UNIVERSITY OF OUM EL BOUAGHI LARBI BEN M'HIDI
FACULTY OF SCIENCES AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL ENGINEERING

Tiny ML for Gesture Recognition

Thesis submitted in fulfillment of the requirements for the degree of

MASTER of SCIENCE in Electronics of Embedded Systems

By:

Nourelhouda LAMRAOUI
Cheyma TOUATI

Proposed and led by:

Mr Khaled.MENDACI

June 2023

Thanks

First of all, we thank Almighty God for giving us the strength, will, and privilege to study and follow the path of science.

We would also like to thank in particular: My supervisor Mr. MENDACI KHALED, on orientation, who made me a considerable contribution without which this work could not have been carried out in the right port. With constant interest and great competence.

Then we thank the members of the jury, and for agreeing to pass judgment on this work.

We also thank with all our hearts all the teachers who have contributed to our learning from a young age to date.

Finally, we thank all those who, directly or indirectly, have contributed to the realization of this work.

Thank you all

Dedications

The biggest thanks go to allah

I dedicate this work to my mother and father, who filled me with their support and love,

To My father who gave so much support that i can be proud to find the results of long years of sacrifice,

To my lovely sisters " sarah, yasmine & mariam "

To my dearest brother "islam",

Finally, I dedicate it to All family without exception and all my friends and all my teachers, of the specialty one by one .and to all who knows me.

Nourelhouda

Dedications

Thank God for success

I dedicate this work to my mom and dad, may God prolong their lives.

To my father who gave so much support and the sacrifices I made thanks,

To my little sister Soumya.

To my dear brothers “Elias and Zakaria”,

To my nephews “Nour Al-Eman”, “Abdul Basit” and “Jawad”

Finally, have mercy on the soul of my dear brother “Sofyan” and my late uncle “Al-Rubaie” who live there and will remain so.

Cheyma

CONTENT TABLE

Figures list

Table list

Abbreviations List

NN	**N**eural **N**etwork
ML	**M**achine **L**earning
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etworks
MLP	**M**ulti**l**ayer **P**erceptron
MCU	**M**icro**C**ontroller **U**nit
IMU	**I**nertial **M**easurement **U**nit
MLP	**M**ulti**L**ayer **P**erceptron
SGD	**S**tochastic **G**radient **D**escent
Edge ML	**E**dge **M**achine **L**earning
IoT	**I**nternet **o**f **T**hings
RAM	**R**andom **A**ccess **M**emory
ROM	**R**ead **O**nly **M**emory
I/O	**I**nput **/O**utput
Tiny ML	**T**iny **M**achine **L**earning
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphics **P**rocessing **U**nit
API	**A**pplication **P**rogramming **I**nterface
TF Lite	**T**ensor**F**low **L**ite
TPU	**T**ensor **P**rocessing **U**nit
DSP	**D**egital **S**ignal **P**rocessing
MNIST	**M**odified **N**ational **I**nstitude of **S**tandards and **T**echnology database

INTRODUCTION

General introduction

ML has emerged as a fascinating field with the potential to revolutionize human-computer interaction. Tiny ML is a sub class of ML that targets contrained devices with limited ressources at the edge like microcontrollers for battery based or low power applications.

Our project focuses on developing an advanced and intuitive gesture recognition system by harnessing the combined power of the Edge Impulse Tiny ML platform, Arduino/PlateformIO, and the XIAO BLE SENSE board and its embedded Inertial Measurement Unit (IMU) sensor .

We aim to leverage the capabilities of the XIAO BLE SENSE board for a wearable application, by harnessing the potential of cutting-edge technologies such as neural networks (NN), machine learning (ML), TensorFlowLite, and Keras. By capturing motion data using the IMU sensor, we will preprocess and annotate the collected datasets using the comprehensive Edge Impulse platform. Through this platform, we will train and fine-tune NN models using TensorFlow and Keras, enabling the system to accurately recognize and classify different gestures.

Throughout the project, we will emphasize optimizing the performance and memory usage of the NN models to ensure efficient operation on the resource-constrained microcontroller. This will involve exploring various techniques i, including feature extraction, model compression, and optimization, to achieve a balance between accuracy and resource efficiency.

The thesis is organized as follows:

In the first chapter, we present the general concepts related to AI and ML. In the second, a brief summary of theory related to neural networks is presented. In the third chapter, Tiny ML is presented with the frame works behind it. In the last chapter, we present a wearable application for gesture recognition. We summarize the achieved results in a conclusion.

CHAPTER I:

Introduction to artificial intelligence and machine learning

I.1 Introduction to Artificial Intelligence (AI)

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their behavior. The term can also be applied to any machine that exhibits properties associated with the human mind, such as B. learning and problem solving. [1]

A desirable characteristic of AI is its ability to rationalize and take actions that are most likely to achieve a specific goal. A subset of artificial intelligence is machine learning (ML), which refers to the concept that computer programs can automatically learn and adapt to new data without human assistance. Deep learning techniques enable this automatic learning by ingesting large amounts of unstructured data such as text, images or videos. [1]

When most people hear the term artificial intelligence, robots are usually the first thing that comes to mind. That's because big-budget movies and novels weave stories of humanoid machines ravaging the planet. but it is not the truth. [1]

Artificial intelligence is based on the principle that human intelligence can be defined in such a way that machines can easily imitate it and perform tasks ranging from the simplest to even more complex. One of the goals of artificial intelligence is to mimic human cognitive activity. Researchers and developers in the field have made surprisingly rapid progress in mimicking activities such as learning, thinking, and cognition, so long as those activities can be specifically defined. Some believe that innovators may soon be able to design systems that surpass the ability of humans to learn or reason. But others remain skeptical, because all cognitive activity pervades important value judgments that are subject to human experience. [1]

As technology advances, previous benchmarks that defined artificial intelligence become outdated. For example, machines that calculate basic functions or recognize text through optical character recognition are no longer considered to embody artificial intelligence, since this function is now taken for granted as an inherent computer function. [1]

AI is continuously evolving to benefit many different industries. Machines are wired using a cross-disciplinary approach based on mathematics, computer science, linguistics, psychology, and more. [1]

I.2 Machine learning definition

Machine learning is a subset of artificial intelligence (AI). It focuses on teaching computers to learn from data and improve with experience rather than being explicitly programmed to do so. In machine learning, algorithms are trained to find patterns and correlations in large data sets and make the best decisions and predictions based on this analysis. Machine learning applications improve with use, and the more data they have, the more accurate they become. Applications of machine learning are all around us –in our homes, our shopping carts, our entertainment media, and our healthcare. [2]

Figure 1.1: machine learning process

I.3 Applications of Artificial Intelligence

The uses of artificial intelligence are endless. The technology can be applied across many different sectors and industries. AI is being tested and used in the healthcare industry to tailor drug dosing and dispense various treatments for specific patients, and to support surgical procedures in the operating room. [1]

Other examples of machines with artificial intelligence are chess-playing computers and self-driving cars. Each of these machines must weigh the consequences of each of its actions, as each action affects the bottom line. In chess, the end result is winning the game. In a self-driving car, the computer system must consider and calculate all external data in order to take action to avoid a collision. [1]

Artificial intelligence also has application in the financial industry, where it is used to detect and flag activity in banking and finance such as unusual debit card usage and large account deposits—all of which help a bank's fraud department. Applications for AI are also being used to help streamline and make trading easier. This is done by making supply, demand, and pricing of securities easier to estimate. [1]

I.4 Types of AI

Artificial intelligence can be categorized into four types.

I.4.1 Reactive AI

uses algorithms to optimize output based on a set of inputs. For example, chess-playing AIs are reactive systems that optimize the best strategy to win the game. Reactive AI tends to be fairly static, unable to learn or adapt to new situations. Therefore, given the same input, it produces the same output. [1]

I.4.2 AI with limited memory

Can adapt to past experience or update itself based on new observations or data. The amount of updates is usually limited (hence the name) and the storage length is relatively short. For example, self-driving cars can "read the road" and adapt to new situations, even "learning" from past experience. [1]

I.4.3 Theory of Mind AI

Is fully adaptive, with a wide range of abilities to learn and store past experiences. These types of artificial intelligence include advanced chatbots that can pass the Turing test and make people believe that the artificial intelligence is human. While this AI is advanced and impressive, it's not confident. [1]

I.4.4 Confident AI,

As the name suggests, become sentient and aware of their own existence. Still in the realm of science fiction, some experts believe that an AI will never become conscious or "alive".[1]

I.5 Special Considerations

Artificial intelligence has attracted the attention of scientists and the public since its inception. A common theme is the notion that machines will become so complex that humans will not be able to keep up, and will take off and redesign themselves exponentially. [1]

Another reason is that machines can invade people's privacy and even be armed. Other debates debate the ethics of artificial intelligence and whether intelligent systems should be treated like robots, with the same rights as humans. [1]

Self-driving cars have been controversial because their machines are often designed to minimize risk and loss. In the event of a collision with one person or another at the same time, the cars calculate the option that causes the least damage. [1]

Another contentious issue for many AI people is how it will affect human employment. As many industries seek to automate certain jobs through the use of intelligent machines, there are fears that people will be squeezed out of the workforce. Self-driving cars could render taxi and car-sharing programs obsolete, while manufacturers could easily replace humans with machines, rendering human skills obsolete. [1]

I.6 How Is AI Used Today?

Today, artificial intelligence is widely used in a variety of applications with varying degrees of maturity. Recommendation algorithms that suggest what you might like next are popular AI implementations, as are chatbots that appear on websites or in the form of smart speakers such as Alexa or Siri. AI is used to make forecasts related to weather and financial forecasts, streamline production processes, and reduce various forms of redundant cognitive work (such as tax accounting or editing). AI is also used to play games, control self-driving cars, process speech, and more. [1]

For example: in healthcare, artificial intelligence is used to support diagnosis. AI is very good at identifying small abnormalities in scans, and is even better at triangulating diagnoses based on a patient's symptoms and vital signs. AI is also used to triage patients, maintain and track medical records, and process health insurance claims. Future innovations are expected to include AI-assisted robotic surgery, collaborative clinical judgment, and virtual nurses or doctors.[1]

I.7 The relation between machine learning and AI

Machine learning and its components of deep learning and neural network fit as concentric subsets of artificial intelligence. Artificial intelligence processes data to make decisions and predictions. Machine learning algorithms allow artificial intelligence not only to process this data, but also to learn and get smarter without additional programming. Artificial intelligence is the overarching element of all fundamental subsets of machine learning. The first subset is machine learning; within that is deep learning, then neural networks. [2]

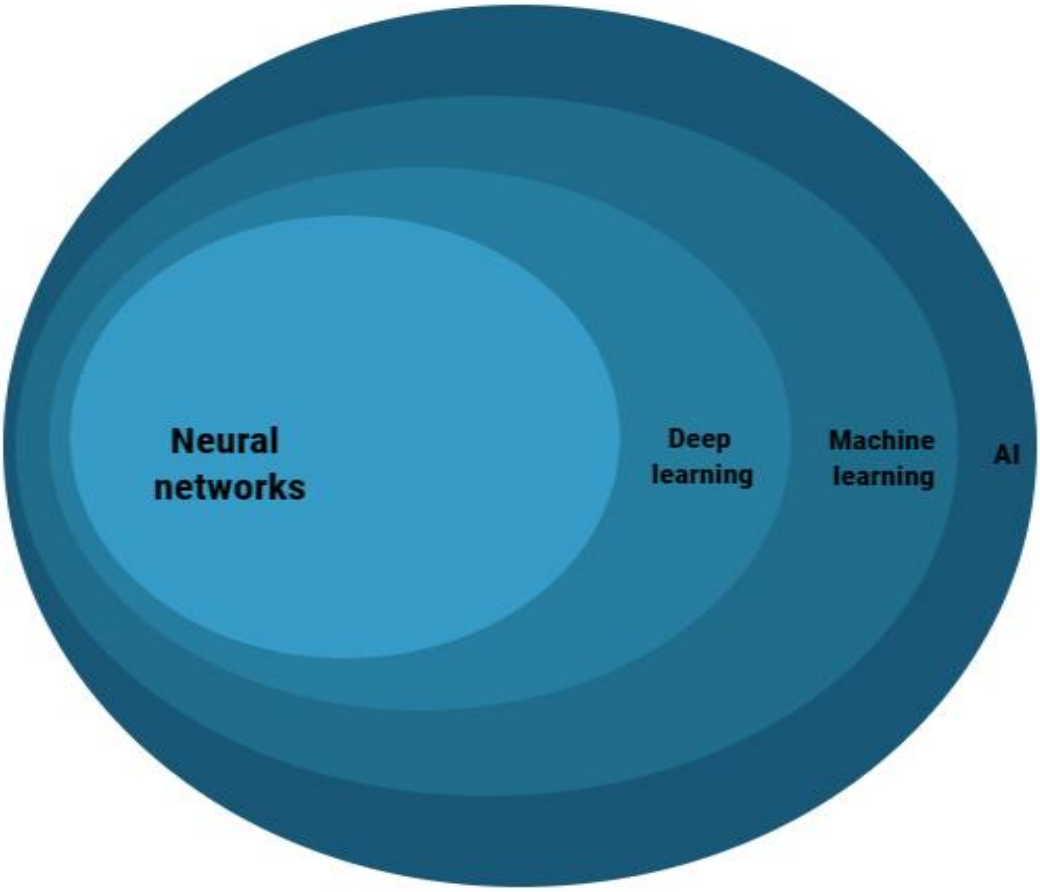


Figure 1.2: Diagram of the relationship between AI and machine learning

I.8 Neural network

Artificial neural networks are modeled from neurons in biological brains. The artificial neurons are called nodes and are divided into layers that work in parallel. When the artificial neuron receives a digital signal, it processes it and sends a signal to other neurons connected to it. Like the human brain, neural reinforcement improves pattern recognition, ability, and overall learning. [2]

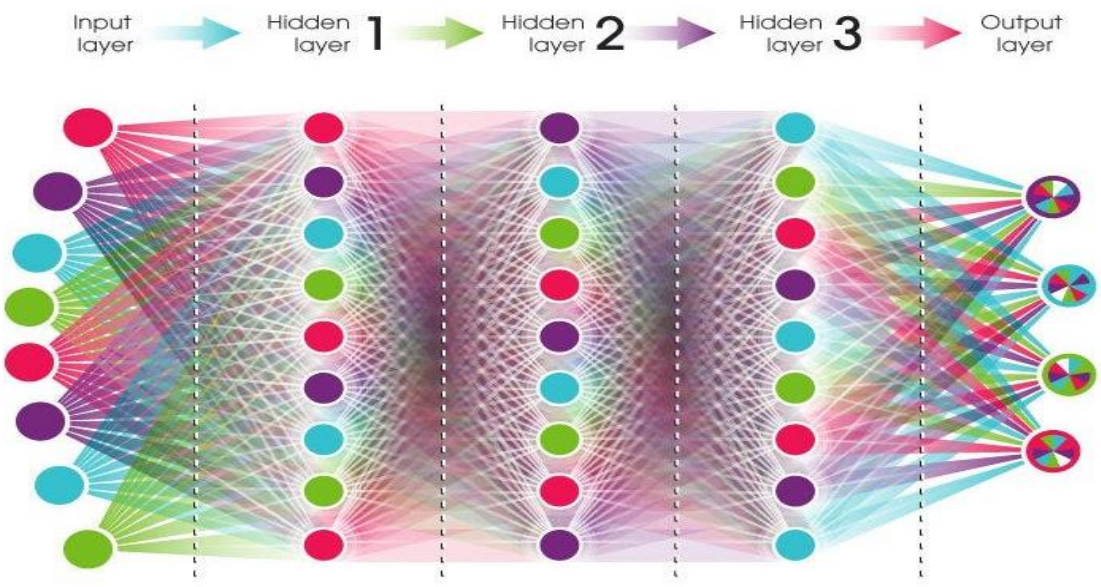


Figure 1.3: Illustration of neural network

I.9 Deep learning

This type of machine learning is called "deep" because it involves many layers of neural networks and large amounts of complex and diverse data. To enable deep learning, the system collaborates with multiple layers in the network and consistently extracts higher quality results. For example, a deep-learning system that processes natural images and searches for Gloriosas will identify plants at the first level. As it moves through the neural layers, it recognizes a flower, then a daisy, and finally a gloridium. Examples of deep learning applications include speech recognition, image classification, and pharmaceutical analysis. [2]

I.10 How does machine learning work?

ML consists of different types of ML models using different algorithmic techniques. Depending on the type of data and the desired outcome, one of four learning models can be used: supervised, unsupervised, semi-supervised, or reinforcement. In each of these models, one or more algorithmic techniques can be applied with respect to the dataset used and the expected results. Essentially, ML algorithms are designed to classify things, discover patterns, predict outcomes, and make informed decisions. Algorithms can be used alone or in combination for optimal accuracy when dealing with complex and more unpredictable data. [2]

I.10.1 supervised learning

Supervised learning is the first of four machine learning models. With supervised learning algorithms, machines are taught using examples. A supervised learning model consists of "input" and "output" data pairs, with the output labeled with an expected value. For example, suppose the goal is to have a machine tell the difference between daisies and violets. The binary input data pair contains images of daisies and pansies. The desired outcome of this particular pairing is to pluck the daisies so that this is pre-determined as the correct outcome. [2]

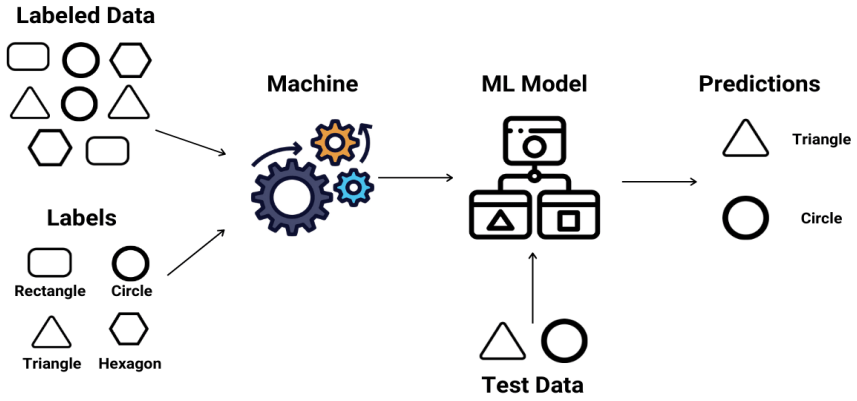


Figure 1.4: supervised learning

Using an algorithm, the system brings together all this training data over time and begins to identify relevant points of similarity, difference, and other logical points—until it can predict the answer to the daisy or pansy question on its own. It's the equivalent of giving a child a set of questions and answers, then asking them to show their work and explain their logic. Supervised learning models are used in many of the applications we interact with every day, such as recommendation engines for products and traffic analysis apps like Waze, which predict the fastest route at different times of day. [2]

I.10.2 unsupervised learning

Unsupervised learning is the second of the four machine learning models. In an unsupervised learning model, there is no solution key. The engine examines incoming data—much of it unlabeled and unstructured and begins using all relevant, accessible data to identify patterns and correlations. In many ways, unsupervised learning is modeled on the way people observe the world. We use intuition and experience to group things. As we experience more and more of something, our ability to classify and identify it becomes more accurate. For machines, "experience" is defined by the amount of data input and provided. Common examples of unsupervised learning applications include facial recognition, genetic sequencing analysis, market research, and cybersecurity. [2]

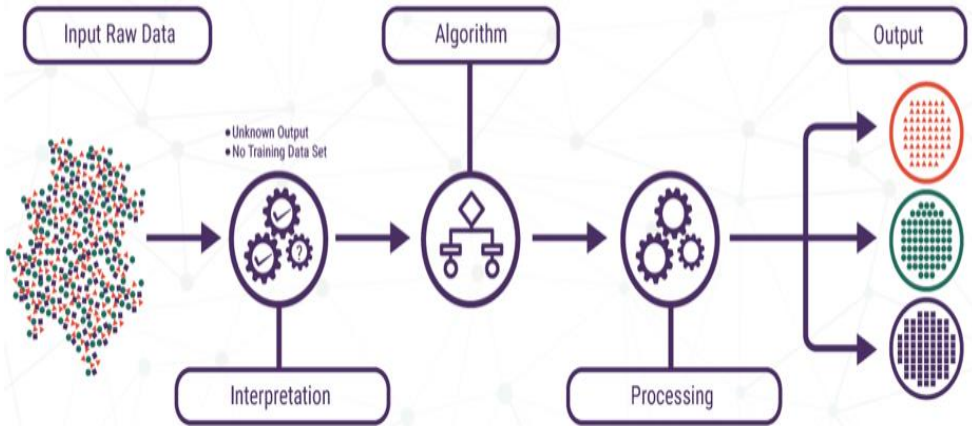


Figure 1.5: unsupervised learning

I.10.3 semi-supervised learning

Semi-supervised learning is the third of the four machine learning models. In a perfect world, all data would be structured and tagged before entering the system. However, since this is clearly not feasible, semi-supervised learning becomes a viable solution when there is a large amount of raw unstructured data. The model consists of feeding in a small amount of labeled data to augment unlabeled records. Essentially, labeled data can give the system a jumpstart and can greatly improve learning speed and accuracy. Semi-supervised learning algorithms guide machines to analyze labeled data to obtain relevant properties that can applied to unlabeled data. [2]

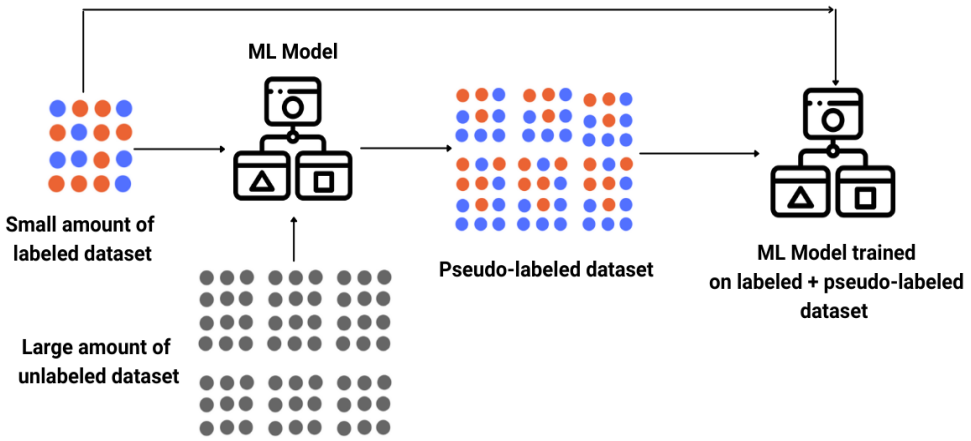


Figure 1.6: semi-supervised learning

I.10.4 reinforcement learning

Reinforcement learning is the fourth machine learning model. In supervised learning, a machine is given an answer key and learns by finding correlations between all correct answers. A reinforcement learning model does not include an answer key, but instead inputs a set of allowed actions, rules, and potential final states. When the intended goal of the algorithm is fixed or binary, the machine can learn from examples. But with variable expected outcomes, the system must learn through experience and rewards. In reinforcement learning models, the "rewards" are numbers and are programmed into the algorithm as something the system is trying to gather. [2]

Figure 1.7: reinforcement learning

In many ways, this model is similar to teaching someone how to play chess. It's certainly impossible to try and show them every possible move. Instead, you explain the rules and they develop skills through practice. Rewards don't just come in the form of winning the game, but

also getting your opponent's checkers. Applications of reinforcement learning include automated price bidding for buyers of online advertising, computer game development, and high-stakes stock market trading. [2]

I.11 Conclusion

AI (Artificial Intelligence) and ML (Machine Learning) have revolutionized numerous industries and have immense potential for further advancements. These technologies enable computers to learn from data, make predictions, and perform tasks with minimal human intervention. AI and ML have significantly improved efficiency, accuracy, and decision-making processes across various domains such as healthcare, finance, transportation, and more. However, ethical considerations and responsible development must accompany their implementation to address potential challenges like bias, privacy concerns, and job displacement. As AI and ML continue to evolve, collaborative efforts between humans and machines will be crucial for maximizing their benefits while ensuring a fair and inclusive future.

CHAPTER II:

Artificial Neural Network

II .1 Introduction to neural network

Artificial neural networks (ANNs) are a class of computer algorithms inspired by the functioning mechanisms of the brain. There are many programs capable of performing more complex tasks, but their ability to compete with the human brain is extremely rare. In this chapter, we'll look at neural networks from a different angle.

II.2. Biological neurons

A neuron is a cell that can transmit information to other neurons through its different connections (synapses). The human brain is the best model of an extremely fast multifunctional machine. [3]

II.2.1. Characteristics

In their general organization and biochemical system, neurons have many similarities with other cells. Here are the characteristics of biological neurons:

- Receive signals from nearby neurons.
- Integrate these signals.
- Generate nerve impulses (nerve message).
- Driving it
- Transmit it to another neuron capable of receiving it. [4]

II.2.2. Structure

- A neuron consists of three parts:
- Dendrites: recipients of messages
- The cell body: generates the action potential (the response)
- The axon: transmits the signal to the next cells.
- Synapse: which allows cells to communicate with each other, in addition it plays a role in modulating the signals that pass through the nervous system [4]

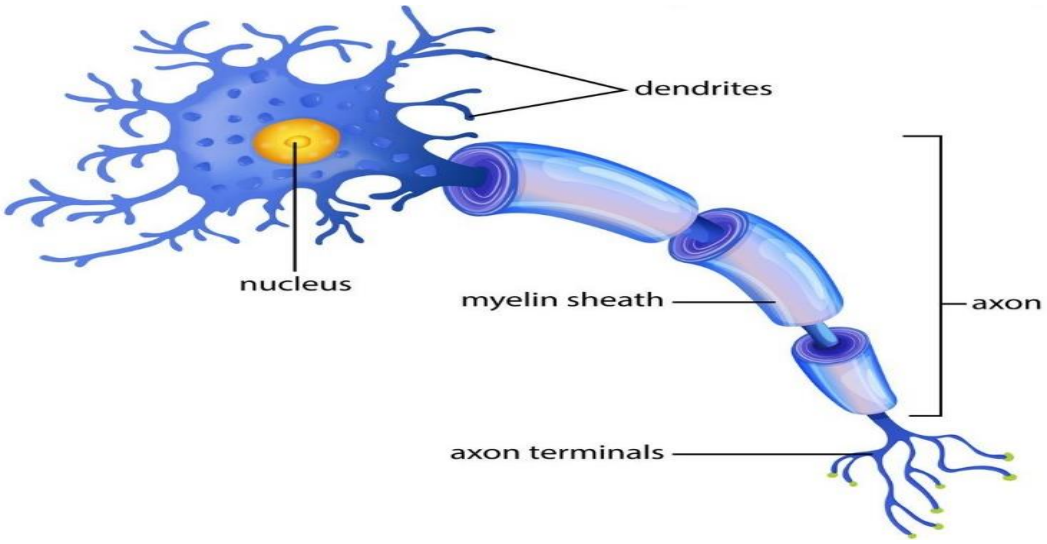


Figure 2.1: biological neuron

II.3. Artificial neurons

The first systematic study of artificial neurons came from neuropsychiatrist McCulloch and logician Pitts, who were inspired by the study of biological neurons [3]

It sums the weighted inputs

- **U=$\sum W_i X_i$ + Θ** **(2.1)**
- **Y=f(U)** **(2.2)**

(U) : potentiel du neurone. Neuron potential

- Y : neuron output
- X : input signal i. (Inputs can be Boolean, binary (0.1), bipolar (-1.1), or real).
- W_i : the weights i.
- **Θ** : bias.
- f : activation fonction.

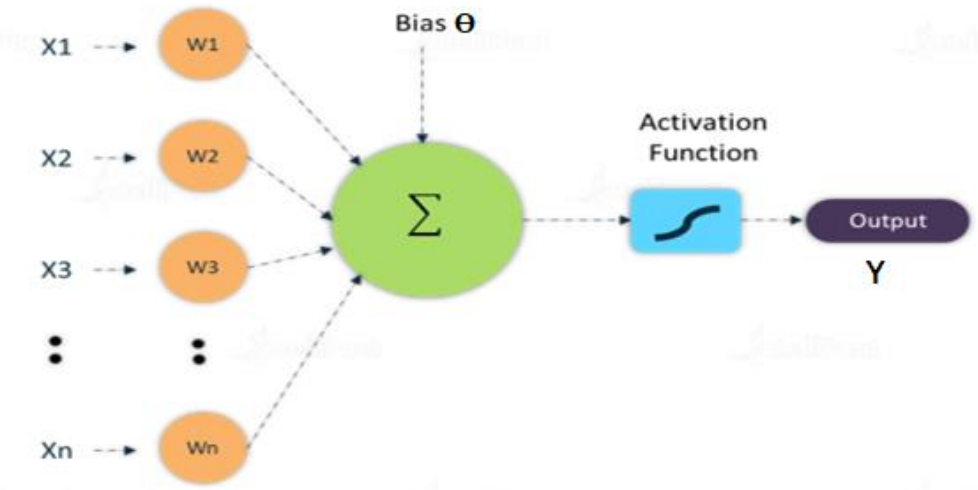


Figure 2.2: Schematic representation of an artificial neuron

The modelling of the biological nervous system is based on the following correspondence:

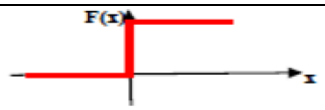
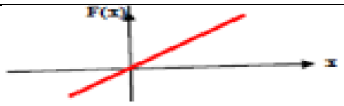
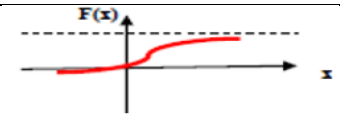
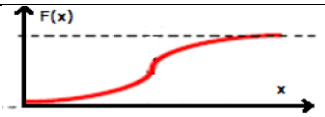
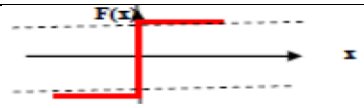
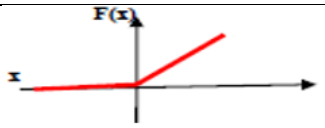
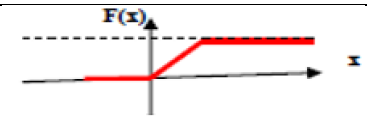
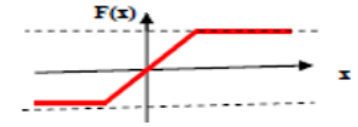
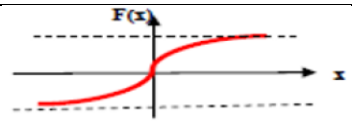
Table 2.1: the difference between biological neurons and artificial neurons

Biological neurons	Artificiel neural network
Cell body	Input function (sum)
dendrites	inputs
Axon	Activation function
Axon terminals	outputs

II.4. Activation functions

There are many possible forms for the activation function. The most common are summarized in the following table:

Table 2.2: Activation functions types

Function name	Mathematical expression	Plot
Binary	$f(x) = \begin{cases} 0 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases}$	
Identity	f (x) = x	
Sigmoid	f (x) $= \frac{1}{1+e^{-x}}$	
SoftMax	f_i $(x) = \frac{exp(x_i)}{\sum_j \exp(x_j)}$	
Symmetric threshold	$f(x) = \begin{cases} -1 \; si \; x < 0 \\ 1 \; si \; x \geq 0 \end{cases}$	
Relu	$f(x) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases}$	
Saturated linear	$f(x) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \; 0 \leq x \leq 1 \\ 1 \; si \; x \geq 0 \end{cases}$	
Symmetric saturated linear	$f(x) = \begin{cases} -1 \; si \; x < -1 \\ x \; si \; -1 \leq x \leq 1 \\ 1 \; si \; x > 0 \end{cases}$	
Hyperbolic tangent	f $(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	

II.5. Neural network topology

The connections between the neurons that make up the network describe the topology of the model. There are several network architectures, we can mention few: [3]

II.5.1. Multilayer network

Neurons are arranged in layers. There is no connection between neurons of a same layer and connections are only made with neurons of following layers.

II.5.2. Recurrent neural network

These are networks in which information propagates from layer to layer with possible backtracking.

II.6. Single layer and Multilayer perceptron

This is the first operational ANN. It is a fully interconnected network with only two layers (input and output). It is composed of threshold neurons. Supervise the learning process and gradually increase its weight. [3]

II.6.1. Single layer Perceptron

The single-layer perceptron is the first of three arrays that can be used with binary or continuous input. Different algorithms can be used to scale weights and connection thresholds in a single-layer perceptron [1]

II.6.2. Multilayer Perceptron (MLP)

The multilayer perceptron is a point network with one or more layers of neurons between the input and output layers [1]

II.7. Learning neural networks

Learning neural networks is a step in the development of neural networks. At this point, the parameters of the network will evolve until the desired behavior is obtained. The idea is to adjust the weight of the ANN to reduce the gap between the actual output and the desired output.

II.7.1. neural networks supervised learning

Since supervised learning is an adaptation of the synaptic coefficients of the network, the output of the network corresponds to the desired output in each case. Supervised learning is the most common type of learning. Whenever you want to adjust your weight,

- Every time you try, the error is calculated.
- The weight is replaced by the smallest error, if there is one.

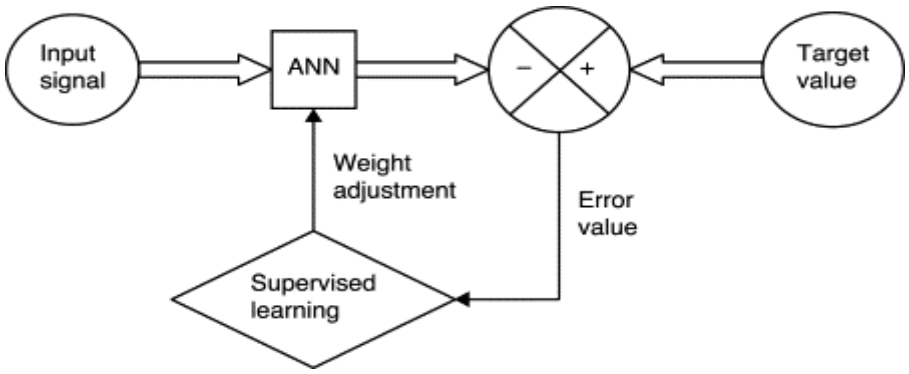
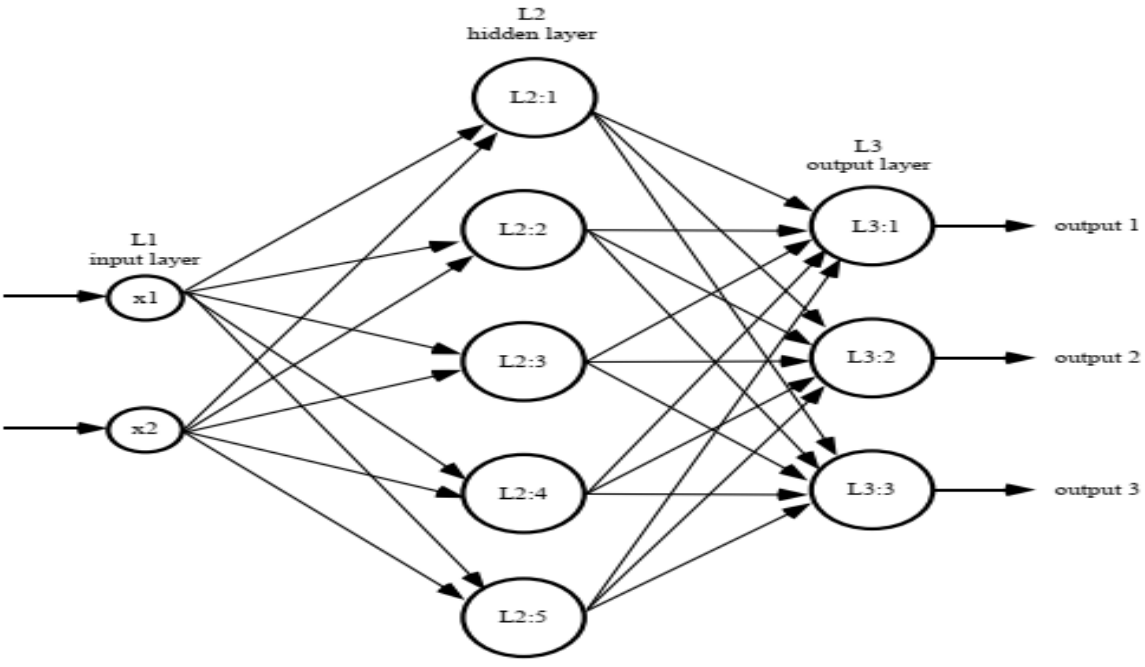


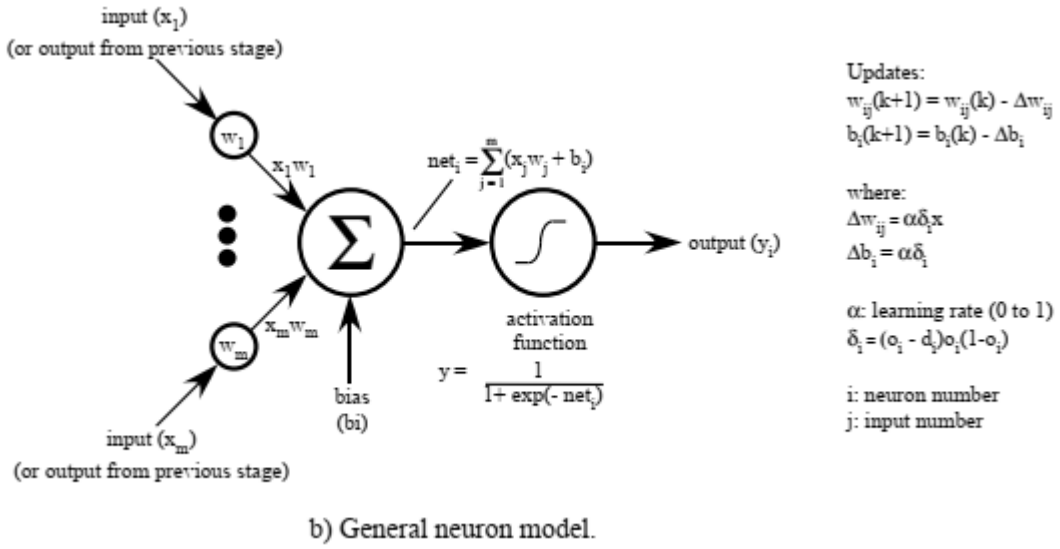
Figure 2.5: supervised learning

II.8 Learning algorithm

There are many learning algorithms to adjust the weight and biases on an ANN, like : Gradient Descent , Stochastic Gradient Descent (SGD), Batch Gradient Descent, Adagrad, Momentum Back propagation, Adam, … etc



a) Three layer ANN



b) General neuron model.

Figure 2.6: a three-layer ANN model b generalized single neuron model [9]

The general way how the training process is made in the case of three-layer network can be summarized by the following steps:

The training steps can be summarized:

1. Node designation: Llayer# : neuron#

2. Number of neurons in Layers L1, L2, and L3 - n, m, l

3. Weight designation: wlij wlayer# neuron# input#

Training process:

1. Initialize all weights and biases in all layers to small random values.

2. Provide training set input to x1 and x2 at Layer L1.

3. Output calculation proceeds layer-to-layer.

- Start at layer L2, calculate L2 net value for each neuron.

- Process L2 net values to activation functions to determine L2 layer neuron outputs.

- Layer 2 outputs now serve as Layer 3 inputs.

- Calculate L3 net value for each neuron.

- Process L3 net values to activation functions to determine L3 layer neuron outputs.

$$net_i = \sum_{j=1}^{m} X_{ij} W_J + b \qquad O_i \frac{1}{1+exp(-net_i)}$$

4. Calculate change in weights and biases between Layers 2 and 3

- Compare Layer 3 actual output (O) to desired output (d).

- Calculate change in weight: $\Delta wij = \alpha \delta oj$, $\Delta bi = \alpha \delta i$

α: learning rate (0 to 1) $\delta i = (oi - di)oi (1-oi)$

oj : output of neuron j in Layer L2

5. Calculate change in weights and biases between Layers 1 and 2:

$\Delta wij = \beta \delta HIoj$, $\Delta bi = \beta \delta HI$ $\delta HI = oi (1 - oi)$

$$\delta_{HI} = o_i(1 - o_i \sum_{k=1}^{m} \delta_k w_{ik}$$

β: learning rate (0 to 1)

oj : output of neuron j in Layer L1

oi : output of neuron i in Layer L2

δk: from neurons in Layer L3

wik: weight connecting a specific neuron (k) in the L2 layer

to a specific neuron (i) in the L3 layer

6. Update weights and biases between Layers L2 and L3 and Layers L1 and L2:

wij(k+1) = wij(k) - Δwij, bi (k+1) = bi (k) - Δbi

7. Obtain the error for neurons in Layer L3: E = 1

8.If error term has reached desired goals or maximize of epochs reached

else

continue the training at step 2.

Running - Categorizing

1. Obtain stored weights and bias from training.

2. Apply new input/output pair from outside (or inside) the training set.

3. Calculate net and output. 4. Place input pair into appropriate category. [10]

II.8.1. Backpropagation

Backpropagation is an algorithm used to train artificial neural networks. It involves propagating error gradients from the output layer to the input layer. It consists of a forward pass, where inputs are processed and an output is generated, and a backward pass, where the error is calculated and used to update the network's weights and biases. By iteratively adjusting the parameters based on these gradients, the network learns to minimize the difference between its predictions and the target outputs. Backpropagation is a key technique for optimizing neural networks and enabling them to learn from labeled data.

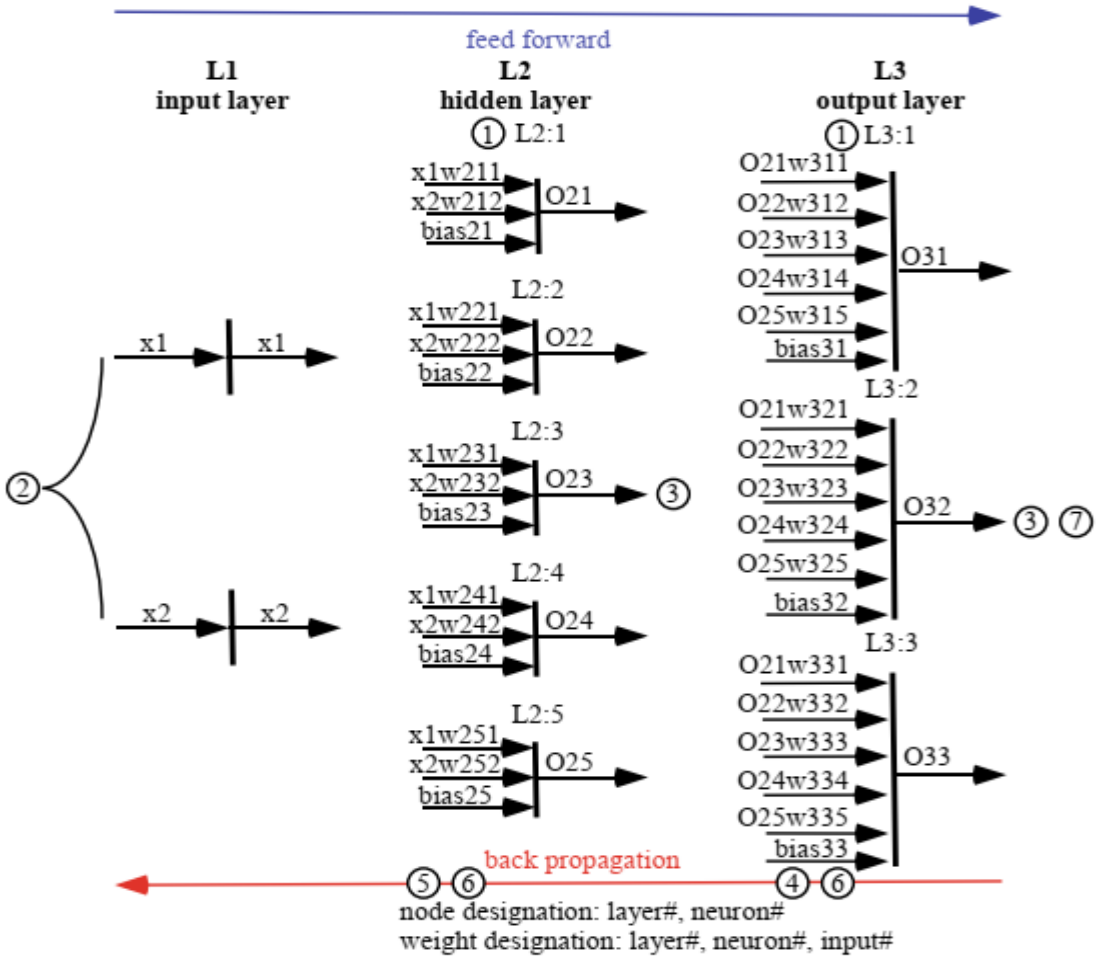


Figure. 2.7: Simplified ANN representation [10]

II.9. The advantages and disadvantages of neural networks

Advantage of neural networks

- Robustness with noisy data.
- Allows the simulation of a wide variety of behaviors.
- Fault tolerance (eliminate neurons ...).
- Automatic calculation of weights.
- Generalization. [4]

Disadvantages

- Unlike the decision tree method, the result is extremely opaque.
- The time required to set up a training program can be quite long.
- Complex representation.
- The study period can be long.
- There is a local minimum for the cost function.
- is difficult to explain the results when you have no prior knowledge [4]

II.10. Conclusion

In this chapter, we have seen how neurons are an essential component of the human brain, and how they are modeled after prototypes of biological nerve cells. Neural networks with parallel processing, learning, and approximation functions can be considered totally different from traditional computers.

CHAPTER III:

Tiny Machine Learning

III.1. Introduction

This part talks about machine learning at the edge and tiny machine learning, its advantages and areas of use.

III.2. Machine Learning at the Edge

Edge Machine Learning (Edge ML) is one of the most talked-about tech advancements since the Internet of Things (IoT), and for a good reason. With the rise of IoT came an explosion of Smart Devices connected to the Cloud, but the network was not yet ready to support this surge in demand. Cloud networks were congested, and companies overlooked key issues with Cloud computing, such as security. The solution: Edge ML.

So, what is Edge ML anyway? Edge ML is a technique by which Smart Devices can process data locally (either using local servers or at the device-level) using machine and deep learning algorithms, reducing reliance on Cloud networks. The term edge refers to processing that occurs at the device- or local-level (and closest to the components collecting the data) by deep- and machine-learning algorithms.

III.3. Definition of Tiny ML

Tiny Machine Learning (Tiny ML) is a field of study at the intersection of machine learning (ML) and embedded systems that enables running ML models on devices with extremely low-power microcontrollers. Let's explain some terms.

Embedded systems: are hardware and software systems designed to perform a dedicated function. They are computers, but in contrast to general-purpose computers such as a pc, a smartphone, or a tablet, embedded systems aim to perform specific tasks. Electronic calculators, digital cameras, printers, home appliances, ATMs are all examples of embedded systems.

Microcontrollers: constitute the hardware part of an embedded system. These are chips consisting of a processor, RAM, ROM, and Input/output (I/O) ports, enabling embedded systems to perform their task.

III.4. The benefits of Tiny ML

III.4.1. Low Latency

Data collected by edge device sensors is processed by an on-device ML model to generate the results. Tiny ML models are optimized to have a reduced number of parameters and there is no time wasted in transmitting the data to a datacenter server or cloud for processing. This in turn results in a very low latency (i.e., fast turnaround time) from data collection to result generation. Tiny ML applications can work in real time due to their low latency for mission critical applications.

III.4.2. Privacy

Machine learning consumes data. Privacy has been a major concern against the adoption of machine learning technologies since consumers are oftentimes unwilling to share their data or transport it over the internet due to security concerns. Tiny ML enables on-device processing of data in real time. In Tiny ML applications, since data is neither transmitted out of device nor stored anywhere, there are no privacy concerns.

III.4.3. Low Power

Tiny ML applications are expected to run on resource constrained devices, process real time data and produce results within data sampling intervals. Driven by all these requirements Tiny ML models are optimized to run with fewer parameters, use reduced computations and consume low power.

Tiny ML applications mostly use on-device sensors' data. Majority of the sensor data (other than microphone audio data and camera video data) volumes are small (less than 1K bytes/sec). Consequently, machine learning models to process these data are also light.

III.4.4. Reduced Bandwidth

Tiny ML applications cater to small, resource constrained devices with poor to no internet connectivity. On-device sensors capture the data, data is processed on-device and hence there is no raw sensor data transmission bandwidth involved. Sometimes Inference or analytics results limited to 100s of bytes are transmitted to cloud requiring very low bandwidth.

III.5. Current Landscape and Challenges in Tiny ML

III.5.1. Current Landscape

a. Hardware and Software Heterogeneity

Heterogeneous computing refers to systems that use more than one kind of processor or core. These systems gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar coprocessors, usually incorporating specialized processing capabilities to handle particular tasks.

b. Lack of Benchmarking Tools

Benchmarking is a valuable quality management tool that allows organizations to compare their performance with that of other companies or industry standards. This is what the little machine is missing.

c. Lack of Datasets

Lack of data means more AI hidden doubts and inadequate outcomes. It leads us to the fact that without access, it is impossible to prepare data for machine learning, and even the most outstanding software won't matter without sufficient data filling.

d. Lack of Popularly Accepted Models

Often, in tiny machine learning software, it is difficult to find models that are compatible with the lived reality.

III.5.2. Challenges in Tiny ML

While Tiny ML presents immense potential, it also faces several challenges that must be addressed to fully realize its capabilities:

a. **Model Optimization:**

   Developing highly efficient models that can perform complex tasks with limited resources remains a significant challenge. Further research into model compression techniques and architecture design is crucial.

b. **Hardware Limitations:**

   The development of dedicated, low-power hardware accelerators for Tiny ML is still in its infancy. Continued innovation in hardware design will play a vital role in advancing Tiny ML capabilities.

c. **Energy Efficiency:**

As devices become more intelligent, managing power consumption becomes even more critical. Developing energy-efficient ML algorithms and hardware will be key to the long-term success of Tiny ML.

d. Privacy and Security:

With more devices processing sensitive data, ensuring privacy and security becomes increasingly important. Researchers and developers must address these concerns as they work on new Tiny ML applications.

III.6. Applications of Tiny ML

III.6.1. Industries:

Tiny ML, when used on low-powered devices, can detect faults in a machine ahead of time constantly. It implies maintenance based on predictions. One such example is introducing an IoT device by an Australian start-up, Ping Services that monitors wind turbines by attaching itself to the turbine's exterior. It alerts the authorities if it senses any potential issue or malfunction.

III.6.2. Agriculture:

Tensor Flow Lite tool enables farmers to detect diseases in a plant when they take a picture of it. It works on any device and needs no internet connection. The process allows for the protection of agricultural interest and is a crucial requirement for remote farmers.

III.6.3. Hospitals:

A project called the Solar Scare Mosquito uses Tiny ML to stop the spread of diseases like dengue, malaria, etc. It runs on solar power, detects mosquito breeding conditions, and signals the water to prevent mosquito breeding.

III.6.4. Aquatic Conservation:

Tiny ML-powered devices monitor whales in real-time and alert them during strikes in busy shipping lanes.

III.6.5. Smart Homes:

Tiny ML can improve the efficiency and responsiveness of smart home systems by enabling local decision-making, reducing latency, and increasing privacy.

III.7. Future of Tiny ML

As the demand for smart devices continues to grow, the future of tiny ML looks bright. Advances in hardware and software are making it easier to develop and deploy tiny ML models on a wide range of devices.

In the coming years, we can expect to see more innovative applications of tiny ML, particularly in areas such as autonomous vehicles, robotics, and augmented reality. As the technology continues to evolve, it has the potential to revolutionize the way we interact with and perceive the world around us.

- **New Dimension:** We emphasize the future growth and adoption of the Tiny ML paradigm shall depend on several aspects. For example, an intelligent edge system requires the edge software to integrate online learning, real-time learning, distributed training, coherence between edge-device intelligence, sophisticated and data-network management. Such cooperation should be extended for localized security and privacy improvement so that end-user context can be preserved at edge devices. Furthermore, we should focus on edge infrastructure development, edge platform orchestration, and low-cost knowledge sharing capabilities into the edge systems.

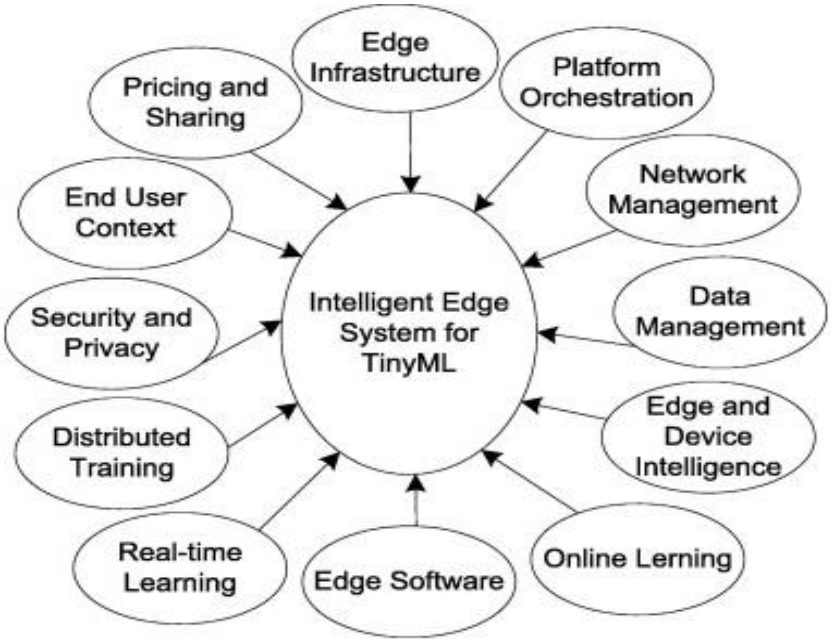


Figure 3.1: Basics key enablers of Tiny ML

III.8. Machine-Learning Tools and Process Flow

Machine learning workflows define which phases are implemented during a machine learning project. The typical phases include data collection, data pre-processing, building datasets, model training and refinement, evaluation, and deployment to production.

Machine learning involves showing a large volume of data to a machine so that it can learn and make predictions, find patterns, or classify data. The three machine learning used type in our application is a supervised learning.

The general steps for a machine learning model creation can be summarized as follows

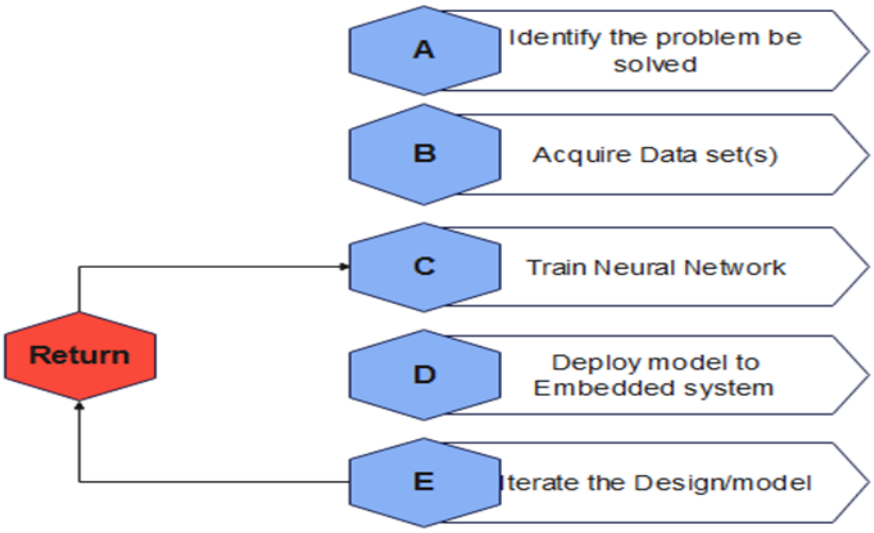


Figure 3.2: Steps for a machine learning model creation.

Remark:

When the studied model fails to give accurate results when deployed on target device, the developer returns to step C for model re-work/adjustment.

III.9. General High-Level Software Architecture and tasks

Machine Learning architecture is defined as the subject that has evolved from the concept of fantasy to the proof of reality. As earlier machine learning approach for pattern recognitions has lead foundation for the upcoming major artificial intelligence program. Based upon the different algorithm that is used on the training data machine learning architecture is categorized into three types.

The following figure represents the general architecture for machine learning:

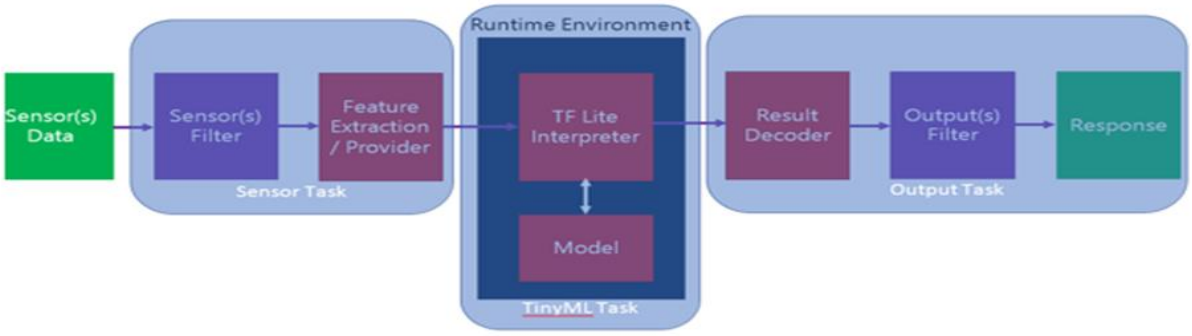


Figure 3.3: General Architecture for machine learning

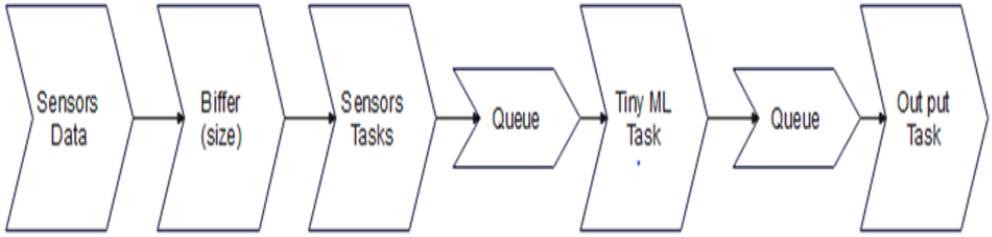


Figure 3.4: Software flow tasks for ML

III.10. Options for getting Training Data

- **Online Data Sets:** consist of open source data sets. That can be downloaded from the net for example:
  - ✓ MNIST for classifying hand written digits.
- **Generate it :**(From a mathematical model): consists of generating data from a known model.
- **Collect it:** by using an appropriate hardware and software setup (MCU, Sensor, and Firmware that sends data via serial) and save it in a useful format like: json, bin, csv, dat……etc.
- **Buy it:** from vendor when the model of the system is unavailable or complex, and data collection is expensive or hard to setup.

III.11. Machine Learning Frameworks

A machine learning framework is a tool that lets software developers, data scientists, and machine learning engineers build machine learning models without having to dig into the underlying working principle (math and stat) of the machine learning algorithms. It streamlines the development process where the programmers do not have to reinvent the wheels for creating a specific application. Machine learning frameworks have a bunch of similar working libraries that simplify the development of machine learning models.

Figure 3.5: List of some machine learning frameworks

There are many popular machine learning frameworks for managing machine learning projects, two of them will be presented:

III.11.1. Tensor Flow

Tensor Flow is an open-source machine learning framework developed by Google. It provides a platform for building and training various types of machine learning models, particularly deep neural networks. Tensor Flow's building blocks include:

a. **Tensors:**

Tensors are the fundamental data structures in Tensor Flow. They are multi-dimensional arrays that represent the input, output, and intermediate data in a computation.

b. Operations:

Tensor Flow allows you to define a series of mathematical operations on tensors, such as addition, multiplication, convolution, etc. These operations build a computational graph, which defines the flow of data through the model.

c. **Variables:**

Variables hold the weights and biases of a model and are updated during training. They are used to optimize the model's performance by minimizing a defined loss function.

d. **Graph:**

A Tensor Flow computation is represented as a computation graph, where nodes represent operations and edges represent tensors. This graph enables Tensor Flow to efficiently parallelize and distribute the computations across different hardware devices.

e. **Sessions:**

A Tensor Flow session is an environment for executing operations and evaluating tensors. Sessions can be run on CPUs, GPUs, or distributed across multiple machines, providing flexibility and scalability in the execution of computations.

f. **High-level APIs:**

Tensor Flow also offers high-level APIs like Keras and Estimators, which provide a more user-friendly interface for building and training models, abstracting away the lower-level details.

These building blocks together make Tensor Flow a powerful and versatile framework for developing and deploying machine learning models across a range of applications.

Pros of Using Tensor flow:

- It can render subtle computational graph visualizations.
- Since Google backs it, it is efficient, provides seamless performance, and caters to quick updates.
- It also allows executing subparts of a graph, which enable machines to extract discrete data.

III.11.2. Tensor Flow Lite

Tensor Flow Lite (TF Lite) is a lightweight and efficient machine learning framework developed by Google. It is specifically designed for deploying machine learning models on mobile and embedded devices. Here is a simplified schematic to illustrate the key components of Tensor Flow Lite:

a. Model Conversion:

Tensor Flow Lite supports conversion of Tensor Flow models (trained using Tensor Flow) into a more optimized format suitable for mobile devices. This conversion process reduces the model size and improves its performance.

b. Model File Format:

The converted Tensor Flow model is saved in a file format specifically designed for Tensor Flow Lite, such as .TF lite. This format is optimized for efficient inference on mobile devices.

c. **Tensor Flow Lite Interpreter**:

At the core of Tensor Flow Lite, there is an interpreter that loads the converted model file and runs the machine learning inference tasks on the device. The interpreter provides an interface to execute the computations required for inference.

d. **Neural Network Operations:**

The interpreter consists of a set of pre-defined neural network operations, such as convolution, pooling, fully connected layers, activation functions, and more. These operations are optimized for running on mobile hardware, ensuring efficient execution.

e. **Hardware Acceleration:**

Tensor Flow Lite leverages hardware acceleration whenever possible to further optimize the performance. It utilizes the specific capabilities of the device's hardware, such as GPUs (Graphics Processing Units), TPUs (Tensor Processing Units), or dedicated AI accelerators, to speed up the inference process.

f. **Application Integration:**

Tensor Flow Lite provides APIs and libraries that allow developers to integrate machine learning models into their mobile applications seamlessly. This enables various use cases like image recognition, natural language processing, recommendation systems, and more.

Overall, Tensor Flow Lite serves as a framework that enables efficient deployment of machine learning models on resource-constrained devices, making it ideal for running AI applications on mobile phones, IoT devices, and other embedded systems.

Pros of Using Tensor flow:

- Tensor Flow models can be efficiently transferred into Tensor Flow Lite models for mobile-friendly deployment.
- It's offers a relatively simple way for mobile developers to build applications using Tensor flow machine learning models on iOS and Android devices
- Tensor Flow Lite decreases inference time, which means problems that depend on performance time for real-time performance are ideal use cases of Tensor Flow Lite.

III.11.3. Keras

is an open-source framework developed on top of Tensor Flow long research and adaptation phase, Keras became the choice of high-level neural network. François Chollet is a Google engineer who designed it to be fast, easy to implement, and modular by nature. ML developers can apply it in different domains like healthcare, corporate. It is written in Python and can efficiently run on GPUs and CPUs. After going through an insights, sales predictions, customer support, virtual assistants, etc.

Pros of Using Keras:

- It has a collection of pre-trained models.
- Keras is a high-level API that supports multi-platform and multi-backend integrations.
- Developers use it for rapid prototyping.

III.11.4. Tensor flow Lite for Microcontrollers

Tensor Flow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices.

It is characterized by:

•Runs machine learning models on microcontrollers.

•Core run-time is ~16kB

•Does not require an OS (can run bare metal)

•Written in C++ 11

•Several example cases already available:

•Hello World

• Keyword spotting (Micro speech)

•Gesture detection.

III.12. Edge Impulse

Edge Impulse was designed for software developers, engineers and domain experts to solve real problems using machine learning on edge devices without a specialized skill in machine learning.

Edge Impulse is a cutting-edge technology that enables the development of embedded machine learning for edge devices. As the demand for intelligent and autonomous devices grows, the need for efficient and scalable machine learning solutions becomes paramount. Edge Impulse is a promising platform for developing embedded machine learning for edge devices. Edge Impulse provides a user-friendly interface, efficient algorithms, and powerful hardware integration that makes it the best option for developers and researchers in this field.

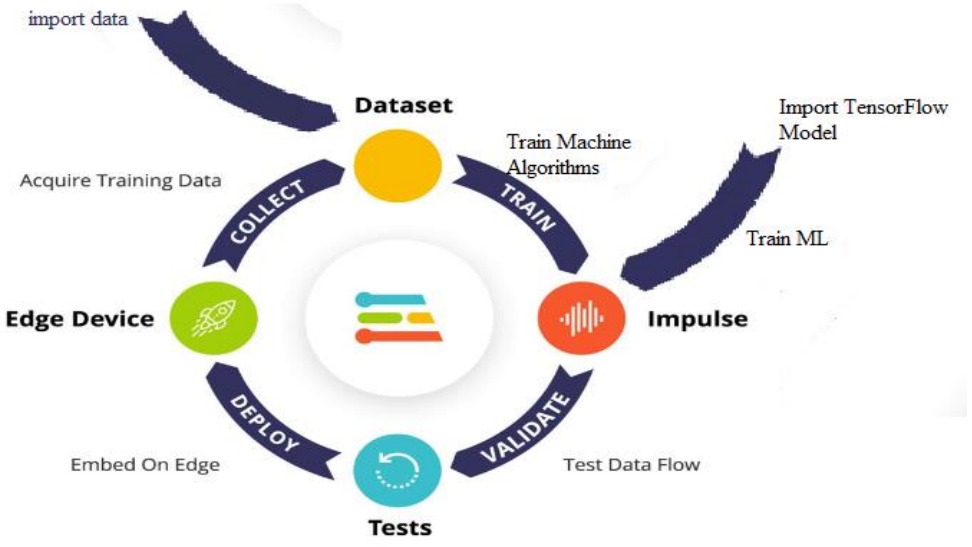


Figure 3.6: Edge Impulse ML design flow

III.12.1. Edge Impulse – Project Design flow

- **Project Creation**

After entering your email, name, and password correctly, you can start your experience on edge impulse:

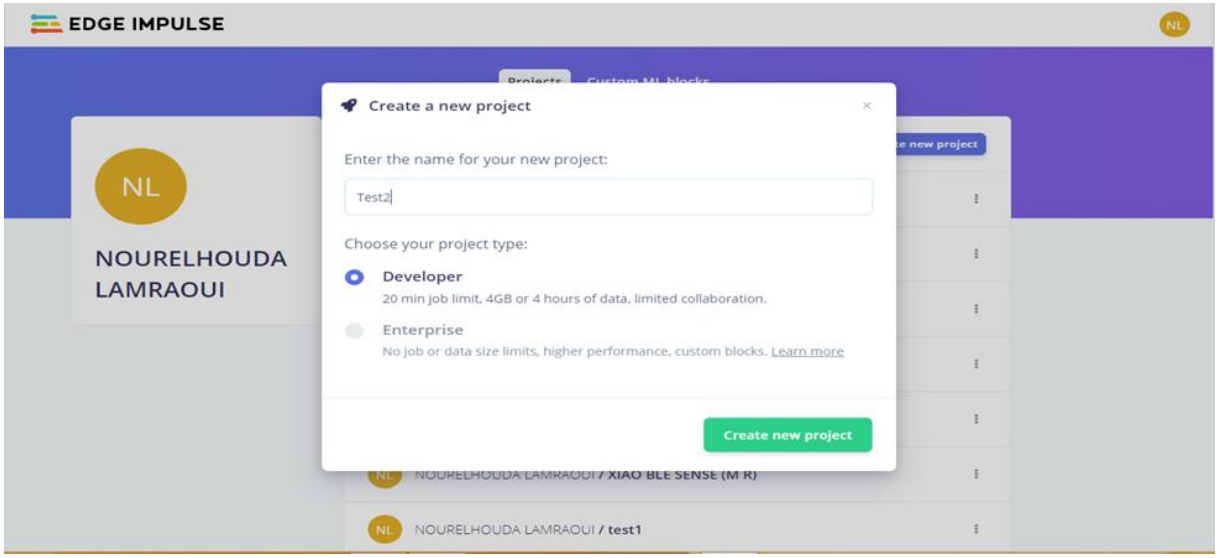


Figure 7.4: Project creation on edge impulse

III.12.2. Components of the edge impulse window

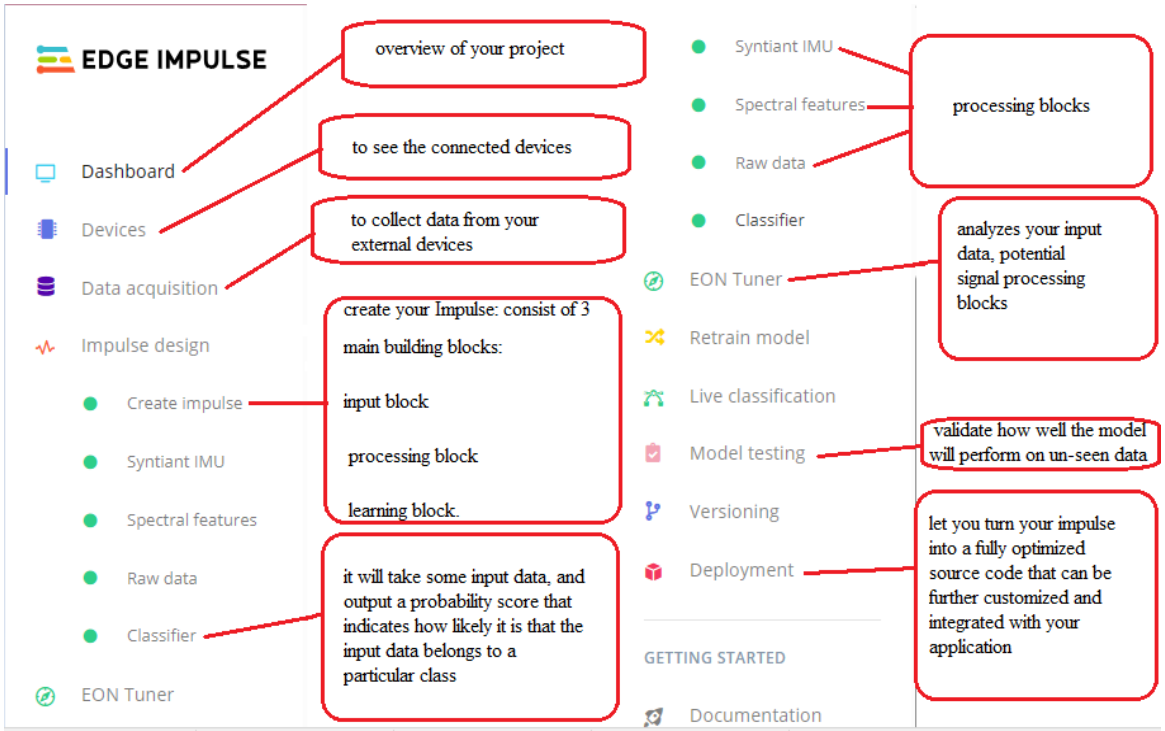


Figure 3.8: some edge impulse blocks

a. Dashboard

After creating your Edge Impulse Studio project, you will be directed to the project's dashboard. The dashboard gives a quick overview of your project such as your project ID, the number of devices connected, the amount of data collected, the preferred labeling method, among other editable properties. You can also enable some additional capabilities to your project such as collaboration, making your project public, and showcasing your public projects using Markdown READMEs.

The figure below shows the various sections and widgets of the dashboard that we will cover here:

b. Devices

There is a wide variety of devices that you can connect to your Edge Impulse project. These devices can help you collect datasets for your project, test your trained ML model and even deploy your ML model directly to your development board with a pre-built binary application. On the Devices tab, you'll find a list of all your connected devices and a guide on how to connect new devices that are currently supported by Edge Impulse.

c. Data sources

The data sources page is actually much more than just adding data from external sources. It let you create complete automated data pipelines so you can work on your active learning strategies.

From there, you can import datasets from existing cloud storage buckets, automate and schedule the imports, and, trigger actions such as explore and label your new data, retrain your model, automatically build a new deployment task and more.

d. Data Acquisition and Preprocessing

All collected data for each project can be viewed on the Data acquisition tab. You can see how your data has been split for train/test set as well as the data distribution for each class in your dataset. You can also send new sensor data to your project either by file upload, Web USB, Edge Impulse API, or Edge Impulse CLI.

e. Dataset train/test split ratio

The train/test split is a technique for training and evaluating the performance of a machine learning algorithms. It indicates how your data is split between training and testing samples. For example, an 80/20 split indicates that 80% of the dataset is used for model training purposes while 20% is used for model testing.

This section also shows how your data samples in each class are distributed to prevent imbalanced datasets which might introduce bias during model training.

f. Raw data

The Raw Data block generates windows from data samples without any specific signal processing. It is great for signals that have already been pre-processed and if you just need to feed your data into the Neural Network block.

g. Data explorer

The data explorer is a visual tool to explore your dataset, find outliers or mislabeled data, and to help label unlabeled data. The data explorer first tries to extract meaningful features from your data (through signal processing) and then uses a dimensionality reduction algorithm to map these features to a 2D space. This gives you a one-look overview of your complete dataset patterns.

h. Impulse design

After collecting data for the project, you can now create your Impulse. A complete Impulse will consist of 3 main building blocks: input block, processing block and a learning block.

This last one is the most important, where the own machine learning pipeline will be built.

The following figure shows an example for the first step in Impulse design for movement classification using accelerometer data.

•Window Size – the size of the data that we will slide across to analyze data features

•Window Increase – the size of the window steps

•Zero-pad – If the sample data set is smaller than the window, pad remaining space with 0’s.

i. EON Tuner

The EON Tuner helps you find and select the best embedded machine learning model for your application within the constraints of your target device. The EON Tuner analyzes your input data, potential signal processing blocks, and neural network architectures - and gives you an overview of possible model architectures that will fit your chosen device's latency and memory requirements.

j. Retrain model

The Retrain model feature in the Edge Impulse Studio is useful when adding new data to your project. It uses already known parameters from your selected DSP and ML blocks then uses them to automatically regenerate new features and retrain the Neural Network model in one single step. You can consider this a shortcut for retraining your model since you don't need to go through all the blocks in your impulse one by one again.

k. Live classification

Live classification lets you validate your model with data captured directly from any device or supported development board. This gives you a picture on how your model will perform with real

World data. To achieve this, go to live classification and connect the device or development board you want to capture data from.

III.13. Conclusion

In conclusion, Tiny ML represents a major advance in the field of machine learning, enabling intelligent decision-making on low-power devices. While there are many challenges to overcome, tiny machine learning offers many advantages, including real-time processing, greater privacy and security, and lower energy consumption.

As the technology continues to evolve, we can expect more innovative applications of micro-ML in various industries that will change the way we live and work.

CHAPTER IV:

TinyML for Gesture Recognition Application Based on XIAO BLE SENSE and Edge Impulse

IV.1. Introduction

In this part, we will introduce a ML application for gesture recognition on a constrained device at the edge, the selected device is XIAO BLE sense board. We will present the steps and the work flow process for developing a tiny ML working application on an MCU.

IV.2. The Seeed Studio XIAO nRF52840 BLE sense

Seeed Studio XIAO Series are diminutive development boards, sharing a similar hardware structure, where the size is literally thumb-sized. The code name "XIAO" here represents its half feature "Tiny", and the other half will be "Puissant".

Figure 4.1: SEEED STUDIO XIAO Nrf52840 BLE sense

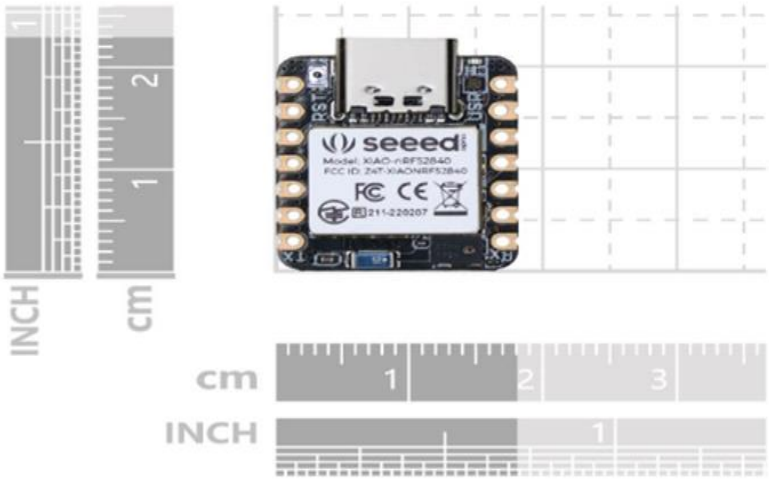


Figure 4.2: Dimensions of SEEED STUDIO Nrf52840 BLE sense

IV.2.1. Hardware Description

The development board equipped with Nordic nRF52840 MCU chip, which integrates Bluetooth 5.0 connectivity. Its small dimensions make it suitable for use in wearable devices and IoT projects. It is dedicated to surface mounting, and the built-in Bluetooth antenna can greatly facilitate the rapid implementation of IoT projects.

Seeed XIAO BLE Sense is equipped with a digital microphone with pulse density modulation (PDM). It can receive real-time audio data, which allows you to use it for audio recognition. There is also a 6-axis Inertial Measurement Unit (IMU) on the board, which can be very useful in Tiny ML projects such as gesture recognition.

The user has a Near Field Communication (NFC) interface, a reset button and a 3-in-1 LED (user LED) with a charge LED that indicates the charging status when the battery is connected. There are 11 digital I / O that can be used as PWM pins and 6 analog I/O used as ADC pins. It supports all three popular serial interfaces such as UART, I2C and SPI.

Besides, the on-board MCU has several I/O s that can be used as ADC pins and communication interfaces:

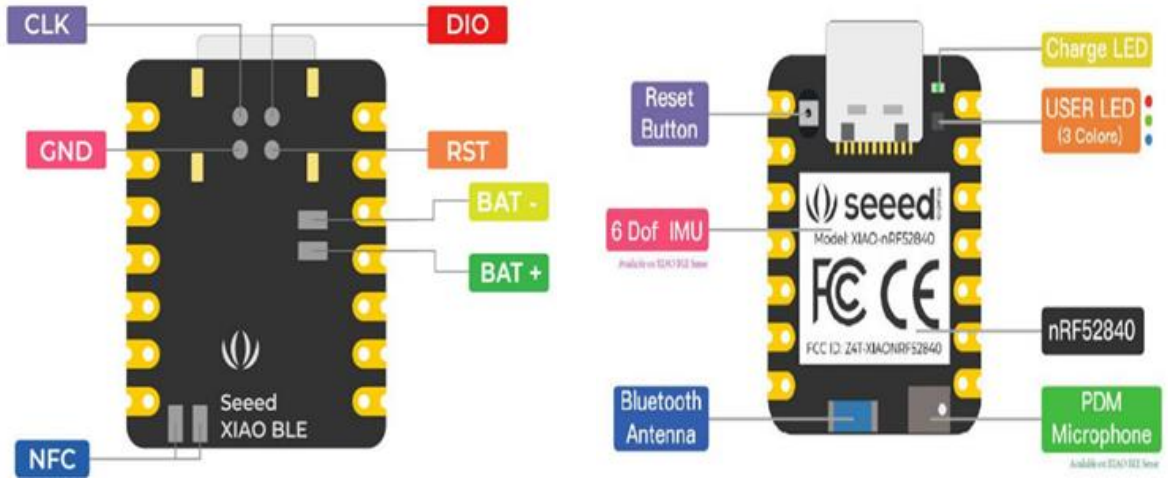


Figure 4.3: Description of seeed studio Xiao Nrf52840 BLE sense development board

IV.2.2. Hardware Pin out

There are 11 digital I/O that can be used as PWM pins and 6 analog I/O that can be used as ADC pins. It supports UART, IIC, and SPI all three common serial ports.

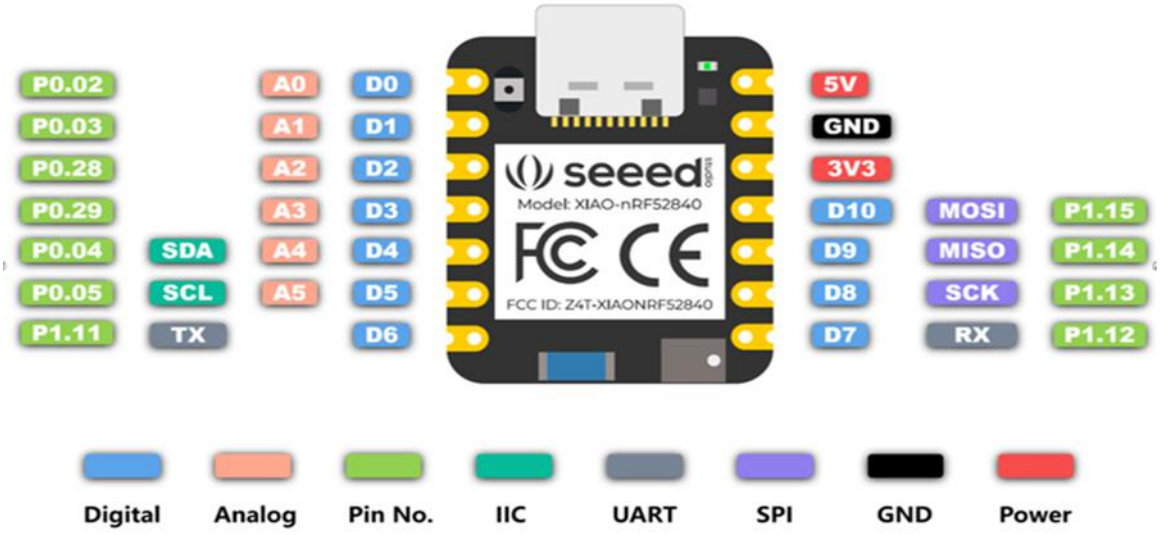


Figure 4.4: Description of card Nrf52840 BLE sense Pin out.

IV.2.3. Features of the SEEED Studio nRF52840 BLE sense board

- Powerful wireless capabilities: Bluetooth 5.0 with onboard antenna
- Powerful CPU: Nordic nRF52840, ARM® Cortex®-M4 32-bit processor with FPU, 64 MHz
- Ultra-Low Power: Standby power consumption is less than 5μA
- Battery charging chip: Supports lithium battery charge and discharge management
- Onboard 2 MB flash
- Onboard PDM microphone
- Onboard 6-axis LSM6DS3TR-C IMU
- Ultra Small Size: 20 x 17.5mm, Seeed Studio XIAO series classic form-factor for wearable devices
- Rich interfaces: 1xUART, 1xI2C, 1xSPI, 1xNFC, 1xSWD, 11xGPIO (PWM), 6xADC
- Single-sided components, surface mounting design
- 3-color user LED
- Size: 21 x 18mm
- Weight: 32g

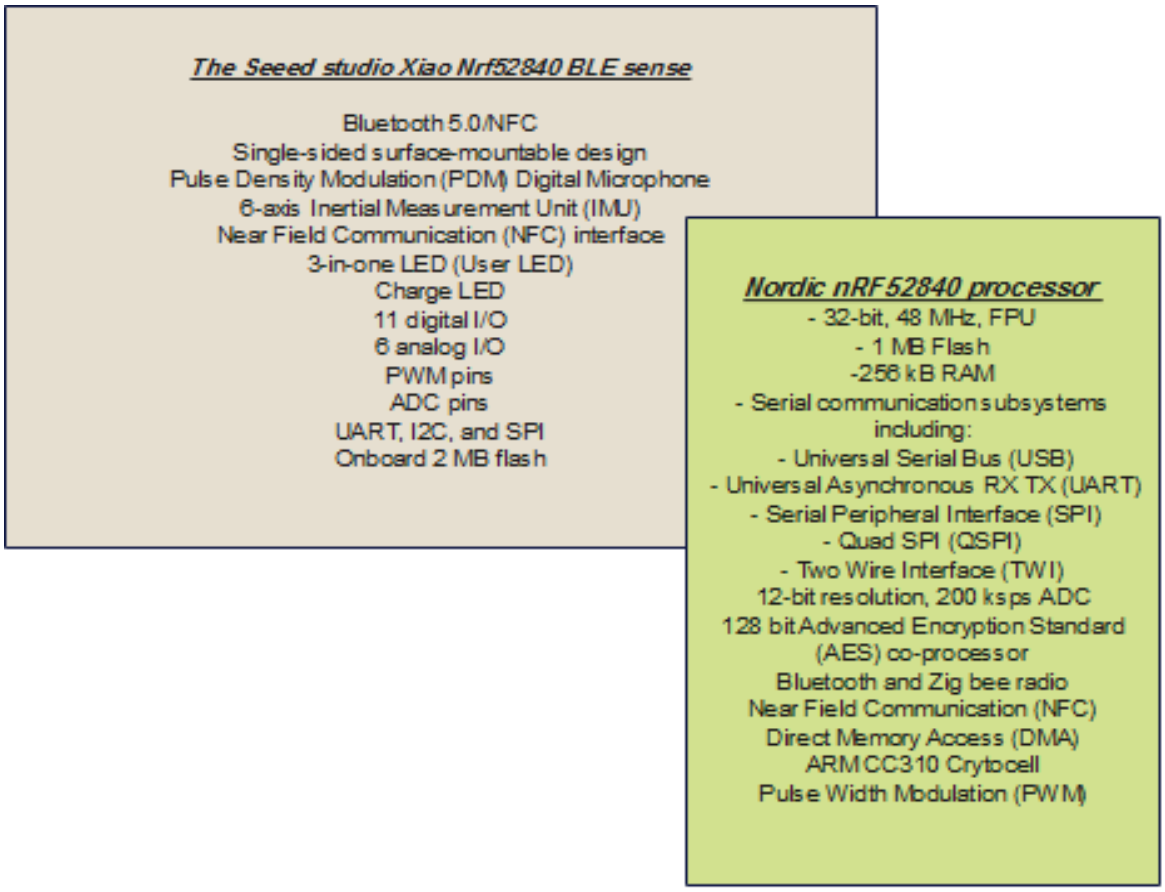


Figure 4.5: XIAO BLE Sense and it's on board nRF52840 Microcontroller

IV.3. Gesture recognition application based on Tiny ML

IV.3.1. Hardware setup

The hardware used in the gesture recognition application is listed below:

- ✓ 1 x or Studio XIAO nRF52840 Sense (or Seeed Studio XIAO nRF5284 with external accelerometer shield/board)
- ✓ 1 x Computer (with the right tools installed)
- ✓ 1 x USB Type-C cable
- ✓ 1 x 0.96 Oled Display
- ✓ Lithuim battery

IV.3.2. Steps to test project hardware parts

To have a successful project from hardware point of view we must test the board and each used hardware component from IMU sensor and oled display

a. Blink:

first, we are going to connect the Seeed Studio XIAO BLE nRF2840 sense to the computer via a USB Type-C cable and upload a simple code from Arduino IDE to check whether the board is functioning well.

Software Setup :

Download and Install the latest version of Arduino IDE according to your operating system. Then launch the Arduino application.

There is no pre-installed package for NRF52840 Board. So we need to first install the board on Arduino IDE. To do that Navigate to File > Preferences, and fill “Additional Boards Manager URLs” with the URL below:

https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.json

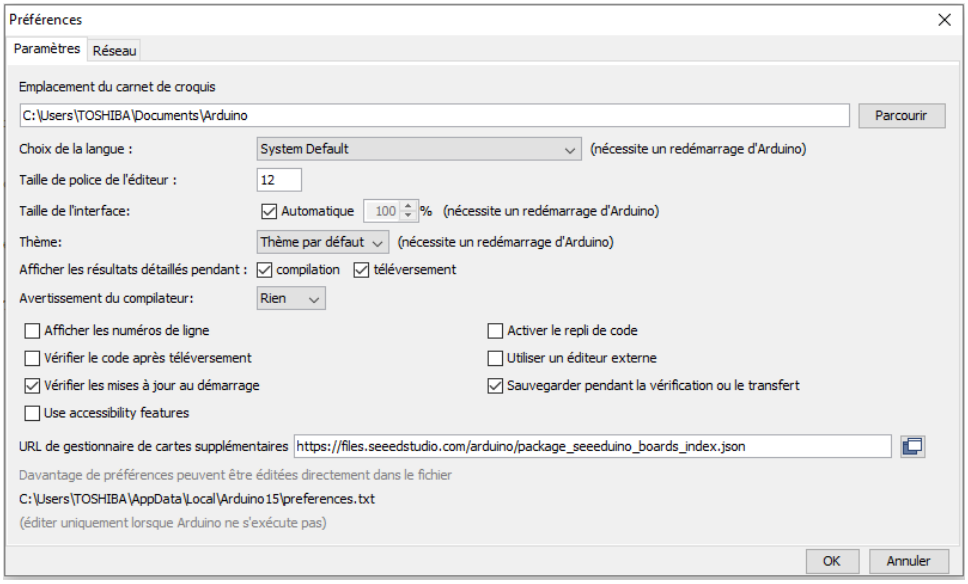


Figure 4.6: Arduino references

Navigate to *Tools* > *Board* > *Boards Manager*, type the keyword "*seeed nrf52*" in the search box, select the latest version of Seeed nRF52 Boards, and install it.

Figure 4.7: Arduino board manager

After installing the board package, navigate to *Tools* > *Board* > *Seeed nRF Boards* and select "*Seeed XIAO BLE – nRF52840 Sense*". Now we have finished setting up the XIAO BLE Sense for Arduino IDE.

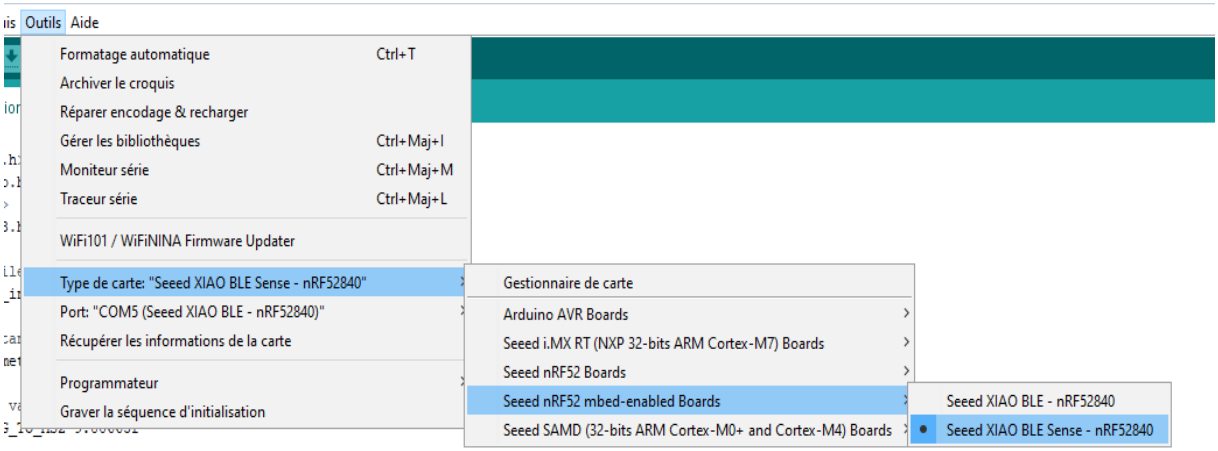


Figure 4.8: connect the board

Navigate to *Tools > Port* and select the **serial port**. You can get the port number from the Device Manager.

Navigate to *File* > *Examples* > *Basics* > *Blink* to open Blink example. Then upload the code.

After uploading the code, the on-board will turn-ON and OFF for 1 second.

Figure 4.10: blink the led

b. **0.96 Oled**

Displaying a message (gesture recognition) on 0.96″ I2C OLED Display

Now let us interface SSD1306 0.96″ I2C OLED Display with Seeed XIAO nRF52840 Sense.

The connection is fairly simple as shown in image below.

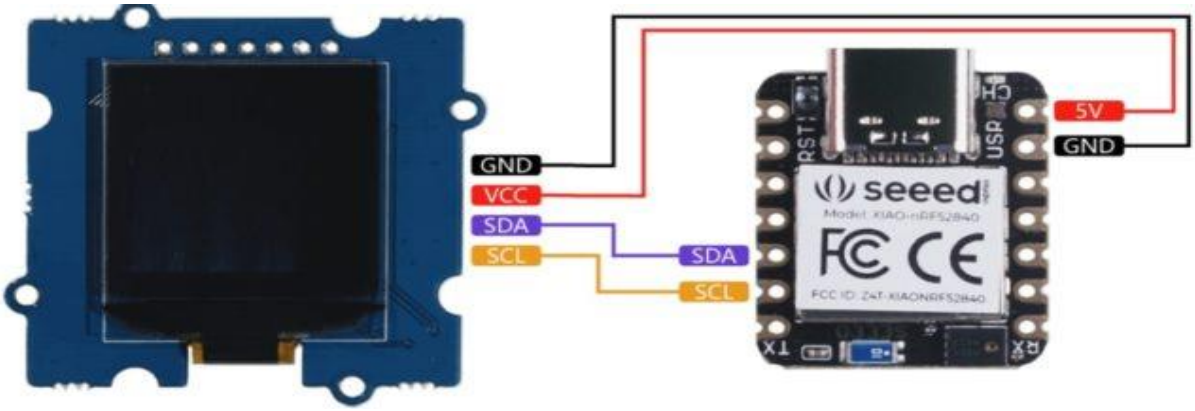


Figure 4.11: Connect the Seeed XIAO BLE SENSE with the OLED

Connect the VCC & GND pin of OLED Display to nRF5840 3.3V & GND Pin. Also, connect the SDA & SCL Pin of OLED to nRF5840 SDA & SCL Pin.

To use the code with Arduino IDE, you need to install **Adafruit GFX Library** and also **Adafruit SSD1306 Library**.

Now upload the following code to the Seeed XIAO nRF52840 board.

As soon as the code gets uploaded the OLED Display will start displaying the text .

Figure 4.12: displaying the text

c. IMU capture

The XIAO BLE Sense is equipped with a high-precision 6-Axis Inertial Measurement Unit (IMU) which includes a 3-axis accelerometer and a 3-axis gyroscope. There is also an embedded temperature sensor on this module.

First, we will read the value of the LSM6DS3 IMU Sensor from Seeed XIAO BLE nRF52840 Sense.

- Connect the Type C USB Cable to the XIAO BLE Board. Then connect the other end of the cable to your PC. Hence the Serial communication with PC will establish.

- To use the IMU Sensor, we need the LSM6DS3 library. You can download the **LSM6DS3 Library**.
- Once the library is downloaded, open your Arduino IDE. Then add the library through add zip folder.

- upload the code to your XIA BLE Sense Board.

- **Testing IMU Sensor:**

After the code is uploaded, open the Serial Monitor.

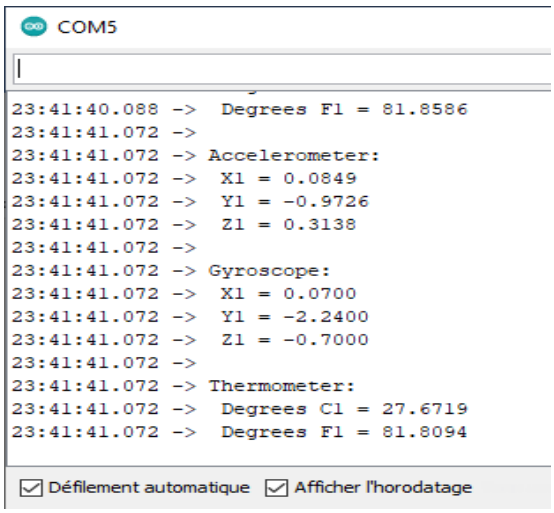


Figure 4.13 : the IMU capture results in the serial monitor

IV.3.3. Software setup

The required libraries for Arduino IDE are listed below. It is highly recommended that use the codes here to check whether the hardware is functioning well.

Seeed_Arduino_LSM6DS3-master

U8g2 or ardafruit_GFX + ardafruit_SSD1306

To set Seeed Studio XIAO nRF52840 Sense up in Edge Impulse, you will need to install the following software:

Node.js v12 or higher.

Arduino CLI

The Edge Impulse CLI and a serial monitor. Install by opening command prompt or terminal and run:

a. Connecting to Edge Impulse

Step 1. Connect the Seeed Studio XIAO nRF52840 Sense to your computer via a USB Type-C cable.

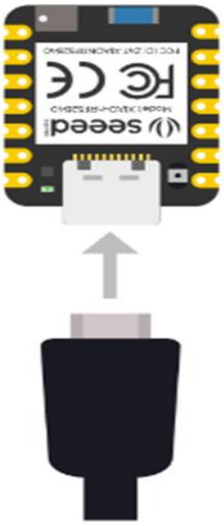


Figure 4.14. Cennect Seeed studio

Step 2. Create a new project in Edge Impulse

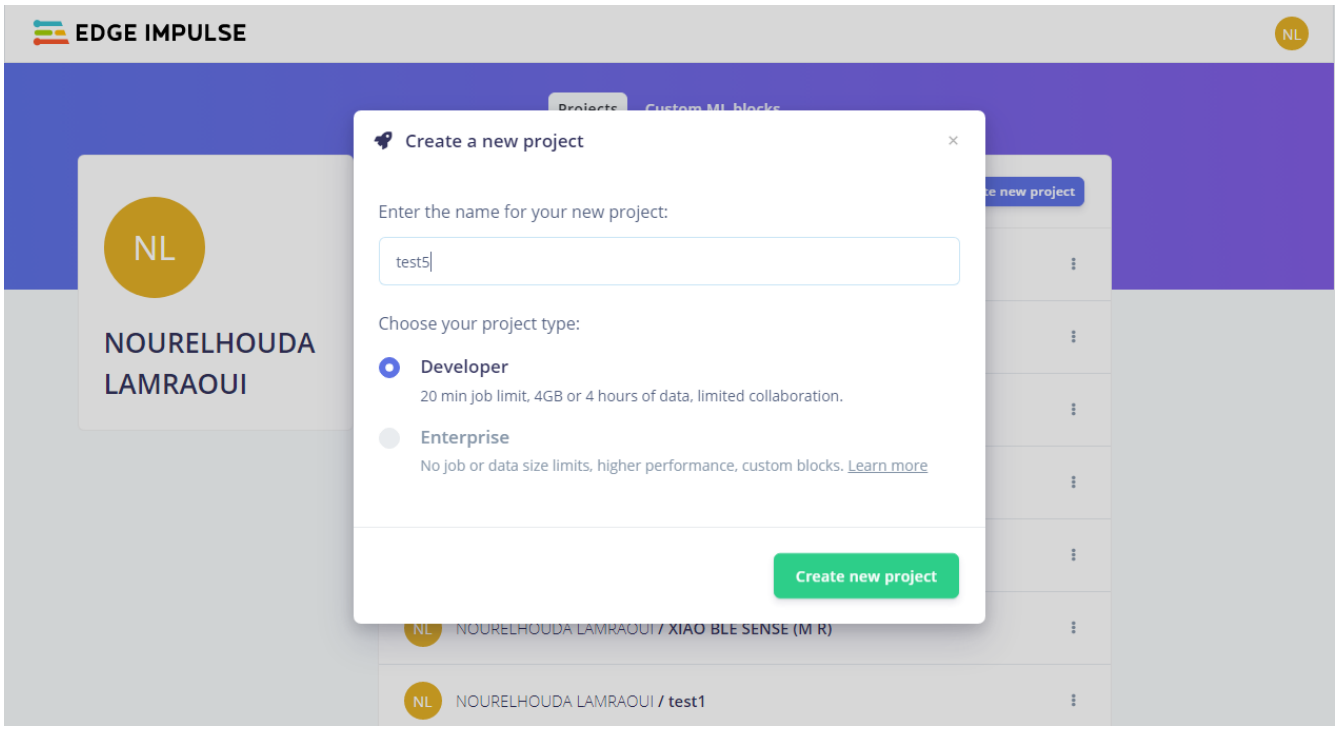


Figure 4.15: Project creation on edge impulse

Step 3. Choose "Accelerometer data" and click "Let's get started!"

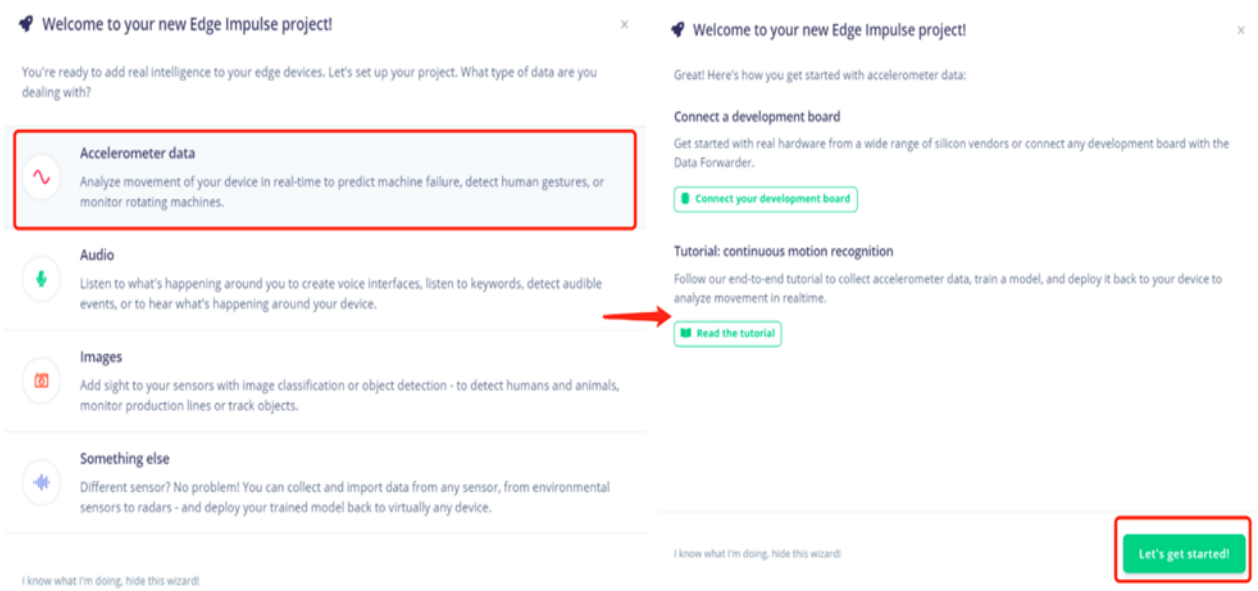


Figure 4.16: Choosing the type of data

Step 4. Connect the XIAO nRF52840 Sense to Edge Impulse

Move to Edge Impulse "Data acquisition" page, the outcome should be like this if the connection is successful. The XIAO BLE SENSE board is not yet supported by the Edge impulse platform we did not find it in the device list shown on the right of the page.

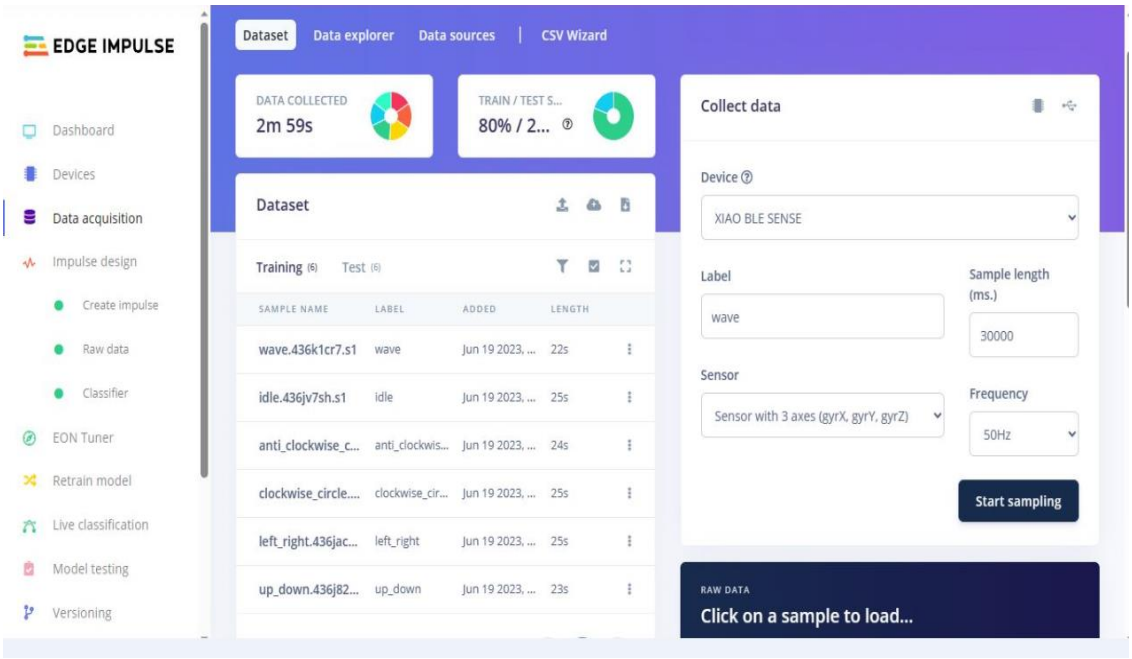


Figure 4.17: Data Acquisition for our project

b. Training (and test) data acquisition

Step 5. Run the command in your terminal or cmd or PowerShell to start it.

Because The XIAO BLE SENSE board is not yet supported by the Edge impulse platform, we proceed IMU data acquisition by using the serial terminal connected to Edge impulse server using a data forwarder.

Run the command in your terminal or cmd or PowerShell to start it.

Edge-impulse-data-forwarder

Step 6. We need to use the CLI to connect the Seeed Studio XIAO nRF52840 Sense with Edge Impulse. First, login your account and choose your project.

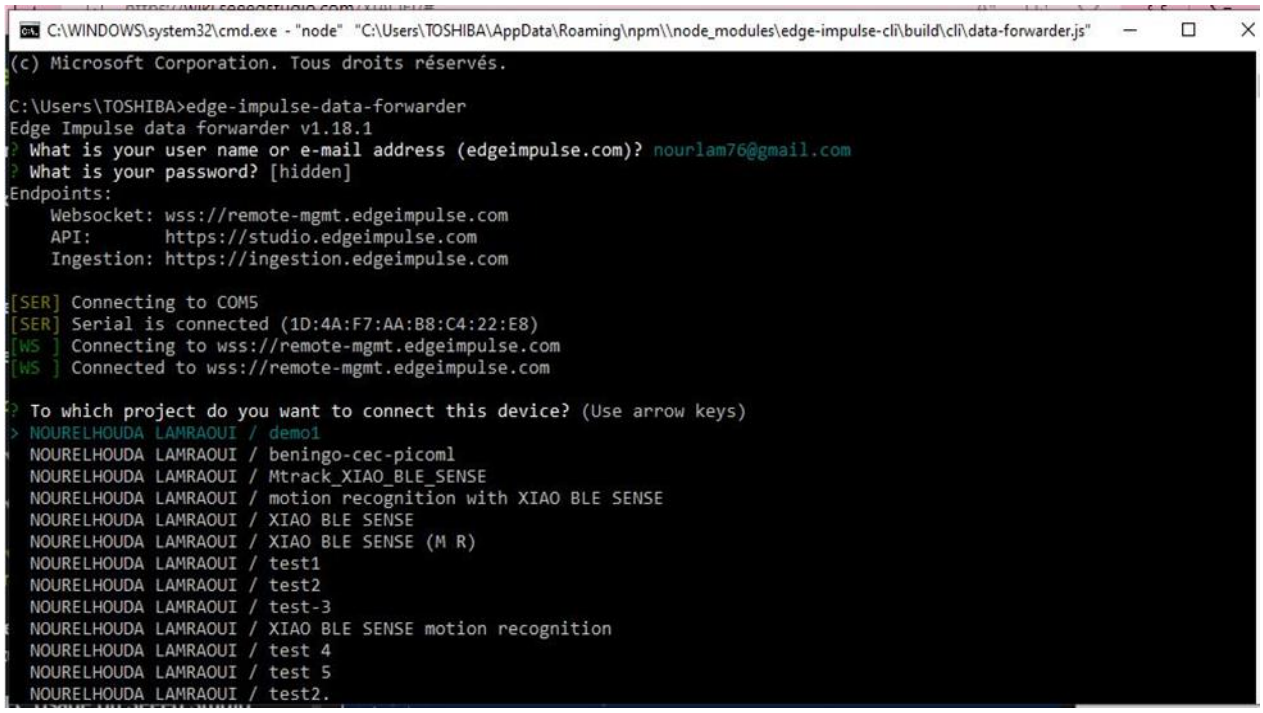


Figure4.18. opening the CMD and connect the Seeed Studio XIAO nRF52840 Sense with Edge Impulse

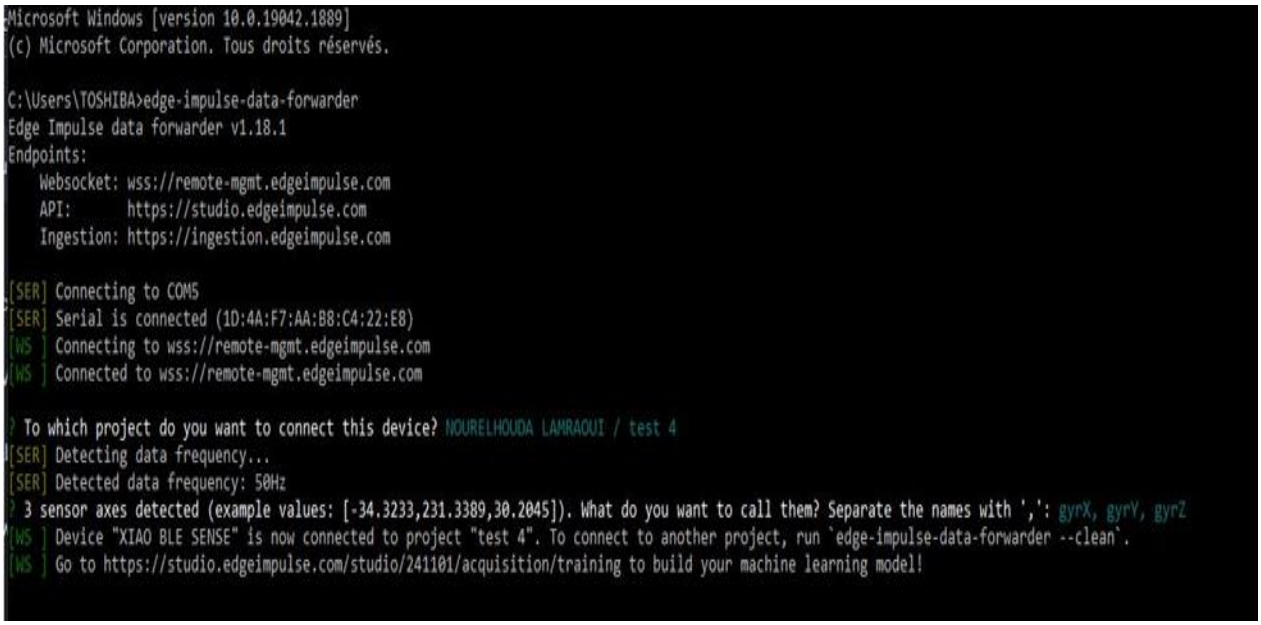


Figure4.19: choosing the project and and give the sensors name

Step 7. Select the sensor as "3 axes". Name your label as up and down, modify Sample length (ms.) To 30000 and click start sampling.

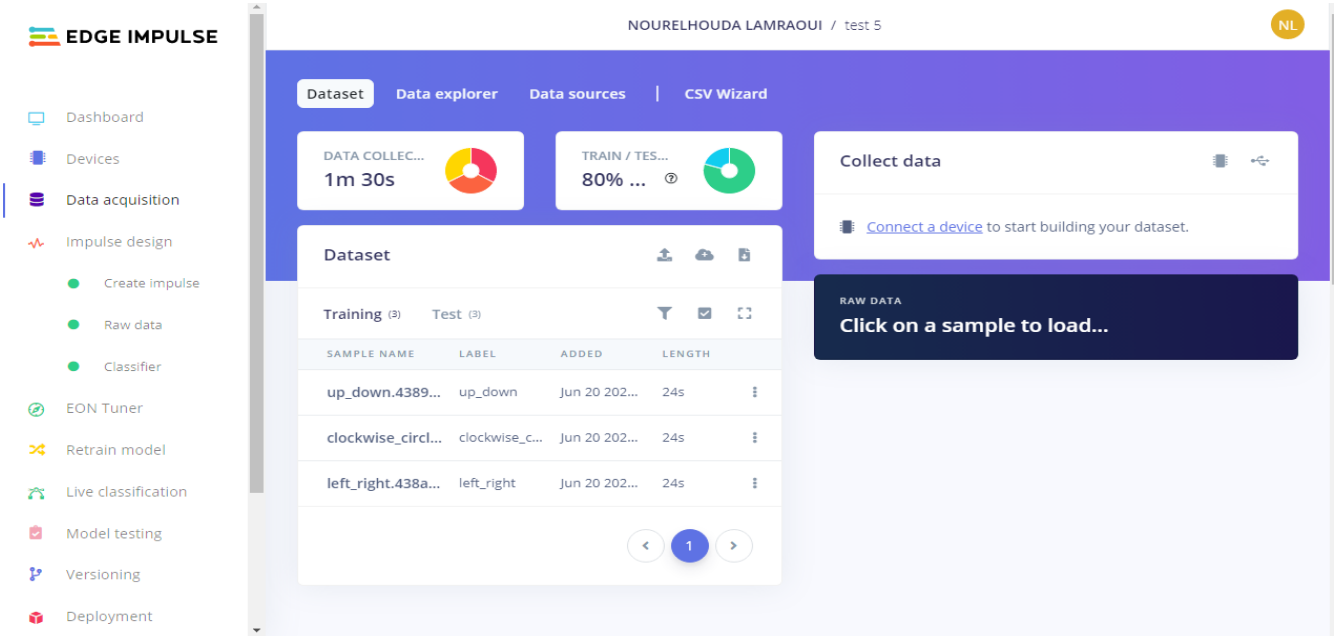


Figure 4.20: taking samples using XIAO BLE SENSE

Step 8. Swing the Seeed Studio XIAO nRF52840 Sense up and down and keep the motion for 30 seconds. You can find the acquisition is shown up like this:

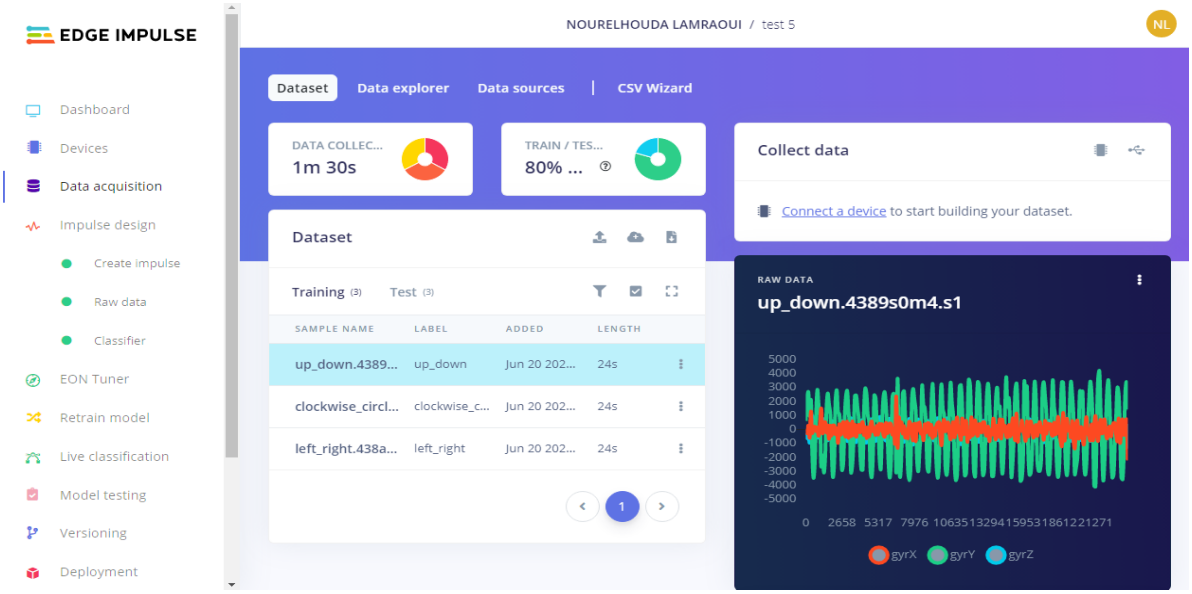


Figure 4.21: choose the sample to see the raw data

Step 9. Split the data by clicking the raw data right top and choose "Split Sample". Click +Add Segment and then click the graph. Repeat it more than 20 time to add segments. Click Split and you will see the the sample data each for 1 second. (24s training and 6s test)

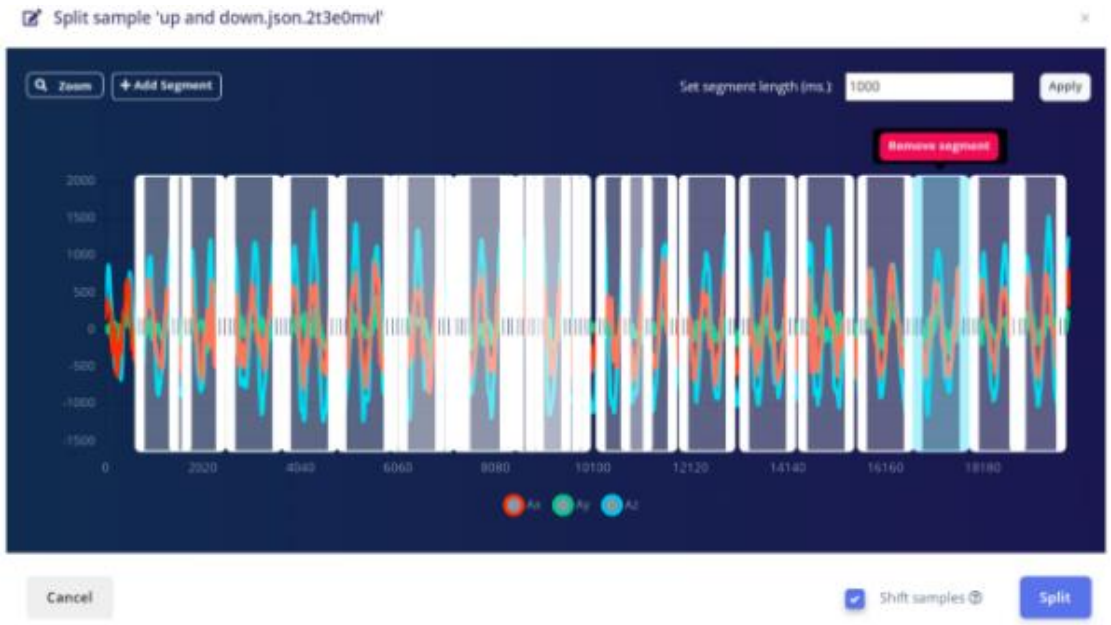


Figure 4.22: add segments and split the samples

Step 10. Repeat step 8 and step 9. Add new styles in a new project: left, idle, anticlockwise, and Wave.

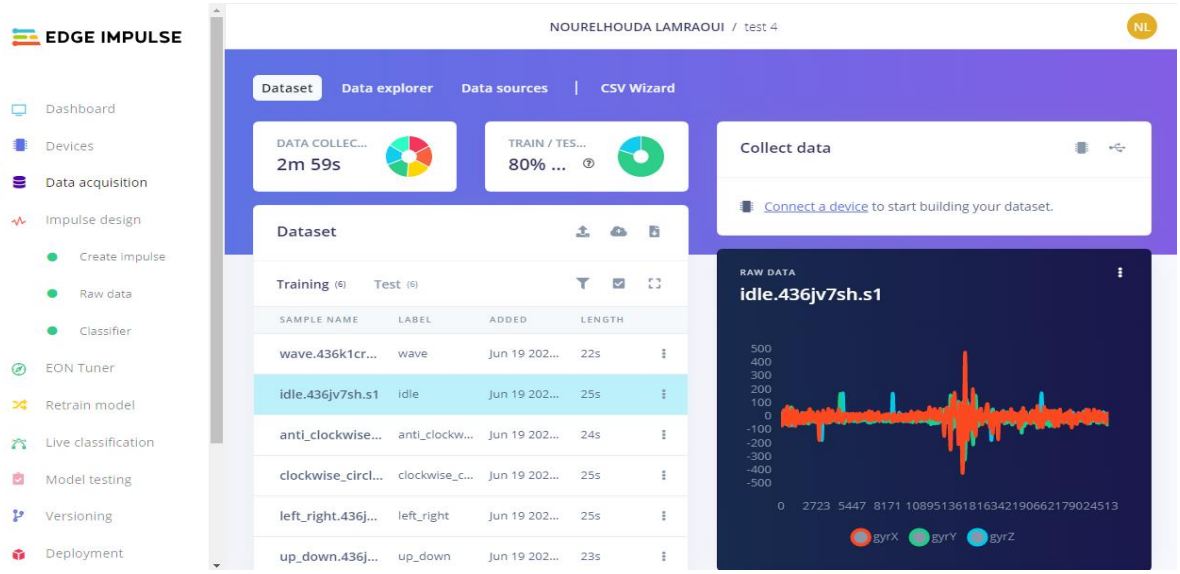


Figure 4.23: add more samples such as idle anticlockwise and wave

c. Create Impulse

Step 12: Click Create impulse -> Add a processing block -> Choose Spectral Analysis -> Add a learning block -> Choose Classification (Keras) -> Save Impulse

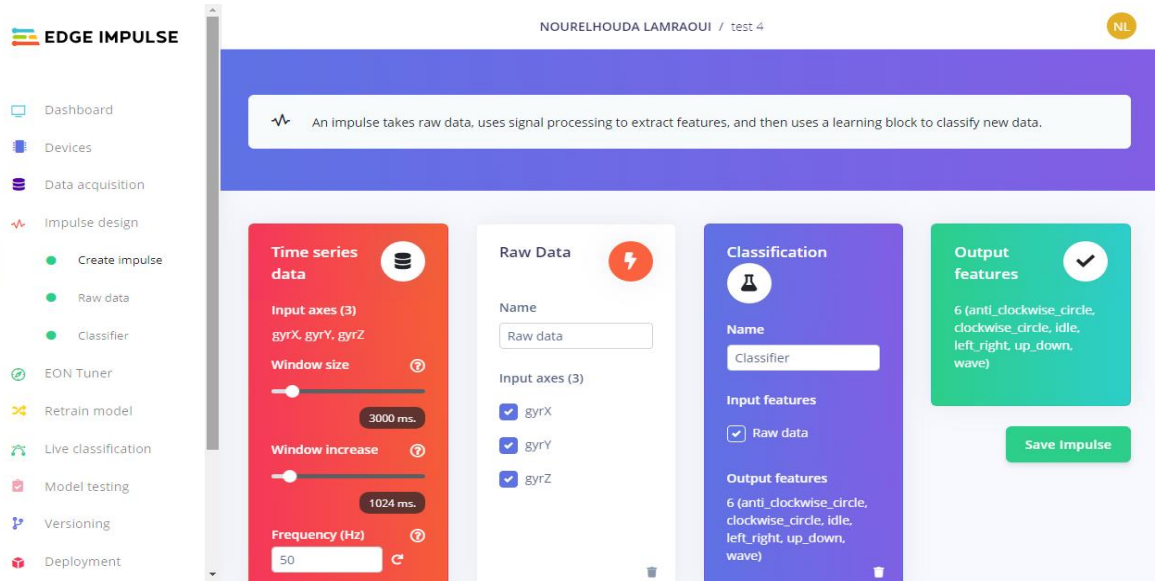


Figure 4.24: add processing & spectral analysis and learning blocks

d. Collected Sensor Data for Gesture Classification

In the first part of the project, we try to classify this types of gestures which are: UP_DOWN, LEFT_RIGHT and Clockwise_CIRCLE and in the second experiment we added three other gestures which are: Idle, wave Anti_Clockwise_CIRCLE .

First we begin by collecting 30 second for each pattern.

Label 1: Up and Down

We first start by collecting 30 seconds for the Up and down pattern, the following figure represent the plot of the measurements:

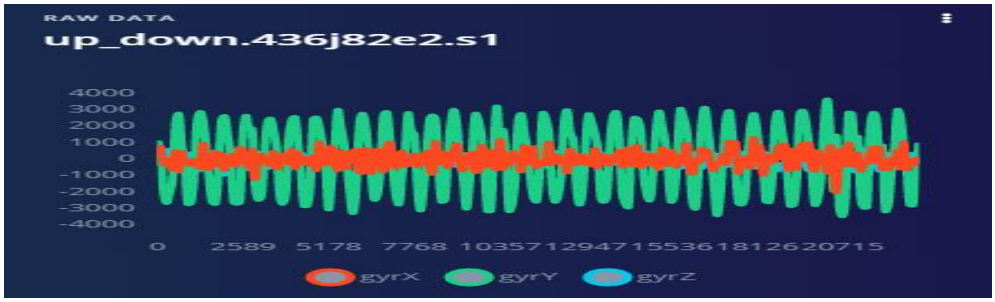


Figure 4.25: The Up and Down plot

Label 2: Clockwise_CIRCLE

We move to the second mode, which is circle in clockwise orientation, and we record 30 seconds:

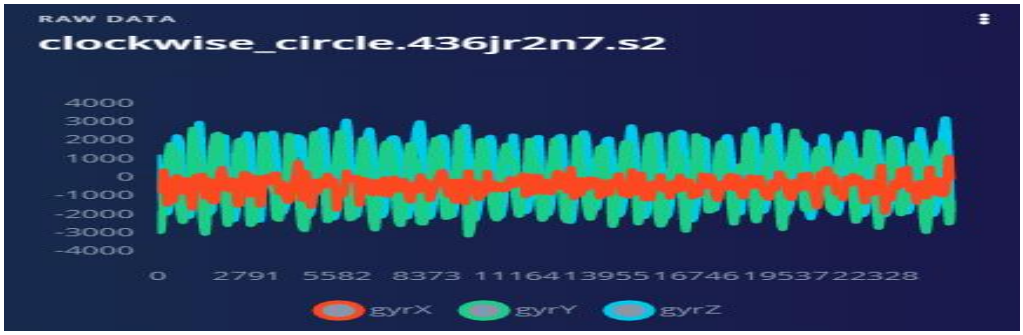


Figure 4.26: The Clockwise_CIRCLE plot

Label 3: Anti_Clockwise_CIRCLE

We move to the next mode, which is circle in anticlockwise orientation, and we record 30s

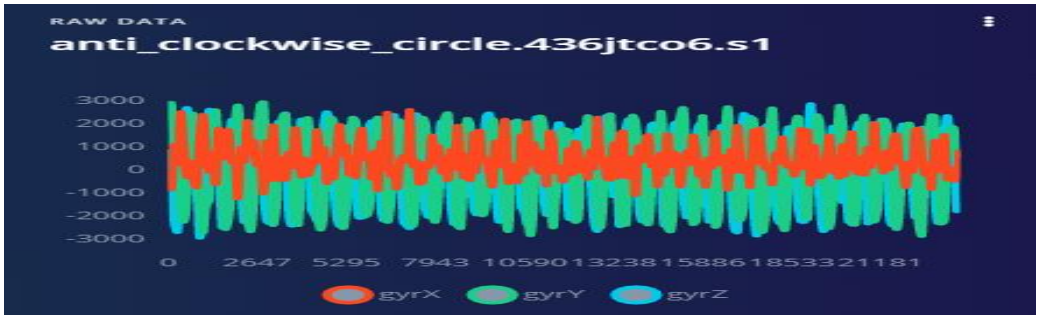


Figure 4.27: The Anti_Clockwise_CIRCLE plot

Label 4: Right and Left

We move to the next mode, which is right and left, in which we record 30 seconds

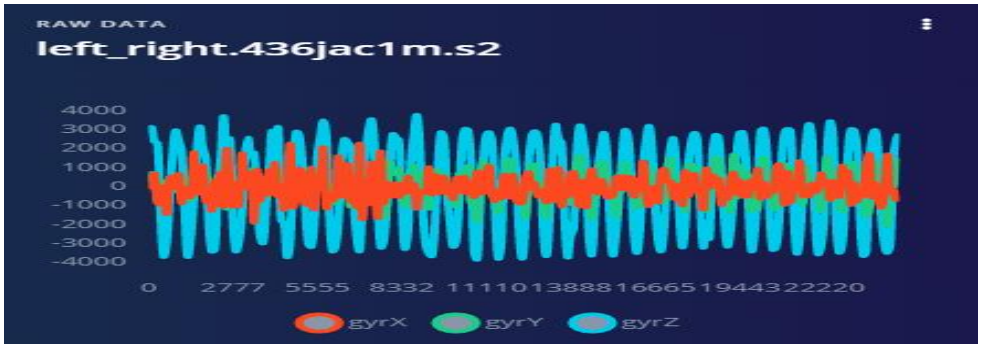


Figure 4.28: The Right and Left plot

Label 5: Idle

We move to the next mode, which is idle in which we record 30 seconds

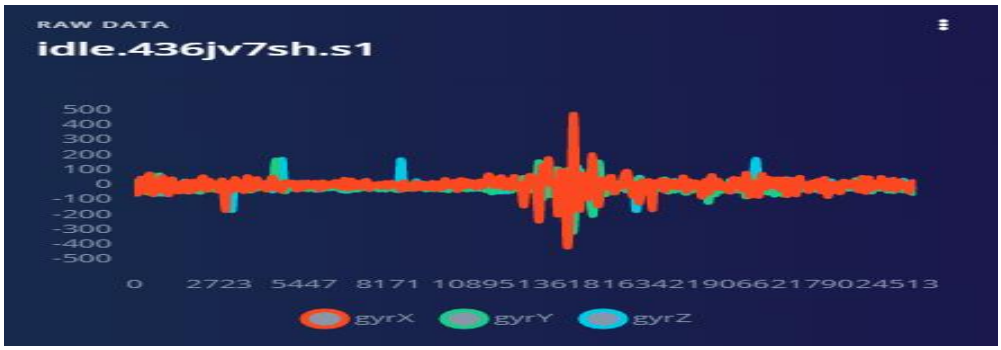


Figure 4.29: The Idle plot

Label 6: Wave

We move to the last mode, which is idle in which we record 30 seconds

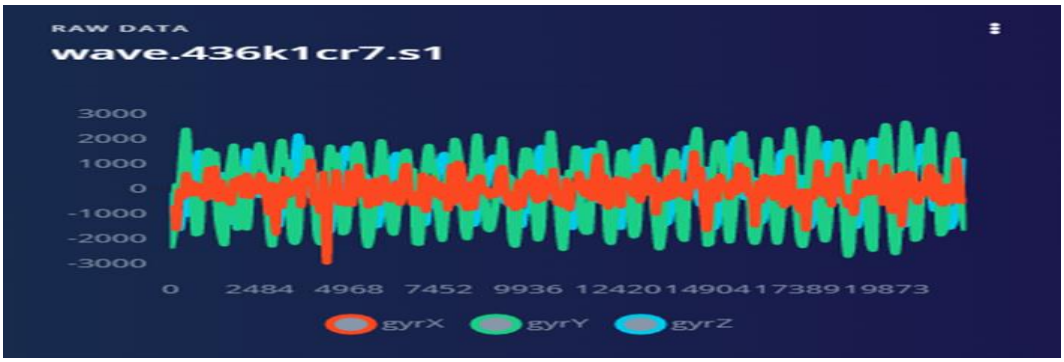


Figure 4.30: The Wave plot

e. Classifier your model

Classifier is a cutting-edge technology that uses machine learning algorithms to classify data at the edge, or on the device itself. This means that the device can make decisions without relying on a cloud server, which is particularly useful in scenarios where internet connectivity is limited or unreliable.

Unlike traditional machine learning models, Edge Impulse Classifier is designed to work with small datasets and limited computing resources. It uses a technique called transfer learning, which involves taking pre-trained models and fine-tuning them for specific use cases. This allows for faster training times and more accurate predictions.

Step 13: Click Classifier -> Click Start training -> Choose Unoptimized (float32)

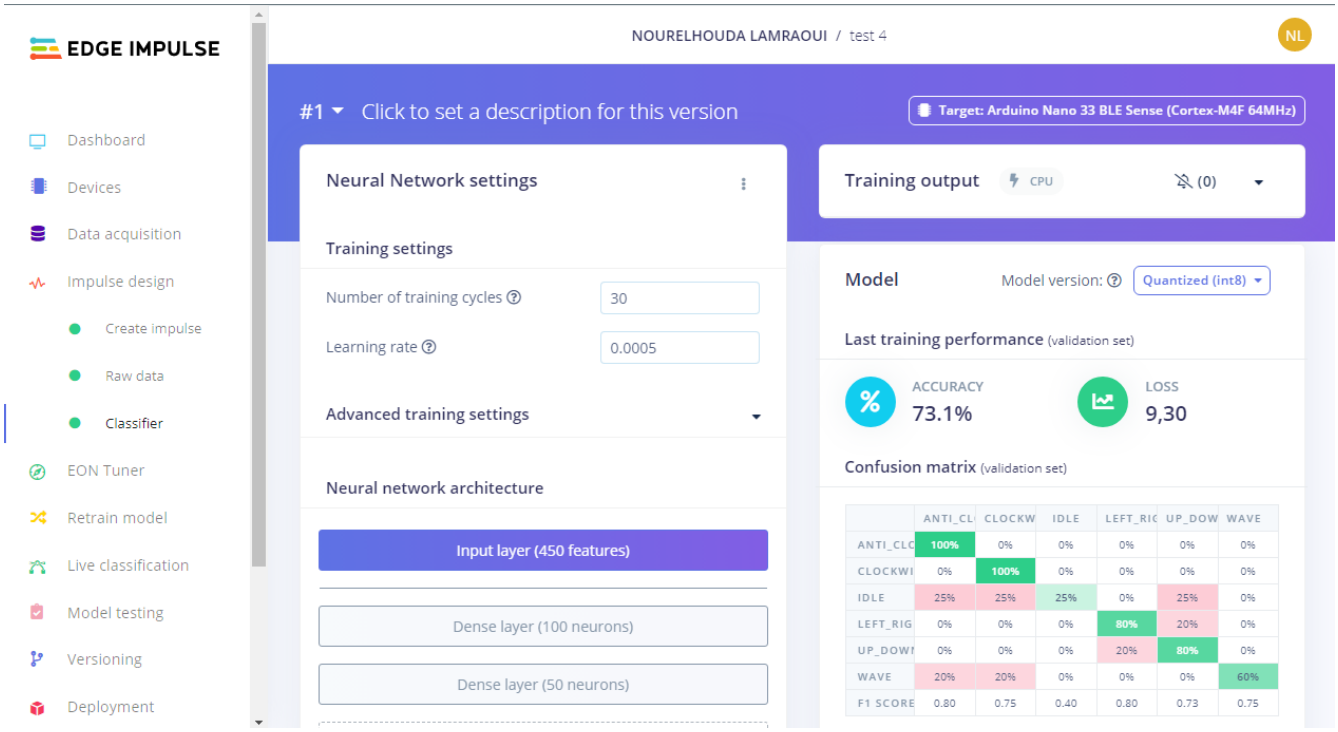


Fig4.31: start training the model

f. Data explorer

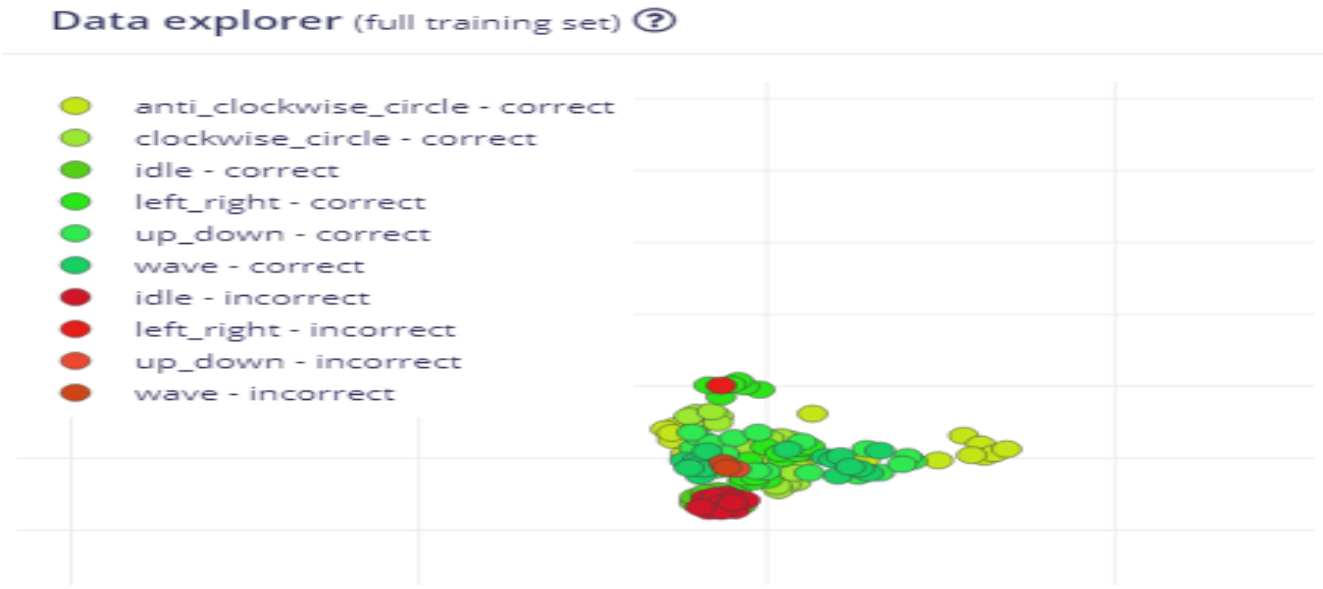


Figure 4.32: A result of Data explorer in the case of 6 classes

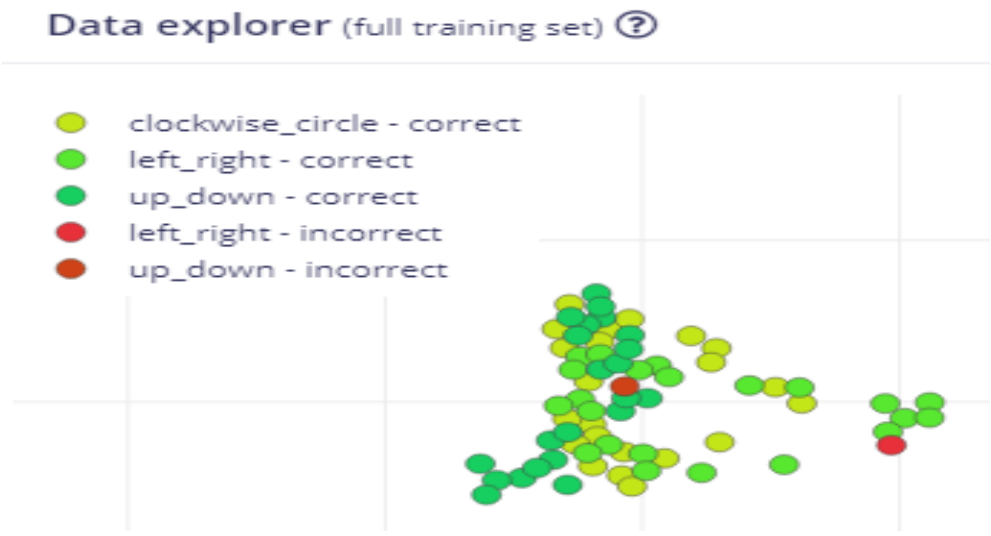


Figure 4.33: A result of Data explorer in the case of 3 classes

i. Choose Neural Network settings

This section is intended for choosing the desired NN type and architecture, either by using the graphical interface or introducing the python Keras code in the “switch to Keras (expert) mode”.

To add a specific layer in the graphical mode, the user can click on add extra layer and choose form the types available by default in edge impulse.

The figure below lists some of the available types:

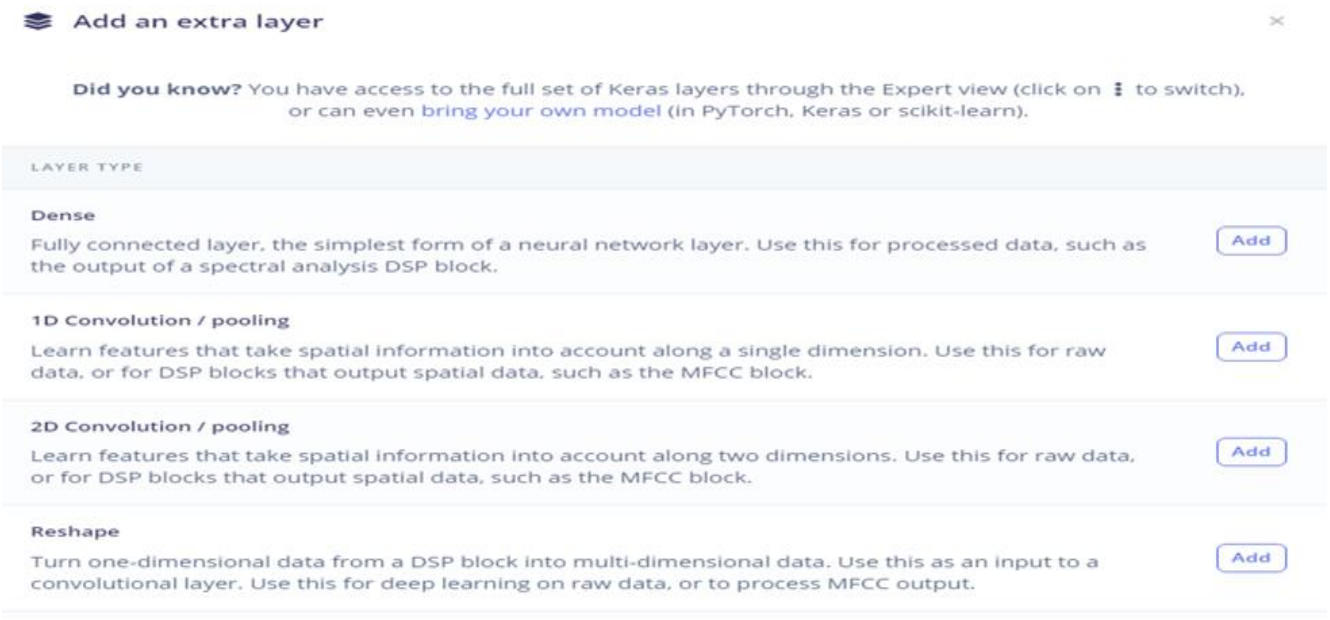


Figure 4.34: add an extra layer

The used NN architecture and its setting is shown in the figure below

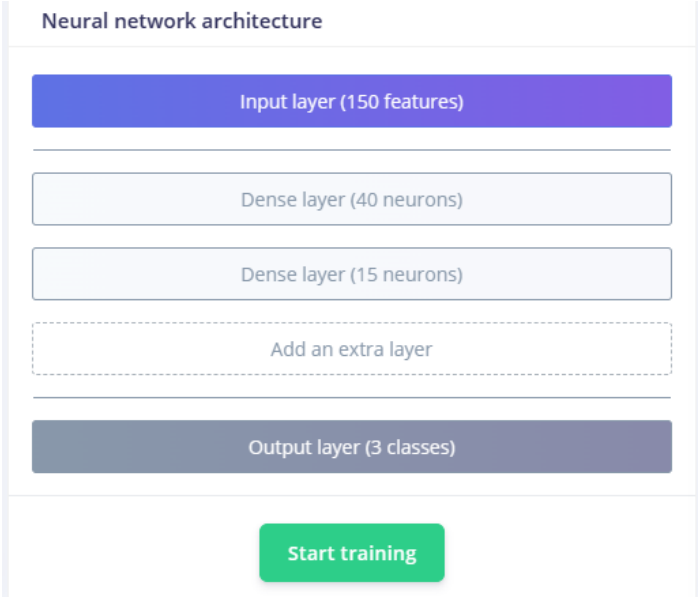


Figure 4.35: NN architecture for 3 classes of motion

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv1D, Conv2D, Flatten, Reshape, MaxPooling1D, MaxPooling2D, AveragePooling2D, BatchNormalization, TimeDistributed, Permute, ReLU, Softmax
from tensorflow.keras.optimizers import Adam
EPOCHS = args.epochs or 30
LEARNING_RATE = args.learning_rate or 0.0005
# this controls the batch size, or you can manipulate the tf.data.Dataset objects yourself
BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder=False)

# model architecture
model = Sequential()
model.add(Dense(40, activation='relu', activity_regularizer=tf.keras.regularizers.l1(0.00001)))
model.add(Dense(15, activation='relu', activity_regularizer=tf.keras.regularizers.l1(0.00001)))
model.add(Dense(classes, name='y_pred', activation='softmax'))

# this controls the learning rate
opt = Adam(learning_rate=LEARNING_RATE, beta_1=0.9, beta_2=0.999)
callbacks.append(BatchLoggerCallback(BATCH_SIZE, train_sample_count, epochs=EPOCHS))

# train the neural network
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
model.fit(train_dataset, epochs=EPOCHS, validation_data=validation_dataset, verbose=2, callbacks=callbacks)

# Use this flag to disable per-channel quantization for a model.
# This can reduce RAM usage for convolutional models, but may have
# an impact on accuracy.
disable_per_channel_quantization = False
```

Figure 4.36: Corresponding Keras code

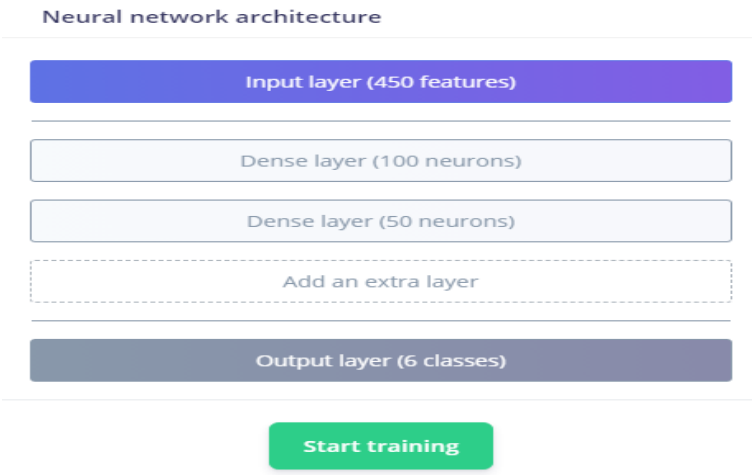
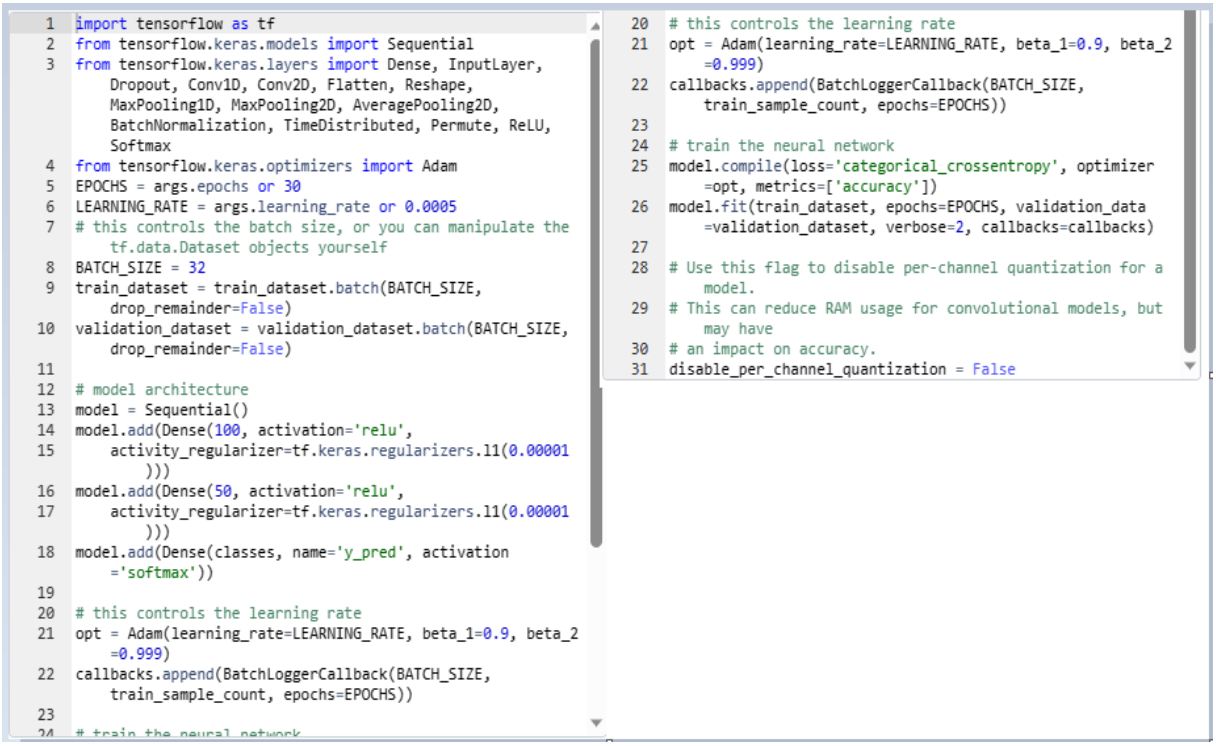


Figure 4.37: NN architecture for 6 classes of motion



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer,
    Dropout, Conv1D, Conv2D, Flatten, Reshape,
    MaxPooling1D, MaxPooling2D, AveragePooling2D,
    BatchNormalization, TimeDistributed, Permute, ReLU,
    Softmax
from tensorflow.keras.optimizers import Adam
EPOCHS = args.epochs or 30
LEARNING_RATE = args.learning_rate or 0.0005
# this controls the batch size, or you can manipulate the
    tf.data.Dataset objects yourself
BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE,
    drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE,
    drop_remainder=False)

# model architecture
model = Sequential()
model.add(Dense(100, activation='relu',
    activity_regularizer=tf.keras.regularizers.l1(0.00001
        )))
model.add(Dense(50, activation='relu',
    activity_regularizer=tf.keras.regularizers.l1(0.00001
        )))
model.add(Dense(classes, name='y_pred', activation
    ='softmax'))

# this controls the learning rate
opt = Adam(learning_rate=LEARNING_RATE, beta_1=0.9, beta_2
    =0.999)
callbacks.append(BatchLoggerCallback(BATCH_SIZE,
    train_sample_count, epochs=EPOCHS))

# train the neural network
model.compile(loss='categorical_crossentropy', optimizer
    =opt, metrics=['accuracy'])
model.fit(train_dataset, epochs=EPOCHS, validation_data
    =validation_dataset, verbose=2, callbacks=callbacks)

# Use this flag to disable per-channel quantization for a
    model.
# This can reduce RAM usage for convolutional models, but
    may have
# an impact on accuracy.
disable_per_channel_quantization = False
```

Figure 4.38: Corresponding Keras code

The different layer functions and ops are:

- **Activity regularizer:** Regularize to apply a penalty on the layer's output

The value returned by the activity regularize object gets divided by the input batch size so that the relative weighting between the weight regularizes and the activity regularizes does not change with the batch size.

- **Dropout:** is a technique to reduce over fitting when training the model.

The term “Dropout” refers to the removal of neurons in the layers of a Deep Learning model.

Its use is really simple on all libraries. With Keras & Tensor flow all you need to do is add a Dropout layer and indicate the desired dropout probability.

The probability of default is 0.5, here in takes 0.2: tf .keras .layers .Dropout (0.2) It is used as a neurosis sofa, it means that after (or before) each sofa you can add a Dropout which will deactivate certain neurons.

- **Dense implements the operation:** output = activation (dot (input, kernel) + bias) where

Activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use bias is True). These are all attributes of Dense.

j. Training performance

For activation functions, we used “ReLu” type for the fisrt two layers and “Softmax” in the last, while the Adam algorithm is used for training the NN.

Figure 4.39 : confusion matrix for 3 classes

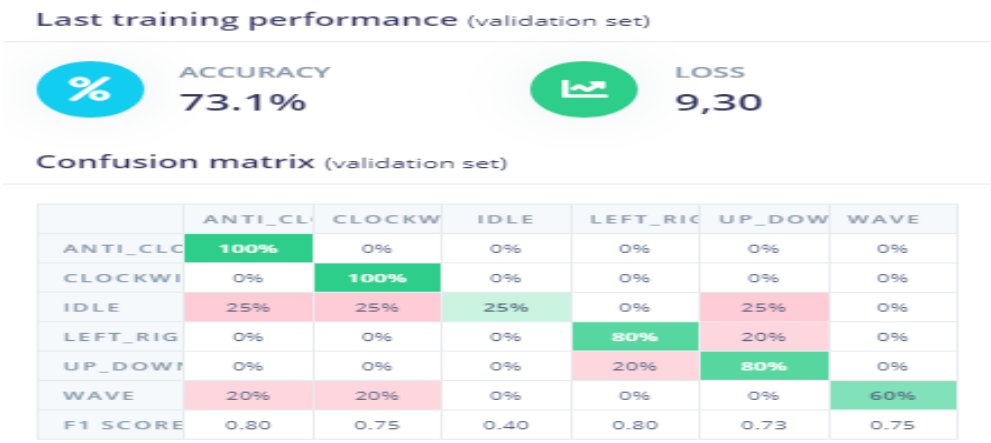


Figure 4.40: Confusion matrix for 6 classes

Discussion

we noticed that when we put more classes, we increase the loss and decrease the accuracy , which means that the model for a higher number of classes needs more tunning.

k. Target performance and footprint

The Target performance and footprint is shown in the figure below

Figure 4.41: target performance and memory footprint for 3 classes

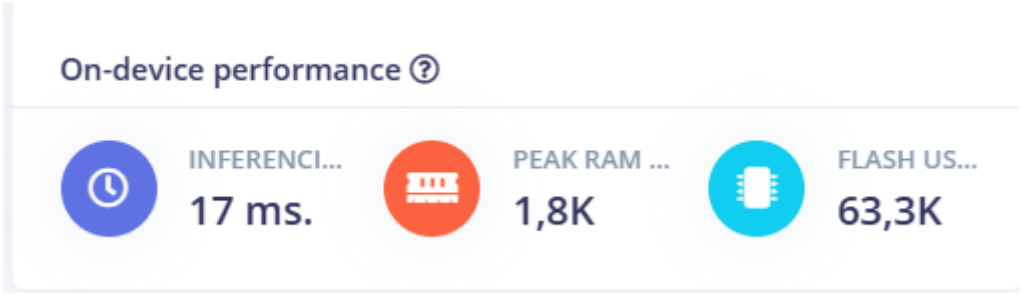


Figure 4.42: target performance and memory footprint for 6 classes

Discussion

we noticed that when we work with more classes, inference time increases and the case is for the memory footprint of the MCU.

l. Deployment

The Deployment page consists of a variety of deployment options to choose from depending on your target device. Regardless of whether you are using a fully supported development board or not, you can deploy your impulse to any device.

From the Deployment page, select the Search deployment options search box to select and configure a deployment option:

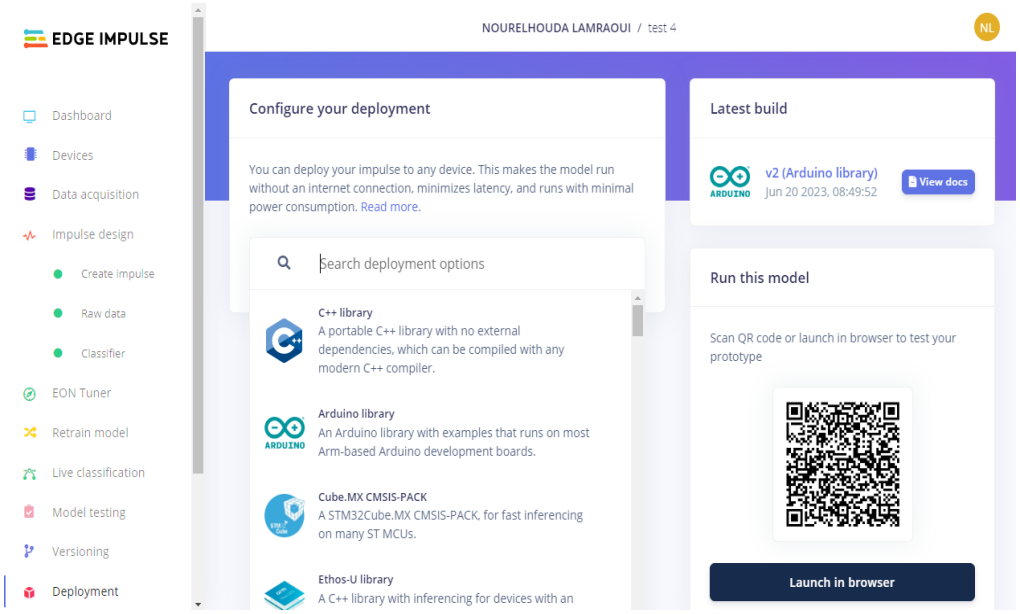


Figure 4.43: Configuration of deployment platform target

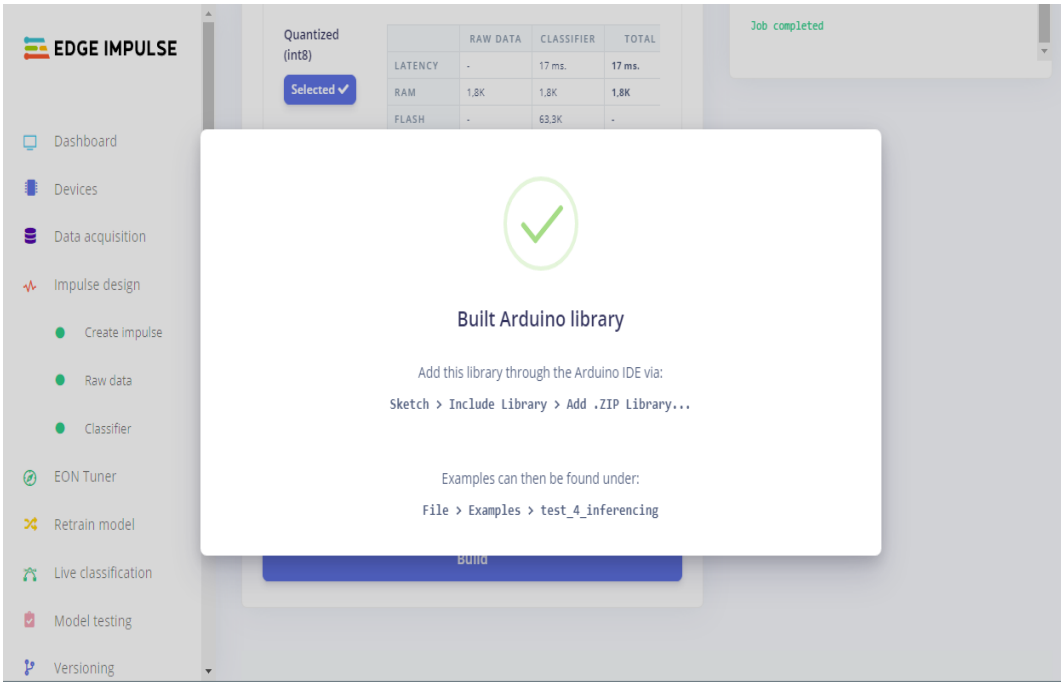


Figure 4.44: Arduino platform files generated

In our case we selected Arduino IDE as target platform and Arduino NANO BLE Sense as device target, on which we modified the main source file to work with XIAO BLE sense board and its IMU unit (LSM6DS3) which is different from LSM9DS1.

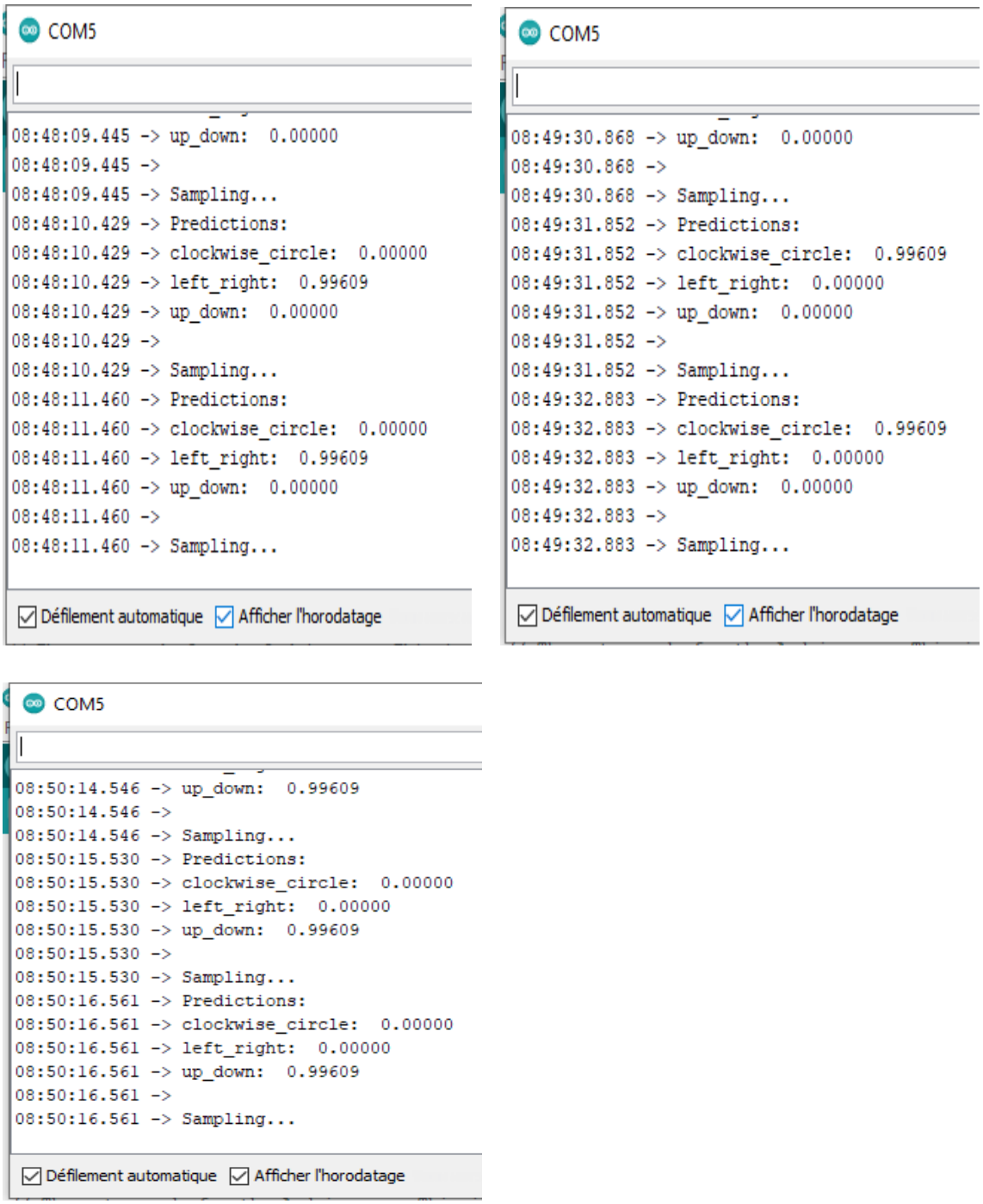


Fig4.45: Serial monitor inference results for gesture recognition: 03 classes

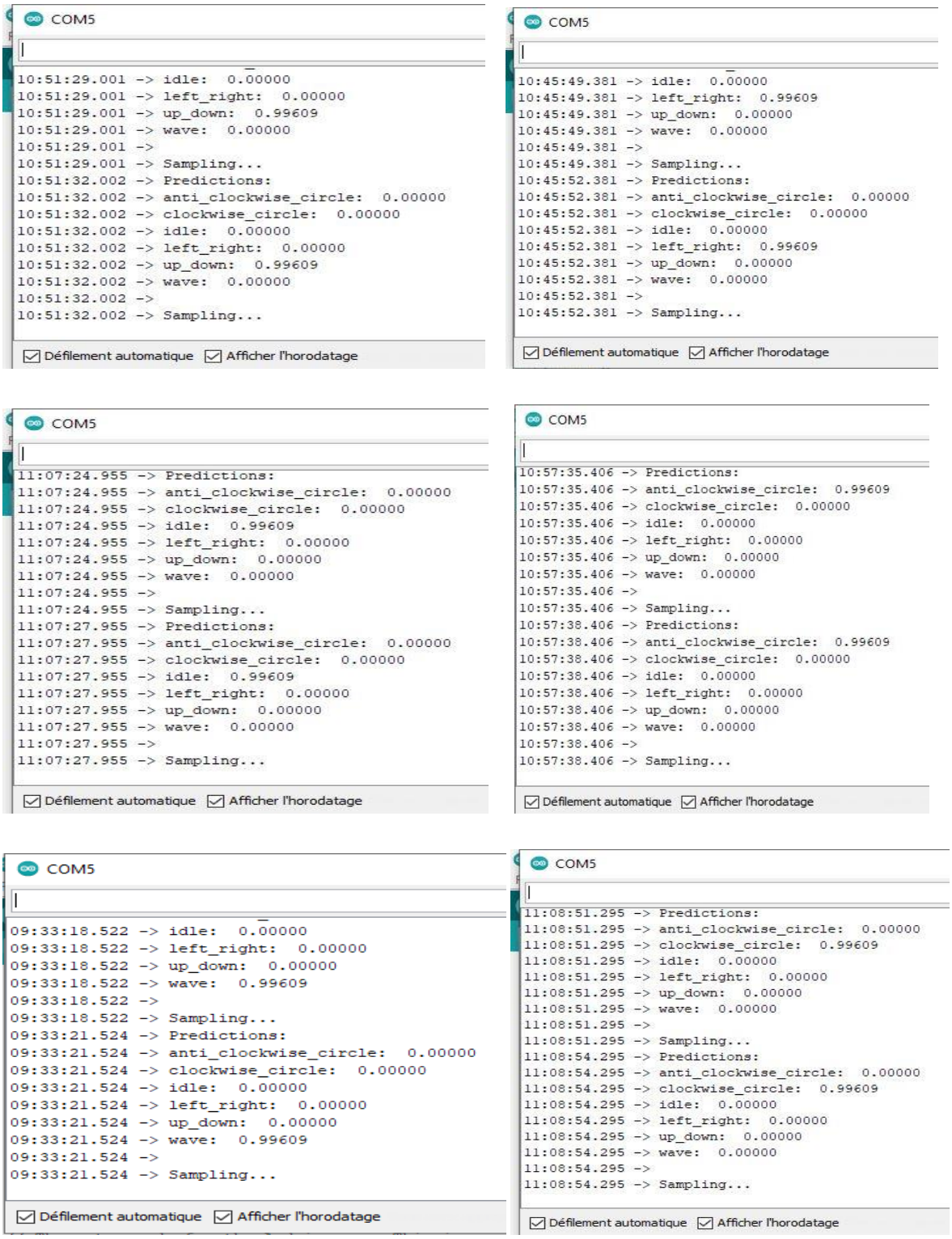


Figure 4.46: Serial monitor inference results for gesture recognition: 06 classes

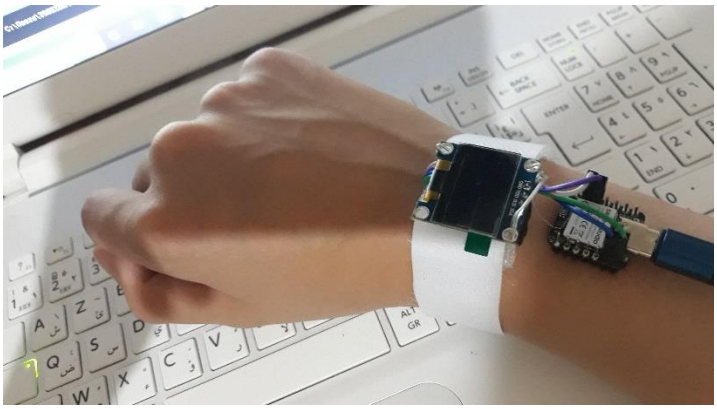


Figure 4.46: Real wearable application for gesture recognition

IV.4 Conclusion

Through this chapter, we presented gesture recognition application based on XIAO BLE SENSE board and TinyML/ Edge Impulse application. We concluded that the recognition presents better rates in case of fewer number of classes; the same thing can be noticed in terms of accuracy , inference time and memory footprint. So, for more classes, the NN model might be tuned more to enhance the results.

XIAO BLE SENSE board, and Edge Impulse's TinyML, utilizing the IMU, offer a powerful wearable solution for gesture recognition applications. The XIAO BLE SENSE board's compact design and Bluetooth Low Energy connectivity make it suitable for various devices. With Edge Impulse, developers can easily collect IMU data, label gestures, and train ML models. TinyML's lightweight algorithms enable real-time inferencing on the XIAO BLE SENSE board, delivering low-latency, low-power gesture recognition. This combination empowers developers to create accurate and efficient gesture recognition applications using IMU data.

CONCLUSION

General conclusion

Our project on gesture recognition using XIAO BLE SENSE board integrating the Inertial Measurement Unit (IMU) sensor, Arduino IDE and the Edge Impulse platform which employes several aspects and advanced technologies like neural networks (NN), machine learning (ML), TensorFlowLite, and Keras, has been a remarkable endeavor. Through meticulous data preprocessing, training and fine-tuning NN models, and optimizing for resource-constrained device through the EdgeImpulse plateform, we have successfully developed a robust and intuitive gesture recognition system.

The Edge Impulse platform has provided a comprehensive framework for model development and evaluation, while the integration of the XIAO BLE SENSE board on Arduino IDE has enabled seamless interaction, were we faced difficulties to make the on board IMU unit working with the generated model files on the Arduino IDE.

This project not only provides hands-on experience, and has not only enhanced our practical skills in implementing ML concepts but also deepened our understanding of gesture recognition techniques and the potential of TensorFlow and Keras frameworks, by showcasing the capabilities of NN and ML in real-world applications. This project sets the stage for future innovations and advancements in the exciting domain of gesture recognition, paving the way for more intuitive and engaging human-computer interactions.

As a future work, we aim to enhance the application for more gesture types and integrates the model in a complete wearable device application for motion sensing or controlling by gestures.

References

[1] Jake FRANKENFIELD Reviewed by GORDON SCOTT Artificial Intelligence: What It Is and How It Is Used (investopedia.com) ,April 24, 2023

[2] What is machine learning? (sap.com)

[3] S.Michelle, "Neural networks", course support, 2001.

[4] Kadous Djamila, Use of neural networks as a datamining tool: Generation of behavioral model of a physical process from data, University of Tlemcen, 2012.

[5] Miloudi.Lalia"Heuristic methods applied to the optimization of the orientation of a photovoltaic solar panel", thesis presented for the obtaining of the Doctoral degree in university of M'Hamed Bougara -Boumerdes, 2017.

[6] S. Faysal et B. Abderrahmane ,"Recognition of handwritten figures by SVMs" Bachelor's thesis, Universityof Ahmed Draïa D'Adrar, 2015.

[7] A. Rebiai, "A hybrid approach to the recognition of Arabic script Manuscrite", Mémoire de magister. University of Constantine, 2007.

[8] TRAHI.Fatiha Memory Magister scratching neuron reaseau estimation

[9] Dan, Single Layer Perceptron Explained, ML Corner, October 13, 2020.

[10] Pete Warden, Daniel Situnayake - TinyML_ Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers (2019, O'Reilly Media) - libgen.lc

[11] By Cabe Atwell: https://www.fierceelectronics.com/electronics/what-edge-

machine-learning Nov 24, 2020

[12] Cem Dilmegani, 3. MINUTE READ SEPTEMBER 27, 2021

[13] Article on the site : https://www.projectpro.io/article/machine- learning-frameworks/509

[14] Journal of King Saud University - Computer and Information Sciences ,Volume 34, Issue 4, April 2022

[15] Antoine Tardif,:https://www.unite.ai/tinyml-the-future-of-machine-learning-on-a-minuscule-scale/ April 12, 2023

[16] BroutonLab, Data Science, in 2017.

[17] Rohit Sharma, , Title Book :Introduction to Tiny ML. Published on: July 09, 2022

[18] Supriti Tripathy, Article on the site :https://www.opengrowth.com/resources/tinyml-fundamentals-applications-deployment , 25th July 2021

[19] Steven F. Barrett Arduino V: Machine Learning, Department of Electrical and Computer Engineering University of Wyoming Laramie, WY, USA, January 2023

Abstract

This project aims to develop a gesture recognition application for 06 classes and more using TinyML technics and a tiny board for a wearable application. Gesture recognition has become a captivating field of study and application, driven by advancements in technologies like edge impulse platform , devices such as Arduino and the XIAO BLE SENSE board. By integrating sensors like IMU (Inertial Measurement Unit) with microcontrollers (MCUs), complex motions can be captured and analyzed. The combination of machine learning and microcontrollers, known as TinyML, has enabled the deployment of neural networks (NNs) directly on these small devices. Popular frameworks like TensorFlow, TensorFlow Lite, and Keras empower developers to train and deploy efficient models for gesture recognition, revolutionizing the way we interact with machines. With the power of these tools, accurate and real-time recognition of gestures can now be achieved, opening up exciting possibilities in various domains such as gaming, robotics, and human-computer interaction.

Keywords : gesture recognition, edge impulse platform, Arduino, XIAO BLE SENSE board, IMU NN, AI, TinyML, TensorFlow, keras .

ملخص

يهدف هذا المشروع إلى تطوير تطبيق التعرف على الإيماءات لـ 06 فصول وأكثر باستخدام تقنيات TinyML ولوحة صغيرة لتطبيق يمكن ارتداؤه. أصبح التعرف على الإيماءات مجالًا آسرًا للدراسة والتطبيق، مدفوعًا بالتقدم في تقنيات مثل منصة الاندفاع الحافة وأجهزة مثل Arduino ولوحة XIAO BLE SENSE. من خلال دمج أجهزة الاستشعار مثل IMU (وحدة القياس بالقصور الذاتي) مع أجهزة التحكم الدقيقة (MCUs)، يمكن التقاط الحركات المعقدة وتحليلها. مكّن الجمع بين التعلم الآلي والسيطرات الدقيقة، المعروفة باسم TinyML، من نشر الشبكات العصبية (NNs) مباشرة على هذه الأجهزة الصغيرة. تعمل الأطر الشائعة مثل TensorFlow و TensorFlow Lite و Keras على تمكين المطورين من تدريب ونشر نماذج فعالة للتعرف على الإيماءات، وإحداث ثورة في طريقة تفاعلنا مع الآلات. مع قوة هذه الأدوات، يمكن الآن التعرف الدقيق في الوقت الفعلي على الإيماءات، مما يفتح إمكانيات مثيرة في مجالات مختلفة مثل الألعاب والروبوتات والتفاعل بين الإنسان والحاسوب.

Résumé

Ce projet vise à développer une application de reconnaissance gestuelle pour 06 classes et plus en utilisant des techniques TinyML et une petite planche pour une application portable. La reconnaissance gestuelle est devenue un domaine d'étude et d'application captivant, alimenté par les avancées technologiques telles que la plateforme d'impulsions de bord, les appareils tels qu'Arduino et la carte XIAO BLE SENSE. En intégrant des capteurs comme IMU (Inertial Measurement Unit) avec des microcontrôleurs (MCUs), les mouvements complexes peuvent être capturés et analysés. La combinaison de l'apprentissage automatique et des microcontrôleurs, connus sous le nom de TinyML, a permis le déploiement de réseaux neuronaux (NNs) directement sur ces petits appareils. Des cadres populaires comme TensorFlow, TensorFlow Lite et Keras permettent aux développeurs de former et de déployer des modèles efficaces pour la reconnaissance gestuelle, révolutionnant la façon dont nous interagissons avec les machines. Avec la puissance de ces outils, la reconnaissance précise et en temps réel des gestes peut maintenant être réalisée, ouvrant des possibilités passionnantes dans divers domaines tels que le jeu, la robotique et l'interaction homme-ordinateur.

Mots-clés : reconnaissance gestuelle, plate-forme d'impulsion, Arduino, carte XIAO BLE SENSE, IMU NN, AI, TinyML, TensorFlow, keras .