with precise control, comprising approximately 100 individual input samples for each of the five predefined activity classes. During data collection, subjects were consistently positioned at an approximate distance of 1 meter directly in front of the active radar sensor. Each data acquisition session was configured to capture a precise 4-second window of continuous movement. Importantly, unlike methodologies that involve segmenting longer continuous recordings in post-processing, our data was specifically collected as discrete, self-contained 4-second single acquisitions, ensuring consistency and minimizing potential artifacts from manual segmentation. The raw, complex-valued data streaming from both radar units are intrinsically spatial-temporal matrices, stored efficiently as NumPy (‘.npy‘) files. These raw matrices represent the fundamental radar signatures of human movement.

To render these complex radar signals amenable to advanced machine learning algorithms, particularly Convolutional Neural Networks (CNNs), a dedicated Python-based pre-processing program was developed. This program plays a pivotal role in transforming the raw radar data into a visual, image-like format. The core functionality of this program involves:

- Loading the raw NumPy files from the radar sensors.
- Computing the magnitude of the complex-valued radar signals to obtain intensity information.
- Critically, selecting the most relevant "range bins" which correspond to the spatial area of interest (the first 40 range bins where human movement at 1 meter is most prominent).
- Combining the processed data streams from specific antenna configurations. For instance, the program is capable of concatenating data from RX0 and RX1 to form an 80-pixel wide image for preliminary analyses, or, more comprehensively, merging data from all three antennas (RX0, RX1, and RX2) to create a 120-pixel wide input image. This dynamic combination allows for flexible experimentation with different sensor data fusion strategies.
- Ensuring that the resulting images are appropriately sized (e.g., 125 pixels in height, truncating longer sequences, and 80 or 120 pixels in width depending on the antenna combination) while preventing the introduction of artificial padding or "black bars" by dynamically adjusting image dimensions based on actual data length.

The subsequent sections of this report will delve into the detailed methodologies employed for data collection, the intricate processes within the radar data pre-processing pipeline, the design and comprehensive training of the machine learning models using the Edge Impulse platform, and a rigorous evaluation of their real-world performance metrics. Finally, we will conclude with a concise summary of our key findings, offer insights into the limitations encountered, and propose promising avenues for future research and development.

1 | Data Pre-processing and intput Charateristic

Raw data from UWB radar sensors (SR250Mate and Infineon) are complex-valued, multi-dimensional arrays in NumPy (.npy) format, representing radar reflections across range bins over time. For CNN processing, these signals are transformed into grayscale images. This section details our custom Python scripts and the rationale for input characteristics.

1.1. Raw Data Format and Acquisition Details

Each .npy file from an RX antenna (RX0, RX1, RX2) typically has a shape of (N frames, 8, 128): N frames (4-second acquisition), 8 internal channels, and 128 range bins. All data was collected as discrete 4-second single acquisitions with subjects 1 meter from the radar.

1.2. Custom Python Pre-processing Pipeline

Our Python script processes these raw .npy files into grayscale images. While variations exist for different antenna combinations and radar units, their core functionality is consistent. We detail the comprehensive version for all three Infineon RX antennas.

The pipeline involves:

- Loading the npy file and computing the magnitude

  Raw complex-valued .npy files for each RX antenna are loaded. The magnitude of these complex numbers is computed to extract signal intensity, resulting in a real-valued array.

- Selection of Relevant Internal Channels and Range Bins

  To optimize data and reduce noise, we select only the most informative parts:

- Internal Channel Selection: Only the first internal channel (index 0) from the 8 available is used. This channel empirically contains the most salient motion data.
- Range Bin Selection (First 40 Bins): Only the first 40 bins (0:40) from the 128 are retained. At 1 meter, relevant motion information is concentrated here; remaining bins are largely irrelevant background noise ("redundant pixels"). This reduces each antenna's data to (N frames, 40).

• Horizontal Concatenation of Antenna Data

Processed data from selected RX antennas are horizontally concatenated to form a single image, leveraging spatial information from multiple perspectives:

- For RX0 and RX1 (80-pixel width): Concatenated as `np.hstack((data_float_rx0, data_float_rx1))`, resulting in an 80-pixel wide image.
- For RX0, RX1, and RX2 (120-pixel width): Concatenated as `np.hstack((data_float_rx0, data_float_rx1, data_float_rx2))`, yielding a 120-pixel wide image. This width captures comprehensive spatial data from three channels.

We ensure all RX antenna arrays have consistent N frames; inconsistencies lead to data exclusion.

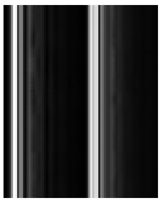
• Dynamic Image Sizing and Black Bar Prevention

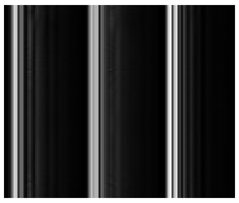
To avoid artificial padding or "black bars," image height dynamically adjusts to the actual N frames (up to 125 pixels, for 4 seconds at 25 Hz). If N_frames exceeds 125, data is truncated.

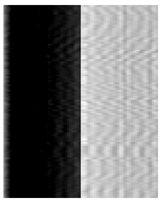
• Image Saving

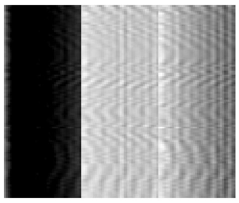
The final image data is saved as a grayscale PNG using `matplotlib.pyplot`. The figure is configured for exact pixel dimensions, without axes or borders, ensuring only radar data is visible. Filenames include the activity label and original acquisition ID.

This pipeline consistently transforms raw radar data into high-quality, normalized grayscale images, optimized for deep learning models on edge devices, while effectively managing dimensionality.


(a) SR250Mate RX0 + RX1


(b) SR250Mate RX0 + RX1+RX2


(c) Infineon RX0+RX1


(d) Infineon RX0+RX1+RX2

2 | Model Development and Initial Classification Trials

The core of our human activity recognition system relies on robust Machine Learning models capable of accurately classifying movements from radar data. For the development, training, and deployment of these models, we utilized Edge Impulse, a leading platform for TinyML that streamlines the entire machine learning pipeline for edge devices.

2.1. Edge Impulse Platform Setup

Upon generating the pre-processed image datasets using our custom Python scripts (as detailed in Section 2), these images were uploaded to the Edge Impulse studio. During the upload process, each image was automatically assigned a corresponding label based on its filename (e.g., `"left_arm_up"`, `"right_arm_up"`, `"left_leg_up"`, `"right_leg_up"`, `"still_position"`). This systematic labeling, embedded within our Python script's output, ensured that the dataset was correctly categorized for supervised learning.

Once uploaded, the entire dataset, comprising hundreds of labeled images for each of the five activity classes, was automatically split within Edge Impulse into an 80% training set and a 20% testing set. This standard split ensures that the model is trained on a substantial portion of the data while reserving a distinct, unseen portion for unbiased performance evaluation.

Our primary target deployment device was the Arduino Nano 33 BLE Sense. However, due to the complexity of our dataset, and considering the computational demands of the neural networks, initial assessments by our tutor suggested that the Arduino Nano 33 BLE Sense might be slightly underpowered. This led to the consideration of more capable, high-end Microcontroller Units (MCUs) such as those from the STM32 series for final deployment, which offer greater memory and processing power.

Within Edge Impulse, the machine learning pipeline is structured around "Impulses."

An Impulse represents a complete signal processing and machine learning pipeline, where each Impulse is essentially dedicated to a specific model configuration. For all our initial model iterations, including the first one described below, certain standard blocks were configured:

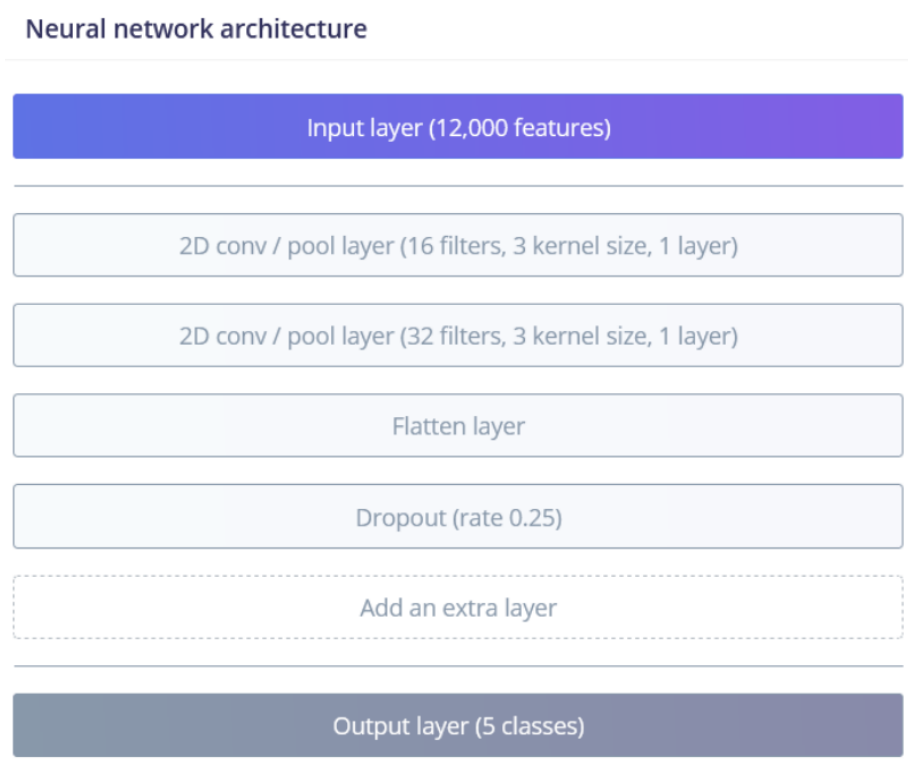
- Image Data Block: this block defines the input characteristics for the neural network. For all our models, the input images were configured as grayscale. The configuration specified how the input features (raw pixels from our pre-processed images) would be fed into the learning block.
- Classification Block: this block defines the Neural Network Classifier that will be trained on the processed features. The specific architecture of this classifier varies for each model iteration, as detailed in the following subsections.

2.2. First model architecture and training configuration

Our initial approach involved developing a Convolutional Neural Network (CNN) with a straightforward architecture, designed to establish a baseline performance for our radar-based Human Activity Recognition. This first model's architecture and training settings were configured as follows within the Edge Impulse NN Classifier block:

Neural Network Architecture: the model's architecture, visualized in the Edge Impulse environment (as shown in figure 2.1), consists of a series of convolutional and pooling layers, followed by dense layers for classification.

- Input Layer: the network accepts grayscale images.
- Convolutional Layer 1 (Conv2D): this layer applies 16 filters with a 3x3 kernel size, using a Rectified Linear Unit (ReLU) activation function.
- Convolutional Layer 2 (Conv2D): another convolutional layer with 32 filters, a 3x3 kernel, and ReLU activation.
- Flatten Layer: the output from the convolutional and pooling layers is flattened into a 1D vector.
- Dropout Layer: a Dropout layer with a rate of 0.25, applied to prevent overfitting by randomly setting a fraction of input units to zero during training.
- Dense Layer 2 (Output Layer): The final fully connected layer with 5 neurons, corresponding to our five activity classes (left arm up, right arm up, left leg up, right leg



Neural network architecture

Input layer (12,000 features)

2D conv / pool layer (16 filters, 3 kernel size, 1 layer)

2D conv / pool layer (32 filters, 3 kernel size, 1 layer)

Flatten layer

Dropout (rate 0.25)

Add an extra layer

Output layer (5 classes)

Figure 2.1: First Neural Network attempt

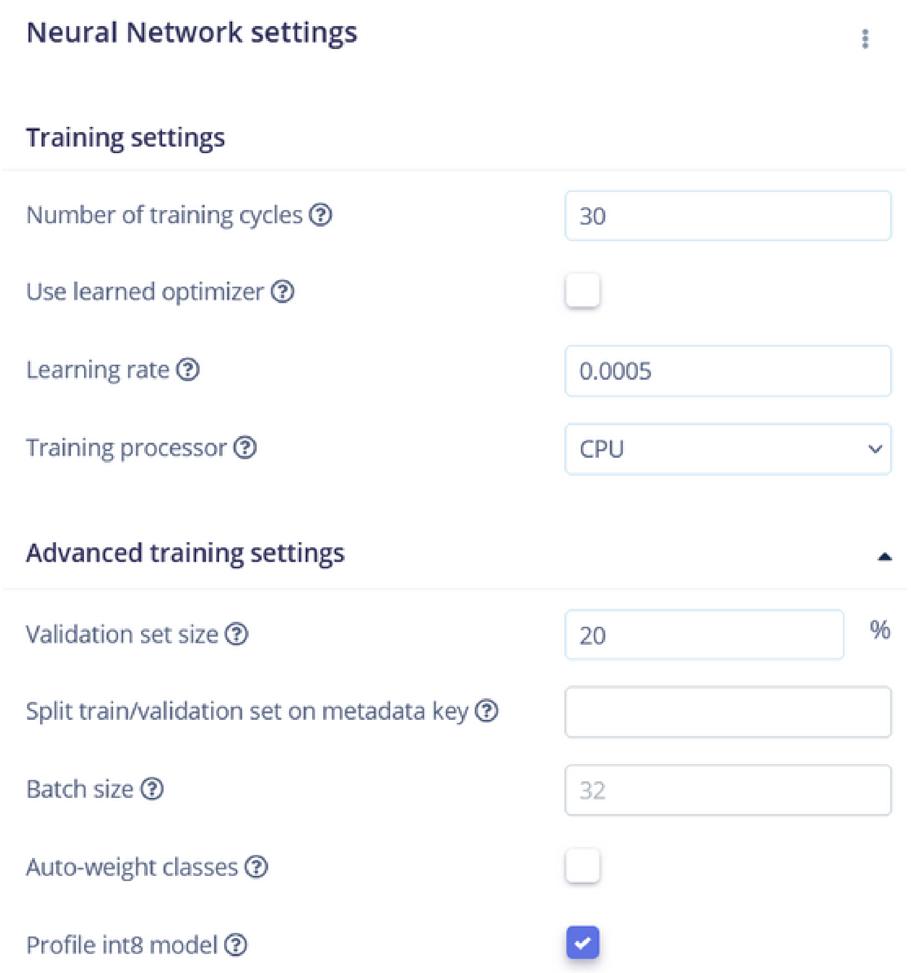
up, still position), and a Softmax activation function for multi-class classification.

Training Settings: The training process for this model was configured with the following parameters (as shown in figure 2.2):

- Number of Training Cycles (Epochs): 30
- Learning Rate: 0.0005
- Validation Set Size: 20% (split from the training data for validation during training)

2.3. Performance evaluation of the first model across datasets

This specific neural network model configuration, as detailed previously, was systematically evaluated against four distinct datasets, each representing different radar units and antenna configurations. For each dataset, the model's performance was assessed based on classifier (training/validation) accuracy, training loss, inference time, peak RAM usage, and Flash usage, as well as the crucial model testing accuracy on unseen data. The detailed results, including confusion matrices and accuracy/loss plots from Edge Impulse,



Neural Network settings

Training settings

Number of training cycles	30
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Figure 2.2: Parameter settings

are presented below.

Results for dataset 1 (SR250Mate RX0+RX1 Data)

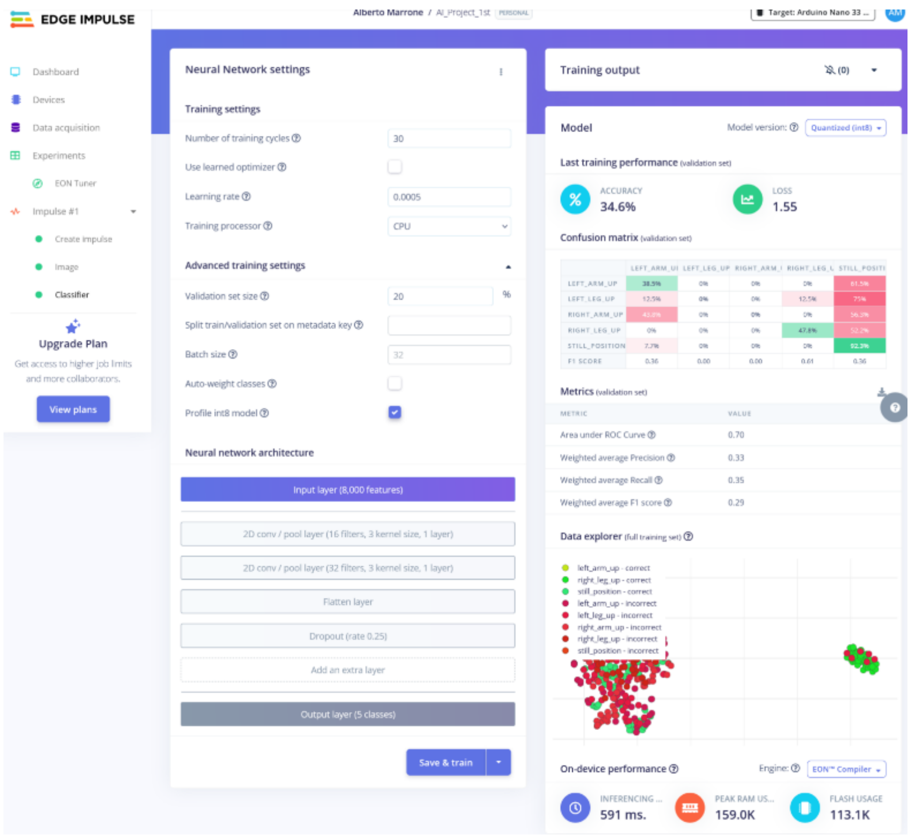
For the SR250Mate radar using data from RX0 and RX1 antennas:

- Classifier Performance: The model achieved a training accuracy of 34.6% with a loss of 1.55 (see 2.3).
- On-Device Performance (Estimated by Edge Impulse):
  - Inference Times : 591ms
  - Peak RAM usage: 159 kB
  - Flash usage: 113.1 kB
- Model Testing Accuracy: Unfortunately, the model exhibited a 0% accuracy on the unseen test set.
- Conclusion: the model is not working well on this dataset but all the resource limits have been respected (see 2.4).

Results for Dataset 2 (SR250Mate RX0+RX1+RX2 Data)

For the SR250Mate radar using data from all three RX0, RX1, and RX2 antennas:

- Classifier Performance: The model achieved a training accuracy of 38.3% with a loss of 1.30. This indicates a slight improvement in training over the RX0+RX1 configuration.
- On-Device Performance (Estimated by Edge Impulse) (see 2.5):
  - Inference Time: 864 ms
  - Peak RAM Usage: 237.1 KB
  - Flash Usage: 152.2 KB
- Model Testing Accuracy: Similar to the previous dataset, the model showed a 0% accuracy on the unseen test set (see 2.6).
- Conclusion: Same as for the first training/test. The the peak RAM usage and the flash usage limits are respected but the inference time is not respected.



Neural Network settings

Training settings

Number of training cycles	30
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (8,000 features)
2D conv / pool layer (16 filters, 3 kernel size, 1 layer)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
Flatten layer
Dropout (rate 0.25)
Add an extra layer
Output layer (5 classes)

Training output

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 34.6%   LOSS 1.55

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_I	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	38.5%	0%	0%	0%	61.5%
LEFT_LEG_UP	12.5%	0%	0%	12.5%	75%
RIGHT_ARM_UP	43.8%	0%	0%	0%	56.3%
RIGHT_LEG_UP	0%	0%	0%	47.8%	52.2%
STILL_POSITION	7.7%	0%	0%	0%	92.3%
F1 SCORE	0.36	0.00	0.00	0.61	0.36

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	0.70
Weighted average Precision	0.33
Weighted average Recall	0.35
Weighted average F1 score	0.29

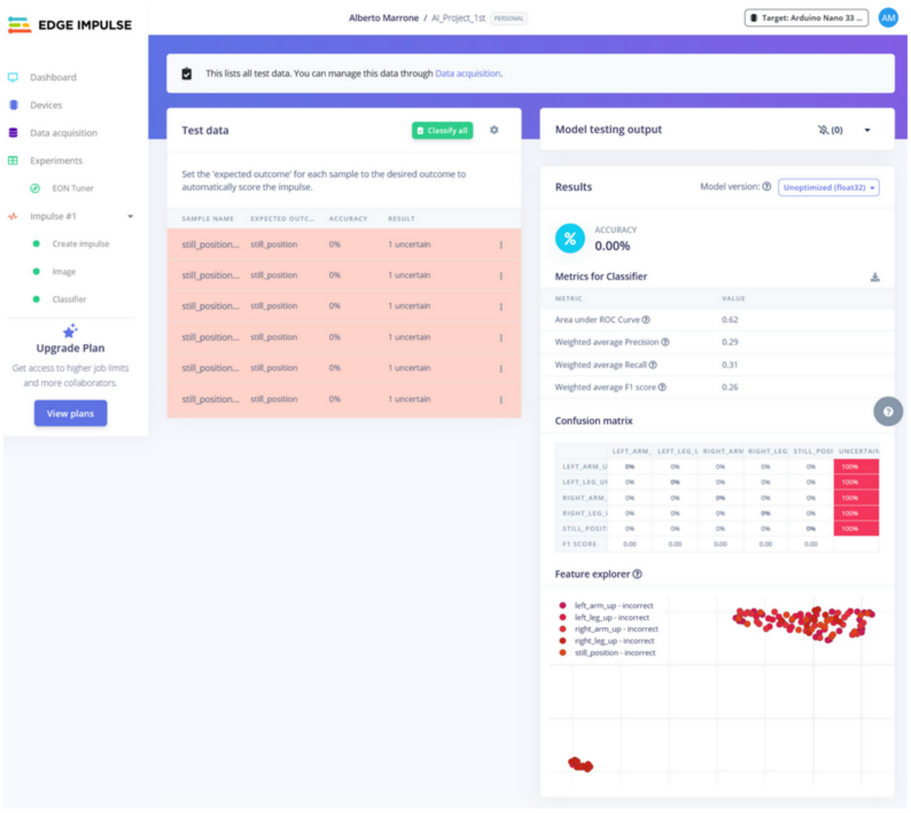
Data explorer (full training set)

- ● left_arm_up - correct
- ● right_leg_up - correct
- ● still_position - correct
- ● left_arm_up - incorrect
- ● left_leg_up - incorrect
- ● right_arm_up - incorrect
- ● right_leg_up - incorrect
- ● still_position - incorrect

On-device performance — Engine: EON™ Compiler

INFERENCING ... 591 ms.   PEAK RAM US... 159.0K   FLASH USAGE 113.1K

Figure 2.3: Results for the training on dataset 1



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain

Model testing output

Results

ACCURACY
0.00%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.62
Weighted average Precision	0.29
Weighted average Recall	0.31
Weighted average F1 score	0.26

Confusion matrix

	LEFT_ARM_	LEFT_LEG_U	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	0%	0%	0%	0%	0%	100%
LEFT_LEG_UI	0%	0%	0%	0%	0%	100%
RIGHT_ARM_	0%	0%	0%	0%	0%	100%
RIGHT_LEG_I	0%	0%	0%	0%	0%	100%
STILL_POSIT	0%	0%	0%	0%	0%	100%
F1 SCORE	0.00	0.00	0.00	0.00	0.00	

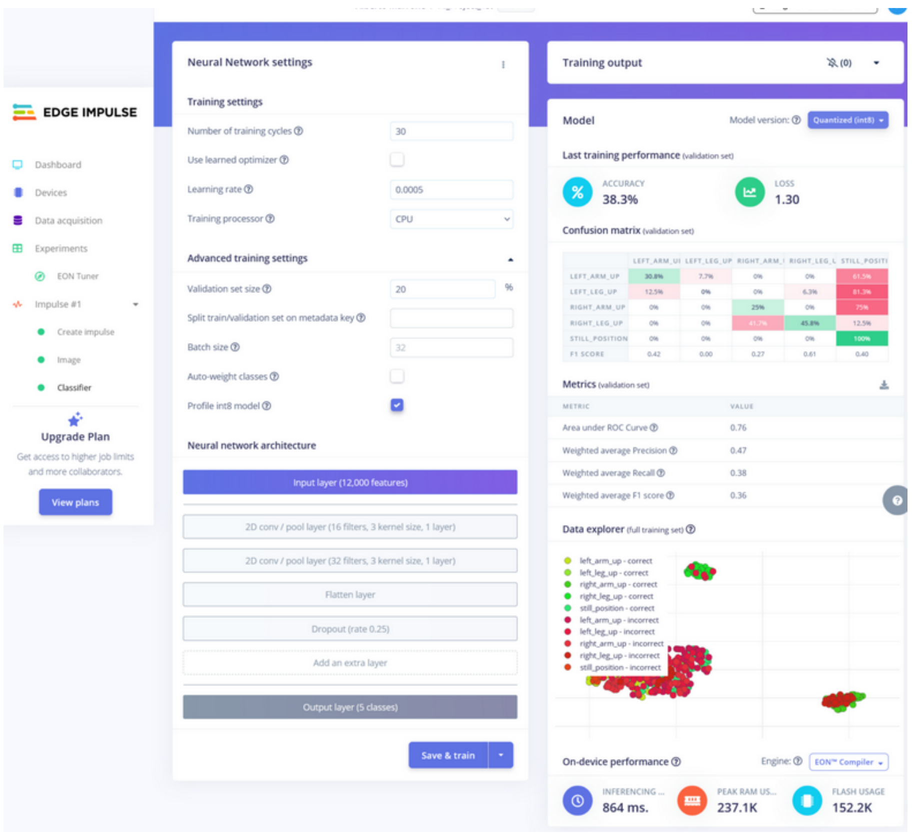
Feature explorer

Figure 2.4: Results for the test on dataset 1

Results for Dataset 3 (Infineon RX0+RX1 Data)

For the Infineon radar using data from RX0 and RX1 antennas:

- Classifier Performance: The model achieved a training accuracy of 80.2% with a loss of 0.86, indicating a much stronger learning phase compared to SR250Mate.
- On-Device Performance (Estimated by Edge Impulse) (see 2.7):
  - Inference Time: 591 ms
  - Peak RAM Usage: 159 kB
  - Falsh Usage: 113.1 kB
- Model Testing Accuracy: The model yielded a test accuracy of 16.83%. While significantly better than SR250Mate, it is still relatively low (see 2.8).
- Conclusion: even if the training accuracy is high, the score obtained for the test is quite low. All resource requirements have been met.



EDGE IMPULSE

Neural Network settings

Training settings

Number of training cycles	30
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (12,000 features)
2D conv / pool layer (16 filters, 3 kernel size, 1 layer)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
Flatten layer
Dropout (rate 0.25)
Add an extra layer
Output layer (5 classes)

Training output (0)

Model Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 38.3%

LOSS 1.30

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_I	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	30.8%	7.7%	0%	0%	61.5%
LEFT_LEG_UP	12.5%	0%	0%	6.3%	81.3%
RIGHT_ARM_UP	0%	0%	25%	0%	75%
RIGHT_LEG_UP	0%	0%	41.7%	45.8%	12.5%
STILL_POSITION	0%	0%	0%	0%	100%
F1 SCORE	0.42	0.00	0.27	0.61	0.40

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	0.76
Weighted average Precision	0.47
Weighted average Recall	0.38
Weighted average F1 score	0.36

Data explorer (full training set)

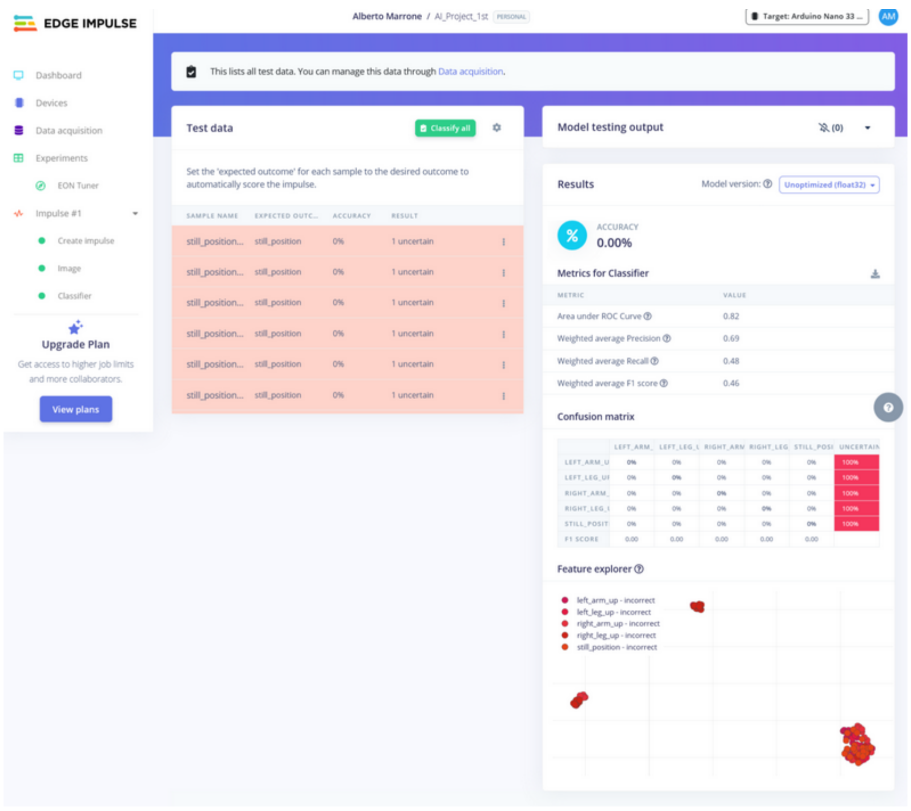
On-device performance Engine: EON™ Compiler

INFERENCING ... 864 ms.

PEAK RAM US... 237.1K

FLASH USAGE 152.2K

Figure 2.5: Results for the training on dataset 2



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain

Model testing output

Results

ACCURACY
0.00%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.82
Weighted average Precision	0.69
Weighted average Recall	0.48
Weighted average F1 score	0.46

Confusion matrix

	LEFT_ARM_	LEFT_LEG_L	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	0%	0%	0%	0%	0%	100%
LEFT_LEG_UI	0%	0%	0%	0%	0%	100%
RIGHT_ARM_	0%	0%	0%	0%	0%	100%
RIGHT_LEG_I	0%	0%	0%	0%	0%	100%
STILL_POSIT	0%	0%	0%	0%	0%	100%
F1 SCORE	0.00	0.00	0.00	0.00	0.00	

Feature explorer

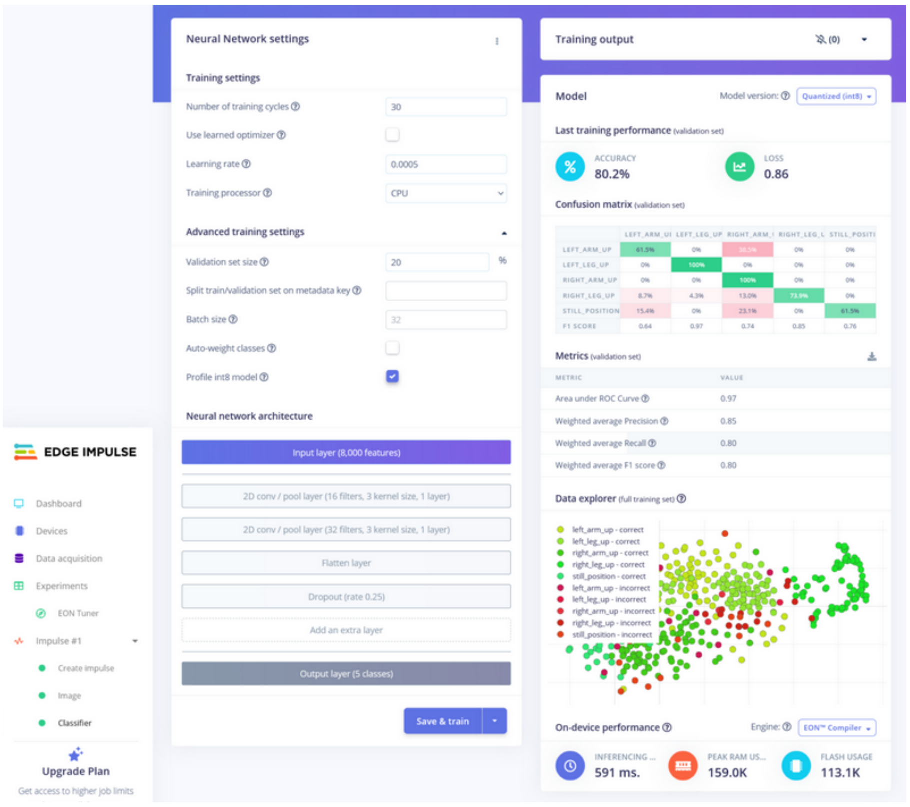
- ● left_arm_up - incorrect
- ● left_leg_up - incorrect
- ● right_arm_up - incorrect
- ● right_leg_up - incorrect
- ● still_position - incorrect

Figure 2.6: Results for the test on dataset 2

Results for Dataset 4 (Infineon RX0+RX1+RX2 Data)

For the Infineon radar using data from all three RX0, RX1, and RX2 antennas:

- Classifier Performance: The model achieved an improved training accuracy of 85.0% with a loss of 0.71, marking the best training performance so far.
- On-Device Performance (Estimated by Edge Impulse) (see 2.9):
  - Inference Time: 591 ms
  - Peak RAM Usage: 159 kB
  - Flash Usage: 113.1 kB
- Model Testing Accuracy: The test accuracy reached 39.25% (see 2.10), representing a substantial improvement over other datasets and the best performance among all initial tests.
- Conclusion: same as for the data set 3. But now the accuracy is much more better than before. It shows that adding an antenna might be a key element in order to get better data and so a better classifier.



Neural Network settings

Training settings

Number of training cycles	30
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	☑

Neural network architecture

Input layer (8,000 features)
2D conv / pool layer (16 filters, 3 kernel size, 1 layer)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
Flatten layer
Dropout (rate 0.25)
Add an extra layer
Output layer (5 classes)

Training output (0)

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 80.2%
LOSS 0.86

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_I	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	61.5%	0%	38.5%	0%	0%
LEFT_LEG_UP	0%	100%	0%	0%	0%
RIGHT_ARM_UP	0%	0%	100%	0%	0%
RIGHT_LEG_UP	8.7%	4.3%	13.0%	73.9%	0%
STILL_POSITION	15.4%	0%	23.1%	0%	61.5%
F1 SCORE	0.64	0.97	0.74	0.85	0.76

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	0.97
Weighted average Precision	0.85
Weighted average Recall	0.80
Weighted average F1 score	0.80

Data explorer (full training set)

On-device performance — Engine: EON™ Compiler

INFERENCING ... 591 ms.
PEAK RAM US... 159.0K
FLASH USAGE 113.1K

Figure 2.7: Results for the training on dataset 3

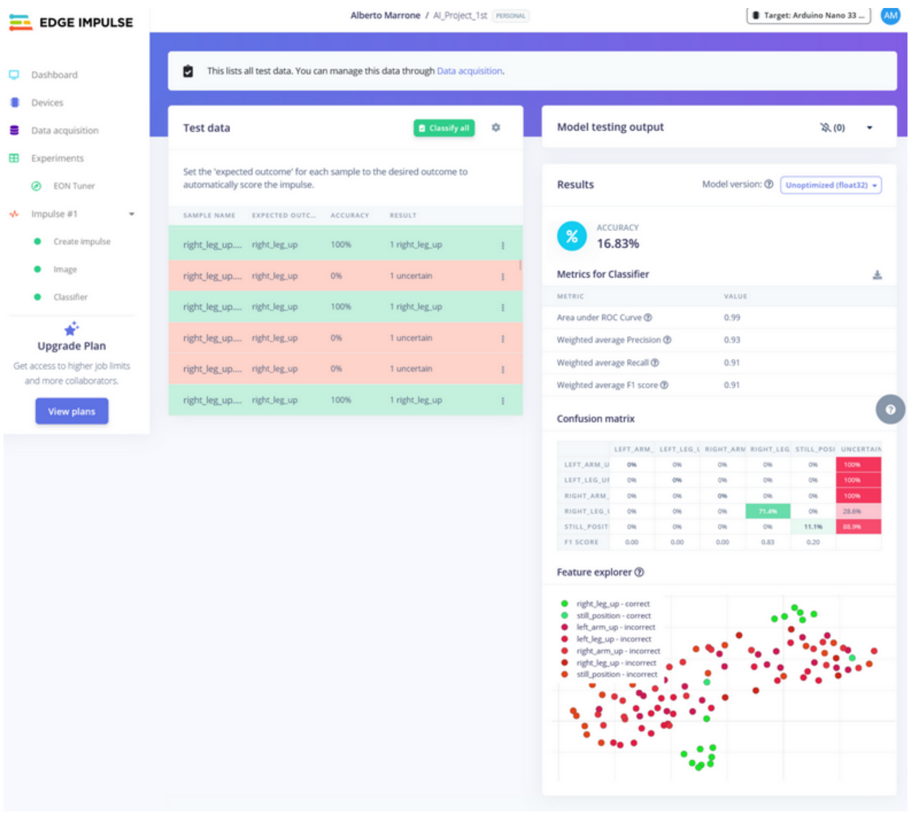
2.4. Performance Analysis

The initial evaluation of our first neural network model across the four diverse datasets reveals a critical dichotomy in performance, primarily dictated by the radar hardware used, rather than the number of receiving antennas processed (i.e., RX0+RX1 versus RX0+RX1+RX2). This analysis will clarify the distinction between classifier and model testing accuracy, discuss the causes for the observed performance, and highlight the implications for edge device deployment.

Classifier Accuracy vs. Model Testing Accuracy

It is crucial to differentiate between the "classifier accuracy" reported during the training phase in Edge Impulse and the "model testing accuracy" obtained on the unseen test set.

- Classifier Accuracy (Training/Validation Accuracy): this metric reflects how well the model is learning from the data it has been exposed to during training. It is typically a higher value, indicating the model's ability to classify patterns within the training dataset itself, including a validation subset.
- Model Testing Accuracy: this is the most critical metric for evaluating a model's



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
right_leg_up....	right_leg_up	100%	1 right_leg_up
right_leg_up....	right_leg_up	0%	1 uncertain
right_leg_up....	right_leg_up	100%	1 right_leg_up
right_leg_up....	right_leg_up	0%	1 uncertain
right_leg_up....	right_leg_up	0%	1 uncertain
right_leg_up....	right_leg_up	100%	1 right_leg_up

Model testing output

Results

ACCURACY 16.83%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.93
Weighted average Recall	0.91
Weighted average F1 score	0.91

Confusion matrix

	LEFT_ARM_	LEFT_LEG_U	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	0%	0%	0%	0%	0%	100%
LEFT_LEG_UI	0%	0%	0%	0%	0%	100%
RIGHT_ARM_	0%	0%	0%	0%	0%	100%
RIGHT_LEG_I	0%	0%	0%	71.4%	0%	28.6%
STILL_POSIT	0%	0%	0%	0%	11.1%	88.9%
F1 SCORE	0.00	0.00	0.00	0.83	0.20	

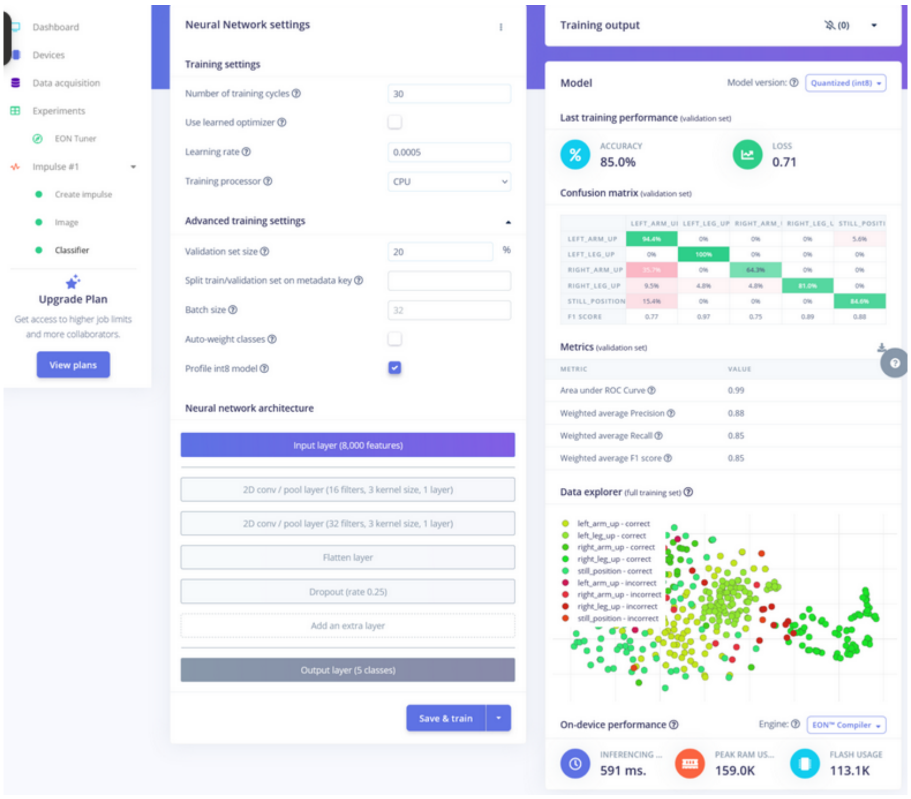
Figure 2.8: Results for the test on dataset 3

real-world applicability. It represents the model's performance on a completely new, unseen dataset (the 20% test split). A low testing accuracy, even with high classifier accuracy, indicates overfitting (the model has memorized the training data rather than learning generalizable patterns) or that the training data itself is not sufficiently representative or discriminative for the task.

On-Device resource consumption: A noteworthy observation across all four datasets and their respective trials is the consistency in the estimated on-device resource usage and inference time.

- Inference Time: 591 ms (for RX0+RX1 models) and 864 ms (for RX0+RX1+RX2 models).
- Peak RAM Usage: 159 KB (for RX0+RX1 models) and 237.1 KB (for RX0+RX1+RX2 models).
- Flash Usage: 113.1 KB (for RX0+RX1 models) and 152.2 KB (for RX0+RX1+RX2 models).

These values demonstrate that, for this initial model architecture, the resource require-



Neural Network settings

Training settings

Number of training cycles	30
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (8,000 features)
2D conv / pool layer (16 filters, 3 kernel size, 1 layer)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
Flatten layer
Dropout (rate 0.25)
Add an extra layer
Output layer (5 classes)

Training output

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 85.0% LOSS 0.71

Confusion matrix (validation set)

	LEFT_ARM_UP	LEFT_LEG_UP	RIGHT_ARM_UP	RIGHT_LEG_UP	STILL_POSITION
LEFT_ARM_UP	94.4%	0%	0%	0%	5.6%
LEFT_LEG_UP	0%	100%	0%	0%	0%
RIGHT_ARM_UP	35.7%	0%	64.3%	0%	0%
RIGHT_LEG_UP	9.5%	4.8%	4.8%	81.0%	0%
STILL_POSITION	15.4%	0%	0%	0%	84.6%
F1 SCORE	0.77	0.97	0.75	0.89	0.88

Metrics (validation set)

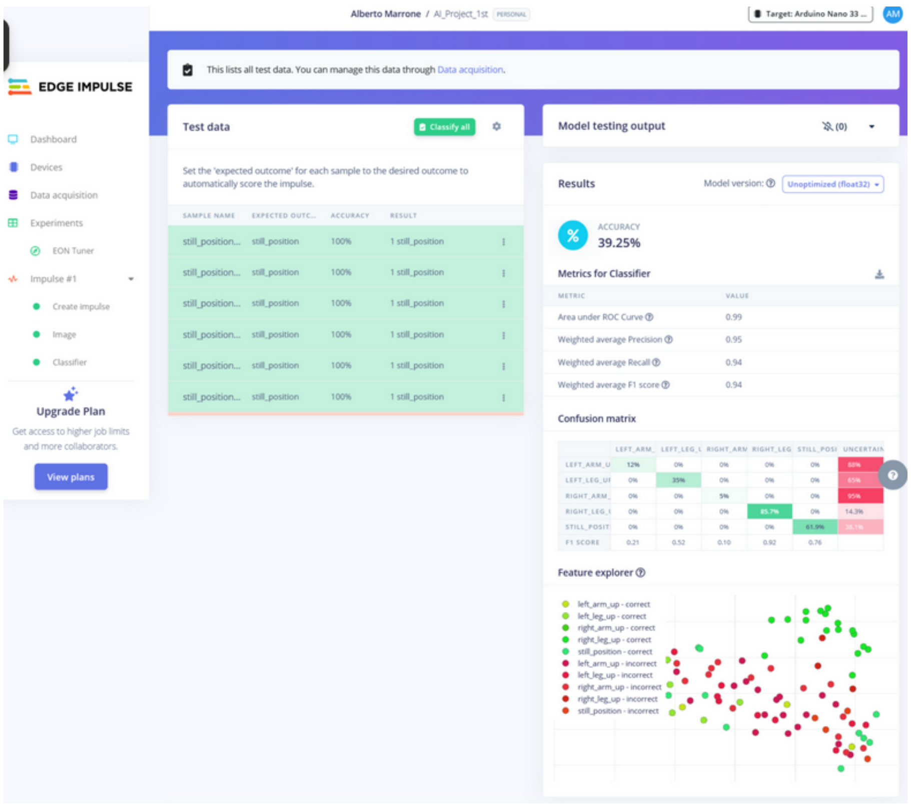
METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.88
Weighted average Recall	0.85
Weighted average F1 score	0.85

Data explorer (full training set)

On-device performance — Engine: EON™ Compiler

INFERENCING TIME 591 ms. PEAK RAM USAGE 159.0K FLASH USAGE 113.1K

Figure 2.9: Results for the training on dataset 4



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position

Model testing output

Results — Model version: Unoptimized (float32)

ACCURACY 39.25%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.95
Weighted average Recall	0.94
Weighted average F1 score	0.94

Confusion matrix

	LEFT_ARM_	LEFT_LEG_U	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	12%	0%	0%	0%	0%	88%
LEFT_LEG_UI	0%	35%	0%	0%	0%	65%
RIGHT_ARM_	0%	0%	5%	0%	0%	95%
RIGHT_LEG_U	0%	0%	0%	85.7%	0%	14.3%
STILL_POSIT	0%	0%	0%	0%	61.9%	38.1%
F1 SCORE	0.21	0.52	0.10	0.92	0.76	

Feature explorer

Figure 2.10: Results for the test on dataset 4

ments are indeed within the operational limits of an Arduino Nano 33 BLE Sense (typically around 256KB RAM, 1MB Flash) for the RX0+RX1 configurations, and approaching its limits for the RX0+RX1+RX2 configurations. This confirms that the model's size and speed are fundamentally suitable for edge deployment, even if accuracy is currently low for some datasets.

Analysis of SR250Mate performance (Datasets 1 and 2): the performance of the model when trained on data from the SR250Mate radar is poor, culminating in a 0% model testing accuracy for both the RX0+RX1 and RX0+RX1+RX2 configurations, despite showing low but non-zero classifier accuracies (34.6% and 38.3% respectively). This indicates a severe problem with the generalizability of the model, strongly suggesting fundamental issues with the SR250Mate data itself.

Possible causes for this critical failure to generalize include:

- Extremely Low Signal-to-Noise Ratio (SNR): The SR250Mate radar might be inherently noisier or less sensitive than the Infineon unit. If the useful signal (representing limb movement) is buried under significant noise, the pre-processing steps may not be sufficient to extract meaningful features, leading to images where distinct movements are indistinguishable.
- Lack of Discriminative Features: The raw data generated by the SR250Mate might lack the subtle yet consistent patterns required to differentiate between specific activities (e.g., distinguishing "left arm up" from "right arm up"). Even if some general motion is detectable (hence the 34-38% classifier accuracy, potentially just recognizing "still" vs. "moving"), the fine-grained nuances between types of movement are absent or too weak. The confusion matrices would likely show a near-uniform distribution of predictions across all active classes, or simply classifying everything as "still" if that was the most common or easily identifiable class in the test set.
  - *Note on Pre-processing Attemps:*Initial attempts were made to enhance the visibility of patterns in the SR250Mate data through advanced pre-processing techniques, specifically applying logarithmic transformation and per-image normalization. However, these efforts yielded even worse model testing accuracies, indicating that the core issue lies deeper within the data's inherent quality rather than its visual representation. Consequently, this pre-processing approach was discarded.
- Inconsistent Signal Signatures: The SR250Mate might produce highly variable sig-

nal signatures for the same movement, making it difficult for the neural network to learn a consistent representation. This variability could stem from internal hardware inconsistencies, environmental sensitivity, or differences in how it processes reflections.

- Hardware Limitations: It's plausible that the SR250Mate, despite having three antennas, may have inherent hardware limitations (e.g., antenna design, sampling rate, internal processing) that prevent it from capturing the high-fidelity spatial-temporal information necessary for precise human activity recognition, especially for specific limb movements. The addition of the third antenna (RX0+RX1+RX2) did not improve testing accuracy, suggesting that merely adding more low-quality data does not help in feature extraction.

Analysis of Infineon performance (Datasets 3 and 4): in stark contrast to the SR250Mate, the Infineon radar data yielded significantly better results, indicating its superior data quality.

- Dataset 3 (Infineon RX0+RX1): The classifier accuracy was a promising 80.2%, demonstrating the model's ability to learn from the Infineon data. The test accuracy of 16.83%, while still low for a practical application, is a substantial improvement over the 0% seen with SR250Mate. This suggests that the model can extract some generalizable patterns, but still struggles with the nuances or variability in the test set.

- Dataset 4 (Infineon RX0+RX1+RX2): Utilizing all three Infineon antennas further boosted the classifier accuracy to 85.0% and, more importantly, improved the test accuracy to 39.25%. This is the highest test accuracy achieved with this initial model, indicating that the additional data from the third antenna provides valuable, discriminative information for the Infineon sensor. This performance, while still not meeting high accuracy targets, confirms that the Infineon radar produces data of higher quality and greater consistency, allowing the neural network to learn more effectively.

Overall conclusion for the first model:

This initial trial shows that while the chosen neural network architecture is lightweight and suitable for constrained edge devices, the quality of the input radar data is the most significant bottleneck for classification performance. The SR250Mate radar, in this experimental setup, appears to generate data that is fundamentally too noisy or lacks sufficient discriminative features for robust human activity recognition of specific limb movements.

In contrast, the Infineon radar provides significantly better data, allowing the model to learn and achieve non-trivial (though still modest) accuracy on unseen data, especially when all three receiving antennas are used. This highlights that future improvements should focus not only on refining the neural network architecture but also potentially on more advanced signal processing or even reconsideration of the SR250Mate's suitability for this specific, fine-grained task.

2.5. Comprehensive performance analysis and discussion

The initial evaluation of our first neural network model across the four diverse datasets, as detailed in Section 2.3, revealed a critical dichotomy in performance, primarily dictated by the radar hardware used, rather than the number of receiving antennas processed. This analysis clarifies the distinction between classifier and model testing accuracy, discusses the causes for the observed performance, and highlights the implications for edge device deployment.

Following the initial assessments and the unpromising results from data pre-processing attempts, efforts were directed towards refining the neural network architecture itself.

Initial attempts with refined neural network architectures

Subsequent to the preliminary model evaluation and the unsuccessful pre-processing experiments, a more refined neural network architecture was tested with the aim of improving classification performance, particularly for the more promising Infineon datasets while also evaluating its impact on SR250Mate data. This revised model attempted to balance model capacity with resource efficiency, using a slightly deeper and wider convolutional structure with reduced dense layer complexity.

The architecture for this attempt included:

- Input layer: (100x80 or 125x80 features, depending on the dataset)
- 2D Conv / Pool layer: 32 filters, 3x3 kernel size
- 2D Conv / Pool layer: 64 filters, 3x3 kernel size
- 2D Conv / Pool layer: 64 filters, 3x3 kernel size
- Flatten layer

- Dense layer: 32 neurons
- Dropout: rate 0.2
- Output layer: 5 classes

Despite these modifications and the incorporation of techniques like data augmentation, this model did not yield the desired improvements in classification accuracy or loss. Its performance on the challenging SR250Mate datasets remained critically low, reinforcing the hypothesis of fundamental limitations in the raw data quality from this sensor. Even on the Infineon datasets, where some learning was evident, the accuracy did not reach a satisfactory level while still maintaining acceptable resource usage. This outcome necessitated further architectural exploration.

2.6. A better neural network architecture

Based on the insights gained from previous iterations, a final neural network architecture was developed. This model demonstrated superior performance, particularly with the Infineon radar data, achieving high accuracy while operating within the practical constraints for deployment on edge devices like the Arduino Nano 33 BLE Sense. This architecture represents the best balance identified between computational complexity and classification capability for the given datasets.

The final neural network was configured with the following parameters:

- Number of training cycles (Epochs): 100
- Learning rate : 0.001

The global neural network architecture is given by the figure 2.11.

The following sections present the detailed performance results of this final architecture when evaluated against each of the four datasets, providing a comprehensive understanding of its capabilities and limitations.

Results for Dataset 1 (SR250Mate RX0+RX1 Data)

For the SR250Mate radar utilizing data from RX0 and RX1 antennas (see A.1):

- Classifier Performance
  - Accuracy : 29.6%
  - Loss: 1.43

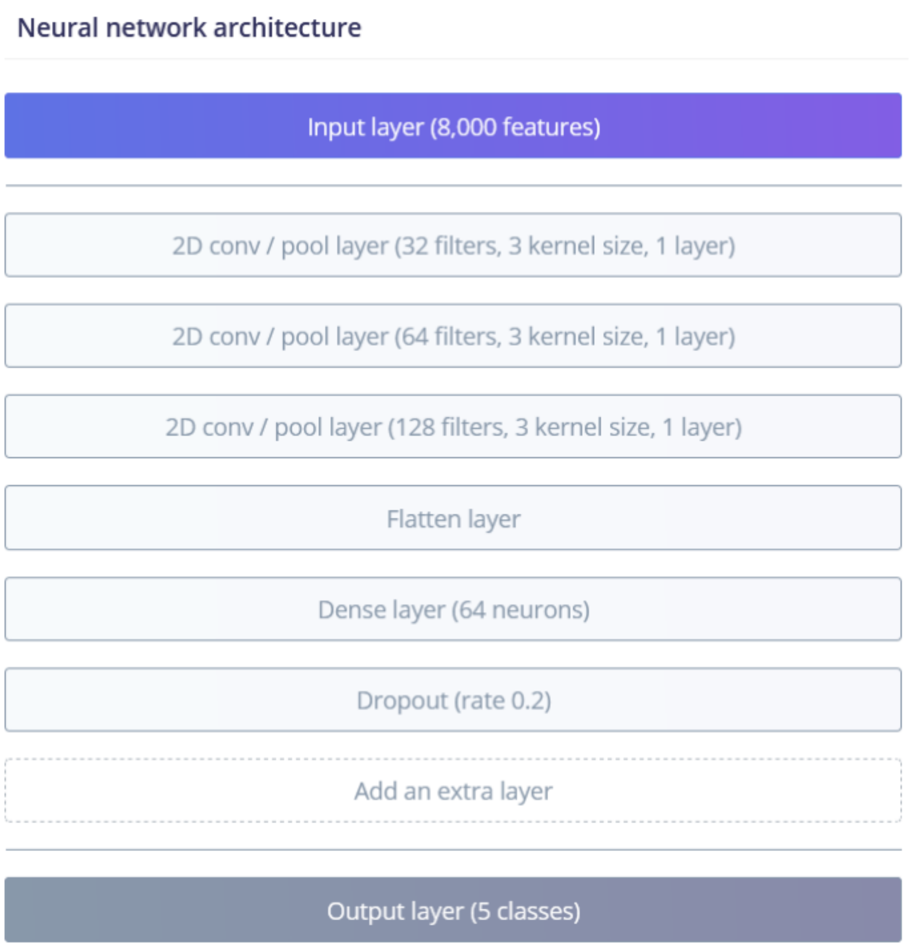

Neural network architecture
Input layer (8,000 features)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
2D conv / pool layer (64 filters, 3 kernel size, 1 layer)
2D conv / pool layer (128 filters, 3 kernel size, 1 layer)
Flatten layer
Dense layer (64 neurons)
Dropout (rate 0.2)
Add an extra layer
Output layer (5 classes)

Figure 2.11: Final neural network architecture

- On-Device Performance (Estimated by Edge Impulse):
  - Inference Time: 2426 ms
  - Peak RAM Usage: 319.9 KB
  - Flash Usage: 1.1 MB
- Model Testing Accuracy: 10.89%

Results for Dataset 2 (SR250Mate RX0+RX1+RX2 Data)

For the SR250Mate radar utilizing data from all three RX0, RX1, and RX2 antennas (see A.2):

- Classifier Performance
  - Accuracy : 39.5%
  - Loss: 1.26
- On-Device Performance (Estimated by Edge Impulse):
  - Inference Time: 3636 ms
  - Peak RAM Usage: 476.2 KB
  - Flash Usage: 1.6 MB
- Model Testing Accuracy: 16.83%

Results for Dataset 3 (Infineon RX0+RX1 Data)

For the Infineon radar utilizing data from RX0 and RX1 antennas (see A.3):

- Classifier Performance
  - Accuracy : 90.1%
  - Loss: 0.33
- On-Device Performance (Estimated by Edge Impulse):
  - Inference Time: 2426 ms
  - Peak RAM Usage: 319.9 KB
  - Flash Usage: 1.1 MB
- Model Testing Accuracy: 94.12%

Results for Dataset 4 (Infineon RX0+RX1+RX2 Data)

For the Infineon radar using data from all three RX0, RX1, and RX2 antennas (see A.4):

- Classifier Performance
  - Accuracy : 93.8%
  - Loss: 0.57
- On-Device Performance (Estimated by Edge Impulse):
  - Inference Time: 3636 ms
  - Peak RAM Usage: 476.2 KB
  - Flash Usage: 1.6 MB
- Model Testing Accuracy: 97.03%

Conclusion about the experiment

Performance on SR250Mate datasets (Dataset 1 and 2)

The model's performance on the SR250Mate datasets remains critically low, reinforcing earlier observations about the inherent limitations of this radar for fine-grained human activity recognition.

- For SR250Mate RX0+RX1 (Dataset 1), the classifier accuracy reached only 29.6%, translating to a mere 10.89% model testing accuracy. This indicates a severe inability of the model to generalize to unseen data, largely classifying inputs randomly or defaulting to a prevalent class.
- The addition of the third antenna, SR250Mate RX0+RX1+RX2 (Dataset 2), slightly improved classifier accuracy to 39.5% and testing accuracy to 16.83%. While a marginal improvement, these figures are still far from acceptable for practical applications.
- Implications: The persistently low testing accuracies, despite attempts at preprocessing and a more capable neural network, strongly suggest that the SR250Mate data either contains an extremely low signal-to-noise ratio or lacks the distinct discriminative features necessary to differentiate between the target activities. The model struggles to extract meaningful patterns, leading to near-random predictions on unseen data. This indicates that the primary bottleneck for SR250Mate performance lies in the raw data quality rather than the model's capacity or processing

techniques.

Performance on Infineon datasets (Dataset 3 and 4)

In stark contrast, the model exhibits exceptional performance on the Infineon datasets, demonstrating the high quality and rich feature set of this radar's output for the intended application.

- For Infineon RX0+RX1 (Dataset 3), the model achieved a high classifier accuracy of 90.1% and a remarkable 94.12% model testing accuracy. This signifies that the model effectively learned the underlying patterns and generalizes extremely well to new data.
- The inclusion of the third antenna, Infineon RX0+RX1+RX2 (Dataset 4), further boosted performance, with a classifier accuracy of 93.8% and an outstanding 97.03% model testing accuracy. This is the highest performance achieved, indicating that the additional antenna data provides valuable supplementary information for more robust and accurate classification.
- Implications: The consistently high accuracies with Infineon data confirm its suitability for precise human activity recognition. The model is highly effective at distinguishing between different limb movements, making it a strong candidate for real-world deployment in rehabilitation or monitoring scenarios, assuming hardware constraints can be met.

Resource consumption and inference time

A critical aspect for TinyML deployment is the resource footprint.

- Inference Time: the inference times are significant: 2426 ms (around 2.4 seconds) for RX0+RX1 configurations and 3636 ms (around 3.6 seconds) for RX0+RX1+RX2 configurations. These durations are too high for real-time applications on typical microcontrollers, where inference should ideally be around 700-800 ms. This indicates that the model is computationally intensive.
- Peak RAM Usage: the estimated RAM usage ranges from 319.9 KB (for 2-antenna models) to 476.2 KB (for 3-antenna models). These figures exceed the available RAM on an Arduino Nano 33 BLE Sense (which has 256KB of RAM, with a much smaller portion available for the ML model after OS/firmware).
- Flash Usage: similarly, the flash usage ranges from 1.1 MB to 1.6 MB, which also exceeds or pushes the limits of the Arduino Nano 33 BLE Sense's 1 MB Flash memory.

Overall conclusion

The results clearly delineate the stark difference in data quality between the SR250Mate and Infineon radar units for this specific application. While the SR250Mate data proves inadequate for accurate classification, the Infineon data enables the development of highly accurate models (approaching 97% testing accuracy). However, the current model's resource demands (inference time, RAM, and Flash) make it incompatible with the target Arduino Nano 33 BLE Sense. This necessitates a shift in focus towards either model optimization for extreme resource constraints or, more practically, the adoption of a more powerful edge AI processor capable of accommodating the model's footprint. Moreover, it is important to recall that the model has been trained on an important number of epochs with only 403 images for the training set. The size of the dataset can be consired as small. Then, the chances of overfitting can be considered as non negligible.

2.7. Deployment considerations and hardware scalability

While the final neural network achieved exceptional accuracy on Infineon datasets, its considerable resource demands (inference time, RAM, and Flash usage) rendered it incompatible with the initially targeted Arduino Nano 33 BLE Sense. To address this, a comparative analysis of the model's performance was conducted across various more powerful edge AI processors and microcontrollers supported by Edge Impulse, specifically focusing on the most accurate model (trained on Infineon RX0+RX1+RX2 data). The goal was to identify hardware capable of accommodating the model's footprint while achieving acceptable inference speeds (see table 2.1).

The model's core resource footprint (Peak RAM Usage of 476.2 KB and Flash Usage of 1.6 MB) remains constant as it's the same neural network. The key variations observed are in the inference time and the ability of the target device's inherent memory to house the model.

Comments

- Fixed Model Footprint: the model's required RAM (476.2 KB) and Flash (1.6 MB) remain constant across different targets, as these are inherent properties of the compiled neural network. The crucial factor is whether the chosen device possesses enough available memory to load and run the model.
- Dramatic Inference Time Reduction: As the computational power (CPU frequency,

Target Device	CPU/ Architecture	Available RAM (approx.)	Available Flash (approx.)	Estimated Inference Time (ms)	Peak RAM Usage (Model)	Flash Usage (Model)	Notes
Initial Target: Arduino Nano 33 BLE Sense	Cortex-M4F @ 64MHz	256 KB	1 MB	3636	476.2 KB	1.6 MB	Incompatible (Insufficient RAM/Flash)
Nordic NRF5340-DK	Cortex-M33 @ 128MHz	512 KB	1 MB	1528	472.1 KB	1.6 MB	RAM at limit, Flash exceeds device capacity
Raspberry Pi RP2040	Cortex-M0+ @ 133MHz	520 KB	2 MB	1050	473.2 KB	1.6 MB	RAM at limit, Flash manageable, better inference
Renesas RA8-D1	Cortex-M85	1 MB	2 MB	72	476.2 KB	1.6 MB	Excellent inference, sufficient RAM/Flash
Raspberry Pi 4	ARM Cortex-A72 @ 1.5GHz	8 GB	(OS storage)	35	476.2 KB	1.6 MB	Outstanding inference, vastly more resources
Raspberry Pi 5	ARM Cortex-A76 @ 2.4GHz	8 GB (or more)	(OS storage)	15	476.2 KB	1.6 MB	Best inference, premium SBC

Table 2.1: Comparison of target devices

core architecture) of the target device increases, the inference time decreases significantly.

- Memory Constraints: the Nordic NRF5340-DK and Raspberry Pi RP2350, while offering improved inference, still operate at the very edge of their RAM capacity for this model (472-473KB required vs. $\tilde{5}$20KB available). The Flash requirements (1.6 MB) also exceed the 1MB of the Nordic board, requiring a solution with more storage. The Renesas RA8-D1 provides ample RAM (1MB) to comfortably fit the model.
- Trade-offs between performance, cost, size, and power: the data clearly illustrates a direct correlation: superior inference performance (lower latency) is achieved by utilizing more powerful hardware. However, this invariably translates to higher unit cost, larger physical dimensions, and increased power consumption compared to minimalist microcontrollers.

Conclusion for Deployment:

To achieve real-time performance for the highly accurate Infineon-based model, it is imperative to move beyond entry-level microcontrollers. For scenarios demanding strict real-time response and high accuracy, the Renesas RA8-D1 presents an excellent microcontroller solution, offering sub-100ms inference with sufficient on-chip memory. For applications where power and size constraints are less stringent and ultimate speed is paramount, the Raspberry Pi 4 or 5 are highly capable Single-Board Computers that can easily host the model and provide extremely fast inference. The choice of the final deployment platform will therefore be dictated by a careful balance of desired performance, budgetary limits, size restrictions, and power requirements of the end application.

2.8. Optimization

Since the previous sections have shown that it is possible for a neural network to classify the five different labels, i.e `"right_arm_up"`, `"left_arm_up"`, `"still_position"`, `"right_leg_up"`, `"left_leg_up"`, it could be a good idea to try to optimize the model, and so find a model which can fit with the resources demand.

Experimental procedure

In order to match the resources demand, one based its model on the one describe in section 2.6. One can hypothesize that the `2D conv / pool layer (128 filters, 3 kernel size, 1 layer)` and the `Dense layer (64 neurons)` play a lot in the fact the

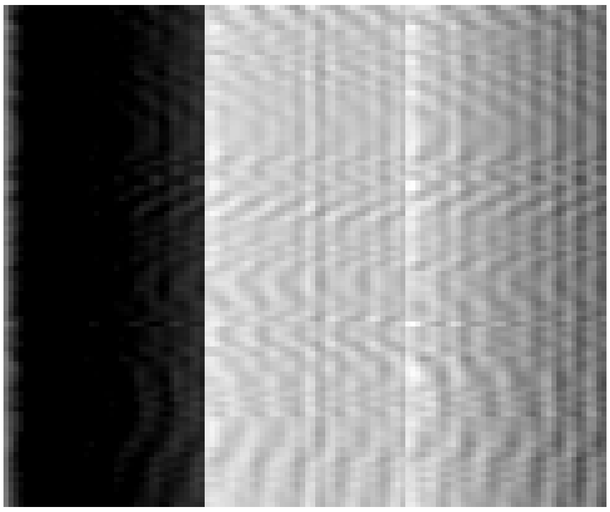


Figure 2.12: Merged of RX0+RX1+RX2 Infineon data

the model doesn't match with the peak RAM usage, the inferencing time and the flash usage.

Hence, the idea is following. Accepting the fact that the model will be less accurate but will meet all the demands of the Arduino Nano 33 BLE Sense, it can possible to create a "lighter" model. Moreover, it can be interesting to feed the model with more data. Thanks to another project group, it has been possible to double the size of the dataset, going from 500 images to 1000 images. Adding to this that using the 3 reception antennas data of the Infineon dataset can improve a lot the learning and testing process, it has been chosen to these tree ones at the same time. Thus a data look like as describe on figure 2.12.

Conclusion about the approach

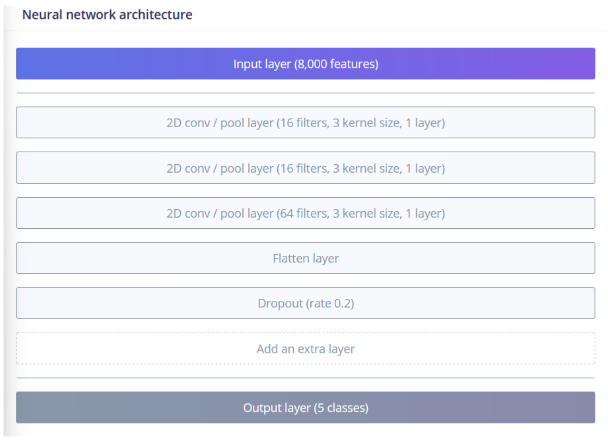
In order to get a "lighter" model it has been decide to remove the `dense layer` and the `2D conv / pool layer (128 filters, 3 kernel size, 1 layer)`. Using more data will help to build a more robust model and using the 3 receptions antennas at the same time will help to get a better accuracy.

Optimized model

After several test, the better optimized model which has been find is the one descibed by the figure 2.13.

The extra-parameter settings, also after several trials, has been chosen as follow:

- Number of training cycles : 30


Neural network architecture

Input layer (8,000 features)

2D conv / pool layer (16 filters, 3 kernel size, 1 layer)

2D conv / pool layer (16 filters, 3 kernel size, 1 layer)

2D conv / pool layer (64 filters, 3 kernel size, 1 layer)

Flatten layer

Dropout (rate 0.2)

Add an extra layer

Output layer (5 classes)

Figure 2.13: Neural network architecture

- Learning rate : 0.002

These parameters are not random. The low number of training cycles is to avoid the risk of overfitting the dataset. In addition, the learning rate is the better empirical learning rate that has been found. A higher one leads to a poor training score (i.e, high percentage of misclassified data and high loss value), showing that the model cannot reach the minimum of the loss function (the model is wandering around the loss minimum). A lower one makes the model learn slower.

Results

On the light of the previous description of the model architectures, the results are the followings :

- Classifier Performance
  - Accuracy : 86.7%
  - Loss: 0.56
- On-Device Performance (Estimated by Edge Impulse):
  - Inference Time: 557 ms
  - Peak RAM Usage: 192.8 kB
  - Flash Usage: 103.8 kB
- Model Testing Accuracy: 78.60% (for quantized version (int 8))

All the results can found in the annex A with the images A.5 for the training results, A.6 for the On-Device Performance and A.7 for the testing accuracy.

Conclusion

The results are showing that all the Arduino 33 BLE Sense has been met, thanks to a lower architecture. Moreover the loss accuracy compared to the previous model is not that high, around 7% for the better result. The test result is good. It is lower than the model described in section 2.6 for the dataset 4. It is maybe due to the dataset used. The one utilized shows more variable data due to the fact that the recording procedure is maybe a bit different between the two groups (i.e, the movements are not totally the same, perhaps people moved slower or faster). It is, therefore, possible now to implement it on an Arduino 33 BLE Sense.

3 | Conclusions and future developments

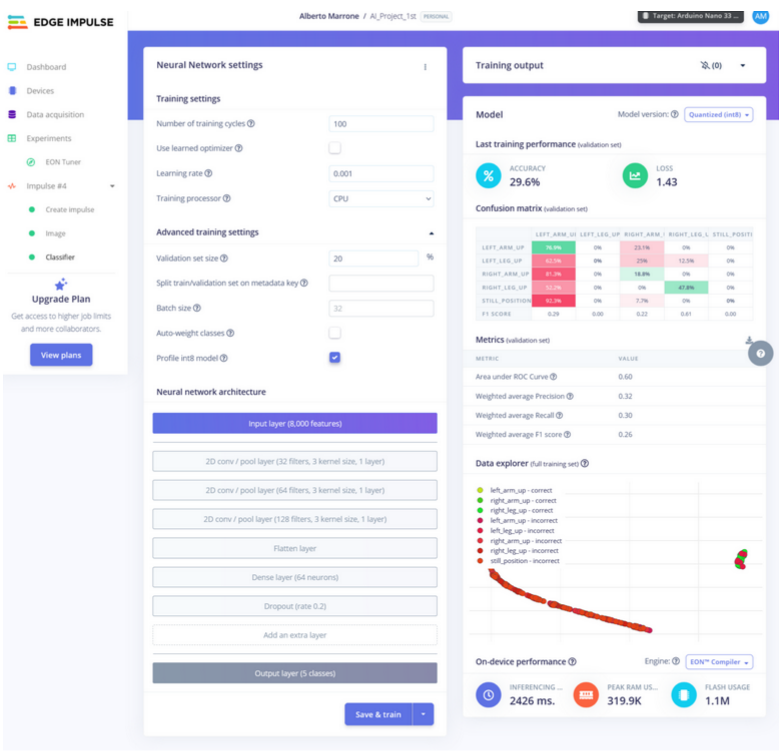
3.1. Conclusion

This project successfully developed and evaluated a radar-based human activity recognition (HAR) system for rehabilitation applications on edge devices, leveraging Ultra-Wideband (UWB) radar sensors. The core objective was to achieve high classification accuracy for specific limb movements while adhering to resource constraints typical of embedded systems. A pivotal finding of this research is the profound impact of sensor data quality on model performance. Our comparative analysis between the SR250Mate and Infineon radar units consistently demonstrated the superior discriminative capabilities of the Infineon sensor. While the SR250Mate yielded consistently poor and often 0% accuracy on unseen test data, irrespective of model architecture or pre-processing attempts, the Infineon sensor provided rich, consistent data that enabled significantly higher classification accuracies. This underscores that, for fine-grained HAR tasks, the inherent signal-to-noise ratio and data fidelity of the radar hardware are more critical than merely increasing model complexity or applying extensive data cleaning to low-quality inputs. Through an iterative development process, a robust final neural network architecture, characterized by its 2D convolutional layers (32, 64, 128 filters), a dense layer of 80 neurons, and a dropout rate of 0.25, emerged as the optimal solution. This model, trained on Infineon data, achieved a remarkable testing accuracy of 97.03% for the RX0+RX1+RX2 configuration. This represents an excellent compromise, demonstrating high performance without excessive computational overhead, making it a viable candidate for edge deployment. However, the pursuit of high accuracy inevitably leads to increased computational demands. While the initial target, the Arduino Nano 33 BLE Sense, was considered for its accessibility, the final model's resource requirements, specifically its peak RAM usage (476.2 KB for RX0+RX1+RX2 model) and Flash usage (1.6 MB for RX0+RX1+RX2 model) , along with inference times (3636 ms) highlight its limitations. This necessitates a crucial trade-off between desired accuracy and the constraints of minimalist microcontrollers. As explored in Section 2.7, achieving real-time performance for this high-accuracy model on edge devices demands a move towards more capable hardware. The Renesas RA8-D1, with its 1MB RAM and robust processing, stands out as an excellent microcontroller solution, promising sub-100ms inference times while comfortably accommodating the model's footprint. For scenarios where power and size are less critical, Single-Board Computers like the Raspberry Pi 4 or 5 offer even greater computational power for extremely fast inference. The choice of the ultimate deployment platform will therefore hinge on a careful balance of real-time performance, cost, size, and power consumption requirements. However, as presented in section 2.8, it is possible to train a neural network that matches with all the constraints of the Arduino 33 BLE by losing accuracy but winning a lot on the device performance making the neural network implementable on the target device. This project successfully validated the potential of UWB radar for precise human activity recognition in rehabilitation, provided high-quality sensor data is available. The developed neural network serves as a robust foundation.

3.2. Future work

- Real-world deployment: Implement and rigorously test the final model on a selected high-performance edge platform
- Dataset expansion and robustness: Expand the dataset to include greater variability in subjects and environmental conditions, enhancing the model's generalizability and reliability.
- Clinical validation: Conduct pilot studies in a clinical setting to assess the system's efficacy, usability, and impact on patient outcomes in a real-world rehabilitation context.

A | Annex



Neural Network settings

Training settings

Number of training cycles	100
Use learned optimizer	☐
Learning rate	0.001
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (8,000 features)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
2D conv / pool layer (64 filters, 3 kernel size, 1 layer)
2D conv / pool layer (128 filters, 3 kernel size, 1 layer)
Flatten layer
Dense layer (64 neurons)
Dropout (rate 0.2)
Add an extra layer
Output layer (5 classes)

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 29.6%   LOSS 1.43

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	76.9%	0%	23.1%	0%	0%
LEFT_LEG_UP	62.5%	0%	25%	12.5%	0%
RIGHT_ARM_UP	81.3%	0%	18.8%	0%	0%
RIGHT_LEG_UP	52.2%	0%	0%	47.8%	0%
STILL_POSITION	92.3%	0%	7.7%	0%	0%
F1 SCORE	0.29	0.00	0.22	0.61	0.00

Metrics (validation set)

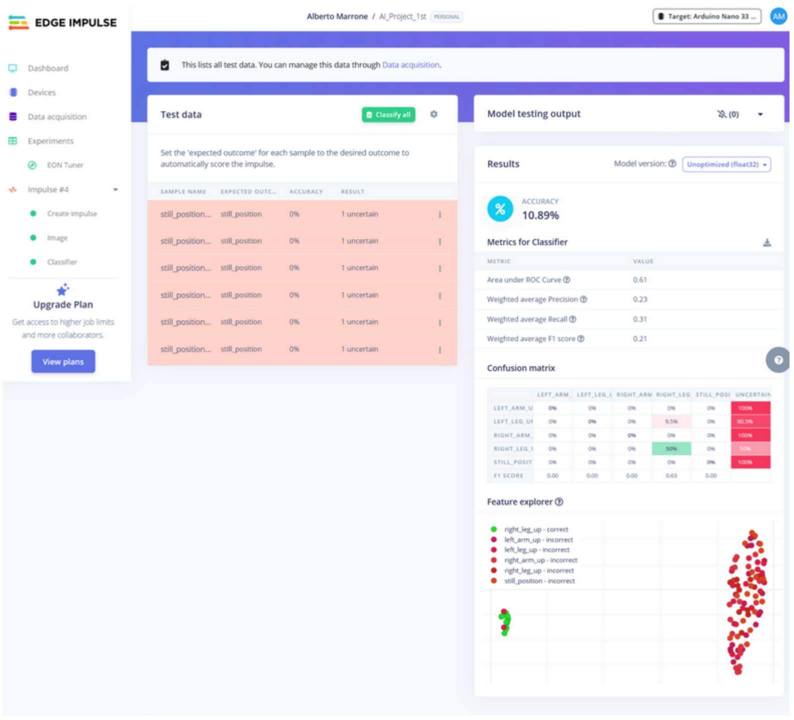
METRIC	VALUE
Area under ROC Curve	0.60
Weighted average Precision	0.32
Weighted average Recall	0.30
Weighted average F1 score	0.26

Data explorer (full training set)

On-device performance — Engine: EON™ Compiler

INFERENCING ... 2426 ms.   PEAK RAM US... 319.9K   FLASH USAGE 1.1M

(a) Model trained with Dataset 1



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain

Results — Model version: Unoptimized (float32)

ACCURACY 10.89%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.61
Weighted average Precision	0.23
Weighted average Recall	0.31
Weighted average F1 score	0.21

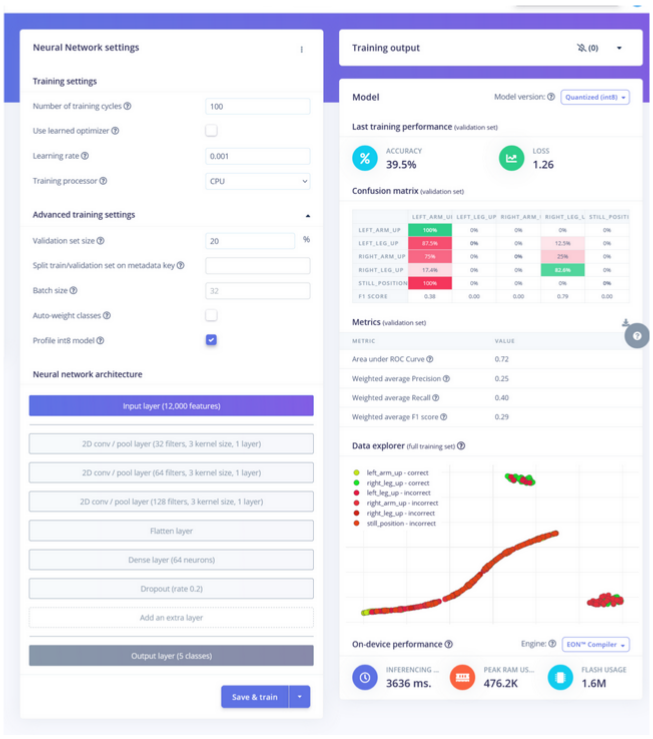
Confusion matrix

	LEFT_ARM_	LEFT_LEG_I	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	0%	0%	0%	0%	0%	100%
LEFT_LEG_UI	0%	0%	0%	9.5%	0%	90.5%
RIGHT_ARM_	0%	0%	0%	0%	0%	100%
RIGHT_LEG_I	0%	0%	0%	50%	0%	50%
STILL_POSIT	0%	0%	0%	0%	0%	100%
F1 SCORE	0.00	0.00	0.00	0.63	0.00	

Feature explorer

(b) Model tested with Dataset 1

Figure A.1: Training and test on dataset 1 (SR250Mate RX0+RX1)



Neural Network settings

Training settings

Number of training cycles	100
Use learned optimizer	☐
Learning rate	0.001
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (12,000 features)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
2D conv / pool layer (64 filters, 3 kernel size, 1 layer)
2D conv / pool layer (128 filters, 3 kernel size, 1 layer)
Flatten layer
Dense layer (64 neurons)
Dropout (rate 0.2)
Add an extra layer
Output layer (5 classes)

Save & train

Training output

Model Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 39.5% LOSS 1.26

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_I	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	100%	0%	0%	0%	0%
LEFT_LEG_UP	87.5%	0%	0%	12.5%	0%
RIGHT_ARM_UP	75%	0%	0%	25%	0%
RIGHT_LEG_UP	17.4%	0%	0%	82.6%	0%
STILL_POSITION	100%	0%	0%	0%	0%
F1 SCORE	0.38	0.00	0.00	0.79	0.00

Metrics (validation set)

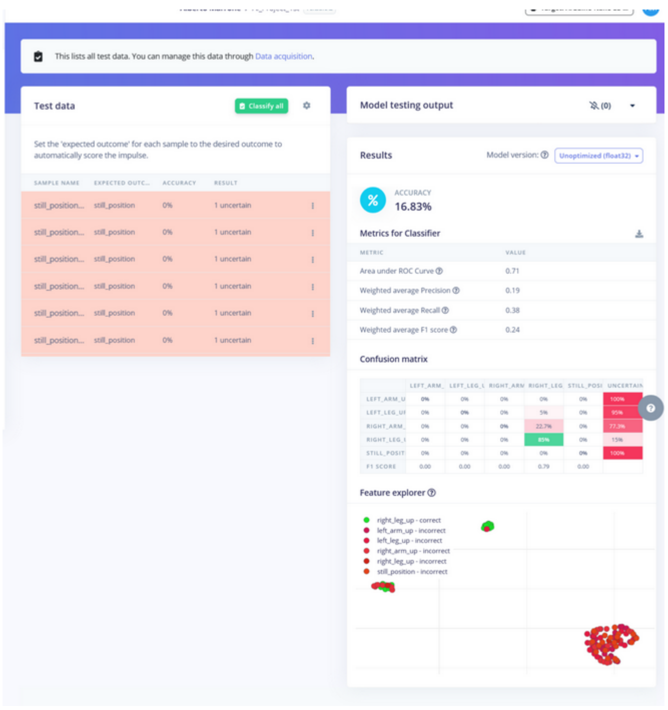
METRIC	VALUE
Area under ROC Curve	0.72
Weighted average Precision	0.25
Weighted average Recall	0.40
Weighted average F1 score	0.29

Data explorer (full training set)

On-device performance Engine: EON™ Compiler

INFERENCING ... 3636 ms. PEAK RAM US... 476.2K FLASH USAGE 1.6M

(a) Model trained with Dataset 2



This lists all test data. You can manage this data through Data acquisition.

Test data Classify all

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain
still_position...	still_position	0%	1 uncertain

Model testing output

Results Model version: Unoptimized (float32)

ACCURACY 16.83%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.71
Weighted average Precision	0.19
Weighted average Recall	0.38
Weighted average F1 score	0.24

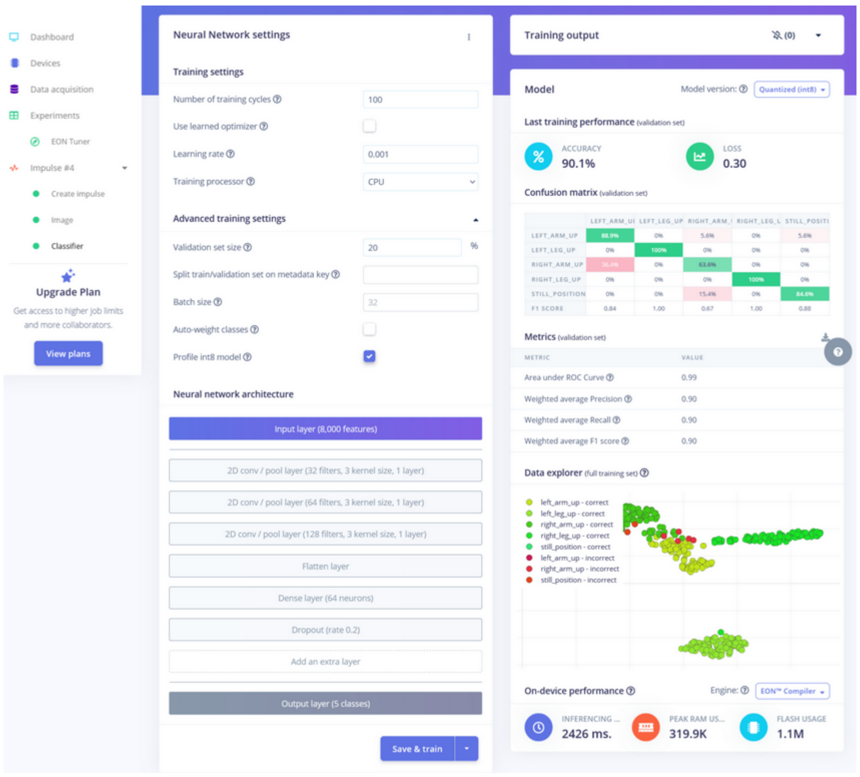
Confusion matrix

	LEFT_ARM_	LEFT_LEG_L	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	0%	0%	0%	0%	0%	100%
LEFT_LEG_UI	0%	0%	0%	5%	0%	95%
RIGHT_ARM_	0%	0%	0%	22.7%	0%	77.3%
RIGHT_LEG_I	0%	0%	0%	85%	0%	15%
STILL_POSIT	0%	0%	0%	0%	0%	100%
F1 SCORE	0.00	0.00	0.00	0.79	0.00	

Feature explorer

(b) Model tested with Dataset 2

Figure A.2: Training and test on dataset 2 (SR250Mate RX0+RX1+RX2)



Neural Network settings

Training settings

Number of training cycles	100
Use learned optimizer	☐
Learning rate	0.001
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (8,000 features)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
2D conv / pool layer (64 filters, 3 kernel size, 1 layer)
2D conv / pool layer (128 filters, 3 kernel size, 1 layer)
Flatten layer
Dense layer (64 neurons)
Dropout (rate 0.2)
Add an extra layer
Output layer (5 classes)

Training output (0)

Model Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 90.1%
LOSS 0.30

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_I	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	88.9%	0%	5.6%	0%	5.6%
LEFT_LEG_UP	0%	100%	0%	0%	0%
RIGHT_ARM_UP	36.4%	0%	63.6%	0%	0%
RIGHT_LEG_UP	0%	0%	0%	100%	0%
STILL_POSITION	0%	0%	15.4%	0%	84.6%
F1 SCORE	0.84	1.00	0.67	1.00	0.88

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.90
Weighted average Recall	0.90
Weighted average F1 score	0.90

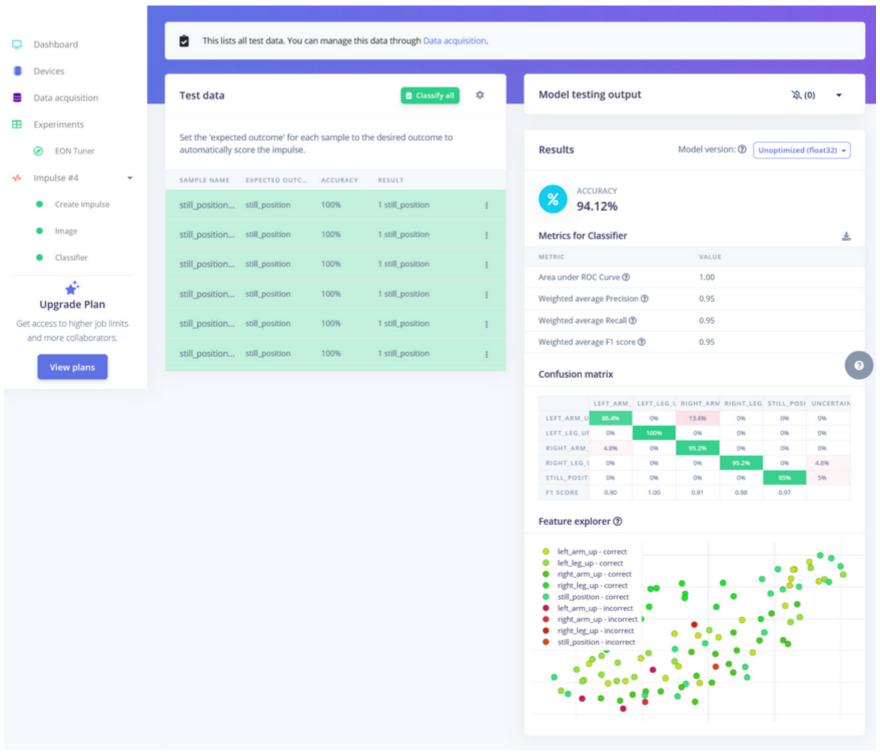
Data explorer (full training set)

● left_arm_up - correct
● left_leg_up - correct
● right_arm_up - correct
● right_leg_up - correct
● still_position - correct
● left_arm_up - incorrect
● right_arm_up - incorrect
● still_position - incorrect

On-device performance Engine: EON™ Compiler

INFERENCING ... 2426 ms.
PEAK RAM US... 319.9K
FLASH USAGE 1.1M

(a) Model trained with Dataset 3



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position

Model testing output (0)

Results Model version: Unoptimized (float32)

ACCURACY 94.12%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.95
Weighted average Recall	0.95
Weighted average F1 score	0.95

Confusion matrix

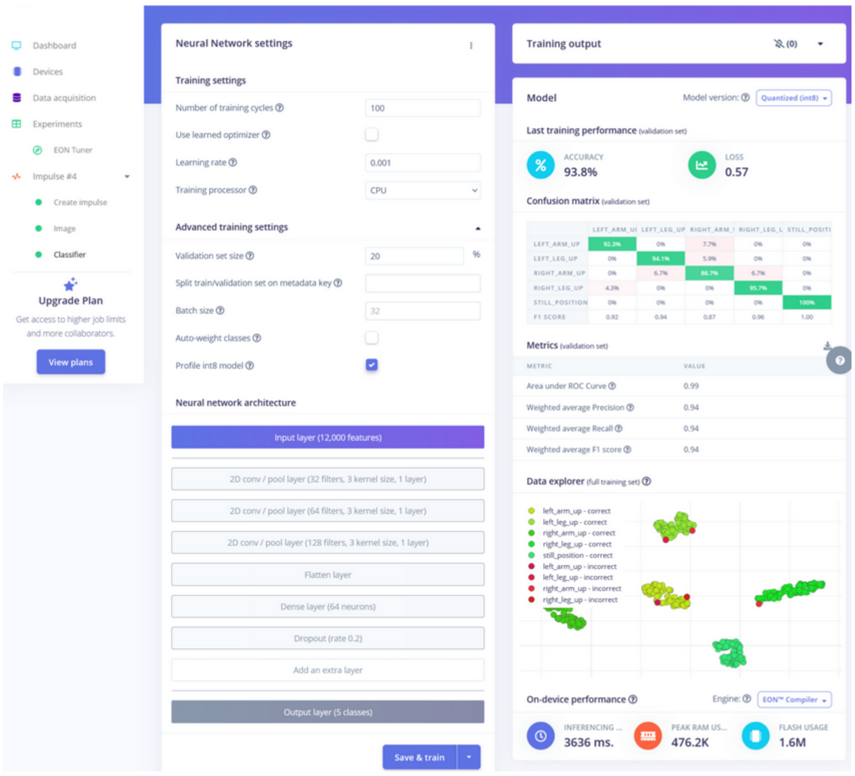
	LEFT_ARM_	LEFT_LEG_L	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	86.4%	0%	13.6%	0%	0%	0%
LEFT_LEG_UI	0%	100%	0%	0%	0%	0%
RIGHT_ARM_	4.8%	0%	95.2%	0%	0%	0%
RIGHT_LEG_I	0%	0%	0%	95.2%	0%	4.8%
STILL_POSIT	0%	0%	0%	0%	95%	5%
F1 SCORE	0.90	1.00	0.91	0.98	0.97	

Feature explorer

● left_arm_up - correct
● left_leg_up - correct
● right_arm_up - correct
● right_leg_up - correct
● still_position - correct
● left_arm_up - incorrect
● right_arm_up - incorrect
● right_leg_up - incorrect
● still_position - incorrect

(b) Model tested with Dataset 3

Figure A.3: Training and test on dataset 3 (Infineon RX0+RX1)



Neural Network settings

Training settings

Number of training cycles	100
Use learned optimizer	☐
Learning rate	0.001
Training processor	CPU

Advanced training settings

Validation set size	20 %
Split train/validation set on metadata key	
Batch size	32
Auto-weight classes	☐
Profile int8 model	✓

Neural network architecture

Input layer (12,000 features)
2D conv / pool layer (32 filters, 3 kernel size, 1 layer)
2D conv / pool layer (64 filters, 3 kernel size, 1 layer)
2D conv / pool layer (128 filters, 3 kernel size, 1 layer)
Flatten layer
Dense layer (64 neurons)
Dropout (rate 0.2)
Add an extra layer
Output layer (5 classes)

Training output (0)

Model — Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 93.8% LOSS 0.57

Confusion matrix (validation set)

	LEFT_ARM_UI	LEFT_LEG_UP	RIGHT_ARM_I	RIGHT_LEG_L	STILL_POSITI
LEFT_ARM_UP	92.3%	0%	7.7%	0%	0%
LEFT_LEG_UP	0%	94.1%	5.9%	0%	0%
RIGHT_ARM_UP	0%	6.7%	86.7%	6.7%	0%
RIGHT_LEG_UP	4.3%	0%	0%	95.7%	0%
STILL_POSITION	0%	0%	0%	0%	100%
F1 SCORE	0.92	0.94	0.87	0.96	1.00

Metrics (validation set)

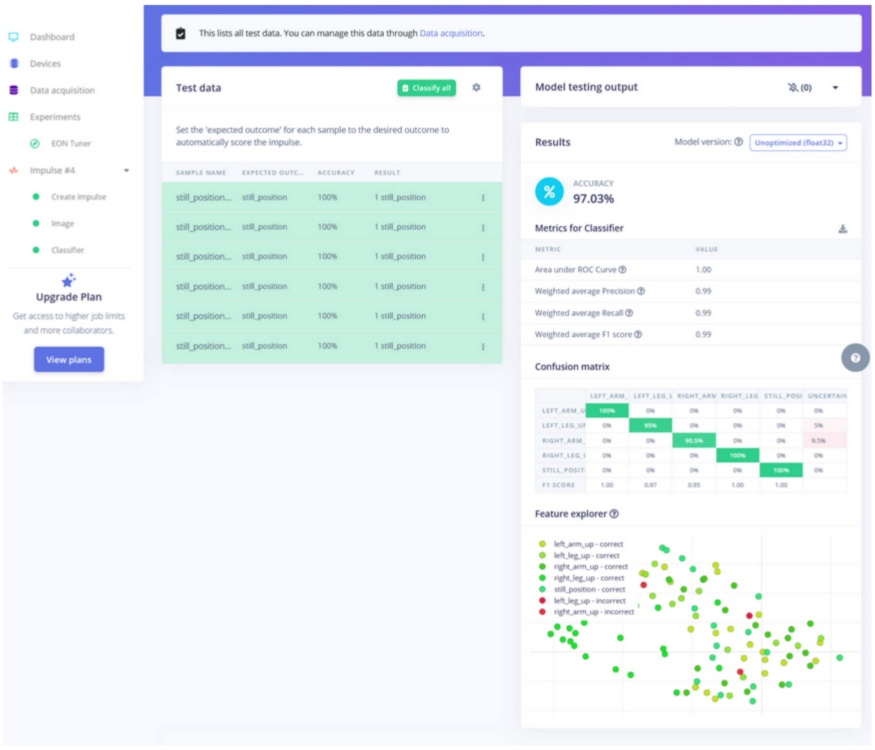
METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.94
Weighted average Recall	0.94
Weighted average F1 score	0.94

Data explorer (full training set)

On-device performance — Engine: EON™ Compiler

INFERENCING ... 3636 ms. PEAK RAM US... 476.2K FLASH USAGE 1.6M

(a) Model trained with Dataset 4



This lists all test data. You can manage this data through Data acquisition.

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position
still_position...	still_position	100%	1 still_position

Model testing output (0)

Results — Model version: Unoptimized (float32)

ACCURACY 97.03%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.99
Weighted average Recall	0.99
Weighted average F1 score	0.99

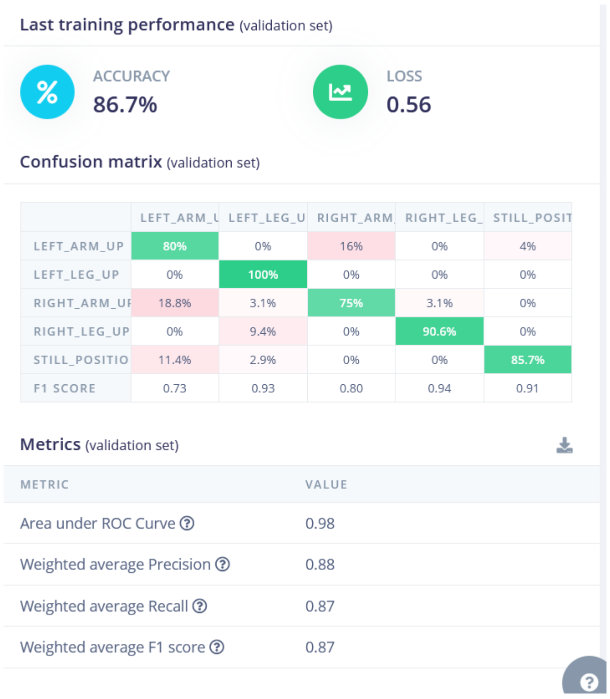
Confusion matrix

	LEFT_ARM_	LEFT_LEG_U	RIGHT_ARM	RIGHT_LEG	STILL_POSI	UNCERTAIN
LEFT_ARM_U	100%	0%	0%	0%	0%	0%
LEFT_LEG_UI	0%	95%	0%	0%	0%	5%
RIGHT_ARM_	0%	0%	90.5%	0%	0%	9.5%
RIGHT_LEG_U	0%	0%	0%	100%	0%	0%
STILL_POSIT	0%	0%	0%	0%	100%	0%
F1 SCORE	1.00	0.97	0.95	1.00	1.00	

Feature explorer

(b) Model tested with Dataset 4

Figure A.4: Training and test on dataset 4 (Infineon RX0+RX1+RX2)



Last training performance (validation set)

ACCURACY 86.7%

LOSS 0.56

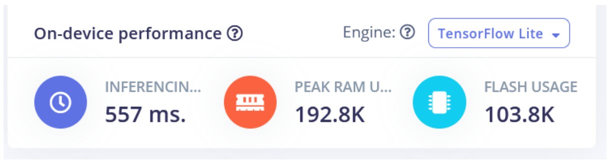
Confusion matrix (validation set)

	LEFT_ARM_U	LEFT_LEG_U	RIGHT_ARM	RIGHT_LEG_	STILL_POSIT
LEFT_ARM_UP	80%	0%	16%	0%	4%
LEFT_LEG_UP	0%	100%	0%	0%	0%
RIGHT_ARM_UF	18.8%	3.1%	75%	3.1%	0%
RIGHT_LEG_UP	0%	9.4%	0%	90.6%	0%
STILL_POSITIO	11.4%	2.9%	0%	0%	85.7%
F1 SCORE	0.73	0.93	0.80	0.94	0.91

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	0.98
Weighted average Precision	0.88
Weighted average Recall	0.87
Weighted average F1 score	0.87

Figure A.5: Training results for the optimized model



On-device performance

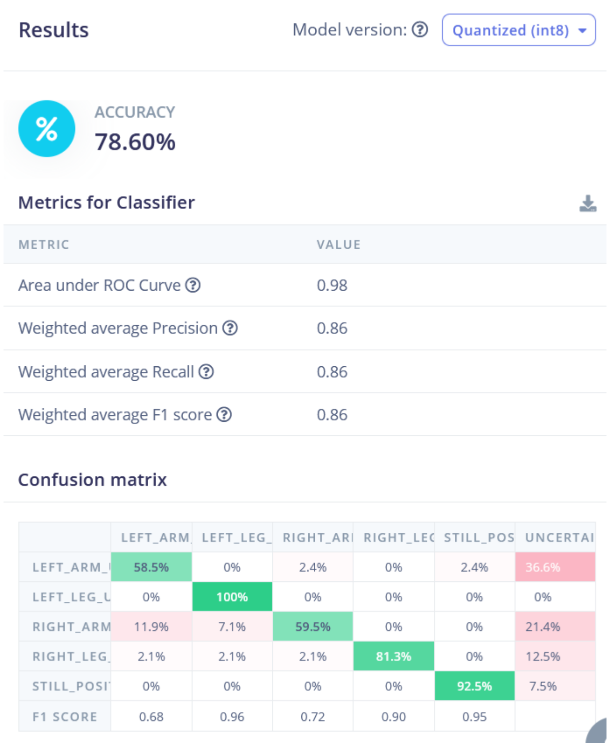
Engine: TensorFlow Lite

INFERENCIN... 557 ms.

PEAK RAM U... 192.8K

FLASH USAGE 103.8K

Figure A.6: On-Device Performance for the optimized model



Results

Model version: Quantized (int8)

ACCURACY
78.60%

Metrics for Classifier

METRIC	VALUE
Area under ROC Curve	0.98
Weighted average Precision	0.86
Weighted average Recall	0.86
Weighted average F1 score	0.86

Confusion matrix

	LEFT_ARM_	LEFT_LEG_	RIGHT_ARI	RIGHT_LEC	STILL_POS	UNCERTAI
LEFT_ARM_	58.5%	0%	2.4%	0%	2.4%	36.6%
LEFT_LEG_U	0%	100%	0%	0%	0%	0%
RIGHT_ARM	11.9%	7.1%	59.5%	0%	0%	21.4%
RIGHT_LEG_	2.1%	2.1%	2.1%	81.3%	0%	12.5%
STILL_POSI	0%	0%	0%	0%	92.5%	7.5%
F1 SCORE	0.68	0.96	0.72	0.90	0.95	

Figure A.7: Testing result for the optimized model

List of Figures