Kufa Journal of Engineering
Vol. 15, No. 3, July 2024, P.P. 148-169
Article history: Received 12 January 2024, last revised 28 March 2024, Accepted 10 May 2024

OBJECT DETECTION ALGORITHMS IMPLEMENTATION ON EMBEDDED DEVICES: CHALLENGES AND SUGGESTED SOLUTIONS

Ruqaya Alaa[1], Ehab A. Hussein[2] and Hilal Al-libawy[3]

[1] Master's degree student, Department of Electrical Engineering, College of Engineering, University of Babylon, Republic of Iraq. Email: eng771.ruqaya.alaa@student.uobabylon.edu.iq

[2] prof., Department of Electrical Engineering, College of Engineering, University of Babylon, Republic of Iraq. Email: dr.ehab@uobabylon.edu.iq

[3] Asst. prof., Department of Electrical Engineering, College of Engineering, University of Babylon, Republic of Iraq. Email: Eng.hilal_al-libawy@uobabylon.edu.iq

https://doi.org/10.30572/2018/KJE/150309

ABSTRACT

Object detection and image classification are among the most important areas to which scientific research is directed, which are commonly used in various applications based on computer vision. The development of low-cost embedded devices with powerful processing is leading to a trend in their use in computer vision, which provides reduced access time, reliability, and data security. That's why Tiny Machine Learning (TinyML) technology appeared, which is field that specifically explores the application of machine learning on highly limited edge devices. Deep learning techniques are increasingly employed in data-intensive and time-sensitive IoT applications. Deploying Deep Neural Network (DNN) models on microcontrollers (MCUs) is challenging due to limited resources such as RAM. Recent advancements in the field of TinyML hold the potential to introduce a novel category of peripheral applications. TinyML enables the development of new applications and services by eliminating the reliance on cloud computing, which consumes power and poses risks to data security and privacy. TinyML currently regarded as a promising artificial intelligence (AI(alternative that specifically targets technologies and applications for devices with very low profiles. In this paper, the most important algorithms used in object detection will be presented, in addition to the challenges in using low-resource embedded systems that support TinyML technology.

KEYWORDS

Tiny Machine Learning; Deep learning; Artificial intelligence; Embedded devices; TensorFlow Lite; MicroPython.

1. INTRODUCTION

Object detection is an advanced version of image classification that goes beyond simply classifying elements in a picture. This task requires properly determining the precise placements of these objects by outlining bounding boxes around them. Object detection is essential for robots to comprehend and engage with the visual environment surrounding them. An example of a object detection as shown in the Fig .1, the process of detecting the bus using the boundary box. A set of training cases is used by object detection systems to build a model for a class of objects. However, it might be difficult to discern models that accurately depict the wide array of differences observed in photographs. An alternative approach is to utilise convolutional neural networks (CNNs), which have the ability to gain understanding of the many categories by analysing large datasets.

In recent times as explained by the author (Murthy et al., 2020), The discipline of deep learning (DL) is an important topic that is receiving a lot of attention in the scientific community, as well as in the business world and the academic world. Unlike traditional and machine learning methods, deep learning approaches have the capacity to efficiently manage enormous volumes of unstructured data and discover complicated patterns within extensive data sets. This is in contrast to the capabilities of traditional methods. In a short amount of time, it is evolving into a tool that can simulate and solve complex and difficult problems in a variety of scientific and technological fields.

Considering the numerous technological developments that have been made, such as the Internet of Things (IoT) and edge computing, it is desirable to include machine learning techniques into embedded devices that have limited resources in order to achieve dispersed and ubiquitous intelligence. This has been the impetus behind the development of the TinyML paradigm, which is an embedded machine learning technique that enables machine learning applications to run on numerous inexpensive devices that are limited in terms of resources and power. With that being said, during this transition towards the suitable application of the TinyML technology, there are various issues that demand prompt answers. These challenges include the optimisation of processing capacity, the improvement of dependability, and the maintenance of the accuracy of learning models.

The TinyML paradigm advocates for the utilisation of microcontroller units (MCUs) to energise miniature objects through machine learning (ML) based procedures. The challenges encountered by TinyML users are substantial: As an illustration, the quantity of components required in contemporary neural networks, which are currently among the most advanced

technologies, has surged to the magnitude of billions. In many different contexts, larger networks are more efficient and versatile than smaller ones. It is unfortunate that the amount of energy that these networks consume rises in direct proportion to the size of the network. As a consequence of this, the process of expanding neural networks cannot be maintained on a large scale. The study of TinyML is not only useful but also important, and this is yet another persuasive argument in support of this position.

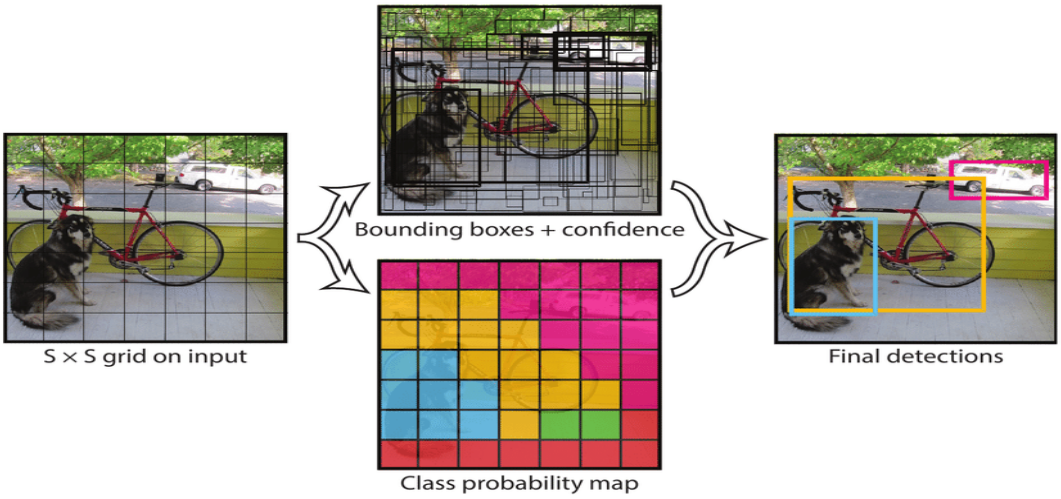


Fig. 1. Object Detection

2. OVERVIEW OF OBJECT DETECTION ALGORITHMS

Tiny machine learning (ML) systems necessitate the use of novel techniques that rely on extremely compact models to prevent excessive memory usage. In summary, TinyML systems need to efficiently optimize machine learning algorithms by utilizing a compact software design while working with high-quality data. Next, the data needs to be sent using binary files that were created by the models trained on a much larger computer.

Object detectors can generally be classified into two primary categories: two-stage object detectors and single-stage object detectors. Two-stage detectors primarily prioritise selective area proposal strategies through intricate architectures, while single-stage detectors prioritise all spatial region proposals for potential object identification using comparatively simpler architectures in a single iteration. The evaluation of an object detector is based on its detection accuracy and inference time. In general, two stage detectors provide higher detection accuracy compared to single stage object detectors. Nevertheless, single stage detectors have superior inference time in comparison to their counterparts (Du, Lixuan,2020).

2.1. Two-Stage Methods:

2.1.1. Region-Based Convolutional Neural Network (R-CNN), Fast R-CNN, and Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network), Fast R-CNN, and Faster R-CNN are all object detection models that have been developed to improve the accuracy and efficiency of detecting objects within an image.

1. R-CNN: R-CNN was the first model in this series and introduced the concept of region proposals. It works by generating a set of potential object regions within an image using selective search algorithm. These regions are then individually classified using a convolutional neural network (CNN). However, R-CNN is computationally expensive as it requires running the CNN for each region proposal separately (Bharati, Puja, Ankita Pramanik, 2020).

2. Fast R-CNN: Through the implementation of a shared CNN backbone, Fast R-CNN was able to achieve an improvement in the computational efficiency of R-CNN. On the other hand, Fast R-CNN takes the full image as input and extracts features by utilising a shared CNN network. This is in contrast to the traditional method of executing the CNN for each region proposal. After that, these characteristics are utilised to categorise every region proposal and to refine the bounding box coordinates of each region. When compared to R-CNN, this method greatly accelerates the course of both the training and inference processes (Rani, Shilpa, 2022).

3. Faster R-CNN: Faster R-CNN The implementation of a Region Proposal Network (RPN) resulted in an increase in both the speed and accuracy of item detection. Obtaining feature maps from the shared CNN backbone network is the responsibility of the RPN, which is responsible for creating region suggestions directly from those maps. Because of this, there is no longer a requirement for internal algorithms such as selective search, which were utilised in earlier models as the author explains (Abbas, Syed Mazhar,2018). After that, the ideas that were generated are categorized and refined in a manner that is comparable to Fast R-CNN. Faster R-CNN is able to achieve even faster inference times while keeping a high level of accuracy: this is accomplished by incorporating the region proposal step into the model itself

2.2. One-Stage Methods:

2.2.1. You Only Look once (YOLO)

The technology that is being presented is a real-time object detecting system that is at the cutting edge of technical advancement and provides exceptional levels of both speed and accuracy. In contrast to earlier methods, such as the R-CNN approach, which required hundreds of

evaluations for a single image, the YOLO (You Only Look Once) algorithm just does a single evaluation of an image. There is a basic factor that leads to the incredible speed of a YOLO model, which is more than one thousand times faster than RCNN models and one hundred times faster than Fast R-CNN models as stated by the author (Lin, Szu-Yin, Hao-Yu Li,2021) . This work reveals the essential mechanism that contributes to this speed . Employing a pre-defined set of bounding box, which are entrusted with the responsibility of identifying items that are located within their respective regions, is the way that is utilized in this approach. YOLO is superior to other object identification algorithms and training method in terms of its computing and processing speed, particularly when it comes to real time processing. YOLO not only has a high computing speed, but it also achieves a high level of accuracy while simultaneously minimizing the background errors that are normally present in competing methods. This is a particularly impressive feature of YOLO. For a relatively short period of time, the architecture of YOLO make it possible for the model to accurately learn and interpret a wide range of different things.

Table 1. Key features of each version of YOLO (Qureshi, Rizwan, et al,2023)

Version	**Date**	**Anchor**	**Framework**	**Backbone**	**AP** (%)
YOLO	2015	X	Darknet	Darknet24	63.4
YOLOv2	2016	X	Darknet	Darknet24	63.4
YOLOv3	2018	X	Darknet	Darknet53	36.2
YOLOv4	2020	X	Darknet	CSPDarknet53	43.5
YOLOv5	2020	X	Pytorch	Modified CSP v7	55.8
PP-YOLO	2020	X	PaddlePaddle	ResNet50-vd	45.9
Scaled-YOLOv4	2021	X	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	X	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	X	Pytorch	CSPDarknet	55.4
YOLOX	2021	X	Pytorch	Modified CSP v5	51.2
PP-YOLOE	2022	X	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	X	Pytorch	EfficientRep	52.5
YOLOv7	2022	X	Pytorch	RepConvN	56.8
DAMO-YOLO	2022	X	Pytorch	MAE-NAS	50.0
YOLOv8	2023	X	Pytorch	YOLO v8	53.9
YOLO-NAS	2023	X	Pytorch	YOLO-NAS	52.2

2.2.2. Single Shot Detection – SSD

An object detection technique known as the SSD algorithm is a method that employs a single neural network to produce predictions about bounding boxes and class probabilities directly from the entirety of an image. Foundational to the architecture of the SSD is a base networks, such as VGG or ResNet as shown in Fig.2. Subsequently, an extensive quantity of convolutional

layers are implemented to generate predictions for class scores and bounding box spanning a broad spectrum of classes and ratings. It is extraordinary reputation has been established on account of its capability to process photographs in real time with a high degree of accuracy.

Furthermore, its capability to detect objects at an extensive range of dimensions renders it well-suited for an extensive variety of applications, such as robotics, surveillance, and autonomous driving, among others. Both SSD and YOLO are developing neural network-based systems with the shared objective of real-time object localization. This objective is being pursued by both of these systems. Additionally, SSD accomplishes this by predicting bounding boxes at numerous dimensions throughout the image, whereas YOLO accomplishes this by partitioning the input image into a grid, as explained in (Zhang, Xi, et al,2023).

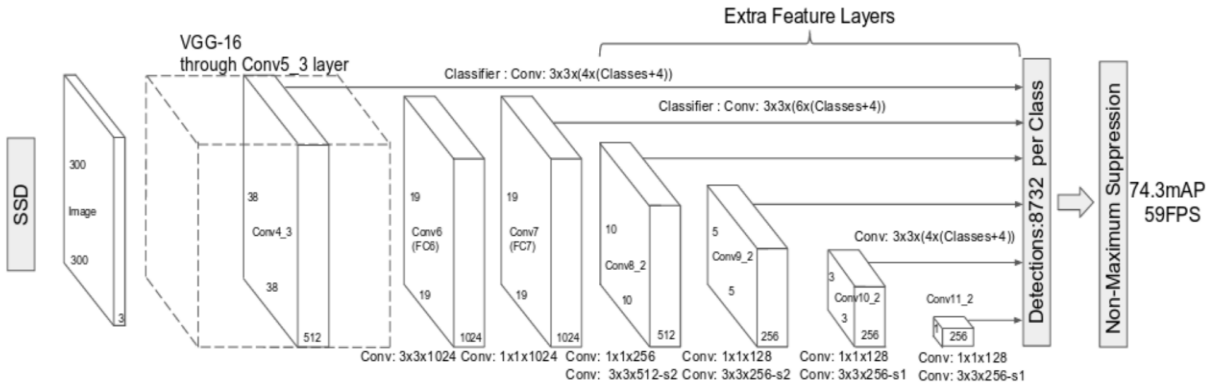


Fig. 2. SSD Architecture

2.2.3. Feature Pyramid Network (FPN) algorithm

FPN is an algorithm that makes use of typical CNN models, this allows for the successful extraction of features. A significant improvement in terms of the representation and output of picture information is represented by its in comparison to traditional CNN networks. In order to accomplish the purpose of improving the representation of each dimension of the input image in the output features, the FPN's objective is to improve the method of feature extraction that is used by the Convolutional Neural Network (CNN) network. This is done in order to meet the goal of improving the outcome. Therefore, the introduction of the FPN structure into the Faster R-CNN algorithm results in an improvement to the feature extraction component of the initial Faster R-CNN method. This enhancement is a desirable outcome. Consequently, this results in the acquisition of extra feature layers as well as a bigger pooling of the ROI as explained in (Lin, Tsung-Yi, et al,2017).

The capabilities of classic CNN models can be improved with the help of FPN, which enables these models to generate feature maps that are more useful for subsequent tasks such as object detection or classification analysis. Essentially, it is a method that may be utilised to enhance the expression of CNN features within the backbone network context. There are three fundamental processes that make up the system: the bottom-up path, which generates dimension features from the bottom to the top; the top-down path, which supplements and enhances features from the top to the bottom, and the correlation expression between the features of the CNN network layer and the features of each dimension of the final output, as shown in Fig.3. As shown in this paper (Fu, Xiang, et al, 2023) FPN is responsible for the production of features that are able to absorb information from every level of the CNN network. After that, the expression feature combination that is ultimately created is created by combining these features

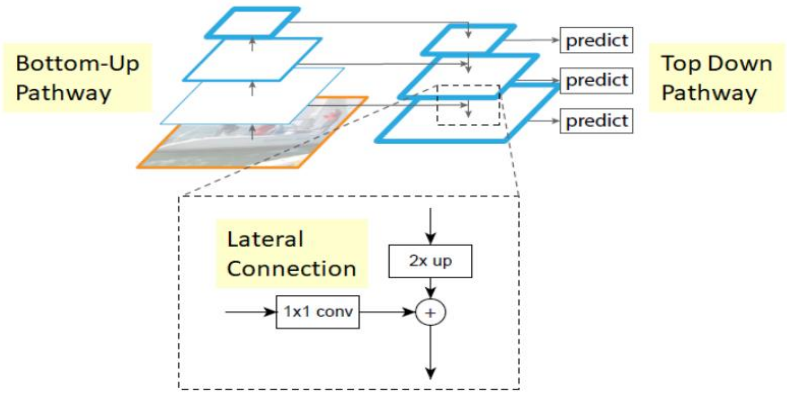


Fig.3 .Bottom-up and top-down architecture of FPN with a building block
(Lin, T.Y., Dollár 2017)

2.2.4. FOMO (Fast Objects, More Objects)

The Faster Objects, More Objects (FOMO) technique is a revolutionary method in machine learning for object detection and it considered as approach for real-time object detection on microcontrollers. Unlike traditional approaches that involve bounding box computations and parameters, FOMO just emphasizes the positions and quantities of each item class. The methodology is based on the process of dividing the input image into tiles of a predetermined size, such as 8x8 pixels, as shown in Fig .4, followed by the application of a classifier on each individual grid. At this juncture, it is possible to build a heat map that visually represents the approximate positions of objects, wherein a smaller tile size is indicative of higher levels of accuracy as explained in (Sakr, F. O. U. A. D.,2023).

The utilization of FOMO allows microcontrollers to execute object detection algorithms with enhanced efficiency in terms of speed and reduced RAM consumption. In the context of FOMO,

it is possible to include numerous objects from distinct classes within a single training image. However, it is crucial to ensure that the bounding boxes of these objects do not overlap and that there exists a sufficient amount of space between them. Consequently, for an image measuring 320x320 pixels, the image would be segmented into a 40x40 grid. A classification algorithm is then applied to each cell within this grid. (R. Banbury, et al.2021)

FOMO is specifically created for scenarios when the precise placements of objects are more important than their sizes. It employs a centroid-based detection approach instead of the conventional bounding box method. It functions effectively with things that have comparable dimensions and should not be positioned in close proximity to one another. The lesser complexity and smaller size of FOMO make it appropriate for devices with limitations, but it may lead to lower precision when compared to more intricate models such as SSD.

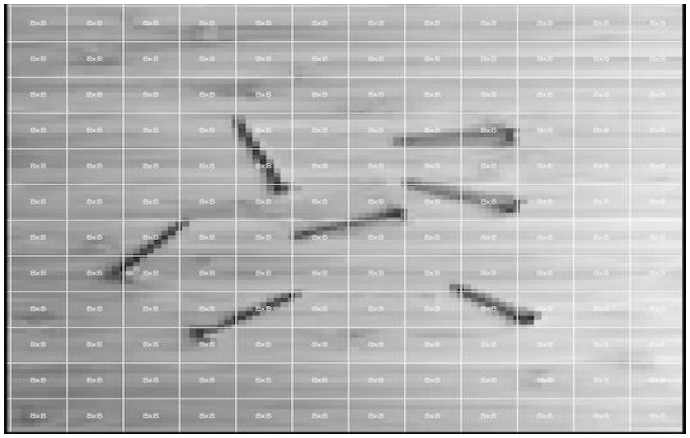


Fig. 4. Explains how FOMO works (Rust, E,2023)

3. OVERVIEW OF MACHINE LEARNING FRAMEWORK FOR EMBEDDED SYSTEM (TINYML)

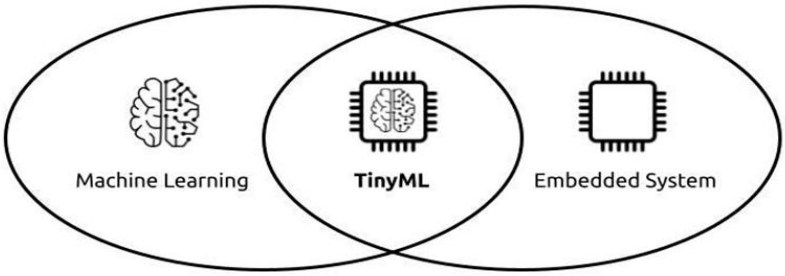


Fig. 5. Definition of TinyML

Tiny Machine Learning (TinyML) enables cognitive functionalities on low-resource IoT devices, specifically Microcontroller units, as explained in (Aoueileyine, Mohamed Ould-Elhassen,2023). To refer to machine learning applications that utilize compact models for very

efficient inference with minimal energy consumption. TinyML is an emerging topic that focuses on implementing machine learning algorithms, such as object detection and classification, specifically for microcontrollers (MCUs). The goal is to leverage the large number of MCUs available to do machine learning tasks at the extreme edge, as shown in Fig.5, with the help of interfacing systems like sensors. Most microcontrollers (MCUs) operate within the Milli-Watt power range, which means that an MCU with similar power consumption levels can run on a coin cell battery for more than a year. TinyML is an emerging interdisciplinary field that focuses on implementing deep neural network models on embedded systems, such as micro-controller-driven devices. It involves the integration of machine learning, software, and hardware.

TinyML will enable the development of innovative services and applications at the edge, relying on distributed edge inference and autonomous decision-making instead of server computation (Schizas, Nikolaos, et al,2022). Specifically, the origins of TinyML may be traced back to the emergence of Edge ML, which addresses the infrastructure requirements for continuous data flow in traditional IoT.

TinyML facilitates the implementation of machine learning applications directly on edge devices, enabling real-time and low-latency inference without dependence on cloud-based servers. This has diverse applications in domains such as IoT, wearable devices, robotics, and other fields. The primary goal of incorporating the TinyML framework into these devices is to achieve decreased latency, optimized bandwidth usage, enhanced data security, heightened privacy, reduced expenses, and total cost reduction in cloud environments. Similarly,as shown in Table 2 TinyML is not regarded as a replacement for the Fog or Cloud paradigms; instead, it is seen as an additional element that operates in conjunction with these paradigms (Sanchez-Iborra, et al,2020).

While TinyML presents immense potential, it also faces several challenges that must be addressed to fully realize its capabilities (Abadade, Youssef, et al,2023):

a. Model Optimization: There is still a substantial obstacle to overcome in the form of the development of highly efficient models that are capable of completing difficult tasks with low resources. The importance of conducting additional research into architecture design and model compression methodologies cannot be overstated.

b. Hardware Limitations: TinyML is still in its infancy compared to the development of dedicated hardware accelerators that are low-power. The advancement of TinyML capabilities will be extremely dependent on the continuation of innovation in hardware design.

c. Energy Efficiency: Managing the amount of electricity that is consumed by gadgets is becoming even more important as technologies continue to advance. Building machine learning algorithms and hardware that are efficient with energy will be essential to TinyML's success in the long run.

d. Privacy and Security: The protection of personal information and data security is becoming increasingly crucial as the number of devices that process sensitive data increases. As they work on new TinyML applications, researchers and developers have a responsibility to solve these concerns.

Table 2. A comparison of hardware for CloudML, MobileML and TinyML (Ray, Partha Pratim,2022)

Platform	Architecture	Memory	Storage	Power	Price
CloudML Nvidia V100	GPU Nvidia Volta	HBM 16GB	SSD/Disk TB~PB	250W	$9K
MobileML Cell Phone	CPU Mobile CPU	DRAM 4GB	Flash 64GB	~8W	~$750
TinyML	MCU Arm4 Arm7 Arm7	SRAM 128KB 320KB 521KB	eFlash 0.5MB 1MB 2MB	0.1W 0.3W 0.3W	$3 $5 $8

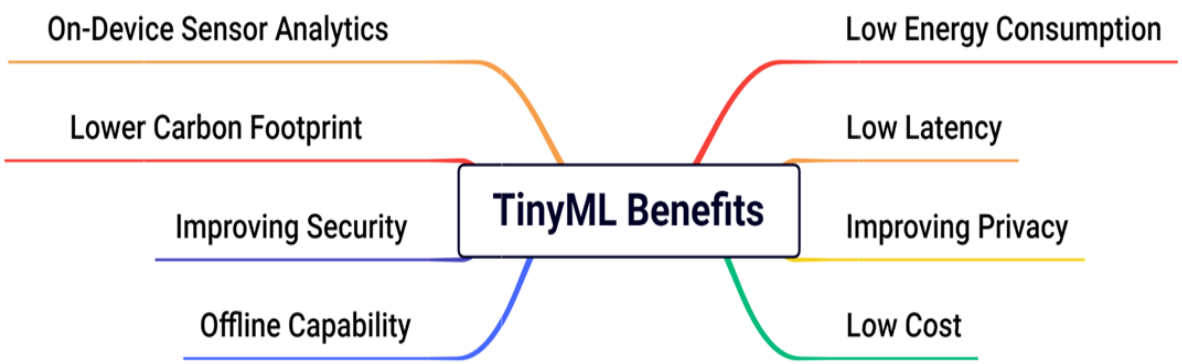


Fig. 6: TinyML benefits (Abadade, Y.et al,2023)

3.1. Framework For TinyML

As a result of the excessive memory requirements, well-known machine learning framework such as TensorFlow and PyTorch are not suited for use inference on MCUs. TensorFlow Lite,

MicroPython, and an online platform for TinyML are just some of the machine learning inference tools that have arisen to meet this demand on MCUs.

3.1.1. TensorFlow Lite

Machine learning models can be executed on edge devices, such as microcontrollers and mobile devices, using a software framework called TensorFlow Lite (TFL), which was developed by Google. LittleML applications are a good fit for it because it is designed to be both lightweight and efficient. To facilitate edge-based machine learning, TFL was developed specifically for this purpose. Consequently, a vast array of efficient algorithms can be executed on resource-constrained periphery devices, such as smartphones, microcontrollers, and other circuits. is a condensed version of the open-source TF machine learning framework developed by Google. (Zim, Md Ziaul Haque,2021). It enables tasks such as image classification, object detection, and natural language processing to be performed directly on the device, without requiring a constant internet connection or relying on cloud-based services.

In order to conduct machine learning in an effective and efficient manner on embedded devices that only have a few kilobyte of memory, TFL is a framework that has been specifically designed for this purpose. The software is compatible with a wide ranges of platforms, including embedded Linux and microcontrollers, as well as Android and iOS devices. Support for a wide range of programming languages is also included, including Java, Swift, Objective-C, C++, and Python. Image recognition, natural language processing, recommendation systems, and other applications are only some of the use cases that are made possible by this (Warden, et al,2019). TinyML relies on TFL, which enables developers to deploy ML models on these devices with little effect on battery consumption and performance. To make machine learning model that have been trained run more effectively on edge devices, TFL offers tools for that. Developer may simply deploy their models across a wide range of devices thanks to the runtime libraries optimized for various hardware platforms. An integral part of the tinyML ecosystem, TFL enables the deployment of machine learning models on edge devices with limited resources.

3.1.2. MicroPython

MicroPython is a free and open-source software environment and programming language optimised for use on embedded systems and microcontrollers. As the author (Bell, Charles,2017) explains The original intent of MicroPython was to offer a high-levels programming language that could be utilized for the purpose of swiftly creating prototypes, developing them, and deploying them on compact devices. MicroPython was an attempt to streamline the process by offering a Python-like language that was both familiar and easy to learn. Arduino, ESP8266/ESP32, STM32, Raspberry Pi Pico, and many more microcontroller

brands are all compatible with it. It has a script mode for executing standalone programs and an interactive shell for real-time experimentation and debugging. Continuous development and improvement have been ongoing processes for MicroPython since its start.

Today, MicroPython is widely used in various applications such as Internet of Things (IoT) devices, robotics projects, home automation systems, sensor networks, educational platforms, and more. Its simplicity and versatility make it an attractive choice for developers looking to work with microcontrollers without sacrificing ease-of-use or power. Similar to TFL, MicroPython includes a runtime interpreter to interpret the bytecode. With MicroPython you can write clean and simple Python code to control hardware instead of having to use complex low-level languages like C or C++ (what Arduino uses for programming). Unfortunately, MicroPython is 101−102 orders of magnitude slower than pure C/C++ , preventing its adoption in time-critical systems(Raza, Wamiq, et al,2021) .

3.1.3. Online Platform For TinyML

Edge Impulse is a service that simplifies the process of generating TinyML models that are specifically targeted for edge devices since it streamlines the process. The training process is carried out on a cloud-based platform, and the trained model that is produced afterwards may be readily transferred to a device located at the edge of the network. In addition, Edge Impulse makes the acquisition of actual sensor data much easier, it enables quick signal processing from raw data to neural networks, and it improves the effectiveness of testing procedures.

3.2. Limited resources embedded systems

Embedded systems are highly diverse and efficient devices that vary in memory size, ranging from a few Kilobytes to a few hundred Kilobytes, and consume less than 3mW of energy. Conventional mobile gadgets, which are experiencing advancements in their processing powers and energy efficiency, nonetheless consume power ranging from hundreds of milliwatts to a few watts for their functioning. The power consumption of these devices is considerably greater than that of microcontrollers (MCUs).

The criterion for TinyML, which requires an energy consumption of less than 1 mW, implies This suggests that embedded devices should be used as hardware platforms (Kamath, Vidya, and A. Renuka,2023). Embedded devices are electronic systems specifically engineered to carry out designated tasks within a larger system or product. These devices are typically small, low-power, and have limited computing resources. Embedded devices can be found in a wide range of applications, including smart appliances, medical devices, security systems,

transportation systems, and many more. They are built with specialized hardware and software to meet the specific requirements of the intended application.

Overall, embedded devices play a crucial role in enabling automation, connectivity, and intelligence in various industries by providing dedicated functionality in a compact form factor (Zhou, Xu, et al,2023). Also, object detection on embedded devices requires a careful balance between accuracy and resource efficiency to enable real-time performance within the limitations of the hardware. These are some types of embedded devices that are used in TinyML application:

3.2.1. Raspberry Pi

The Raspberry Pi, depicted in Fig 7, is an affordable single-board computer (SBC) created by the Raspberry Pi Foundation (raspberrypi.org). The three variants, commonly referred to as 'Raspberry Pi's', are built on a system-on-a-chip architecture. Each model includes an integrated CPU and GPU, on-board memory, and a power input of 5 V DC. Additionally, all models are equipped with a connector specifically designed for connecting a dedicated camera. They also feature a set of general-purpose input/output (GPIO) pins that enable communication with various electrical components, such as LEDs, buttons, servos, motors, power relays, and an extensive selection of sensors. Hardware attached on top (HAT) refers to specialized expansion boards that can be connected to the GPIO pins (Rakkiannan, Thamilselvan, et al,2023). The Raspberry Pi encompasses a wide range of activities, including real-time and independent monitoring of the environment and video documentation of laboratory research, as well as the establishment of measuring stations for long-term field observations. The Raspberry Pi can function as an embedded device for object detection by leveraging its GPIO pins, camera module, and robust processing capabilities (Jolles, Jolle W,2021).

Fig 7. Raspberry Pi

3.2.2. Arduino

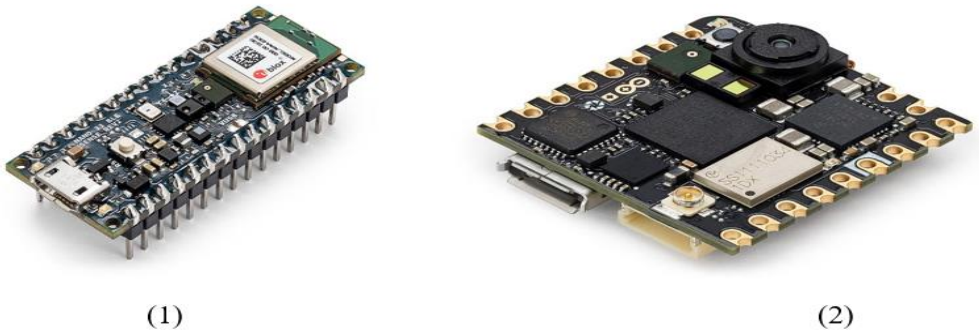
Arduino is a customized microcontroller that is open-source and designed for physical computing. It operates on a single board. The Arduino board possesses an exceptional processing capability of up to 16 MHz, which may vary based on the specific board being used. Arduino made significant progress by introducing a user-friendly integrated development environment (IDE) and standardized hardware. Arduino will come in handy for controlling motors, LEDs, or interfacing sensors (Álvarez,et al,2021) .There are a huge variety of Arduino boards, all with different capabilities. These are some of them that can detect objects:

3.2.2.1. Arduino nano 33 BLE

The Arduino Nano 33 BLE Fig.8 Sense is a tiny development board with a Cortex-M4 microcontroller, motion sensors, a microphone and BLE. Also it is a member of the Nano family depending on the purpose, Arduino Nano, Arduino Nano 33 IoT, Arduino Nano 33 BLE, Arduino Nano 33 BLE Sense, etc. Since there are various boards, a clear distinction is needed. Nano33 BLE Sense is Nordic Semiconductors' nRF52840, 32-bit ARMCortex-M4 CPU running at 64MHz Equipped with 1MB (nRF52840) CPU Flash Memory Consists of. Arduino UNO equivalent to the standard (Kurniawan, Agus,2021). Increased performance and usage compared to R3 (ATmega328P, 16MHz, 32KB). It can be used in a variety of ways, including ML (Machine Learning). As the name of the Nano 33 BLE Sense board suggests, BLE is stands for Bluetooth Low Energy and uses blue in a low-power environment. Tooth communication is possible. Sense features a variety of sensors means.

3.2.2.2. Arduino Nicla Vision

The Nicla Vision Fig 8 is a ready-to-use, standalone camera for analyzing and processing images. Thanks to its 2MP color camera, smart 6-axis motion sensor, integrated microphone, and distance sensor, it is suitable for asset tracking, object recognition, and predictive maintenance. Nicla Vision is a highly capable and adaptable tool specifically engineered to facilitate the development and prototyping of projects centered on image processing and machine vision at the edge. It also enable fast Machine Vision prototyping (Siderius, J. C,2023).



(1) (2)

Fig. 8. 1- Arduino Nano 33 BLE and 2- Arduino Nicla Vision

Table 3. Some devices that can detect objects and support TinyML

Hardware	Processor	CPU Clock	Flash Memory	SRAM Size	Sensors	Power	Organization
Alif Ensemble E7	Cortex-M55 with Ethos-U55 microNPUs	400MHz	4MB	4MB	Camera and microphone	1.71-3.6V	Alif Semiconductor
Arduino Nicla Vision	Dual Ann Cortex M7ZM4	M7: 480MHz and M4: 240MHz	2MB	1MB	Camera, microphone andIMU	3.7V Li-po battery	Arduino
Seeed Grove Vision AI Module	Hi max HX6537-A	400MHz	2MB	2MB	camera, microphone and accelerometer	5V	Seed Studio
Portenta H7 Lite	Dual Cortex M7+M4	M7: 480MHz and M4: 240MHz	16MB	8MB	Camera	3.7V Li-Po	Arduino
Hardware	Processor	CPU Clock	Flash Memory	SRAM Size	Sensors	Power	Organization
Alif Ensemble E7	Cortex-M55 with Ethos-U55 microNPUs	400MHz	4MB	4MB	Camera and microphone	1.71-3.6V	Alif Semiconductor

4. OVERVIEW OF APPLICATION OF TINYML

4.1. Forest Fire Detection

Forest fires pose a substantial peril to the environment, resulting in ecological harm, financial setbacks, and endangering human lives. As computers become smaller, the accuracy and speed of recognition are decreasing because of insufficient computational capacity. In the aforementioned publication (Maslov, D,2022). Real-time fire detection is introduced, this is done using YOLOv4 digital technology. This algorithm replaces CSP Darknet53 in yolov4. The Vision Transformer model is incorporated into the feature extraction network backbone and the yolov4 model to enhance the attention mechanism of the multi-head model for

preprocessing image characteristics with improved articulation. In addition, the YOLOv4 model incorporates the Efficient Channel Attention (ECA) module, which is trained on the channel dimensions of the four feature layers of the head network. This enhances the model's ability to prioritise relevant information (Sobha,et al ,2023). Despite the high accuracy of the generic fire detection algorithm, its complexity renders it unsuitable for efficient execution on small embedded systems. If the algorithm is not able to operate reliably on certain embedded devices, then these methods become impractical.

4.2. TinyML in Healthcare

Individuals with visual impairments utilize several assistive technology in their everyday lives to engage in numerous activities, including navigation and reading texts. Recent technological improvements have allowed developers to efficiently deploy and operate assistive software on embedded devices. In paper (Kunhoth, Jayakanth, et al) introduces a novel wearable technology named VisualAid+ designed to aid those with visual impairments. A portable wearable assistive system has been built for those with visual impairments, utilizing the capabilities of TinyML and Edge AI. This system focuses on object identification and visual scene narration. The VisualAid+ system comprises a Raspberry Pi device, a computer acting as a server, a power source, and two cameras affixed to a wearable glass. One of the cameras is integrated with an ESP32 microcontroller. The camera will record the visuals in front of the user and transmit the images to both the ESP32 and Raspberry Pi. The person detection model will determine the presence of any individual in the user's vicinity. If a person is discovered, the user will be notified through auditory feedback. The object detection model has the ability to identify and classify 80 distinct categories of items depicted in photographs, and it can audibly articulate the names of the objects it detects. In addition, the system will offer the auditory description of visual sceneries (converting image captions into speech) to the user. The server utilizes the visual narration approach and necessitates internet connectivity.

4.3. Agricultural Field

Recently, there has been an increasing fascination with utilizing technology to enhance agricultural practices and tackle the obstacles encountered by the farming sector. However, conventional methods of evaluating and monitoring soil are frequently marked by their lengthy duration, demanding manual procedures, and the requirement for specific expertise. TinyML, a burgeoning discipline, presents auspicious remedies for surmounting these constraints. Deploying energy-efficient and low-latency TinyML models is crucial for enabling real-time decision-making in agricultural contexts that have restricted resources. This TinyML model, which is sensitive to power consumption and delays, can be used in a range of real-time

agricultural scenarios. This technology enables the continuous monitoring of soil quality indices, such as moisture content, pH level, and nutrient levels, across different types of farms. The model uses sensors and low-power microcontrollers to facilitate on-site study and provide quick feedback to farmers (Bhattacharya,et al,2023).

4.4. Reduce Traffic Accidents

The effect of driver drowsiness has a significant role in the occurrence of motor vehicle collisions, hence yielding grave outcomes such as bodily harm or fatality. There exist multiple factors contributing to the occurrence of accidents resulting from driver fatigue. Fatigue can result in a decrease in an individual's reaction time, impairment of decision-making capabilities, and challenges in sustaining attention while driving. Consequently, individuals may experience a deficiency in perceiving crucial visual stimuli, leading to a lack of awareness regarding potential dangers or the manifestation of suboptimal decision-making abilities during the act of operating a vehicle. In the event that the system perceives the driver's eyes to be closed for a duration of two seconds or more, it will initiate an alert with the purpose of arousing the driver from their state of drowsiness and reinforcing the importance of maintaining attentiveness and vigilance while operating the vehicle (Alajlan,et al,2023) .

Table 4. Summary of some applications of TinyML

Reference	Application	ML algorithm	Accuracy	Power	Memory consumption
ARSLAN, S.E. and BOLAT	obstacle detection with multi-zone time of flight sensors	CNN	93.69%	**----**	11.8Kb
Xu, K., Zhang, H., Li, Y., Zhang	Real-time Visual Processing at Edge	**CNN**	66.5%	160mW	----
C. Nicolas, B. Naila, and R.-C. Amar	Detection and counting strawberries	FOMO	90%	120-130 mW	243.9 KB
A. Faraone and R. Delgado-Gonzalo	Edetection of Cardiac Arrhythmai	CNN	78.4%	----	195.6 KB
Simões, Bruno Gobato	Brain Cancer Detection	CNN	94.4%	----	----
V. M. Oliveira and A. H. Moreira	Detecting anomalies on industrial machines	CNN	94%	26 mW	100KB
P. Andrade, I. Silva, G. Signoretti, M. Silva, J. Dias, L. Marques, and D. G. Costa	Detecting road anomalies	SSD	98%	54 mW	----
V. Gutti and R. Karthi	Classification fruits and vegetables	FOMO	98%	10-20 mW	66.1 KB
Sabovic, Adnan	Person Detection	MobileNet V1	81.15 %	----	73KB

5. CONCLUSIONS

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. This paper comprehensively reviews both traditional object detectors starting from RCNN and going all the way to the latest. One of the main goals of the work was to learn and understand with the daily increase in powerful object detectors based on the use of embedded devices, we covered their use in fields such as, forest fire detection, healthcare, agricultural, and Reducing traffic accidents.

6. REFERENCES

A. Faraone and R. Delgado-Gonzalo, ‘‘Convolutional-recurrent neural networks on low-power wearable platforms for cardiac arrhythmia detection,’’ in Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS), Aug. 2020, pp. 153–157.

A. Shawahna, S. M. Sait, and A. El-Maleh, ‘‘Fpga-based accelerators of deep learning networks for learning and classification: A review,’’ ieee Access, vol. 7, pp. 7823–7859, 2018.

Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A comprehensive survey on tinyml. IEEE Access.

Abadade, Youssef, et al. "A comprehensive survey on tinyml." IEEE Access (2023).

Abbas, Syed Mazhar, and Shailendra Narayan Singh. "Region-based object detection and classification using faster R-CNN." 2018 4th International Conference on Computational Intelligence & Communication Technology (CICT). IEEE, 2018.

Alajlan, Norah N., and Dina M. Ibrahim. "DDD TinyML: A TinyML-Based Driver Drowsiness Detection Model Using Deep Learning." Sensors 23.12 (2023): 5696.

Álvarez, José Luis, Juan Daniel Mozo, and Eladio Durán. "Analysis of single board architectures integrating sensors technologies." Sensors 21.18 (2021): 6303.

Aoueileyine, Mohamed Ould-Elhassen. "Tiny Machine Learning for IoT and eHealth Applications: Epileptic Seizure Prediction Use Case." International Conference on Digital Technologies and Applications. Cham: Springer Nature Switzerland, 2023.

ARSLAN, S.E. and BOLAT, B., Tiny machine learning model for obstacle detection with multi-zone time of flight sensors. C. Nicolas, B. Naila, and R.-C. Amar, ‘‘TinyML smart sensor for energy saving in Internet of Things precision agriculture platform,’’ in Proc. 13th Int. Conf. Ubiquitous Future Netw. (ICUFN), Jul. 2022, pp. 256–259.

Bell, Charles. MicroPython for the Internet of Things. Berlin/Heidelberg, Germany: Springer, 2017.

Bharati, Puja, and Ankita Pramanik. "Deep learning techniques—R-CNN to mask R-CNN: a survey." Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019 (2020): 657-668.

Bhattacharya, Saurabh, and Manju Pandey. "Deploying an energy efficient, secure & high-speed sidechain-based TinyML Model for Soil Quality Monitoring and Management in Agriculture." Expert Systems with Applications (2023): 122735.

Du, Lixuan, Rongyu Zhang, and Xiaotian Wang. "Overview of two-stage object detection algorithms." Journal of Physics: Conference Series. Vol. 1544. No. 1. IOP Publishing, 2020.

Fu, Xiang, et al. "DA-FPN: Deformable Convolution and Feature Alignment for Object Detection." Electronics 12.6 (2023): 1354.

Jolles, Jolle W. "Broad-scale applications of the Raspberry Pi: A review and guide for biologists." Methods in Ecology and Evolution 12.9 (2021): 1562-1579.

Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S. and López, O.L., 2023. TinyML: Tools, applications, challenges, and future research directions. Multimedia Tools and Applications, pp.1-31

Kamath, Vidya, and A. Renuka. "Deep Learning Based Object Detection for Resource Constrained Devices-Systematic Review, Future Trends and Challenges Ahead." Neurocomputing (2023).

Kunhoth, Jayakanth, et al. "VisualAid+: Assistive System for Visually Impaired with TinyML Enhanced Object Detection and Scene Narration." 2023 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2023.

Kurniawan, Agus. "Iot projects with arduino nano 33 ble sense." Berkeley: Apress 129 (2021).

Lin, Ji, et al. "Tiny Machine Learning: Progress and Futures [Feature]." IEEE Circuits and Systems Magazine 23.3 (2023): 8-34.

Lin, Szu-Yin, and Hao-Yu Li. "Integrated circuit board object detection and image augmentation fusion model based on YOLO." Frontiers in Neurorobotics 15 (2021): 762702.

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Maslov, D. (2022) Announcing Official Support for the Arduino Nicla Vision, Edge impulse. Available at: https://edgeimpulse.com/blog/announcing-official-support-for-the-arduino-nicla-visionwww.skroutz.gr/s/38491287/Arduino-Nicla-Vision-ABX00051.html (Accessed: 02 January 2024). (Accessed: 03 January 2024).

Murthy, C. B., Hashmi, M. F., Bokde, N. D., & Geem, Z. W. (2020). Investigations of object detection in images/videos using various deep learning techniques and embedded platforms—A comprehensive review. Applied sciences, 10(9), 3280.

P. Andrade, I. Silva, G. Signoretti, M. Silva, J. Dias, L. Marques, and D. G. Costa, ‘‘An unsupervised TinyML approach applied for pavement anomalies detection under the Internet of Intelligent vehicles,’’ in Proc. IEEE Int. Workshop Metrology for Ind. 4.0 IoT (MetroInd4.0IoT), Jun. 2021, pp. 642–647

Qureshi, Rizwan, et al. "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)." Authorea Preprints (2023).

R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J. sun Seo, J. Sieracki, U. Thakker, M. Verhelst, and P. Yadav, “Benchmarking TinyML systems: Challenges and direction,” 2021

Rakkiannan, Thamilselvan, et al. "An automated network slicing at edge with software defined networking and network function virtualization: a federated learning approach." Wireless Personal Communications 131.1 (2023): 639-658.

Rani, Shilpa, Deepika Ghai, and Sandeep Kumar. "Object detection and recognition using contour based edge detection and fast R-CNN." Multimedia Tools and Applications 81.29 (2022): 42183-42207.

Ray, Partha Pratim. "A review on TinyML: State-of-the-art and prospects." Journal of King Saud University-Computer and Information Sciences 34.4 (2022): 1595-1623.

Raza, Wamiq, et al. "Energy-efficient inference on the edge exploiting TinyML capabilities for UAVs." Drones 5.4 (2021): 127.

Sabovic, Adnan, et al. "Towards energy-aware tinyML on battery-less IoT devices." Internet of Things 22 (2023): 100736.

Sakr, F. O. U. A. D. "Tiny Machine Learning Environment: Enabling Intelligence on Constrained Devices." (2023).

Sanchez-Iborra, Ramon, and Antonio F. Skarmeta. "Tinyml-enabled frugal smart objects: Challenges and opportunities." IEEE Circuits and Systems Magazine 20.3 (2020): 4-

Schizas, Nikolaos, et al. "TinyML for Ultra-Low Power AI and Large Scale IoT Deployments: A Systematic Review." Future Internet 14.12 (2022): 363.

Sérgio, André G. Ferreira, and Jorge Cabral. "Machine learning in resource-scarce embedded systems, FPGAs, and end-devices: A survey." Electronics 8.11 (2019): 1289.

Siderius, J. C. Teaching (Tiny) ML using a tangible educational kit. BS thesis. University of Twente, 2023.

Simões, Bruno Gobato, et al. "SPECIALIZED BRAIN TUMOR DETECTION SYSTEM FOR SUPERIOR VIEW OF CRANIAL MRI USING AI AND TINY ML TECHNIQUES." Anais Do Workshop De Micro-ondas. Clube de Autores, 2023.

Sobha, Prathibha, and Shahram Latifi. "A Survey of the Machine Learning Models for Forest Fire Prediction and Detection." International Journal of Communications, Network and System Sciences 16.7 (2023): 131-150.

V. Gutti and R. Karthi, ‘‘Real time classification of fruits and vegetables deployed on low power embedded devices using tiny ML,’’ in Proc. Int. Conf. Image Process. Capsule Netw. Cham, Switzerland: Springer, 2022, pp. 347–359.

V. M. Oliveira and A. H. Moreira, ‘‘Edge AI system using a thermal camera for industrial anomaly detection,’’ in International Summit Smart City 360◦ . Cham, Switzerland: Springer, 2022, pp. 172–187.

Warden, Pete, and Daniel Situnayake. Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers. O'Reilly Media, 2019.

Xu, K., Zhang, H., Li, Y., Zhang, Y., Lai, R. and Liu, Y., 2023. An ultra-low power tinyml system for real-time visual processing at edge. IEEE Transactions on Circuits and Systems II: Express Briefs.

Zhang, Xi, et al. "A Novel SSD-Based Detection Algorithm Suitable for Small Object." IEICE TRANSACTIONS on Information and Systems 106.5 (2023): 625-634.

Zhou, Xu, et al. "A Survey of the Security Analysis of Embedded Devices." Sensors 23.22 (2023): 9221.

Zim, Md Ziaul Haque. "TinyML: analysis of Xtensa LX6 microprocessor for neural network applications by ESP32 SoC." arXiv preprint arXiv:2106.10652 (2021).