Seminar Report

On

TINYML IN AGRICULTURE SECTOR: AWAKENING OF MINDS IN LOW POWER EDGE DEVICES

Submitted by

DANIEL V MATHEW
(Reg No: KTE22EC029)

In partial fulfillment for the award of the Degree of

Bachelor of Technology
in
ELECTRONICS AND COMMUNICATION ENGINEERING

Under the Supervision of

Prof. Sujithamol S

Department of Electronics and Communication Engineering
Rajiv Gandhi Institute of Technology, Pampady
Kottayam 686 501

Sept 2025

Department of Electronics and Communication Engineering
Rajiv Gandhi Institute of Technology
Kottayam-686 501

Certificate

This is to certify that this seminar report entitled **"TINYML IN AGRICULTURE SECTOR: AWAKENING OF MINDS IN LOW POWER EDGE DEVICES"** *is a bonafide record of the work done by* ***DANIEL V MATHEW*** (KTE22EC029) *under our guidance towards the partial fulfillment of the requirements for the award of the degree of* ***Bachelor of Technology*** *in* ***Electronics and Communication Engineering*** *of A P J Abdul Kalam technological university, Trivandrum, during the year 2025 and that this work has not been submitted elsewhere for the award of any degree.*

Prof. Sujithamol S
(Supervisor)

Dr. Sajeev G P & Dr. Nelsa Abraham
(Coordinators)

Dr. David Solomon George
(HoD)

Acknowledgement

The success accomplished in this seminar would not have been possible without the timely help and guidance rendered by many people to whom I feel obliged and grateful.

First of all, I would like to express my sincere thanks to our beloved principal **Dr. Prince A**, for his support and encouragement.

I use this occasion to express my thanks to **Dr. David Solomon George**, Head of Electronics and Communication Engineering Department for his continuous encouragement to fulfill my seminar.

I am thankful to my seminar coordinators **Dr. Sajeev G P**, **Dr. Nelsa Abraham** and my guide **Prof. Sujithamol S**, for their valuable suggestions, inspiration and motivation throughout the seminar work.

I am also thankful to all the teaching and non teaching staff in Electronics and Communication Department for their timely assistance. I thank my parents and friends for their kind cooperation and suggestions which helped me very much for the accomplishment of this seminar work.

Above all, I would like to express my sincere thanks to the Almighty God, who empowered me to fulfill this work, by showering his abundant grace and mercy.

Rajiv Gandhi Institute of Technology, DANIEL V MATHEW
Sept 2025

Abstract

TinyML is a subset of machine learning (ML) that focuses on developing and deploying ML models on resource-constrained devices such as Microcontrollers (MCUs), System-on-Chip (SoCs) and other such devices. Frameworks such as TensorFlow Lite Micro, and Edge Impulse are used for developing ML models for low powered devices. Techniques used in TinyML enables these devices to be more efficient at running ML models. Thereby improving decision-making, reducing cost and power consumption.

This seminar aims to explore what TinyML is, its recent trends and advancements, specifically in its role in the Agriculture sector. TinyML enables low powered devices such as ESP32 CAM modules, STM32 microcontrollers, and Arduinos to be used in ML applications. These devices can then perform growth monitoring, plant disease detection, and mirco-climate management in place of expensive Raspberry Pis and other Mini PCs. Thus, TinyML powered low power edge devices may find applications in smart agriculture in isolated places and also where low power consumption is at top priority.

Contents

List of Figures

List of Tables

chapter 01

Tiny Minds and Immense Data

Tiny minds and immense data are all around us. And these tiny minds are growing in numbers day by day. They consists of low power IOT devices, hand held devices, wearables, DSPs etc.

1.1 Immense Data

These tiny devices are indeed surrounded by a vast amount of Data. From them, they infer about their surroundings and carries out their creator's intentions. At present, these tiny minds are only capable of utilizing a tiny slice of the vast amount of data available to them.

1.2 Doing More

But what if they could do more? What if they could use a much more wider range of this available data? For that we need to make these tiny devices able to process different kinds of data. Some types of data are a bit tricker to be comprehended by these tiny devices. Complex data such as *images* fall into such categories.

This seminar aims to explore how these tiny minds can become able to comprehend such complex data through a new technology called TinyML and how those new capabilities can further provide viable solutions to different problems arising in the *Agriculture sector*.

1.3 Data Processing Systems

All around us we are surrounded by different systems that constantly engaged in different activities. These systems range from a home thermostat to an industry grade quality testing machine. But they all have one thing in common. They all take in some form of data, they process this data, and take some kind of action based on the extracted information.

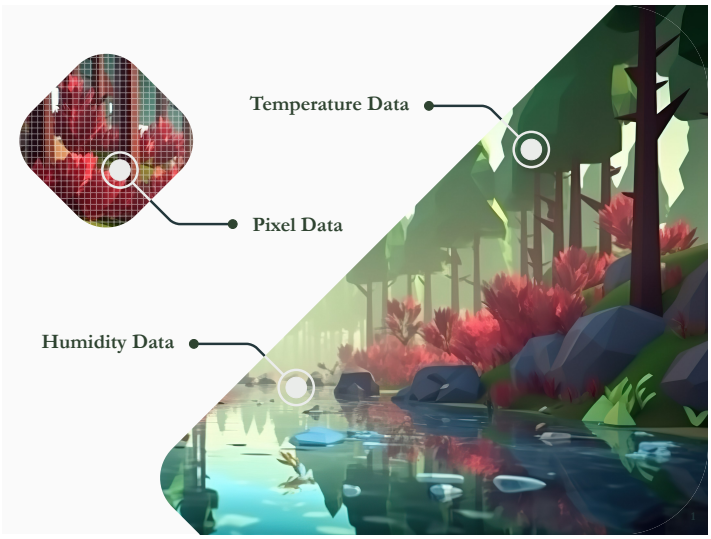

Figure 1.1: Different types of data surrounding us.

As we can see from figure 1.1, there are different kinds of data. Some are easily comprehensible and others that are a bit hard to comprehend. Image data are such complex data forms. In order to properly make use of the data, we need to somehow extract the useful information out them.

1.4 Asking the Right Questions

In order to make sense of the data, we need to ask the *right questions* about the data. Traditionally this is know as an *Algorithm*.

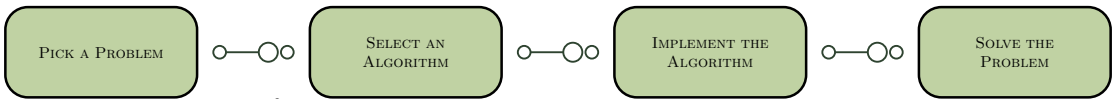

Figure 1.2: Traditional way of solving a problem.

Once we have a problem, we will analyse the problem then pick an appropriate *algorithm* that fits the problem. Then we can simply implement the algorithm and be able to arrive at the solution.

But there is an upper limit this method can take us. What if the problem that we are trying to solve is really complicated and we can't come up with a proper algorithm to solve it? In those cases we need to find another way to solve the problem at hand. But before that let's see some of the different ways we can model our questions.

1.5 Ways of Asking Questions

There are several ways to implement an algorithm. One way is to use an *if-else* to parse through the data available. Figure 1.3 (a) depicts such a method. Like wise, one be able to transform the inputs directly into output[1]. Figure 1.3 (b) shows such a transforming matrix[2].

```
if this_pixel == this_color:
    if this_pixel is at_edge:
        # more and more if statements
elif this_pixel == that_color:
    if this_pixel is in_the_middle:
        # many more
```

(a)

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & \cdots & c_{1n} \\ c_{21} & c_{22} & c_{23} & c_{24} & \cdots & c_{2n} \\ c_{31} & c_{32} & c_{33} & c_{34} & \cdots & c_{3n} \\ c_{41} & c_{42} & c_{43} & c_{44} & \cdots & c_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & c_{m3} & c_{m4} & \cdots & c_{mn} \end{pmatrix}$$

(b)

Figure 1.3: Two ways to model our questions.

In some sense, a matrix transform is much more of a compact version of the *if-else* statement. Matrix operations are parallelizable, therefore operations like these can be accelerated using dedicated hardware like GPUs.

1.6 Learning to Ask Questions

We have seen that we can extract out any kind of information[3] from any kind of data by simply asking it the right questions. And we can model our questions with matrices. In simple terms, with the *right matrix*, we can extract the desired information from any given data.

Now the question becomes, how we are going find the *right matrix* for a particular problem? The techniques and methods we use for that is known under the umbrella term *Machine Learning*. Once we have a pool of data and the desired output[4], we can use machine learning techniques[5] to tweak and tune the *coefficients* of the question matrix, thereby finding the *right question matrix*.

[1] Given that we know the coefficients of transforming matrix.
[2] They need not be 2 dimensional but could be N dimensional.
[3] Obviously, only if it contains the information.
[4] It may be some class, or a particular feature.
[5] Like backpropagation and such.

1.7 ML Models with More and More Capabilities

Day by day these ML models are getting more and more capabilities. They can process large amount of data. And pick up subtle variations in its inputs. LLMs like ChatGPT, Gemini are some examples. Also they are getting better and better at other things like image, video and audio generation etc. Dall E and Sora are some examples.

1.8 Drawbacks of Advanced ML Models

But all of these comes at a cost. The requirement of powerful hardware to run these models. In most cases it would require a whole data center worth of computational power to run a single instance of the model. This makes edge computing really expensive and non feasible. This forces edge systems to offload other data to remote systems for processing. That comes with its own kind of problems.

1.8.1 Major concerns

The larger and more complex the ML model is, the higher its hardware requirements will be. This leads to expensive hardware. These powerful machines consume a large amount of power to run, which increases the carbon footprint. On the other hand, in most cases, cloud computing solutions provide a viable alternative, but they raise other concerns like privacy and latency.

chapter 02

An Introduction to TinyML

TinyML is a subset of machine learning that focuses on developing and deploying ML models on resource constrained devices. TinyML was coined by *Pete Warden* co-author of the *TinyML* book and one of the founding members of Google TensorFlow team.

TinyML possesses great advantages and some drawbacks. In the following chapters we will explore some of these. After a quick tour of the TinyML workflow, we will move on to its applications.

2.1 Target Devices

TinyML mainly targets devices that are resource constrained with kilo-bytes sized RAM and mega hertz clocked CPUs. These devices usually comes with low volume of flash memory[1]. See figure 2.1 for some of the target devices of TinyML.

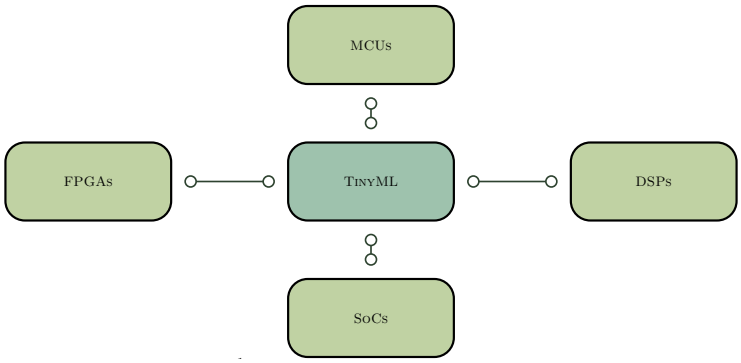

Figure 2.1: TinyML and its target devices.

2.2 "Ok Google" ML Model

In 2014, engineers at Google had to solve a peculiar problem. They wanted the Android phones to be able to wake from sleep mode just using "Ok Google" wake

[1]Few MBs of flash memory.

word(s). In order for this to work, these Android phone needs to continuously process the audio signal from its surroundings and look for the wake words. But unfortunately this can't be carried out by the main CPU, obviously, as it will be in the sleep mode for conserving battery.

So they had to turn to the embedded DSPs in these devices, that only had *tens of kilo-bytes* of memory and *really low* processing power. In short, they had to develop an ML model that will look for the wake words that would fit into the constrains of these tiny DSPs. They in turn, succeeded in this and developed a neural network that does this heavy lifting that sized under just *14 kilo-bytes*. And those DSPs only used few milli-watts of power while running.

2.3 Modern Day's Microcontrollers

That "Ok Google" model was develop around 10 years ago, and utilized those day's DSPs. Let's see some of our day's popular microcontrollers.

2.3.1 ESP32

Flash Memory: 4 MB
CPU Clock Speed: 240 MHz
RAM Available: 520 KB

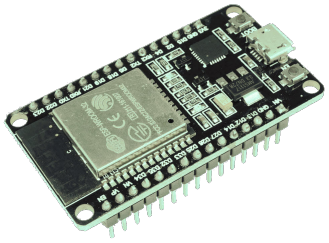

Table 2.1: Table showcasing some of the specifications of ESP32.

2.3.2 Arduino UNO

Flash Memory: 32 KB
CPU Clock Speed: 16 MHz
RAM Available: 2 KB

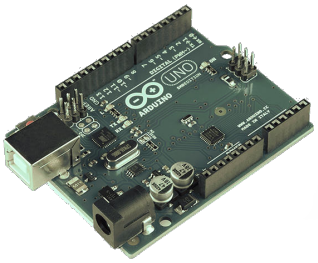

Table 2.2: Table showcasing some of the specifications of Arduino UNO.

2.3.3 Raspberry Pi Pico

Flash Memory: 2 MB
CPU Clock Speed: 125 MHz
RAM Available: 264 KB

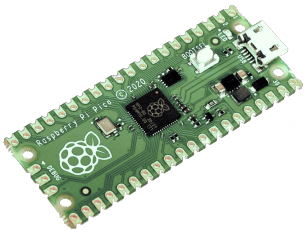

Table 2.3: Table showcasing some of the specifications of Raspberry Pi Pico.

From tables 2.3, 2.2, and 2.1 we can see that modern microcontrollers have came a long way. They are really powerfull at the same time really power efficient. This enables them to host different kinds of ML models without any problems.

chapter 03

Pros and Cons of TinyML

TinyML has several advantages, at the same time it possesses some some disadvantages. In this chapter we will see some of the desirable advantages and one of the major disadvantage. At the end we will take a glance at cascaded systems, and how they can solve one of the major disadvantage of TinyML.

3.1 Benefits of TinyML

There are several reasons to run TinyML on these tiny devices. Table 3.1 show some of them.

Privacy and Security	Economic Viability
Low Power Consumption	Improved Network Latency

Table 3.1: Table showing some of the advantages of TinyML.

We will see each of these in detail in the following sections.

3.1.1 Privacy and Security

The most *secure* connection is *no connection* at all. Since the ML model is running on the device itself, there is no need to sent anything over the wire or air. That is the beauty of *edge computing*, everything stays locally on the device itself. From the privacy and security point of view this is really desirable.

3.1.2 Economic Viability

Modern day's microcontrollers are *extremely cheap*. And they are literally used everywhere. So giving them additional capabilities is much more preferable than replacing them with something that is powerful but expensive. Refer table 3.2 to get an idea about the market price of some of the popular microcontrollers.

From table 3.2 we can see that the average price of these microcontrollers is around 500 INR. At the same time the cheapest SBC[1] costs about 1500 INR. That's just for the board itself. Inorder to run the device one needs to get other accessories[2].

Microcontroller	Price (INR)
ESP32	550
Arduino UNO	570
Raspberry Pi Pico	450

Table 3.2: Table comparing market price of some of the popular microcontrollers.

3.1.3 Low Power Consumption

These tiny devices have advanced much more in the aspect of energy conservation. They can do much more with much less power consumption. Take a look at table 3.3, it gives an idea about the current consumption of some of the microcontrollers.

Comparing to a Raspberry Pi 4, this is really low. Pi 4 consumes about 1.5A of current while running. It is possible to configure these microcontrollers in their respective power saving mode to bring down the current consumption even less. ESP32 has a ULP[3] co-processor that can do various tasks while the main CPU is sleeping for conserving power.

Microcontroller	Current Consumption
ESP32	20-60 mA
Arduino UNO	45-80 mA
Raspberry Pi Pico	18-50 mA

Table 3.3: Table comparing typical current consumption of some of the popular microcontrollers.

3.1.4 Improved Network Latency

This is yet another benefit of *edge computing*. Everything is processed locally, so the latency related with *cloud computing* literally become *zero*. This enables these systems to be deployed to *remote locations* where there have unreliable connections or have no connection at all.

[1]Single Board Computer
[2]Namely, SD card, power supply, etc.
[3]Ultra Low Power.

3.2 Drawbacks of TinyML

We have seen some of the advantages of TinyML, now let's see one of its major disadvantages. The con of a sword is that it's sharp. Just like that, TinyML's greatest advantage is it's greatest disadvantage.

3.2.1 Low Fidelity Nature

It is essential to compress the size of the model to fit with the constrains of the host device. In this process the model could turn into a *low fidelity* one.

But in most cases this will be fine, as the low fidelity one can still infer useful information from the surroundings. Another useful technique to increase the performance of the overall system is to *cascade* the low fidelity one with a high fidelity one.

3.2.2 Cascaded Systems

One way to deal with low fidelity models is to cascade it with a high fidelity model. Figure 3.1 shows such a system.

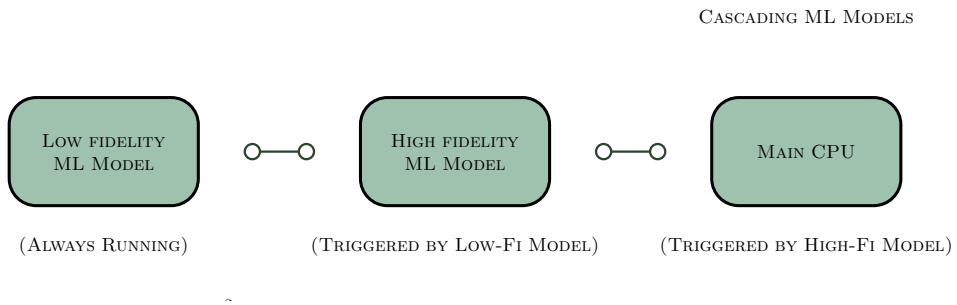

Figure 3.1: Cascading different fidelity ML models.

In this case, the low fidelity one will run continuously and consumes the least amount of energy. Once it picks up something it can wake up the higher fidelity one to run more inferences. If the higher fidelity one confirms the lower fidelity one's inferences, then it can trigger other parts of the system.

In this way, the low fidelity one can help to conserve energy, by preventing higher fidelity one from running continuously.

chapter 04

Essence of TinyML

The techniques and methods used in TinyML makes it stand out from rest of the ML paradigm. This chapter will go through the general work flow and then dive into the core techniques such as *quantization*, *pruning*, and *knowledge distillation*.

4.1 General Workflow

Most of the workflow of TinyML is the same as typical ML workflow. Figure 4.1 depicts a simplified overview of the entire workflow of TinyML.

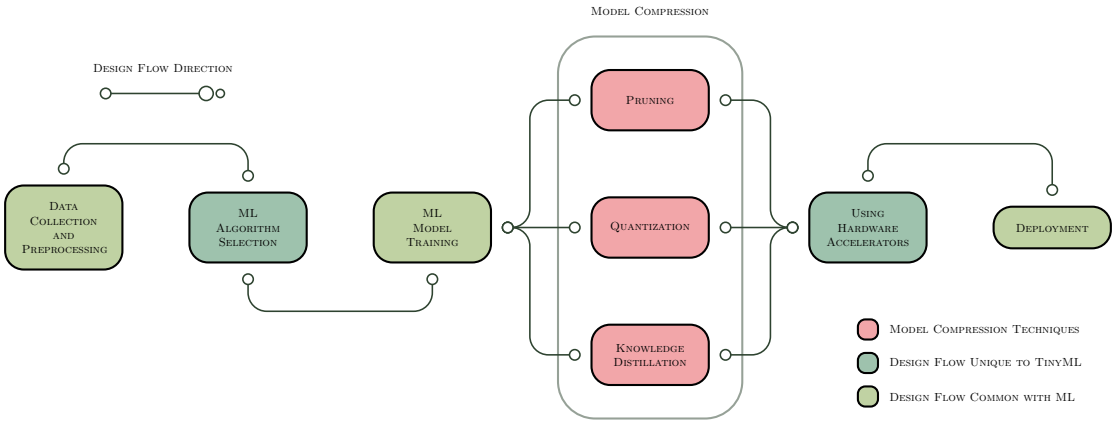

Figure 4.1: Design workflow of TinyML.

At first we need to collect data for the training. After preprocessing we can move on to picking an architecture[1]. Then we can train the model. Once we have the trained model, the core part of the TinyML workflow comes into play, *compressing* the model to fit into the constrains of the deploying device.

Once we have a compressed model we can deploy it to the target device. In some cases[2], certain devices will have hardware accelerators, so we can use them to gain more performance out of the device.

[1]For example, MobileNet, MCUNet, etc.

[2]As in the case of ESP32-S3, it has hardware AI accelerator.

4.2 Bread and Butter of TinyML Workflow

As we can see from figure 4.1, the core part of the workflow is the *compression techniques* we use. Let's see an overview of each of these techniques.

4.2.1 Quantization

One way to shrink the model size is *quantize* the coefficients in our question matrix. The coefficients usually will be 32 bit integers. We need to map them into 8 bit integers. This process is know as *Quantization*.

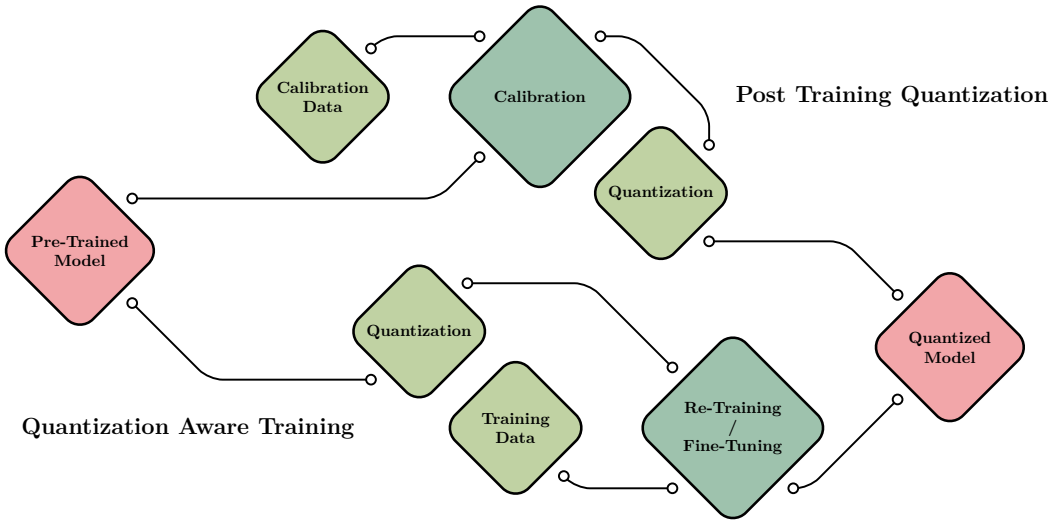

Figure 4.2: Two different ways to do Quantization.

Figure 4.2 show two ways to quantize our trained model. One is *Quantization Aware Training* and the other is *Post Training Quantization*.

Quantization Aware Training

If we have enough data, we can leverage it to tweak and tune the model after the quantization process. What we are doing is essentially re-training the model once more after the quantization. Therefore the name Quantization Aware Training. See figure 4.2 to get a better idea.

Post Training Quantization

In Post Training Quantization, we are simply quantizing after an initial calibration process. Even though this gets the job done, it is preferable to do QAT when ever possible.

4.2.2 Pruning

We can think of our ML model as a bunch of neurons connected to each other through different connections. As shown in figure 4.3, we can think of them having different layers, carrying out different tasks or looking for certain features in the data. But not all of these connections will have a significant influence in next layer. So we can cut these less influencing connection and still keep the performance almost the same. This process is know as *pruning*.

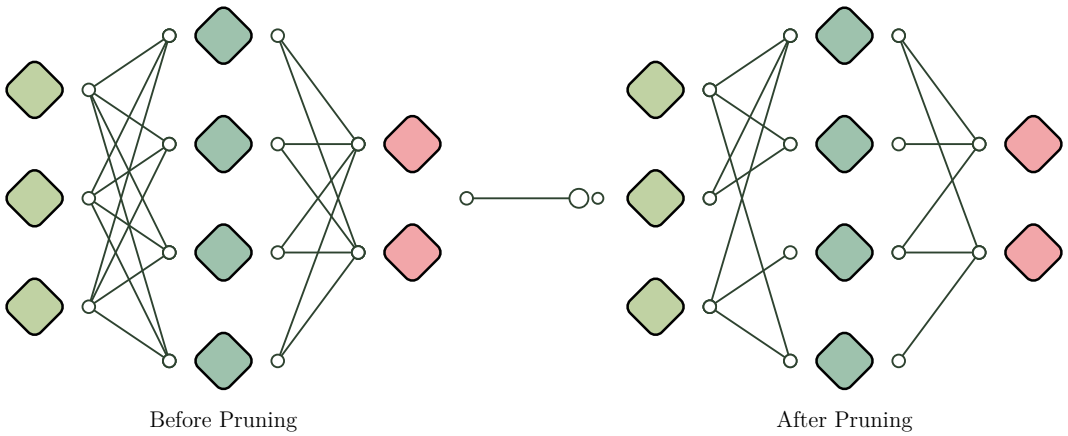

Figure 4.3: Pruning process in TinyML.

What we are essentially doing is zeroing out some of the coefficients in the question matrix that represents the connections. In short, we are generating a more *sparse* question matrix. And we can store these sparse matrices in a more compact way. Thereby decreasing the size of the model.

4.2.3 Knowledge Distillation

In this technique, we will train two models. One is know as the *teacher model* and the other is know as the *student model*. At first we will train the teacher model, then we will use this teacher model to *soft label* the data. Then we will use the hard and soft labelled data to train the student model.

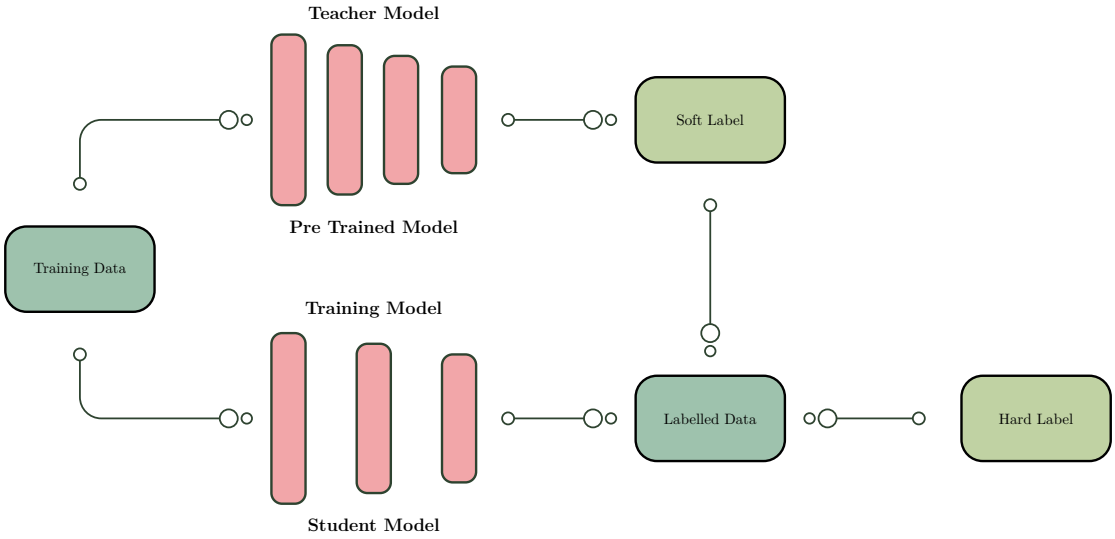

Figure 4.4: Technique of knowledge distillation in TinyML.

Figure 4.4 shows the exact process. This additional help from the teacher model helps the student model to fit to the data quicker. The student model will be much more performant with negligible decrease in accuracy than the teacher model.

4.3 Frameworks and Tool Chains

When it comes to the frameworks and toolchains used in TinyML development, two popular frameworks are widely used:

(a) (b)

Figure 4.5: (a) TensorFlow Lite. (b) Edge Impulse.

4.3.1 TensorFlow Lite

TensorFlow Lite is developed by Google to build and deploy ML models on resource constrained devices. It is an open source framework. And gives more fine grained control to the user.

4.3.2 Edge Impulse

Edge Impulse is an end-to-end development framework for TinyML. It offers a comprehensive set of tools for the entire embedded machine learning lifecycle. It simplifies the entire process of building edge AI solutions. And it makes TensorFlow Lite's capabilities approachable to new and novice users.

chapter 05

Application and Case Study

The benefits of TinyML make it desirable in several fields. This chapter will go through some of the general applications and look at some of the agriculture specific applications. At the end, we will look into some case study.

5.1 General Applications

As we have seen in chapter 3, TinyML has several advantages. And these advantages are desirable in several fields. So application of TinyML is also much wider. Table 5.1 lists several applications of TinyML.

Agriculture: Smart Farming
Industrial: Predictive Maintenance
Smart Homes: Gesture and Voice Control
Healthcare: Smart Wearables
Wildlife Conservation in Remote Areas
Automated Speech Recognition

Table 5.1: Table listing some of the general application of TinyML.

5.2 Agricultural Applications

Now let's see some of the applications that are agriculture specific. Table 5.2 list some of the agriculture specific applications of TinyML. We will see some case study related to these in the following sections.

Plant Disease Detection
Micro Climate Management
Pest Detection in Plants
Fruit Detection

Table 5.2: Table listing some of the agricultural application of TinyML.

5.3 Plant Disease Detection

Developed and field-tested a TinyML-based plant disease detection system for rural West African farmers, enabling offline, low-cost inference. While slightly less accurate than cloud solutions, it outperformed them in accessibility and usability within local constraints.

Figure 5.1

5.3.1 What They Did

In this study[1], researchers from Ghana develop a TinyML base plant disease detection system based on ESP32 CAM. This was aimed to help the farmers in Ghana to detect *Brown Streak* and *Mosaic* diseases in the Cassava[1] plant. They held a comparative study against the already available cloud based solution. Figure 5.1 shows a farmer in Berekuso using the TinyML system.

5.3.2 Results They Got

The TinyML based system gave a *promising performance* even though the cloud based Plantix[2] was better in terms of *accuracy*. But on an *economic bases*, TinyML based system was *more feasible* than the cloud based one because inorder for the farmers to use the Plantix app they need to afford a smartphone. Comparing to that the ESP32 based one was much cheaper.

[1]Which is a common crop in West Africa.
[2]Name of the app.

5.4 Micro Climate Management

Developed a TinyML-based greenhouse monitoring system that is deployable on most modern microcontrollers. Uing sensor data from a custom strawberry greenhouse and a five-action multi-label control strategy, trained and cross-validated 90 MLPs to identify the most efficient model. And resulted with a performant TinyML model.

Figure 5.2

5.4.1 What They Did

In this study[2], researchers developed an optimized tinyML-oriented model for an active machine learning based greenhouse microclimate management system to be integrated in an on-field microcontroller. Inorder to collect multivariate climate data, they designed and deployed an experimental strawberry greenhouse. Then the dataset is used to train and five-fold cross-validate 90 Multi Layer Perceptrons (MLPs) with varied hyperparameters and selected the most performant yet optimized model instance for the addressed task. Figure 5.2 shows the experimental greenhouse.

5.4.2 Results They Got

From their 90 model instances, they selected a model incorporated 2 hidden layers with 7 and 8 neurons respectively and 151 parameters, and it scored a mean accuracy of 97% during the cross-validation phase, then 96% on the supplementary test set. Thanks to the TinyML approach the developed model was a light weight one. So it can be effectively deployed on microcontrollers within real world operating conditions.

5.5 Pest Prevention

Developed a smart insect trap using the ESP-EYE microcontroller and FOMO algorithm, achieving 96% accuracy with low resource use. It enables real-time fruit fly detection, targeted pesticide application, and automatic trap replacement. This sustainable, hands-free solution improves crop protection and can adapt to other pests.

Figure 5.3

5.5.1 What They Did

In this study[3], Developed a TinyML based Fruit fly surveillance system, implemented using ESP-EYE mircrocontroller. They used FOMO (Faster Objects, More Objects) algorithm developed by the Edge Impulse platform.

5.5.2 Results They Got

The system performed with a maximum RAM usage of 2.4Mb and an inference time of 5.694 seconds. The FOMO model only required 53 kilo bytes of flash memory. Furthermore, the trap ran smoothly without human intervention, thanks to its automatic replacement feature for fly-stained adhesive traps

chapter 06

Conclusion

TinyML provides a way to bring ML to low power edge devices. This enables ML to be used in a plethora of embedded applications. And paves a way for the widespread adoption of edge computing. As we have seen, Agricultural Sector could greatly benefit from TinyML, as it encourages to use ML in *remote* and *resource constrained* systems in an *economically viable* manner.

TinyML is still in its early stages, with ongoing experimentation and exploration of its full potential. As research and development continue, advancements in techniques and tools will make TinyML increasingly powerful, efficient, and suitable for a wide range of real-world applications. With continued research and innovation, TinyML is set to become even more accessible and impactful in the years to come, enabling smarter and more connected systems at the edge.

Bibliography

[1] C. Ibegbu and G. A. Korsah, "Tinyml for the detection of plant diseases in resource-constrained areas within west africa," in *2024 IEEE 9th International Conference on Adaptive Science and Technology (ICAST)*, vol. 9, 2024, pp. 1–8. DOI: 10.1109/ICAST61769.2024.10856464.

[2] I. Ihoume, R. Tadili, N. Arbaoui, M. Benchrifa, A. Idrissi, and M. Daoudi, "Developing a multi-label tinyml machine learning model for an active and optimized greenhouse microclimate control from multivariate sensed data," *Artificial Intelligence in Agriculture*, vol. 6, pp. 129–137, 2022, ISSN: 2589-7217. DOI: https://doi.org/10.1016/j.aiia.2022.08.003.

[3] Q. M. Nguyen, V. T. Le, M. N. Lai, and H. B. Vo, "Resource-constrained intelligent trap: Fruit flies surveillance framework with tinyml integration," in *2024 Tenth International Conference on Communications and Electronics (ICCE)*, 2024, pp. 415–420. DOI: 10.1109/ICCE62051.2024.10634657.

[4] C. Nicolas, B. Naila, and R.-C. Amar, "Tinyml smart sensor for energy saving in internet of things precision agriculture platform," in *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2022, pp. 256–259. DOI: 10.1109/ICUFN55119.2022.9829675.

[5] D. A. N. Gookyi et al., "Tinyml for smart agriculture: Comparative analysis of tinyml platforms and practical deployment for maize leaf disease identification," *Smart Agricultural Technology*, vol. 8, p. 100 490, 2024, ISSN: 2772-3755. DOI: https://doi.org/10.1016/j.atech.2024.100490.

[6] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, *A survey of quantization methods for efficient neural network inference*, 2021. arXiv: 2103.13630 [cs.CV].

[7] S. A. R. Zaidi, A. M. Hayajneh, M. Hafeez, and Q. Z. Ahmed, "Unlocking edge intelligence through tiny machine learning (tinyml)," *IEEE Access*, vol. 10, pp. 100 867–100 877, 2022. DOI: 10.1109/ACCESS.2022.3207200.

[8] S. Namratha, R. Bhagya, and R. Bharthi, "Design and implementation of tiny ml model using stm32f platform," in *Trends in Sustainable Computing and Machine Intelligence*, S. Lanka, A. Sarasa-Cabezuelo, and A. Tugui, Eds., Singapore: Springer Nature Singapore, 2024, pp. 169–184, ISBN: 978-981-99-9436-6.