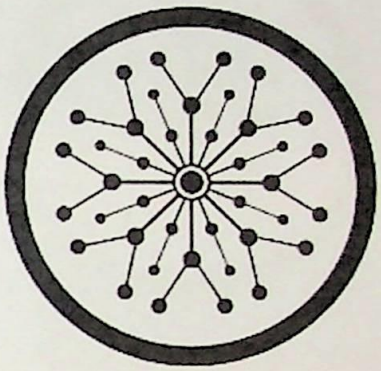
LIGHTWEIGHT CNN DEPLOYMENT ON ESP32-CAM FOR REAL TIME IMAGE CLASSIFICATION



SUPERIOR UNIVERSITY

MS ELECETRICAL ENGINEERING

By:

NAME: ABDUL HANNAN

ROLL NO: SU92-MSEEW-S24-001

SESSION: 2024-2026

DEPARTMENT OF ELECTRICAL ENGINEERING

SUPERIOR UNIVERSITY

LAHORE, PAKISTAN

LIGHTWEIGHT CNN DEPLOYMENT ON ESP32-CAM FOR REAL TIME IMAGE CLASSIFICATION

SUPERIOR UNIVERSITY

A thesis submitted in partial fulfillment of the requirements for the Degree of Master of Science in

Electrical Engineering

Submitted By

Abdul Hannan

SU92-MSEEW-S24-001

Session: 2024-2026

Supervised By:
Dr. Rao Muhammed Asif

DEPARTMENT OF ELECTRICAL ENGINEERING

SUPERIOR UNIVERSITY

Lahore, Pakistan

AUTHOR'S DECLARATION

I, Abdul Hannan a student of MS Electrical Engineering, at Faculty of Engineering and Technology, the Superior University, Lahore, hereby declare that this thesis titled, Lightweight CNN Deployment on ESP32-CAM for Real Time Image Classification is my own research work and has not been submitted, published, or printed elsewhere in Pakistan or abroad. Additionally, I will not use this thesis to obtain any degree other than the one stated above.

I fully understand that if my statement is found to be incorrect at any stage, including after the award of the degree, the university has the right to revoke my MS degree.

Signature of Student: _____

Name of Student: Abdul Hannan

Roll Number: SU92-MSEEW-S24-001

Date: _____

PLAGIARISM UNDERTAKING

I solemnly declare that the research work presented in this thesis titled, Lightweight CNN Deployment on ESP32-CAM for Real Time Image Classification is solely my research work, and that the entire thesis has been completed by me, with no significant contribution from any other person or institution. Any small contribution, wherever taken, has been duly acknowledged.

I understand the zero-tolerance policy of the HEC and Superior University, Lahore, towards plagiarism. Therefore, I, as the author of the above-titled thesis, declare that no portion of my thesis has been plagiarized and that all material used from other sources has been properly acknowledged, cited, and referenced.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis, even after the award of MS degree, the University reserves the right to revoke my degree, and that HEC and the University have the right to publish my name on the HEC/University website for submitting a plagiarized thesis.

Signature of Student: _____

Name of Student: Abdul Hannan

Signature Supervisor: _____

Supervisor Name: Dr. Rao M. Asi

CERTIFICATE OF RESEARCH COMPLETION

It is certified that this thesis titled, **Lightweight CNN Deployment on ESP32-CAM for Real Time Image Classification**, submitted by **Abdul Hannan**, Roll No. **SU92-MSEEW-S24-001**, for **MS Electrical Engineering** at **Faculty of Engineering & Technology**, Superior University, is an original research work and contains satisfactory material to be eligible for evaluation by the Examiners for the award of the above-stated degree.

No part of this dissertation has been submitted anywhere else for any other degree.

Supervisor's Signature: Dr. Rao Muhammed Asif

Supervisor's Name: _____

Designation:

CERTIFICATE OF APPROVAL

It is certified that the research work contained in this thesis, titled **Lightweight CNN Deployment on ESP32-CAM for Real Time Image Classification** was conducted by **Abdul Hannan** under the supervision of Dr. Rao M Asif and is adequate for the award of **MS Electrical Engineering**.

No part of this thesis has been submitted anywhere else to any other degree. This thesis is submitted to the **Faculty Of Engineering and Technology**, The Superior University, Lahore in partial fulfillment of the requirements for the degree.

Student Name: Abdul Hannan Signature: _____

Examination Committee:

a) **Session Chair**: Signature: _____ (Dean FET)

b) **External Examiner**
Name: Dr. Jahangir Arshad Signature: _____
Designation: Associate Professor
University Name:

c) **Internal Examiner**
Name: Dr. Manzor Ellahi Signature: _____
Designation: Associate Professor
University Name:

d) **Supervisor Name:** Dr. Rao M Asif Signature: _____

e) **Head of Department:** Dr. Adnan Yousaf Signature: _____

f) **Name of the Dean:** Dr. Muhammad Zafrullah Signature: _____ (Dean FET)

g) **COE:** Dr. Muhammad Haris Signature: _____

DECLARATION

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. I also declare that this work is the result of my own investigations, except where identified by references and free from plagiarism of the work of others.

ACKNOWLEDGEMENTS

In the name of Allah Almighty, the Most Beneficent, the Most Merciful. First and foremost, I would like to thank Almighty Allah for making all this possible for me and bestowing me with uncountable blessings. I express my sincere gratitude to **Dr. Rao Muhammad Asif** (Superior University), **Dr. Manzoor Elahi** (Superior University) for their invaluable leadership and inspiration throughout my academic tenure at Superior University.

Name: Abdul Hannan

Dated

Table of Contents

ABSTRACT 13
CHAPTER 1 INTRODUCTION 14
1.1 BACKGROUND 14
1.2 INTRODUCTION 15
1.3 PROBLEM STATEMENT 17
1.4 RESEARCH OBJECTIVE 18
1.5 SIGNIFICANCE OF THIS STUDY 19
1.6 SCOPE OF THIS STUDY 20
1.7 LIMITATIONS 21
1.8 RESEARCH ENVIRONMENT AND TOOL USED 22
1.9 ESP32-CAM PLATFORM 23
CHAPTER 2 LITERATURE REVIEWS 25
2.1 INTRODUCTION 25
2.2 EARLY DEVELOPMENT OF CNN ARCHITECTURES 25
2.3 ADVANCEMENT IN DEEP CNN MODELS 26
2.4 LIMITATIONS OF TRADITIONAL CNN MODELS 26
2.5 EMERGENCE OF LIGHTWEIGHT CNN ARCHITECTURES 26
2.6 DEPTHWISE SEPARABLE CONVOLUTION AND MOBILE ARCHITECTURES 27
2.7 EFFICIENT NETWORK DESIGN STRATEGIES 27
2.8 MODEL COMPRESSION AND OPTIMIZATION TECHNIQUES 27
2.9 LIGHTWEIGHT CNNS IN REAL-WORLD APPLICATIONS 28
2.10 RESEARCH GAP IDENTIFICATION 28
2.11 COMPARATIVE ANALYSIS OF EXISTING CNN MODELS 28
2.12 PERFORMANCE TRADE-OFFS IN CNN ARCHITECTURES 29
2.13 ROLE OF OPTIMIZATION TECHNIQUES IN MODEL EFFICIENCY 29
2.14 CNN APPLICATIONS IN REAL WORLD SYSTEMS 29
2.15 IDENTIFIED RESEARCH GAP 29
2.16 SUMMARY OF LITERATURE REVIEW 31
CHAPTER 3 METHODOLOGY 32
3.1 OVERVIEW OF RESEARCH METHODOLOGY 32
3.2 CUSTOM DATASET COLLECTION AND ORGANIZATION 33
3.3 CNN MODEL ARCHITECTURE AND TRAINING 36

3.3.1 OVERVIEW 36
3.3.2 CNN ARCHITECTURE 37
3.3.3 MATHEMATICAL FORMULATION 38
3.3.4 MODEL TRAINING 40
3.4 MODEL QUANTIZATION USING TENSORFLOW LITE INT8 40
3.5 LAPTOP WEB CAM INFERENCE 41
3.6 MODEL CONVERSION TO C ARRAY 42
3.7 ESP32 FIRMWARE IMPLEMENTATION 42
3.8 LIVE STREAMING PREDICTION ON ESP32 43
CHAPTER 4 RESULTS & DISCUSSIONS 45
4.1 EVALUATION MEASURES 45
4.2 RESULTS 46
4.3 K-FOLD CROSS VALIDATION RESULTS 48
4.4 CLASS WISE PERFORMANCE 49
4.5 DISCUSSION 49
CHAPTER 5 CONCLUSION 51
5.1 LIMITATIONS & FUTURE WORK 51
ACKNOWLEDGEMENT 52
CHAPTER 6 REFERENCE 53

Table of Figures

Figure 1-1: ESP32 S3 Microcontroller Configuration Details 23

Figure 1-2: VS Code PlatformIO framework 23

Figure 3-1: Overall Workflow of the Proposed Methodology 32

Figure 3-2: Custom Dataset Directory Structure 33

Figure 3-3: Image Preprocessing Pipeline Flowchart 34

Figure 3-4: Sample Preprocessed Grayscale Images 35

Figure 3-5: The workflow of the model training 36

Figure 3-6: Live Webcam Prediction Output with Zoom ROI (Region of Interest) 41

Figure 3-7: Git Bash Model Conversion Commands 42

Figure 3-8: ESP32 CNN Firmware Deployment & Execution Workflow 43

Figure 3-9: ESP Model Prediction Serial Monitor Logs 44

Figure 3-10: On ESP Model Prediction with TFT Live Streaming 44

Figure 4-1: Training & Validation Accuracy 46

Figure 4-2: Training & Validation Loss 46

Figure 4-3: Quantized CNN Comparison 47

Figure 4-4: Evaluation Metrics of Different Classes 48

List of Tables

Table 2-1: Comparison of Identified Research Gap 30

Table 3-1: Original dataset summary 33

Table 3-2: Preprocessed Dataset Summary 35

Table 3-3: CNN Architecture Summary 38

Table 4-1: Comparison Of Proposed Method With Existing Studies 47

Table 4-2: Model Evaluation Matrix 47

Table 4-3: K-Fold Validation Results 49

Table 4-4: Class Wise Performance 49

ABSTRACT

In recent times, there has been significant progress in intelligent systems and autonomous systems because of the advent of AI tools such as Chatbot and Large-Language models (LLM). The emergence of the need for such AI models and other applications has led to everyone being dependent on smart technologies. Computer vision, which is one of the areas where the use of artificial intelligence is prevalent, uses a lot of computational power; hence, it cannot be implemented using microcontrollers such as ESP32. As a way of showing how far the world of smart technologies has come using controller technologies such as ESP32, this study designed a model of multi-class image classification using an ESP32 microcontroller and achieving a reasonable performance rate in four classes that is, unknown class, person class, fruit class, and car class. For the purpose of training, the images were preprocessed to prevent unnecessary loss of computation. However, the best validation accuracy recorded in the model was 99.34±1% based on five-fold cross-validation with average validation accuracy of 98±1%. Furthermore, the model size was optimized using TF-Lite quantization which is a variation of INT8 quantization scheme. The converted models retained their features by converting them to C array format. In order to increase the accuracy of the model, the training data was provided with larger and more prominent images that had been filtered out from the training set. The ESP32 S3 N16R8 microcontroller together with on-board OV2640 camera were chosen for real-time image acquisition. A 3.2" TFT was also chosen to visualize the execution of computer vision tasks. The model operates perfectly by making accurate predictions with just a few samples per frame to minimize the response delay. The implementation of a multi-class image classification model on an ESP32 microcontroller represents a complete embedded vision system free of cloud computing requirements. Moreover, such a system deployment can be used to show the possibilities that edge computer vision has in terms of being used in an inexpensive intelligent system.

INDEX TERMS: ESP32, Multi-class Image Classification, Deep Learning, Convolutional Neural Network, Tensor flow Lite, Post Training Quantization

CHAPTER 1 INTRODUCTION

1.1 BACKGROUND

AI technologies have made tremendous advances in many fields such as healthcare, transportations, security systems, and industrial automation. The main reason for the advance of these AI fields is referred to as Deep Learning. Machines can discover complex patterns and features through Deep Learning. In Deep Learning, there is no need for handcrafted features because the machines automatically learn complex features from large datasets. Unlike traditional machine learning models, which depend on handcrafted features, Deep Learning learns features directly from input data [1]. Convolutional neural networks (CNN) have been recognized as one of the most effective methods compared to other types of deep learning. CNN is a unique model, which exploits convolution, pooling, and non linearities to exploit the spatial hierarchies in images. CNN consists of various layers in which each layer has a specific role. For instance, convolution layers are responsible for detecting features while pooling layers reduce dimensions. However, while deep layers in CNN can detect complex features such as objects and shapes, shallow layers can only detect lower level features such as textures and edges [2], [3]. Convolutional neural networks' success has been proven through their ability to perform tasks in relation to image classification on a large scale. The emergence of AlexNet network has led to a revolutionary breakthrough in CNN by demonstrating an amazing improvement in image classification performance as compared to other networks [4]. Below are some of the architectures such as VGGNet and ResNet that have made a great contribution to this domain through increasing the depth of the model by employing smaller convolution filters and implementing unique ideas. While the VGGNet applied the idea of smaller convolution filters and increased depth of the model for improved efficiency [5], ResNet employed the use of skip connections to solve the vanishing gradient problem in deep models [6]. The models demonstrated impressive accuracy in benchmarks and have become the mainstay of computer vision applications today. Despite their impressive performance, the models are highly complicated and require substantial memory usage, which makes it impractical to deploy them into practice. For instance, deep learning models such as VGGNet use millions of parameters and make the network bulky, thus complicating computations [7]. Moreover, the rising need for efficient and smart systems to work in real-time environments has brought into focus the limitations of using the classic CNN model. Self-driving cars, security systems, and healthcare systems over the Internet of Things need to function effectively while minimizing power usage and latency. However, the main issue with deep neural networks is that the computational costs are too high, resulting in higher inference times and making them not efficient enough [8],[9]. As such, there arises the necessity to design an effective architecture which can provide high accuracy while minimizing computation costs. There have been some efforts done for this purpose to optimize accuracy, efficiency, and performance. Lightweight CNN architectures are a good solution for this trade-off. In order to optimize this balance, there have been several techniques used such as model compression, pruning, quantization, and architectural efficiency. An extremely effective method of designing small sized CNN's is through use of depth-wise separable convolution layers, whereby there is huge reduction in the number of computations and parameters involved when compared to regular convolutions. The approach of depth-wise separable convolutions is widely applied in models such as the Mobilenet that was specifically designed for mobile/embedded vision [10]. Mobilenet is an extremely efficient approach in its design due to its use of depth-wise and

pointwise convolutions as opposed to regular convolutions. Another efficient architecture approach used for designing small size models is known as SqueezeNet, whereby the model achieves its efficiency by reducing the number of parameters through fire modules, which consists of a squeeze and expand layer [11]. The SqueezeNet architecture delivers comparable performance as AlexNet with far fewer parameters, indicating that efficient architecture design could be an alternative for designing compact and powerful CNNs. Apart from efficient architecture design, pruning and quantization approaches are also widely used techniques to achieve lightweight models from deep neural networks. The former technique involves pruning out superfluous nodes or connections from the architecture to shrink the size of the model whereas the latter involves shrinking the size of model parameters [12]. The rising demand for edge computing has raised interest in lightweight deep learning approaches. This is because edge devices such as mobile phones, watches, and other devices would require light deep learning models that can run inference locally on the device itself without the need for any cloud computing. Not only would that lower latency times but also raise privacy issues. Therefore, lightweight CNNs remain a vital topic in today's research world. One more key feature of lightweight algorithms is their efficiency. Even though the creation of small and efficient networks is important, the problem of ensuring accurate results should be taken into account as well. This could be achieved by correct algorithm design and hyperparameter tuning [13]. Finally, to conclude, the invention of CNNs proved to be groundbreaking in terms of image recognition and computer vision due to the extremely high precision that could be achieved through their utilization. At the same time, their implementation involves the utilization of huge amounts of computing and memory power which makes them unsuitable for usage in situations with limited resources available. This problem can be solved by developing lightweight versions of CNNs, and this is what this project is about.

1.2 INTRODUCTION

Because of the widespread usage of intelligent devices in numerous technical domains, the relevance of the efficient implementation of machine learning systems has become apparent. For example, owing to progress in Computer Vision, automatic devices have become able to analyze visual images with high precision. The success can be explained by the advances in Deep Learning field, according to which Deep Learning techniques excel all others in their ability to recognize patterns [14]. Because of the increasing amount of visual information, it is becoming especially important to create effective machine learning classifiers [14]. However, there are some limitations regarding the functionality of such models that need to be taken into consideration while implementing modern technologies. They involve the absence of computing capabilities, lack of storage capacity, and time-related limitations. The situation has become especially pressing due to the emergence of edge computing technology that implies analyzing data locally without using cloud services. The usage of intelligent technologies based on edge computing principles has resulted in changes in the process of developing ML models where efficiency prevails over effectiveness [15]. The deployment of conventional deep learning methods resulted in a remarkable success of machine learning models in classification due to the use of extensive neural networks that use extensive training sets. However, such algorithms suffer from extremely high computational complexity and a great number of parameters, which makes them inefficient in their usage. The deployment of these models in the edge environment brings up such challenges as high inferencing delay, energy consumption, and more. Thus, the need for developing lightweight algorithms that would

combine efficiency and performance arises [16]. There are promising advancements made in terms of model optimization approaches. Among the most popular approaches to solving the above issue are knowledge distillation, low-rank factorization, and neural architecture search. Knowledge distillation is the technique of extracting the knowledge from a larger and more complex network into a smaller one so as to maintain the latter's performance but reduce its computational overhead [17]. On the other hand, Neural Architecture Search provides a way of automating efficient architectures by exploring many different architectures. In addition to that, another thing that should be considered during model design is designing architectures that are small enough and do not contain any kind of redundancy. For example, the introduction of channel-based operations and grouped convolutions increases the efficiency of feature extraction. Shuffle net is one of the architectures that make use of channel shuffling operation in order to facilitate the passage of data between channels with low computational expenses [18]. On the other hand, efficient net takes advantage of a new approach called compound scaling whereby all parameters are scaled equally leading to improved performance with fewer number of parameters [19]. These approaches have made it possible for efficient and light-weight CNNs to be designed that can be deployed in practice. However, designing such efficient models is a challenging task considering the many aspects that must be considered during their creation. As an illustration, scaling down the model means having to lose the representational ability of the model, and this implies that the ability of the model to perform classification of data will be impaired. Therefore, establishing the perfect balance among the various elements under consideration requires a lot of information about building and analyzing models [20]. Secondly, the diverse applications imply that the models used must be able to tackle any problems arising. For instance, in the medical sector, such models are applied in disease diagnosis, meaning that the aspect of accuracy becomes more important. In the case of real-time surveillance, efficiency becomes vital as decisions need to be made [21]. The development of the IoT world has contributed to a rising need for robust machine learning algorithms. There should be mechanisms that allow for the management of the constraints of hardware to implement deep learning algorithms. Lightweight CNNs are some of the best ways to handle these kinds of challenges. It allows for computation of processes using devices only, ensuring safety, and eliminating delays in communication [22]. Due to the rise in the deployment of these models, the energy efficiency that comes from deep learning has started being seen as a problem. Efficient models not only ensure computational efficiency of the model but also its sustainability as they do not consume much energy. Energy efficiency is crucial especially for battery-powered systems as the energy efficiency factor plays a vital role in making them practical and sustainable [23]. Taking all the above into consideration, the requirement of developing a lightweight CNN classifier is considered as a significant step towards sustainable implementation of AI technology. Not only will the lightweight CNN model be an efficient model in terms of computation and energy usage, but also should match the performance of current efficient models. Furthermore, evaluation of such lightweight models consists of several metrics, including accuracy, precision, recall, computation cost, and inference time. An all-round evaluation system is essential in evaluating the efficiency of the lightweight model. This research attempts to enhance the state-of-the-art techniques of efficient deep learning through the discussion of the above-listed aspects. In conclusion, the change from conventional and compute-intensive deep learning models to lightweight deep learning models is an indication of the changing demands of modern AI technology. It is essential for AI technology to be capable of classifying data with speed and accuracy. Thus, the purpose of

this research is to develop a lightweight CNN model that will strike a balance between both computational efficiency and accurate classification.

1.3 PROBLEM STATEMENT

While there has been great progress made in the area of Deep Learning and while the CNN models have shown remarkable efficiency in performing visual recognition tasks, the problem of balancing between accuracy and computational complexity still remains. In fact, while the latest version of CNN models proved themselves to be outstanding classifiers, the increase in accuracy comes with higher complexity and parameter number, making greater computational costs necessary. This issue represents a major obstacle towards implementing these algorithms in real-life scenarios since, with the current state of deep learning models, they feature rather deep and wide architectures. While this increases their accuracy, it also leads to an increase in computation steps, energy consumption, and memory size. Therefore, deploying these models becomes rather challenging on resource-constrained devices such as smartphones, embedded systems, and edge servers [24]. Another challenge associated with this approach relates to energy consumption required to implement deep neural networks. As the complexity of the model grows, its energy consumption increases as well. Such a consumption of energy is a threat not only to battery-operated devices but also to low-energy ones, as the consumed energy will affect the operational life of those devices [25]. The energy spent by deep neural networks can be harmful to operational life in other kinds of systems as well. Real-time applications should work without any failure because they need reliability. Many contemporary models of CNN do not provide fast processing, especially in terms of real-time applications. In real-time applications, even tiny latencies lead to bad performance. Sometimes, there appear cases of possible safety problems. In an autonomous driving application, it needs to have fast processing in both object detection and object recognition tasks to guarantee safe driving [26]. One more drawback of classical CNN is that they are poorly portable, and it is difficult to adapt them to various conditions of operation. Generally speaking, effective models are usually trained on GPUs within the cloud system. However, after transferring the models to edge devices, they should be somehow transformed, and it affects either accuracy or stability. Finally, yet another relevant issue occurs due to increased complexity and diversity of datasets that require solving. It is crucial to process significant amounts of data as well as high-resolution images along with multi classifications. Classical Convolutional Neural Networks tend to over fit during processing vast amounts of data when optimized. Therefore, it is critical to build efficient architecture capable of handling any volume of data [28]. Some researchers have tried tackling this problem by optimizing classical models using different methods such as pruning, quantization, and knowledge distillation. All of these solutions helped to decrease parameters and made models lightweight and efficient to implement. Yet, it was still necessary to train the models appropriately, making them impractical to use [29]. Another obstacle that may appear is the trade-off between accuracy and efficiency. One must not overlook the fact that light-weighted architecture seeks the lowest number of parameters and minimal computational complexity. However, it always implies the reduction of representational capabilities of the architecture, which cannot recognize particular features of the input. The optimization of both of these factors is one of the most significant challenges in current deep learning research [30]. Yet another problem related to the rapid growth of the Internet of Things ecosystem is the necessity of an efficient learning algorithm. With certain limitations imposed on the computational, memory, and energy abilities of the IoT hardware, it would be unreasonable to

implement traditional deep learning techniques. Thus, it is important to discuss the role of light-weighted convolutional neural networks while performing edge rather than cloud computing [31]. Scalability has also been a challenge that has been contributing to this problem area. As the complexity of applications rises, data sample sizes will continue to grow. Thus, one can say that we need to be able to create models that can scale in their capabilities without the increase in computation costs. Traditional convolutional neural network architectures are known for their limited scalability [32]. Also, other factors, such as those related to the interpretability and deploy ability of the models, make their development challenging. In areas such as healthcare and security, apart from accuracy, the ability of these models to function properly requires them to perform on-device processing. However, big models have problems with doing so [33]. Lastly, one of the aspects that deserve mentioning in terms of CNN models is their environmental impact. Since these models are computationally expensive both for their training and deployment, the latter becomes one of the key sources of carbon emissions [34]. In summary, despite the level of success that has been achieved by the CNN based deep learning algorithms in their classifying tasks, there are still some issues that remain unattended to and pose various problems to the algorithms. Some of these problems include high computational complexity, high memory consumption, low energy efficiency, lack of real-time capability, poor scalability, and lack of resource-constrained environment adaptability [35], [36]. Although the techniques developed for overcoming the above challenges to some extent solve the issues raised, they, on the other hand, introduce some other problems or influence negatively the operation accuracy. As a result, it becomes necessary to develop a lightweight algorithm for classification using CNN algorithms.

1.4 RESEARCH OBJECTIVE

The key objective of this research work is to design an efficient and compact convolutional neural network architecture for image classification tasks that strike the best balance between efficiency and effectiveness. In order to achieve this objective, the following sub-objectives have been formulated:

- To design an efficient lightweight CNN model for image classification tasks. In this case, the main concern will be to design an efficient deep learning architecture from a complexity point of view. This will be accomplished using an efficient convolution technique to reduce the redundancy and enhance the feature extraction abilities of the model.
- To make the proposed architecture more efficient in terms of computing and parameter requirements. To train the proposed architecture efficiently, we need to make sure that the number of the parameters, float-point operations, as well as the amount of memory required during the training process will be reduced without affecting the extraction capabilities of the model.
- To speed up the prediction process using the designed architecture. This can be done by making sure that the proposed network has the ability to provide results in real-time, which means that the latency times will be reduced.
- To design an efficient and accurate model. This will involve the design of a model that is efficient but has an accuracy level equal to that of other advanced CNN models.
- To analyze the performance of the proposed model using conventional performance metrics. The proposed model will be analyzed based on the performance parameters such as accuracy, precision, recall, F1 score, model size, and computation time. These

factors will be considered to analyze both the performance and efficiency of the proposed model.

- To benchmark the proposed model against the current state-of-the-art CNN models. In order to benchmark the efficiency and performance of the proposed model, we will need to compare its performance against existing CNN models.
- To validate the applicability of the proposed model in practical scenarios. Finally, we will need to validate the proposed model to prove its practical viability in real-world use cases such as edge computing systems, IoT, and mobile applications.

1.5 SIGNIFICANCE OF THIS STUDY

As far as the rapid growth in intelligent systems is concerned, deep learning can undoubtedly be considered one of the important technologies used nowadays in computing solutions. Among the systems introduced, the ones using the CNN architecture have shown exceptional performance in terms of image recognition. Despite the fact that the models of this type perform quite well, classic CNN architectures tend to be expensive and complicated to implement. Therefore, there appears to be an urgent need for lightweight models that would combine accuracy and efficiency. The actual relevance of the study derives from the efforts to satisfy the above-listed criteria through introducing a new lightweight CNN-based classifier that would ensure the balance between costs and prediction accuracy [37]. Some of the important characteristics highlighted in the above study include the usefulness of this technology as far as edge computing and mobile intelligence are concerned. Since IoT is being widely used nowadays, there is a need to create intelligent models that do not need so much computation once they have been downloaded onto the local device. For instance, deep learning techniques use GPU and therefore cannot be adopted everywhere. However, with lightweight CNN model, it becomes possible to do computations locally. This helps increase the responsiveness of the whole network in addition to making it reliable and secure [38]. Furthermore, the findings from the study will be useful in enhancing the capability of making decisions in real-time decision-making system. Autonomous vehicles, surveillance systems, factories, and smart healthcare applications among others require decision making within fractions of a second. When the process of making inference takes too much time, the whole decision-making process may fail [39]. In addition to the above mentioned, another important aspect of this paper is that it makes a significant contribution towards resource optimization in the field of deep learning. Because huge amounts of parameters are normally used in deep learning, huge amounts of storage capacity and electrical energy would be required. It might especially be problematic in battery-operated and portable devices. Therefore, having a more effective system with less parameterization would lead to the development of sustainable AI systems. The importance of the creation of energy-efficient AI systems is becoming an important issue due to the proliferation of this technology globally, and this paper makes a significant contribution towards this end [40], [41]. Finally, this paper has academic importance for the future development and innovation of models. Efficient models normally serve as the basis for the future innovation of models in this field, and therefore it is essential to understand their relationship with effectiveness [42]. It becomes evident just how crucial the use of lightweight CNN models is within the healthcare industry. There are many imaging machines which require timely analysis using massive amounts of data consisting of x-ray images, MRI images, CT scans, and other types of data. However, it may not be practical to use heavy models under these circumstances due to the limitation of the computing capacity present within these

healthcare facilities. Lightweight CNN models are key to enabling timely and efficient analysis of data, which would ultimately contribute towards improving patient treatment [43]. Similarly, lightweight CNN models are key to ensuring that object recognition and object classification tasks carried out by surveillance systems are effective. This is especially important within surveillance systems since there may be delays due to the heavy computing requirements [44]. Further, the value of this study pertains to industrial automation and intelligent manufacturing system technologies. In these instances, learning-based models are implemented within the robot and automatic inspection system. Quick response in relation to classification is crucial in order to provide efficient operation. Lightweight CNNs may help in fulfilling this requirement in that they allow efficient implementation of calculations in the embedded systems of industries [45]. From the point of view of technology, the current study pertains to efficient deep learning technologies. Currently, there is much focus among researchers on implementing low-cost yet highly accurate neural network models. Efficient convolution, optimization, etc. are some of the trends pertaining to the development of deep learning models [46]. Moreover, this study helps to improve the availability of AI technologies. Highly efficient AI technologies require high amounts of computer processing power, thus making them only feasible to large corporations and firms having sufficient computer resources. Through lightweight architecture, however, AI technologies would be accessible to smaller firms, research laboratories, and other less economically developed regions lacking sufficient computer capabilities. At the same time, such democratization of technologies will foster innovations in this sphere as well [47]. Last but not least, this proposed research will be able to create a base for the future scalable and adaptable use of AI. Due to the constant emergence of complex data, it is getting more necessary to develop such systems that could function effectively while using minimal computer resources. Lightweight CNN can be considered a platform to develop such systems. To sum up, the significance of this research lies in creating effective DL systems, which would consume minimum computer resources.

1.6 SCOPE OF THIS STUDY

The limitations of this research include the limitation that the research will only cover the development of a lightweight CNN model used in image classification. The research seeks to solve the issue of the computational inefficiency of the current models while still achieving a high degree of accuracy in prediction. This research focuses mainly on improving the architecture of the CNN model as opposed to creating an entirely new algorithm. Since this research focuses on CNNs and image classification, this means that no other type of deep learning model will be considered besides the recurrent neural networks and transformers. This model is intended to be applied in scenarios where there are limited resources, which may include mobile phones, embedded devices, and edge computing. Hence, this project will study how effective this model can be while being under constraints such as computing power, memory, and energy. Distributed training systems and high-performance computing systems, however, are not within the scope of this project. In terms of data, this project shall employ publicly accessible datasets that have been employed in research projects focusing on computer vision and image recognition. Such datasets shall be employed in the training, validation, and testing stages of the model. The datasets that are to be employed for this project include neither proprietary datasets nor any real-time data acquisition systems, except for those necessary for experimentation purposes only. Architecture-wise, the scope of this project is focused on lightweight network architecture via parameter reduction techniques, effective convolution

techniques, and feature extraction techniques. Methods that can be applied after training the model, such as pruning and quantization, are excluded from the scope. In this case, the goal is not to make compression post training but build the effective CNN structure itself. In this regard, performance evaluation measures to be considered during this research include accuracy, precision, recall, F1-score, inference time, and trained models size. Furthermore, computational efficiency will be assessed through floating point operations number and memory utilization. On the other hand, there will be no optimization or design work regarding hardware implementation. Thus, neither FPGA nor ASIC realization of any of the created CNNs will be made. From the other perspective, regarding the field of application, in the current research only applications related to image classification will be studied. Despite the wide usability of convolutional neural networks in tasks like object detection, image segmentation, or video data processing, none of these will be considered. However, when speaking about the real-world applications of the proposed CNNs, there will be several types of practical solutions under investigation. This includes surveillance-related software solutions, mobile image classification systems, healthcare imaging applications, and intelligence products based on IoT infrastructure. On the contrary, however, the use and implementation of such a model would be beyond the scope of academic investigation in this instance. Likewise, this research would be conducted with the constraint of limited computational capacity when it comes to modeling and testing models. To conclude, this research mainly revolves around the development of a computationally efficient CNN classification model. This research would not seek to look into all the possible approaches for optimization, since this could be done with deep learning techniques. Rather, the research seeks to build a computationally efficient model.

1.7 LIMITATIONS

Even though the primary objective of the proposed study was the development of an efficient and light-weighted convolutional neural network model for image classification characterized by low computational complexity, some limitations have been observed, which are necessary to address. The aforementioned limitation indicates the degree to which the research findings could be generalized and utilized. Firstly, it should be acknowledged that one of the key limitations related to the study pertains to its scope limited to the application of convolutional neural networks to the problem of classification rather than other tasks such as detection and segmentation, both of which also could be accomplished with the use of convolutional neural networks. Secondly, the limitation pertains to the issue of data set dependence because the performance of the new algorithm of the light-weighted convolutional neural network was evaluated using benchmark data sets. Even though the data sets are likely to guarantee accurate and unbiased results of testing the algorithm's performance, there might still be issues associated with the diversity of the environment and possible noise or other factors. Additionally, the use of limited computational resources to perform analysis and assessment of the model could be thought of as another limitation. Normal PCs were used to carry out the analysis, and since the limitations of hardware in these PCs made it difficult to delve deeper into the study of very deep networks, as well as hyper-parameter optimization. In addition to architectural designing, there are several other means by which models could be optimized for deep learning algorithms. However, due to limitations in time and computational power, the present study only relied on the architectural designing of models. Another disadvantage is the intrinsic contradiction between efficiency and representational ability that may occur in lightweight models. As the architecture is optimized to be more efficient, there could be a

possibility of reduced representational ability because of parameter reduction and computation reduction. It implies that the model under discussion would fail to learn complex information in comparison with more sophisticated architectures and models. Moreover, the criteria adopted in this research paper are mainly concerned with the accuracy and computational efficiency of the model's architecture, speed, and flops per second. The question of model robustness in case of any attack is not discussed. Furthermore, the deployment of this model at the hardware level, as well as how it performs on GPUs, TPUs, or edge-based AI systems, is beyond the scope of the paper. The next limitation in this paper is that there are no tests that show how the model works in terms of real-time dynamics. This means that while evaluation was done via experiments using controlled situations, they fail to take into account the unpredictable nature of actual conditions, including different lighting conditions, motion blur, and sensor noise. Finally, one of the limitations of the paper is that its focus is narrow as it centers on one specific field within architecture. Though the paper compared their lightweight CNN model to other existing models, they did not compare it to transformer models and similar deep learning architectures for computer vision applications since these fell outside the scope of the study. Overall, while the paper provides a comprehensive way of building a lightweight CNN classifier model, it has a number of limitations as well.

1.8 RESEARCH ENVIRONMENT AND TOOL USED

To conduct an experiment, a research environment was created based on Anaconda Navigator; therefore, the environment was fully configured for machine learning experiments to be conducted in it. Another python environment was created by means of Anaconda prompt as this environment was needed for proper management and configuration of libraries and dependencies. All steps of the research were conducted inside Jupyter Notebook because it was a primary tool of this research. In the process of conducting an experiment, all dataset manipulation operations, such as loading, examining, visualizing, preprocessing, creating a unique architecture for a CNN, training, testing, and converting the model to production state were performed inside Jupyter Notebook. Using the step-by-step procedure of running cells helped to control the entire process of training the model and quickly adjust parameters. The process included image resizing, normalization, one-hot encoding for multiclassification problems, dealing with imbalance in the dataset. The neural network model was created entirely from scratch without any pre-trained networks. Floating-point arithmetic was applied in the training process in order to ensure the stability of the calculations and use gradient-based optimization methods. Training of the model was thoroughly analyzed by studying loss curves and probabilities for multiple outputs.

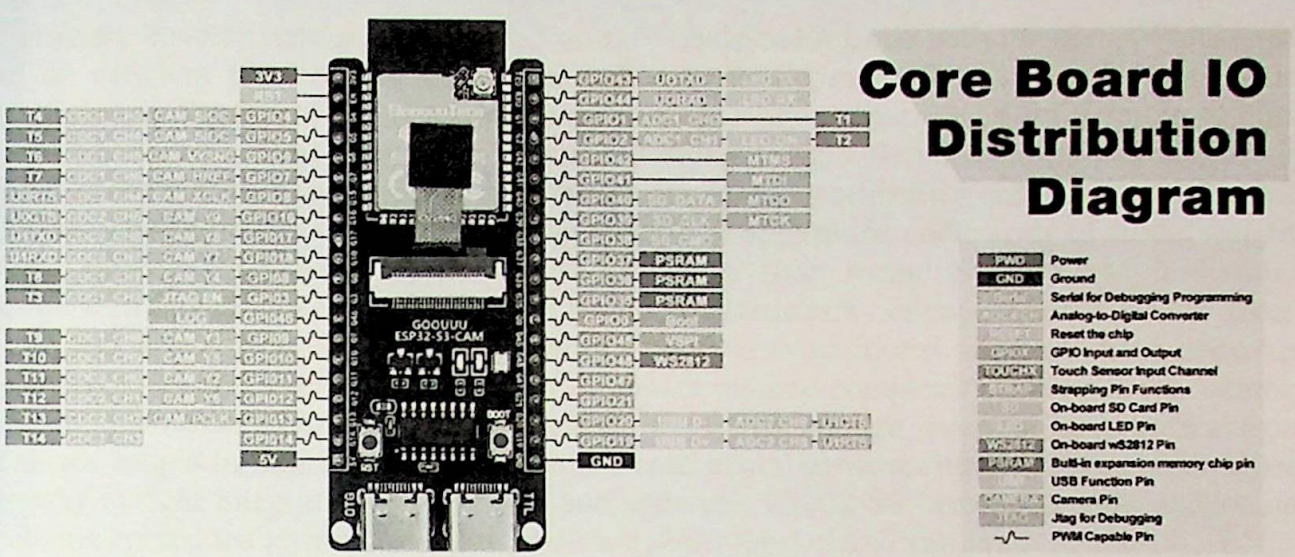


Figure 1-1: ESP32 S3 Microcontroller Configuration Details

After successful training of the model, the model was converted to the INT8 TFLite format because it adhered to the memory constraints of the hardware system used for the embedding process. The conversion to the INT8 format was done by performing the full integer quantization that helps reduce the size and speed up the execution of the model. After performing full integer quantization of the tensor flow lite model, conversion of the model was done to a C model data array through the binary to header technique. This makes it easy to embed the model without using any secondary memory.

1.9 ESP32-CAM PLATFORM

The embedding and assessment of the proposed model was done with the use of ESP32-S3 N16R8 Development Board having the ability to connect the camera module along with increased memory availability: 16 MB of flash memory and 8 MB of external SPI RAM. In spite of increased resources of the board, it can still be regarded as a good example of a resource-limited device and thus be considered as a proper means of evaluating Tiny ML models. Hardware constraints of the ESP32-S3 have significantly influenced the process of development of the model, which includes simplified architecture of the network, reduced input image size, and full integer quantization.



Figure 1-2: VS Code PlatformIO framework

Firmware development was performed using Visual Studio Code with the use of PlatformIO as an efficient platform for embedded software development. Thanks to the capabilities provided by PlatformIO, it was possible to easily manage source files, configuration of build process, memory management and firmware flashing process. Serial terminal was extensively used as an interface to monitor status of the system, memory utilization and results of inference. Memory partition was carefully chosen in order to fit application code, array of model weights and inference memory into the ESP32-S3 device flash memory and RAM limitations. Quantized model was embedded into firmware as a static array of data. Thanks to that, access to model data during inference was performed with zero additional overhead. Such approach allowed to ensure higher reliability and minimize runtime complexity. Methods for memory allocations were carefully considered in order to provide enough space for input buffers from camera, output buffers from inference engines and neural networks models. As a consequence, thanks to tight integration of hardware and software, ESP32-S3 became an independent and reliable embedded vision system for real-time multi-label classification.

CHAPTER 2 LITERATURE REVIEWS

2.1 INTRODUCTION

One of the factors, which contributes to the development of intelligent technologies, is the advancement in the field of artificial intelligence. An example of the development, which took place in this field, can be considered the idea of deep learning approach. It is worth noting that there exist numerous different kinds of algorithms, which can be used during the process of deep learning. For example, it can involve using of the algorithm of convolutional neural network, which is able to produce hierarchical feature representations by itself without manual engineering. This factor was triggered due to the emergence of computing resources, large amount of databases, and other technological improvements. Initially, machine learning involved using of edge detectors, histograms, and scale-invariant features, thus making it possible to extract necessary features by hand. However, this method was characterized by certain limitations since the algorithms, mentioned above, lacked generalization capability [49]. With time, a lot of progress was made regarding CNN architecture that evolved from simpler architectures to more complex and deeper ones. Most improvements in terms of accuracy were obtained by improving problem-solving abilities for complicated benchmarking datasets. However, these improvements have negative sides because increases in accuracy were accompanied by greater computational complexity and higher energy costs. Thus, the interest in optimizing CNN architectures has grown significantly in order to improve their efficiency [50]. In the recent period of time, there has been a noticeable tendency towards creating efficient and compact architectures for deep learning systems that could be used under certain constraints of computational power. These constraints occur in case of using mobile devices, embedded computing, edge computing, and other technologies. It means that the efficient and compact CNNs should involve fewer computational resources while being able to provide sufficient levels of accuracy [51]. The following literature review will analyze CNN architecture, classical deep learning models, and efficient neural networks.

2.2 EARLY DEVELOPMENT OF CNN ARCHITECTURES

Techniques of CNN-based image classifications have come a long way since the time of neural network architectures designed to perform certain tasks in pattern recognition. LeNet-5 is one of the first neural networks that was designed specifically for handwritten digit recognition. This network illustrated the importance of applying convolution operations for extracting spatial features. The model consisted of several alternating convolution and pooling layers, which then went through fully connected layers that classified the dataset. While it was a simple neural network design, it helped form the basis of the current architecture, which employs principles of local receptive field and weight sharing [52]. Despite some great advances, CNNs could not receive adequate attention due to poor hardware support as well as lack of sufficient training sets [52]. However, the advent of improved hardware components and bigger data sets made CNNs more recognized in the realm of computer vision [49, 52]. Therefore, deep CNNs became relevant because they could solve complex tasks of image recognition. Such an outstanding accomplishment occurred when AlexNet was developed and offered a record result in ImageNet benchmark [52]. It brought several useful concepts like ReLU activation, dropout, and GPU training [53]. The success of AlexNet demonstrated the efficiency of deep CNNs,

which resulted in the appearance of many researches in this area. Later, the models began paying more attention to the architecture and depth of neural network.

2.3 ADVANCEMENT IN DEEP CNN MODELS

Following the development of early deep learning models, it became necessary to develop models that were accurate yet with high capability of feature extraction. The development of VGGNet brought about an innovative architecture that enabled one to increase the depth of the model through the use of small 3×3 filters. It became evident from the VGGNet model that deep architecture helps to enhance classification task performance. Regrettably, the VGGNet model suffered from a limitation as well. The model depth increment resulted into an exponential growth in terms of model parameters and hence making the model hard to be trained due to the high memory requirements. The challenge of training deep models led to ResNet coming up with the concept of residual learning that allowed for easy gradient propagation in very deep neural networks [56]. The other major breakthrough in the development of the CNN models was the development of the inception type networks, where the concept of scale was used. It is based on using several sized kernels for convolution within a layer. This technique helped in capturing features from different spaces while not adding significantly to computation. Nevertheless, even with all these advancements, the problem remained with the computational complexity of the deep learning neural network. With improvements in accuracy, the size of neural networks became very large.

2.4 LIMITATIONS OF TRADITIONAL CNN MODELS

Even though deep CNNs have been successful in solving image recognition tasks, there are certain disadvantages in using deep CNNs. Deep CNNs have been found to be computationally complex as they involve lots of parameters and computations during training and testing of deep CNN models, hence making them inefficient in application in systems where computations cannot be adequately performed [58]. Memory requirement can also be viewed as an important constraint when deploying deep learning models due to their bulky nature which requires large memory spaces. Energy requirement becomes another important problem in deep learning as it involves huge energy consumption during training and application, thereby making them inefficient in battery-operated and environmental friendly applications [59]. Traditional CNNs also suffer from slow operation speed which makes them inefficient for use in applications such as autonomous driving and monitoring of sensitive activities since any form of latency can make the entire process fail. Therefore, it becomes important to seek out other solutions which are both efficient and accurate.

2.5 EMERGENCE OF LIGHTWEIGHT CNN ARCHITECTURES

Due to the growing computational cost that comes alongside deep learning models, there has been considerable interest in building different architectures. In light of this, lightweight neural networks, which mainly seek to simplify the complexity of the model while maintaining its efficacy, have gained more popularity recently. Lightweight neural networks perform a very critical function in enabling deployment of models in the context of edge computing because they require few resources such as computation, memory, and energy in order to obtain the best results. While traditional deep learning models aim at making computations more complicated in the convolutional layer, lightweight neural networks try to make the process less complex [60], [61]. One important concept in light-weight CNNs involves the minimization of redundancy within the convolution operation. This concept arises from the fact that standard

convolutions involve excessive computations because of the number of parameters used as well as the number of multiplications and accumulations. Hence, for such difficulties to be tackled, new ways of performing convolutions have been developed, whose effect on accuracy is minimal.

2.6 DEPTHWISE SEPARABLE CONVOLUTION AND MOBILE ARCHITECTURES

Depthwise separable convolution is another significant advancement achieved through lightweight CNNs where computation complexity is significantly reduced. Depthwise separable convolution forms the basis of MobileNet, a popular CNN that is tailored for mobile and embedded devices. In MobileNet, the convolution process is decomposed into depthwise convolution and pointwise convolution [62]. Different variations of the MobileNet architecture have been created. An innovative architecture for MobileNet version 2 featured the usage of inverted residual and linear bottlenecking that enhanced efficiency and efficient feature extraction [63]. Such innovations made MobileNet more efficient in real-time applications such as object detection, face recognition, and mobile vision [64]. Another lightweight neural network architecture is SqueezeNet, which features the use of fire modules. The architecture incorporates squeeze and expand layers to reduce dimensions and keep the same representation capability, respectively. SqueezeNet had comparable performance to AlexNet regarding accuracy despite the latter being larger than the former [65].

2.7 EFFICIENT NETWORK DESIGN STRATEGIES

Apart from MobileNets and SqueezeNets, several other CNN efficient neural networks have been proposed to maximize the efficiency of the Convolutional Neural Networks. One of them is the ShuffleNet, which utilizes channel shuffle so as to facilitate communication between groups of convolutions. The ShuffleNets use the technique of Pointwise Group Convolution to reduce computation costs with the goal to maintain the feature diversity via channel shuffle [66]. One of the other notable contributions to the field of optimizing Convolutional Neural Networks is that of the EfficientNet model, which uses the technique of a new kind of scaling to increase depth, width, and resolution at the same time. Unlike the previous techniques of using random model scaling techniques, the EfficientNet uses a new efficient approach with reduced parameters and low computational cost [67]. Designing of architectures in these models shows advancement in the design of CNN architecture model by focusing on the design of architecture rather than increasing the depth of the model.

2.8 MODEL COMPRESSION AND OPTIMIZATION TECHNIQUES

Alongside advances in architecture, other techniques to compress the model and make the CNNs more efficient have also been developed. In all of these techniques, the basic idea remains the same, i.e., to make the model compact and light-weighted with efficient computations while having low memory usage with acceptable levels of accuracy. One of these techniques is pruning which comprises removing all redundant parameters within the neural network. Based on their extent, prunning can be of different kinds, such as weight prunning, filter prunning, and layer prunning. Filter prunning has shown its effectiveness in minimizing the cost associated with CNN computations [68]. Quantization is another technique where conversion of values from floating point forms into low bits happens. Thus, quantization enables the reduction in computational costs and memory usage of the model [69]. Knowledge distillation is a very powerful technique where a small student network learns from a larger

teacher network. Ultimately, the performance of both networks becomes identical, but the computational cost decreases significantly [70].

2.9 LIGHTWEIGHT CNNS IN REAL-WORLD APPLICATIONS

The application of lightweight CNN models has become an industry standard in multiple practical applications due to the high efficiency and flexibility of such algorithms. In mobile computing, they can provide functions of image recognition, face detection, and augmented reality using very limited computing power. In medical diagnostics, they are able to analyze images obtained via x-ray and MRI scanning for detecting diseases. Due to their ability to run on low-power computers, such solutions may be implemented even in remote regions with very few computing resources available [71]. The field of industrial automation requires application of lightweight CNN algorithms in quality control, defect detection, and robotics. In any case, the performance of such tasks depends directly on the optimized neural network architecture. Moreover, video surveillance system design also requires object detection that is possible only through proper functioning of a neural network. Limited use of computational resources enables deployment on edge devices rather than on centralized clouds.

2.10 RESEARCH GAP IDENTIFICATION

However, there are some challenges that should still be considered despite the recent advancements in the development of lightweight CNNs. First of all, it concerns the challenge associated with the inability of most lightweight CNN models to ensure their efficiency and accuracy at the same time. In fact, there is either a trade-off between their accuracy and efficiency or vice versa. This applies to the connection between their computational complexity and accuracy. Namely, it is evident that an increase in the former leads to an improvement in the latter. Furthermore, many lightweight CNN models today are designed specifically for one particular task while lacking a universal approach to the use of datasets. In addition, dynamic adaptive lightweight CNN models are not common in the domain of deep learning. The optimization of CNNs is performed using pruning and quantization techniques; however, in this case, there are complications connected with the optimization process itself. Therefore, it is important to develop efficient CNNs that are scalable at the same time.

2.11 COMPARATIVE ANALYSIS OF EXISTING CNN MODELS

Nevertheless, improvements in the CNN architecture have led to the creation of various types of architectures with varying levels of complexity, accuracy, and efficiency. Traditional deep learning architectures such as VGGNet and ResNet tend to optimize for accuracy via the development of deep architectures and representations. Despite their high performance on benchmark data sets, these deep neural network architectures are quite resource-intensive hence their limited use in practical settings. On the other hand, lightweight neural networks such as MobileNet, ShuffleNet, and SqueezeNet attempt to achieve computational efficiency through complex ways of implementing convolutions. Various convolution strategies utilized in lightweight neural networks include depthwise separable convolutions, grouped convolutions, and fire modules. A comparison of the above-mentioned architectures shows that achieving computational efficiency comes at the expense of accuracy. For instance, even though the architectures using ResNet produce better results, they are resource-intensive and hence cannot be used in edge devices. In contrast, while architectures like MobileNet and ShuffleNet are resource-efficient, their accuracy levels are lower compared to architectures using ResNet. The recent advancements, EfficientNets, try to address this issue by introducing

a concept called compound scaling. Compound scaling involves achieving a balance between the depth, width, and resolution of the network, and evidence shows that compound scaling produces good results using fewer parameters [73]. However, despite the advancements made so far, there is no single architecture that can satisfy both the criteria in every situation.

2.12 PERFORMANCE TRADE-OFFS IN CNN ARCHITECTURES

One of the key elements when it comes to designing Deep Learning architectures is the trade-off between computing and accuracy. More depth and more width of the model would generally increase its representation capacity. In return, though, an increase in these parameters is always accompanied by higher demands for computation, storage, and processing speed. Lightweight DL architectures try to achieve the right balance in the mentioned problem by simplifying their structure. Simplification, though, results in decreased capacity for information representation, which makes lightweight models less efficient with respect to working with complex data sets. It is generally hard for lightweight models to extract fine features from high-quality images or multiple-class data sets. Solutions include, among others, parameter sharing, reusing the extracted features, and optimization of convolutional processes. Nonetheless, finding a perfect solution is still a challenge [74].

2.13 ROLE OF OPTIMIZATION TECHNIQUES IN MODEL EFFICIENCY

Some optimization methods that were used in order to address computation challenges were mentioned in past works. They include pruning, quantization, and knowledge distillation. The pruning approach implies getting rid of unnecessary links in the neural network. As a result, the neural network becomes smaller and more efficient. The quantization approach implies reducing numerical precision with the aim to increase calculation speed and reduce memory requirements. The knowledge distillation method implies transferring knowledge from a large teacher model to a small student model. This way, the student model achieves better results like those provided by a larger one. All these optimization methods are widely used in modern CNN optimization procedures [75]. However, even though these methods are quite useful, they are applied after training and thus need more optimization.

2.14 CNN APPLICATIONS IN REAL WORLD SYSTEMS

The CNN-based approaches are widely utilized in practice in a variety of different domains. First, CNNs can be applied in the field of medicine in order to diagnose illnesses according to the analysis of the images made using the patient's x-rays, CT-scans, and MRI. In particular, the application of CNN-based methods is beneficial from the perspective of development of light models which will help diagnose the illness remotely [76]. Moreover, CNNs can be utilized in the sphere of security in order to carry out the task of object detection. The latter is a time-intensive process, and therefore, lightweight models are preferable to traditional ones. Another sphere where CNN-based techniques are applicable is mobile technologies because CNNs help achieve the goals of face recognition, object detection, and augmented reality implementation. However, it should be noted that in the sphere, the devices have some limitations concerning processing and energy. As for the field of industrial automation, CNNs can be applied in tasks related to defect detection, quality assessment, and robotics.

2.15 IDENTIFIED RESEARCH GAP

On the other hand, even though there has been significant advancement in developing CNN architecture, there are still certain challenges that have not been resolved. First of all, although

most of the architectures mentioned above are effective in terms of computational costs, they lack precision when applied to resolving complex tasks. Secondly, most of the models created are dedicated solely to performing particular tasks, therefore cannot be used for solving another task or working with a different kind of data. Thirdly, there is currently a need for optimization techniques such as pruning and quantization to be integrated into the architecture. Fourthly, it is required to create models whose complexity depends on available computational capacity. Fifthly, it is crucial to take into consideration several factors including accuracy and inference speed.

Table 2-1: Comparison of Identified Research Gap

Ref.	Method / Model	Hardware Platform	Application Domain	Key Results	Identified Gap
[20]	Quantized CNN	ESP32-S3	Object detection	86% accuracy	Not real-time
[2]	Machine learning review	MCU-class devices	General	Conceptual overview	No deployment
[9]	Quantized CNN	Raspberry Pi	Plant detection	92% accuracy	No MCU validation
[3]	TinyML survey	General MCUs	—	Theoretical analysis	No CV implementation
[4]	Quantized CNN	STM32	Biomedical imaging	120 ms/frame	Binary classification only
[8]	TinyOps framework	STM32H7	ImageNet inference	Efficient execution	Not multi-class
[10]	CNN-based motion detection	Arduino Nano	Motion sensing	500 ms latency	Slow inference
[17]	MCU benchmarking survey	Various MCUs	Performance comparison	—	No model implementation
[19]	Lightweight CNN	MCU	Robotics	85% accuracy	Memory overflow
[16]	CNN-based vision	ESP32	Visual recognition	60% accuracy	Low throughput
[18]	TinyML for computer vision	STM32	Aerial vision	80% accuracy	No multi-class support
[5]	TinyML review	Embedded devices	General	—	Optimization trade-offs
[7]	Energy profiling study	MCU	Power analysis	—	Hardware constraints
[15]	MCU-based AI framework	MCU	System integration	—	Integration challenges

2.16 SUMMARY OF LITERATURE REVIEW

From the literature review, it becomes clear that there have been advancements in CNN architecture from traditional CNN architecture like LeNet to modern CNN architecture for deep learning like ResNet and EfficientNet. Unlike traditional CNNs, the lightweight CNNs, including MobileNet, ShuffleNet, and SqueezeNet, are capable of delivering high efficiencies without compromising on the accuracies of the networks. However, certain limitations are experienced when using these networks with respect to the accuracies and scalability of the network. Pruning, quantization, and knowledge distillation methods are employed to further increase efficiencies; however, these methods come with added complexity. From the literature review, it can be seen that there is a need for lightweight CNN models that offer optimal efficiencies and accuracies. The goal of this research project is to develop a novel lightweight CNN classification model.

CHAPTER 3 METHODOLOGY

3.1 OVERVIEW OF RESEARCH METHODOLOGY

The following research provides a full methodology for creating a lightweight multi-class classification model deployable in an ESP32 cam microcontroller in order to close the above-mentioned research gap. It starts with systematically pre-processing the dataset, then splitting it, training, validating, optimizing, and deploying the model to the embedded system. In order to create an efficient model, the process began with obtaining a complete dataset prepared in a unique way where objects had clearly visible sizes in images; then resizing, greyscaling, and normalizing them to 96 resolutions. The training process consisted of designing a model architecture made up of three convolutional blocks, where increasing numbers of filters were applied to extract features efficiently and with low computation costs. For good results and stability during the training process, optimization methods like Early Stopping, Model Checkpoint, and ReduceLROnPlateau, along with dropout layers, were applied for overcoming the issue of overfitting and generalizing the model. For this, an accurate model was generated in the h5 format and then converted into quantized INT8 TFLite format to make the whole model size smaller, and then test the model using confusion matrix and real-time inference, in the form of live streaming. The entire workflow of the methodology proposed above can be seen in Figure 1.

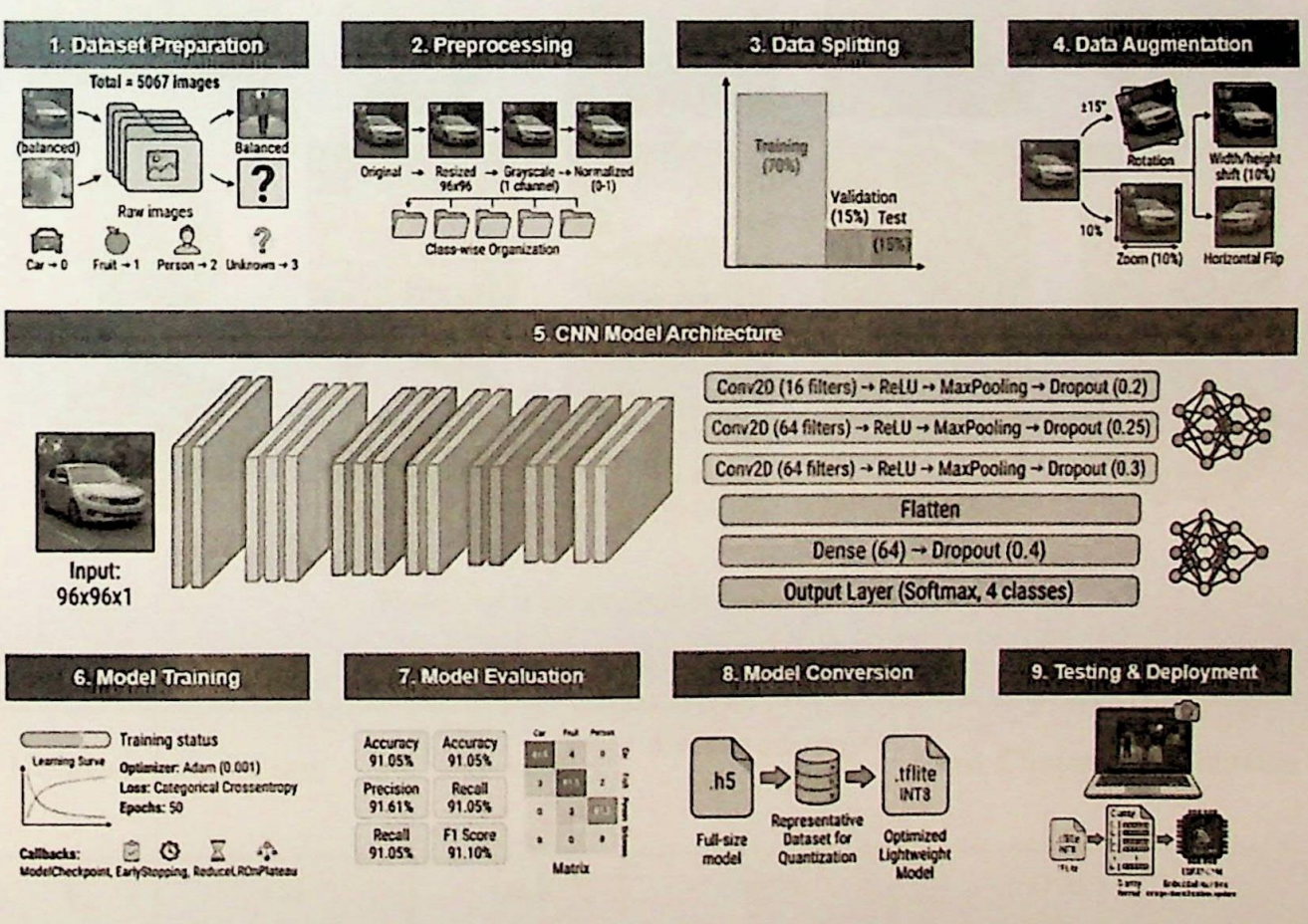


Figure 3-1: Overall Workflow of the Proposed Methodology

3.2 CUSTOM DATASET COLLECTION AND ORGANIZATION

The data used for the training was custom built by choosing a variety of classes including person, fruit, unknown, and car using the "Kaggle" website, which is one of the most popular online datasets sources. Since the goal is to build an accurate predictively trained model that will be deployed on the ESP32 microcontroller, some filtering techniques were conducted to choose those images in which the targets were clearly visible. Using this filtration technique in the dataset helps reduce overfitting and improve the prediction generalization ability during the training process. Through the selected images, a total of 5067 images were chosen in which there is an equal distribution of the same amount of images among all classes as presented in Table 1. While the number of data is very little to train the accurate classification models through the process of computer vision, but it is adequate to train the classification models of ESP-based multi-class classification. In order to ensure that the classifier is able to learn the essential features from low-resolution input, the dataset summary and some images are provided below figure 3-2.

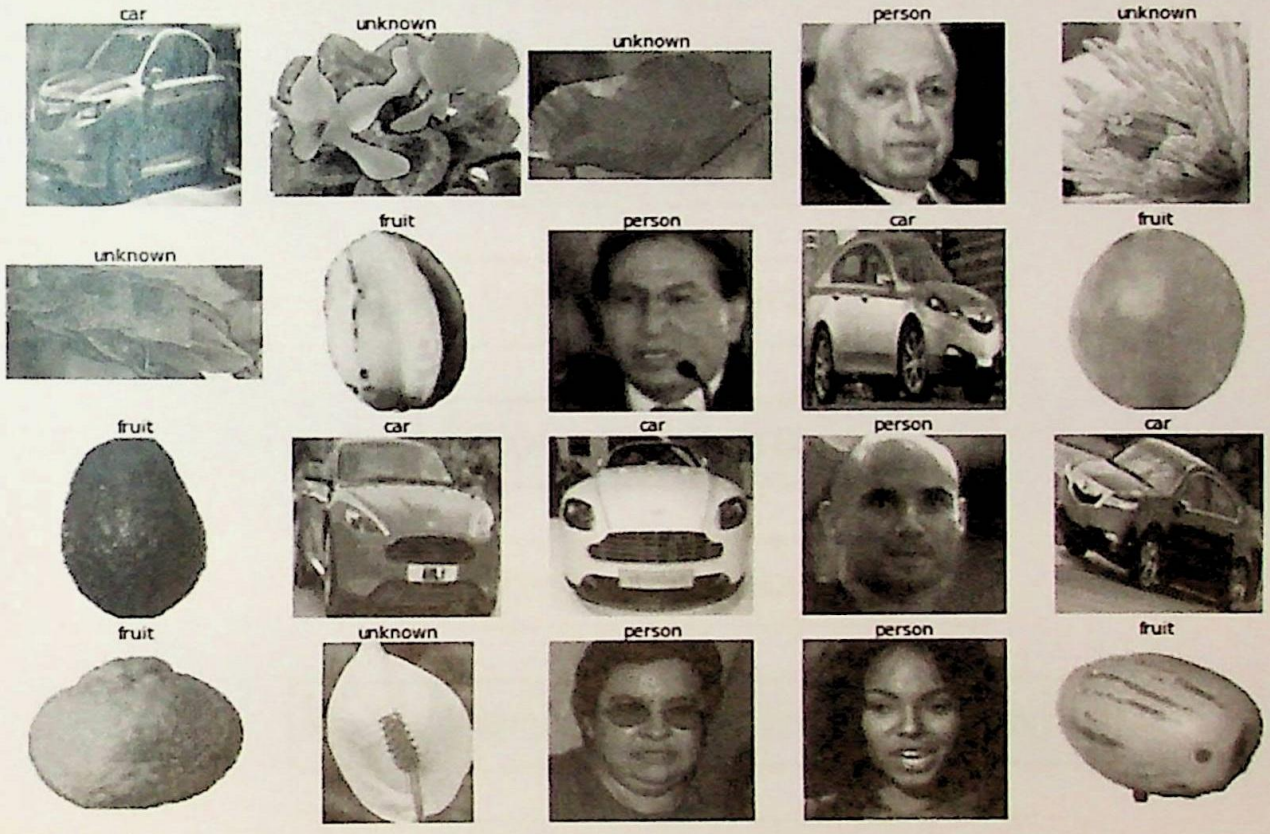


Figure 3-2: Custom Dataset Directory Structure

Table 3-1: Original dataset summary

Number	Class	Images	Average Image Size (MB)	Most Common Resolution
0	Car	1267	0.0115	(100, 100, 3)
1	fruit	1267	0.0045	(100, 100, 3)

Number	Class	Images	Average Image Size (MB)	Most Common Resolution
2	person	1267	0.0156	(256, 256, 3)
3	unknown	1266	0.0759	(500, 375, 3)

In order to decrease the complexity and computation, the following pre-processing was performed for the data before training. The dataset was resized to 96 × 96 pixels and converted to grayscale format with normalization from 0 to 1. This preprocessing decreases the size of input data and makes the dataset more homogeneous for the model. Images that have been preprocessed were saved to another directory in order to increase reproducibility and decrease computation time. Pre-processed data summary and some image visualizations are presented below. Figure 3.3.

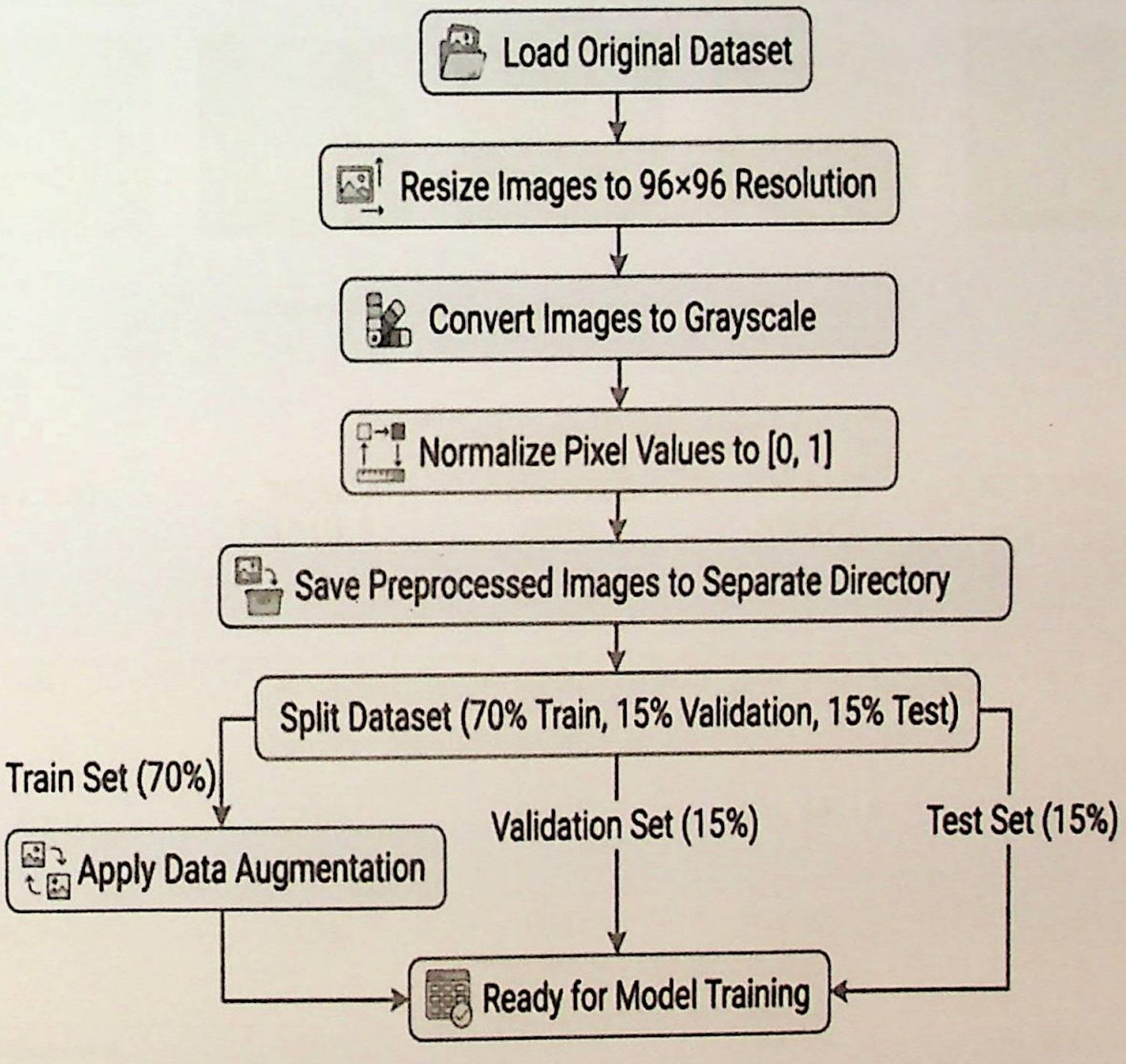


Figure 3-3: Image Preprocessing Pipeline Flowchart

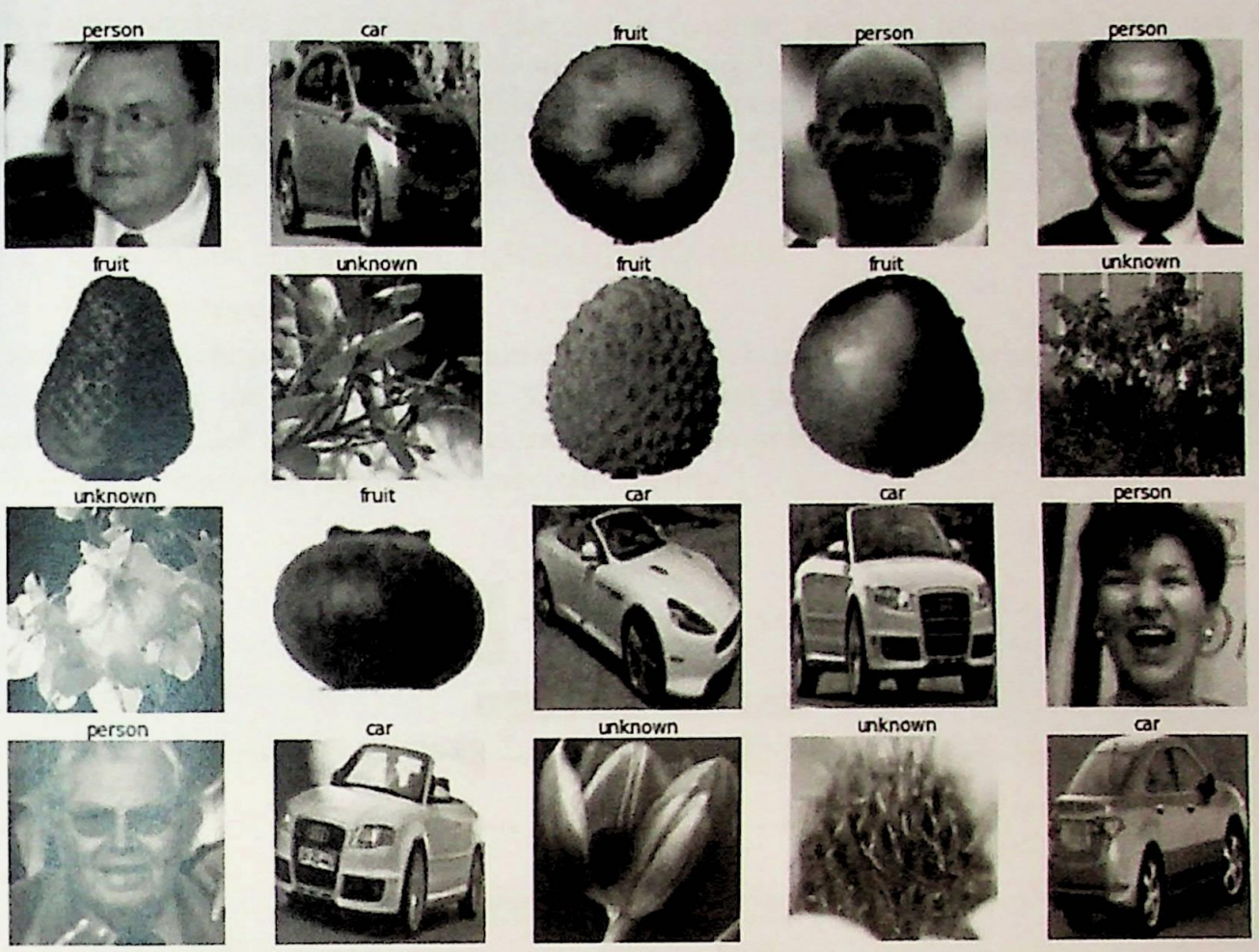


Figure 3-4: Sample Preprocessed Grayscale Images

Table 3-2: Preprocessed Dataset Summary

CLASS	NUM IMAGES	AVG SIZE (MB)	IMAGE SHAPE	EXTENSIONS
car	1263	0.0024	(96, 96, 1)	.jpg
fruit	1270	0.0015	(96, 96, 1)	.jpg
person	1273	0.0018	(96, 96, 1)	.jpg
unknown	1261	0.0024	(96, 96, 1)	.jpg

This table show the preprocessed clean dataset summery in which the dataset loads into 4 dimensional tensor of shape using NumPy for training and evaluation. Corresponding class labels were encoded as integers based on class mapping then split into train and validation with augmentation as x shape is (3797, 96, 96, 1) whereas y shape is (3797,) with class mapping as given "car: 0, fruit: 1, person: 2, unknown: 3".

3.3 CNN MODEL ARCHITECTURE AND TRAINING

3.3.1 OVERVIEW

The parameter chosen to train the multi-class classifier is mainly the major segment because of its influence on the train model size. This is important to make sure that the pre-processing enables the model to identify and extract features from image objects including shape and pattern thus improving the model's prediction ability.

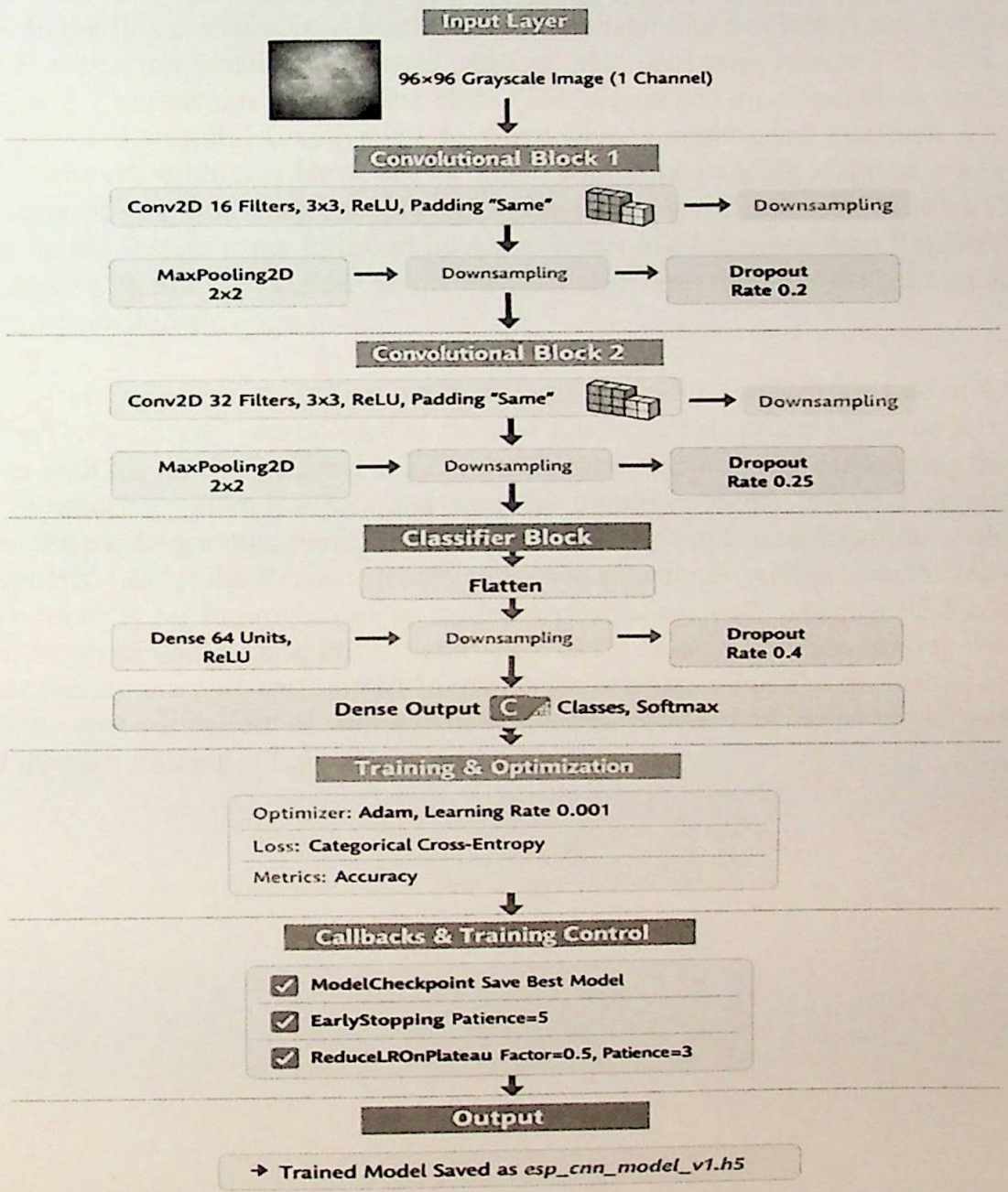


Figure 3-5: The workflow of the model training

3.3.2 CNN ARCHITECTURE

The architecture of the model consists of three convolutional blocks for training of the lightweight classification model that consists of three parameters namely the convolutional layer, a max-pooling layer, and a dropout layer. In this way, there is an inclusion of three convolutional blocks to have a balance between memory capacity and accuracy of the model. Depth of the network becomes higher while extracting more complicated features. Max-Pooling helps to down sample the spatial dimensions for selecting important features. But, the model would become over fit due to these complicated features thus dropout layer helps in avoiding overfitting of the model.

The features extracted by layers are converted into the flattened form to be passed by the fully connected layers using a Softmax activation function. The whole CNN architecture as depicted in Figure 4 starts with an input layer that accepts grayscale images having a resolution of 96×96×1. In the first convolutional block, 16 convolutional filters of kernel size 3×3 were used with ReLU activation function with same padding. Max pool layer of size 2×2 and a dropout layer of rate 0.2 are included in the first block. The second convolutional block has 32 filters of kernel size 3×3 with ReLU activation, followed by max pooling and a dropout layer of rate 0.25. The third convolutional block has 64 filters with same padding and max pooling and a dropout layer of rate 0.3. After convolutional layers, the classification block has a flattening operation on the feature maps followed by dense layers of 64 neurons with ReLU activation. Dropout layer with rate 0.4 is added before the last dense layer that uses the softmax activation function for probabilistic output.

During the training phase, the Adam optimizer was used with the learning rate of 0.001, and categorical cross entropy was utilized as the loss function. This process took up to 50 epochs to be done with the batch size fixed at 32. Further, early stopping was used during the training to keep monitoring validation loss and stop the training process to avoid overfitting and maximize the model training performance, which was not improving anymore, with patience set at 5 epochs. Lastly, the ReduceLROnPlateau was also employed to lower the learning rate whenever there is no improvement in model performance, with patience of 3 epochs and reduction factor of 05. Overall, this structure provides a balance between making the network deeper for feature learning and smaller in size to facilitate training. The evaluation of training and K-Fold cross-validation, as well as test set results, proved that the h5 model architecture obtained through training is lightweight but highly accurate.

Table 3-3: CNN Architecture Summary

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 96, 96, 16)	160
max_pooling2d (MaxPooling2D)	(None, 48, 48, 16)	0
dropout (Dropout)	(None, 48, 48, 16)	0
conv2d_1 (Conv2D)	(None, 48, 48, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 24, 24, 32)	0
dropout_1 (Dropout)	(None, 24, 24, 32)	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 64)	0
dropout_2 (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 64)	589,888
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 4)	260

3.3.3 MATHEMATICAL FORMULATION

The mathematical equations of custom computer vision model training for my application include the model training from scratch without any pertained weights. The model training is start from an input layer representation in 3D matrix of image X as:

$$X_{input} \in R^{height \times width \times depth}$$

Where, height and width is 96x96 (pixels) and depth is 1 (grayscale) is the input of model in the form of network which allow the neurons to extract the features from the image in subsequent layers as convolutional layer operation while maintaining spatial relations.

$$Z^{layer} = \Sigma_{n=1}^{N} W_n^{layer} * A^{layer-1} + bais^{layer}$$

There convolutional layers extract the features such as edges, texture, patterns by sliding the multiple learnable kernel. Each kernel generates a feature map that highlights specific local characteristics of the image. In this process the Ralu activation functions use for the positive signals of layers and non-linearity help the model to complex mappings which prevent the model multiple convolutional layers that could collapse into a single linear operation.

$$A^{layer} = \max(0, Z^{layer})$$

This spatial pooled features maps convert into a feature vector 1D that can be combined perform the high level reasoning by flatten operation as shown:

$$f = Flatten(P^{L})$$

The flattened vector is processed by a dense layer:

$$h = Weights_{fc} \cdot f + bais_{fc}$$

Softmax converts raw network outputs into probabilities, allowing multi-class classification. The network produces class probabilities using Softmax:

$$\hat{y}_c = \frac{e^{h_{class}}}{\Sigma_{j=1}^{Class} e^{h_j}}, for\ class = 1,2,\dots,N$$

Cross-entropy penalizes incorrect predictions more heavily for classes that were assigned high probability, encouraging the model to assign higher probability to correct classes.

$$Loss = -\Sigma_{class=1}^{CLASS} y_{class} \log(\hat{y}_{class})$$

Mini-batch training allows the model to update parameters more frequently than full-batch training while providing smoother and more stable gradient estimates than stochastic updates.

$$L_{batch} = -\left(\frac{1}{BAIS}\right) \Sigma_{i=1}^{BAIS} \Sigma_{class=1}^{C} y_i, class \log(\hat{y}_i, class)$$

The trainable parameters are updated via gradient descent:

$$\theta^{t+1} = \theta^{t} - \eta \nabla_{\theta} L_{batch}$$

Where θ represents all trainable parameters (weights and biases), η is the learning rate and $\nabla_{\theta} L_{batch}$ is the gradient of the loss with respect to the parameters.

However the entire equation for the whole model training is given:

$$\frac{\min_{\theta} 1}{B} \Sigma_{i=1}^{B} \Sigma_{c=1}^{C} y_i, c \log\Big(softmax\big(f(G(X_i); \theta)\big)_c \Big)$$

This single equation captures augmented input images, convolutional neural network feature extraction, soft max classification, cross-entropy loss, mini-batch gradient descent optimization.

3.3.4 MODEL TRAINING

To train the model, 50 epochs was used which is enough recommended training logs for this application and feed the batches of images to the model which is evaluated or validate by validation dataset simultaneously which help the model to adjust their weights and learning performance. The adjustment of the weight is improve as continues callback defines which helps the model to adopt the learning rate when necessary. After the completion of training of the model the file is saved at specific location for evaluation of the model. By this learning rate the model archives 96% accuracy. Training logs, including loss and accuracy curves, should be included in appendix to provide additional insights into the model's learning process. Confusion matrices and other evaluation metrics will be reported separately in chapter 4 results.

3.4 MODEL QUANTIZATION USING TENSORFLOW LITE INT8

After training the CNN model, we converted it into compatible INT8 TFLite model using full integer quantization with a representative dataset. Let the trained CNN parameters be denoted as $\theta_{float32}$ and the quantized parameters as θ_{int8}. The quantization process maps the floating point weights and activations of the model to the INT8 $-128\ to + 127$. This mapping is performed using the equation:

$$q = round\left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} * \left(q_{\max} - q_{\min}\right) + q_{\min} \right)$$

Where `x` is a float32 tensor value, x minand max are the minimum and maximum values of the tensor, and `q_min = -128`, `q_max = 127`. During inference, the quantized tensor is de quantized back to an approximate float value using:

$$\hat{x} = (q - q_{min}) * \frac{x_{\max} - x_{min}}{q_{\max} - q_{min}} + x_{min}$$

This ensures that computations on integer tensors approximate the outputs of the original float32 model. For finding out what min and max values are required in quantization, a small representative set of images X are sampled. The images in X would possibly be augmented by some images obtained through a function G that provides rotation, scaling, shifting, and flipping.

$$X_{i_{aug}} = G(X_i), \qquad i = 1, 2, \ldots, N$$

This methodology will result in a relatively smaller model of CNN, which is suitable for devices with limited memory, with accuracy close to that of the earlier model, as depicted below.

3.5 LAPTOP WEB CAM INFERENCE

For evaluating the performance of the INT8 TFlite model, the software-based real-time webcam was used. In every frame of the video, the image was center cropped and resized to 96x96 pixels before being sent for interpretation by the TFlite model to yield the required probability values and the classification of the object in the zoom area of the frame by displaying the output of the model over the zoom area of the image on the webcam feed.

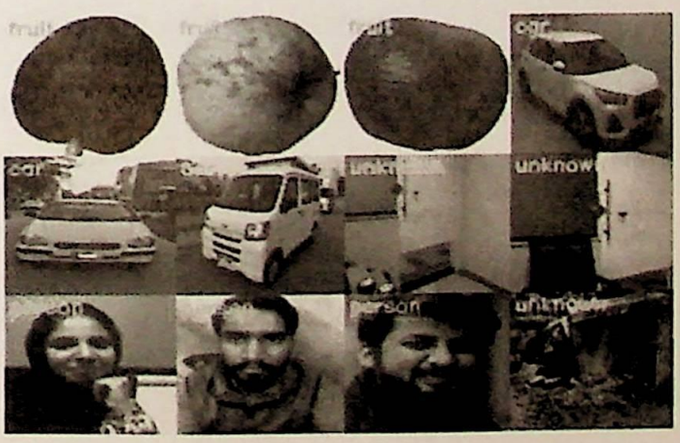


Figure 3-6: Live Webcam Prediction Output with Zoom ROI (Region of Interest)

3.6 MODEL CONVERSION TO C ARRAY

Once the training of the CNN is done and the quantization is performed to use less memory with the help of the data type 'INT8,' the '.tflite file' is converted into a C header file. In the header file, the '.uint8_t data type' is used. Using the data type 'uint8_t data type' has the advantage of 'bulging' the data straight into the memory. In simple words, the ESP32 does not have the 'capability' to read the '.tflite file' from the memory. The transformation process was conducted using 'Git Bash' command line terminal. It is a lightweight 'Windows shell' with Unix command compatibility features like 'xxd'. It is advised since this command line software works well with binary files and avoids Windows command line limits for large 'model' files. The conversion workflow involves navigating to the model directory and enter the command to generate the header.

xxd -i esp_model_int8.tflite > esp_model_int8.h

This generated .h file will be included in the project as part of the Arduino PlatformIO configuration, with TensorFlow Lite for Microcontrollers (TFlite Micro) executing the model in a tensor arena to make predictions on live video from various cameras. This process helps bridge high-level modeling with very low-level embedded development to make AI work on constrained platforms like the ESP32.

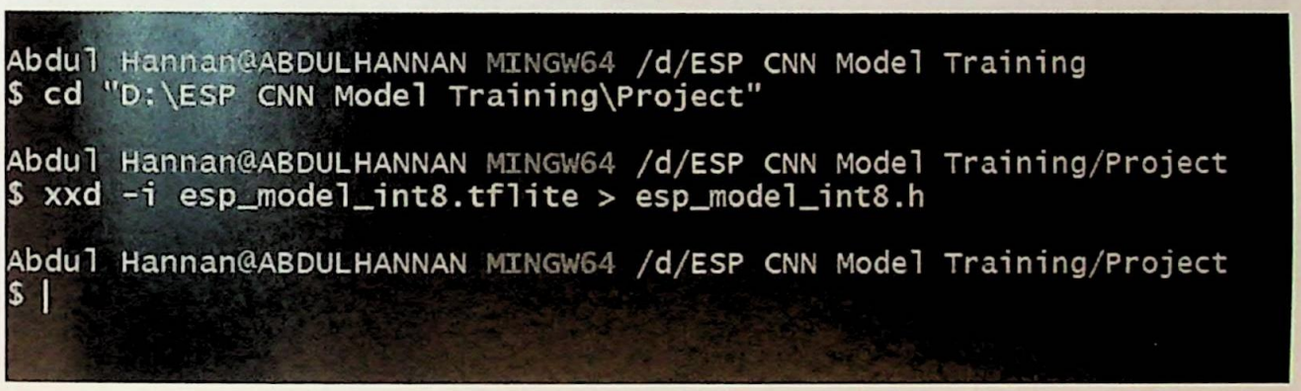


Figure 3-7: Git Bash Model Conversion Commands

3.7 ESP32 FIRMWARE IMPLEMENTATION

After converting this CNN that we've just finished training into a C-array, we now take this array and put it onto our ESP32-S3 DevKitC-1 board, attaching this array onto our board using a combination of Arduino along with PlatformIO, changing our platformio.ini configuration a bit so that we're using a PSRAM-enabled board, choosing our board properly, choosing our framework, adjusting memory, as well as creating a few flags so that TensorFlowLiteMicro operates properly, and while this Arduino code initializes our ESP32 camera to take 96x96 grayscale images, storing frames of memory in PSRAM, we also begin our TensorFlow Lite Micro interpreter, loading in this array that we just created, along with a header (.h). An 8 MB arena for the tensors will be allocated in the PSRAM for both input as well as output tensors to make sure efficient inference on the device without the need to save any data in external storage mediumsThe continuous flow of videos is received within the loop() function, where the video inputs are then transferred to the input tensor and presented to the interpreter for inference. The

output class having the highest probability is determined from the output tensor, with real-time console logging performed either by serial monitor or even on the TFT screen. This configuration enables real-time inference in live situations through embedded hardware, effectively "activating" a machine learning model in the form of a fast-running CNN.

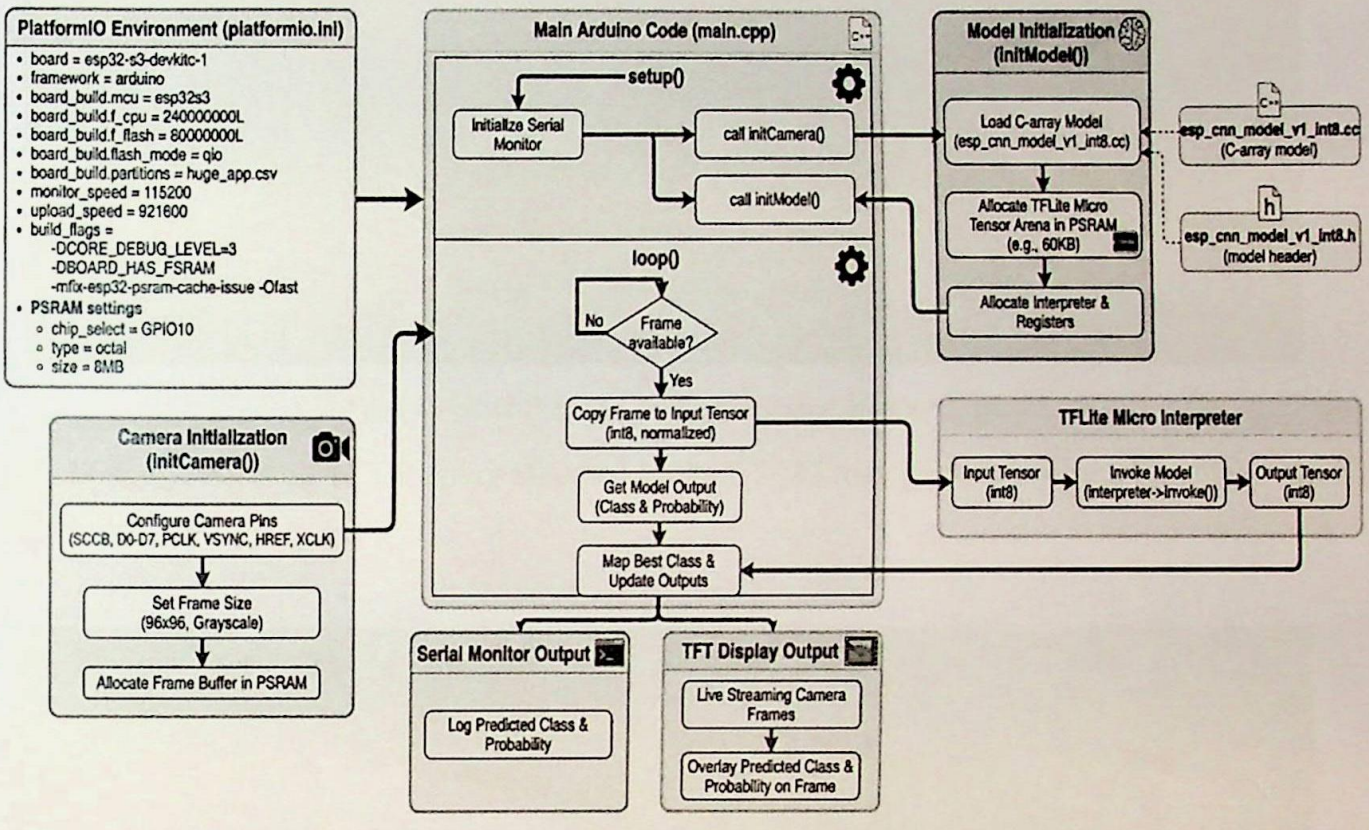


Figure 3-8: ESP32 CNN Firmware Deployment & Execution Workflow

3.8 LIVE STREAMING PREDICTION ON ESP32

After the ESP32 boots up, it immediately engages in the analysis of video capture from the camera in real-time. Each video frame received from the camera will be captured and then placed within the tensor for analysis by the model. After analysis by the computer model through the relevant neural network, the class with the greatest probability will then be displayed in two forms:

Serial Monitor: It prints out the class index along with the confidence percentage for each frame.

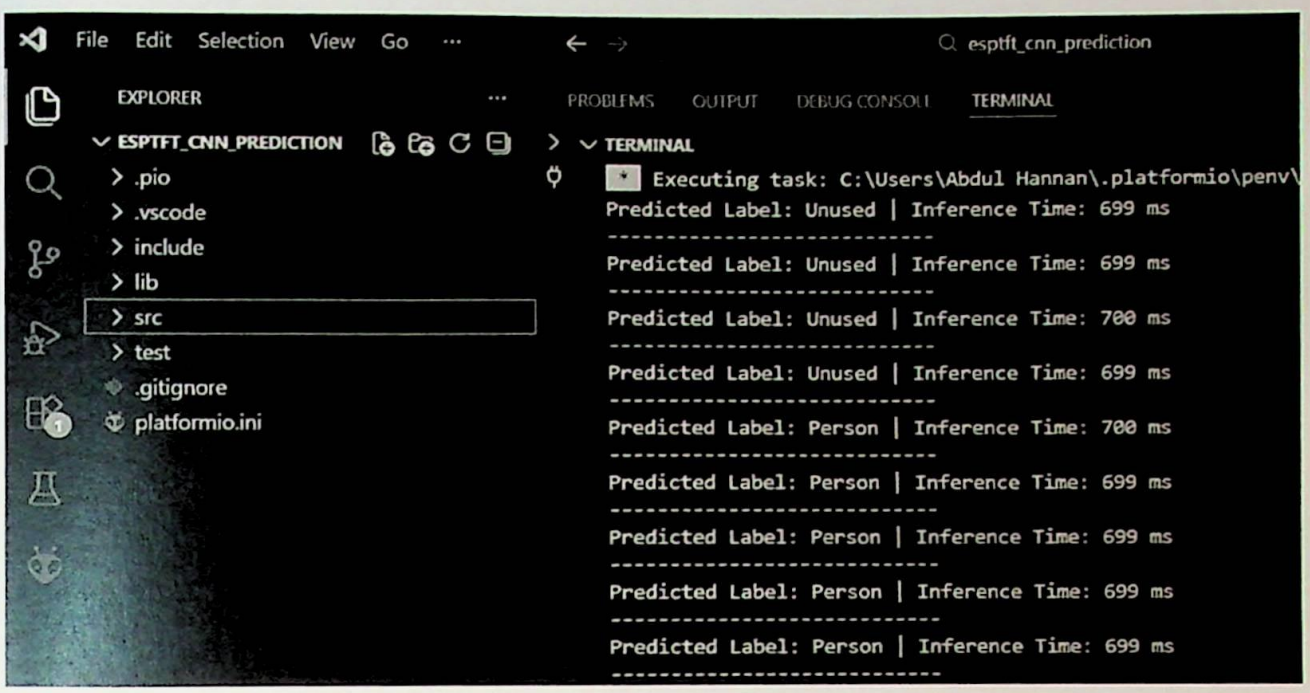


Figure 3-9: ESP Model Prediction Serial Monitor Logs

TFT Display: A small TFT display attached to the ESP32 may provide a visual preview of the predictions.

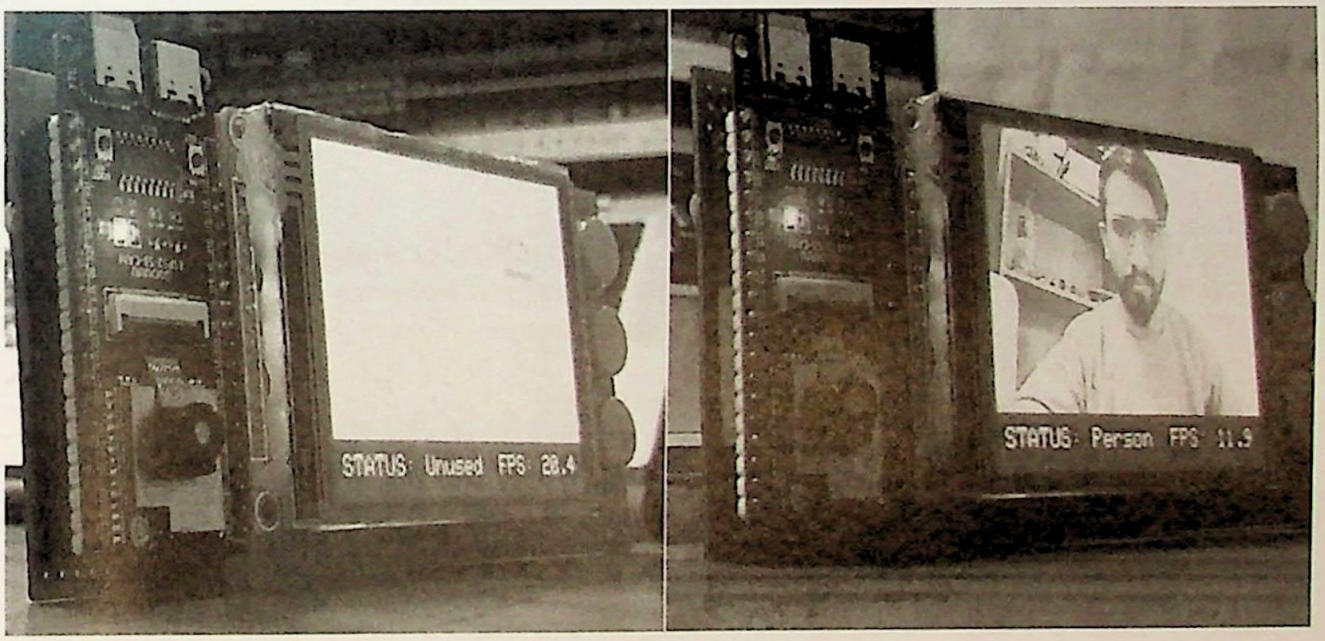


Figure 3-10: On ESP Model Prediction with TFT Live Streaming

CHAPTER 4 RESULTS & DISCUSSIONS

4.1 EVALUATION MEASURES

The following measures have been used in analyzing the results of the multi-class classification model: accuracy, precision, recall, F1 score, confusion matrix, and k-fold cross-validation. Even though all of these measures are quite popular in measuring the performance of any multi-class classification model, they will enable us to see the general performance of our model as well as the class-specific performance measures.

Accuracy:

Accuracy measures the ratio of correct predictions to the total number of predictions. Accuracy is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision:

It defines the ratio of correctly identified positives from all positives that are identified. In essence, precision is used to determine how well the machine-learning algorithm avoids making any mistakes. The calculation for precision is:

$$Precision = \frac{TP}{TP + FP}$$

Recall:

Recall, or sensitivity, denotes the fraction of correctly identified positive samples out of the total number of positive samples. It represents the capacity of the model to find all the important cases. The recall for each class is computed by:

$$Recall = \frac{TP}{TP + FN}$$

F1-score:

The F1 score represents the harmonic mean of precision and recall. This score combines both false positive and false negative cases in a single evaluation, which makes it especially suitable for multi-class classification where different classes may be more difficult to classify than others:

$$F1 - Score = \frac{Precision\ x\ Recall}{Precision + Recall}$$

Confusion matrix:

Confusion Matrix Confusion matrix is a table that presents actual vs. predicted values for a dataset. This can be used to determine the errors made while classifying the data. In a multiclass

image classification task having N classes, the dimensions of the confusion matrix will be n x n where the element at row I and column J will represent the count of samples belonging to class I but labeled as class J.

$$Confusion\ Matrix = \begin{bmatrix} C_{1,1} & \cdots & C_{1,n} \\ \vdots & \ddots & \vdots \\ C_{n,1} & \cdots & C_{n,n} \end{bmatrix}$$

4.2 RESULTS

Evaluating of the Trained Keras Model

The results of the trained keras model were analyzed using standard methods such as accuracy, precision, recall, F1 Score, confusion matrix, and k fold cross-validation for ensuring model performance predictions for all classes. First, the model started with a large loss and high loss of 1.13 and 47.61% accuracy during the training, while the validation accuracy was 76.18%. A few epochs later, the model was stabilizing, improving the training accuracy of 90% while the loss was reduced to 0.26 and 0.11. Finally, at epochs 24, the model restored its best weights when its validation accuracy reached 99.34%, with 0.0339 loss, as demonstrated in figures 5 and 6.

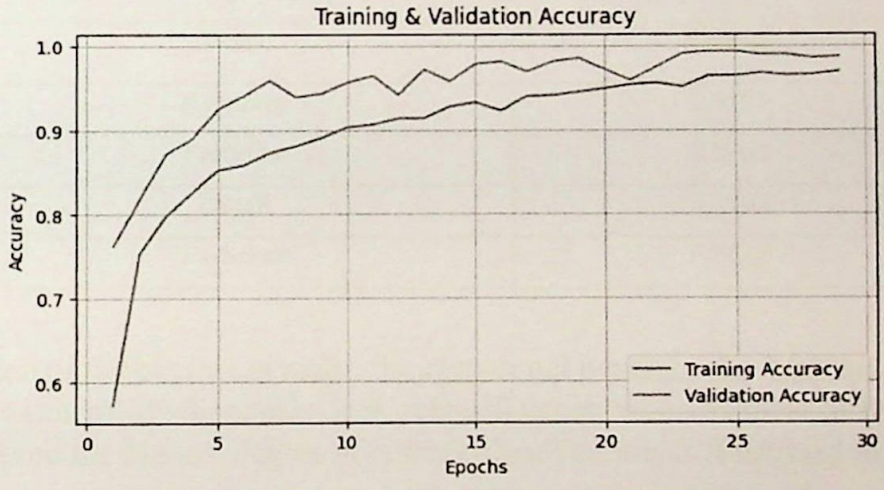


Figure 4-1: Training & Validation Accuracy

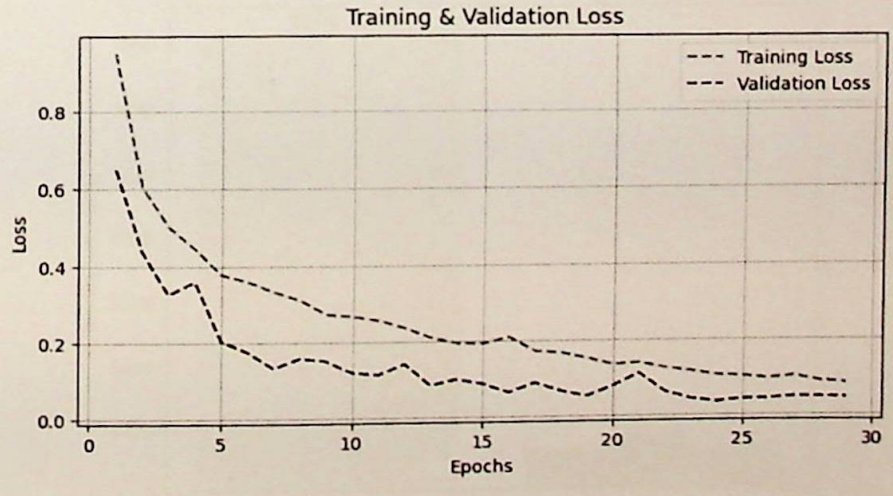


Figure 4-2: Training & Validation Loss

Table 4-1: Comparison of Proposed Method with Existing Studies

Ref	Model	Accuracy	Hardware
[22]	Lightweight CNN	85%	ESP32
[10]	CNN	87%	ESP32
[23]	Fast Region CNN	88%	ESP32 S2
[24]	Quantized CNN	86%	ESP32 S3
[13]	TinyML based CNN	88%	ESP32
Proposed Method	CNN	91%	ESP32

Once training was completed using efficient models, the testing dataset containing totally 760 images belonging to four categories including car, fruit, person, and unknown was tested for accuracy with results of 91.05% accuracy, 91.61% precision, 91.05% recall and 91.10% f1-score that show high efficiency in terms of performance for all categories as depicted in Table 3.

Table 4-2: Model Evaluation Matrix

Metrix	Results
Accuracy	0.9105
Precision	0.9161
Recall	0.9105
F1-Score	0.9110

Visual illustration of the confusion matrix is an essential benchmark which indicated that there are only a few cases of off-diagonals or misclassification, whereas most of the cases of each class have achieved the highest degree of correct classification as illustrated in Figure 8. Table 2 illustrates a comparison between the research method and existing works.

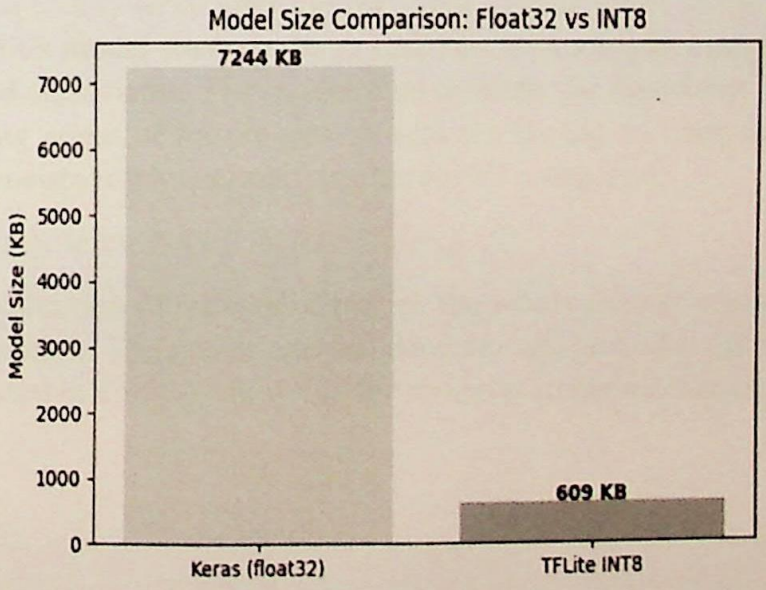


Figure 4-3: Quantized CNN Comparison

Once the evaluation for the trained model was complete, the trained model was then quantized using post-training full integer quantization to obtain the TFLite format of the INT8 model, thereby drastically shrinking its file size to around 609 KB, as shown in figure 7.

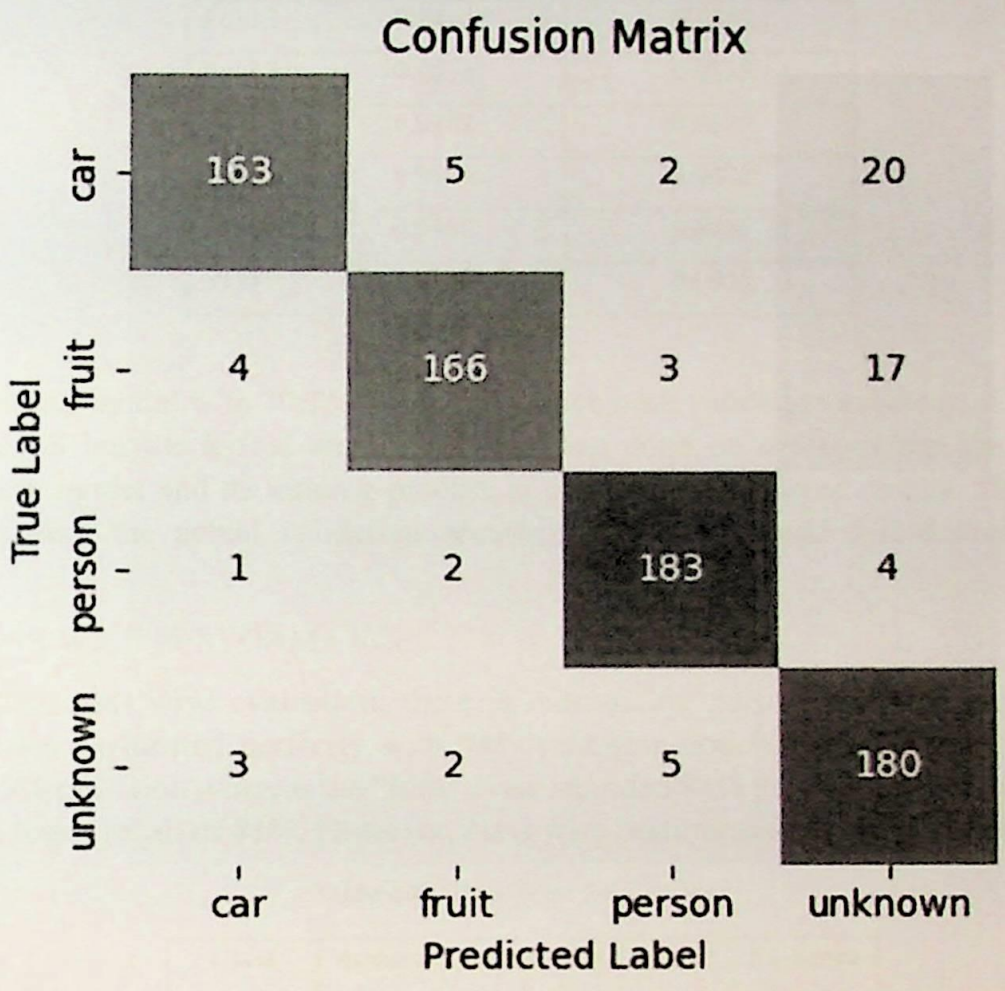


Figure 4-4: Evaluation Metrics of Different Classes

In this respect, this quantized INT8 TFLite model was experimented with using a live webcam setting, where the images were preprocessed to meet the input demands of the model. Besides, model was converted to a C array after the real-time visualization of the performance of this model. In addition, this model was run on an ESP32 CAM with an external TFT display that outputs the predicted class name. The model runs through the feed from the camera on a per-image basis, skipping some of the images in between owing to computing time. However, accuracy remains consistent while classifying across all categories.

4.3 K-FOLD CROSS VALIDATION RESULTS

For validating the evaluation of the trained model, the whole dataset was selected with 5 folds for k-fold cross validation. The means and standard deviations of the accuracy of training and validation, as illustrated in Table 4, show that the model is stable and the classification accuracy is reliable.

Table 4-3: K-Fold Validation Results

Fold	Training Accuracy	Validation Accuracy
Fold 1	0.9513	0. 9908
Fold 2	0.9444	0.9855
Fold 3	0.9536	0.9829
Fold 4	0.9302	0.9829
Fold 5	0.9694	0.9908
Mean	0.9497	0.9865
STD	0.0127	0.0035

The first trained model with 70:15:15 split ratio gave peak validation accuracy of 99.34 with a loss of 0.0338 but the k fold cross validation was done on the complete dataset without changing any model and its training process to give almost the same results. This is evident from comparing the actual validation accuracy and the average 5-fold cross validation accuracy.

4.4 CLASS WISE PERFORMANCE

Based on the class wise evaluation, there is outstanding performance for each class. The "person" class performed perfectly with 94% precision and 96% recall. The "car" classes recorded 95% precision whereas the "fruit" class recorded 94% precision while the "unknown" class had a lower recall of 81%. However, class wise performance as shown in table 5.

Table 4-4: Class Wise Performance

Class	Accuracy	Precision	Recall	F1-Score
Car	0.85	0.95	0.85	0.90
Fruit	0.87	0.94	0.87	0.90
Person	0.96	0.94	0.96	0.95
Unknown	0.94	0.81	0.94	0.87

4.5 DISCUSSION

The usage of artificial intelligence in embedded systems has opened up many more possibilities for edge-driven intelligent systems. Computer vision is one of the main parts of artificial intelligence that plays a vital role in understanding visual information. Implementation of computer vision on constrained devices with microcontroller becomes difficult because deep learning model needs high computational resources, large memory storage, high processing speed. These requirements exist in desktop computers but become impossible to achieve with a microcontroller. Therefore, in this research paper, multi-class image classification model on an ESP32S3 N16R8 CAM microcontroller will be discussed where the work was done independent of any external source or even cloud computing services. The key idea behind this research was to show that the deep learning model could run on a microcontroller even with low computational power. This research shows promising results and it is seen that image classification on four classes such asunknown, person, fruit and car. Which proved the

possibility of implementing computer vision applications in embedded systems with the help of a microcontroller. One of the main difficulties that were encountered during the research was a limitation of hardware resources in embedded systems with microcontrollers. They provide only several megabytes of memory as well as limited RAM. Therefore, performance optimization became crucial since it allowed us to match our neural network to fit within available resources. As a solution to this problem, quantization was applied. It reduced the size of a deep learning model, while still preserving the majority of its ability to make accurate predictions. In particular, the model size was decreased to a suitable value – 609KB for implementation on the ESP32 microcontroller. However, quantization brought some trade-offs between model accuracy and its size. Some of the parameters could lead to loss of accuracy, but this was tolerable in embedded devices where processing speed was more crucial than the prediction accuracy. Based on the observation, it is evident that the quantized models still maintain reasonable accuracy in their predictions, demonstrating that the model compression technique is well-suited for an embedded implementation. Another variable that affects performance is the input image. In this experiment, our model works with images rescaled to 96 pixels in resolution. Reducing the resolution lowers the parameter count in the neural network while reducing computation power and memory usage. However, lowering the resolution also results in feature loss, which could affect prediction accuracy. To counteract this issue, our dataset was preprocessed in such a way as to only include large-sized objects within the images so that they remain identifiable even when presented with lower resolutions. The microcontroller selected for implementation is the ESP32S3 N16R8 model due to its high memory storage compared to other microcontrollers. Using 8MB PSRAM and 16MB flash memory, this microcontroller has sufficient space to store the model data, firmware, and libraries needed for the project. Besides, the inclusion of a TFT LCD interface facilitates real-time camera stream viewing and allows the execution of model predictions.

CHAPTER 5 CONCLUSION

This study involves the training and deployment of the multi-class image classifier model within the constraints imposed by the ESP32 microcontroller. The training process comprises the collection of a custom dataset that contains objects of large size in the images for feature extraction from the normalized low-resolution images. This improved the training accuracy and led to the formation of an effective trained model. The goal of this research is to develop and train the lightweight small model that would be deployed on the ESP32 microcontroller. To accomplish this, a compact model architecture with only three convolutional layers having increased filter sizes (16, 32, 64) was chosen. In order to achieve enhanced performance and robust training optimization techniques like ReduceLROnPlateau, early stopping, and checkpointing were applied along with dropout to avoid overfitting. This trained deep learning model is converted to int8 TFLite through port training quantization which resulted in drastic reduction of model size to nearly 609KB. Then, the model was converted to a C array format for easy deployment in microcontrollers. ESP32S3 N16R8 CAM microcontroller along with an external TFT LCD were utilized to perform the inference operations on the CNN model. Real-time inference of the model is achieved by performing image classification on the ESP without having any reliance on cloud. In order to minimize latency, only selected few samples per frame can be processed because of the computational complexity and prediction capabilities of the model. Using ESP Tensor Flow Lite Micro firmware library on-device inference has been done efficiently which proves that the model can accurately classify images into multiple classes.

5.1 LIMITATIONS & FUTURE WORK

The suggested research highlights the use of multi-class classification in ESP32 microcontroller. Nevertheless, there are some restrictions in the proposed research, one of which is the latency that occurs during the process of model inference since the model needs some time to produce predictions in a resource-constrained microcontroller. Moreover, the model used for training is at a resolution of 96 grayscale which enables the model to be executed in resource-constrained microcontroller but makes the model less effective in recognizing objects because of its low-resolution size. In order to tackle this problem, the process of selecting the images to be trained has been based on the size of the target object in the image; however, this strategy might not be applicable due to the changing distance and sizes of objects when deployed using ESP32-CAM.

In order to overcome these limitations, the future research area that could be considered is the training process for the Multi-class Classification model that consists of very low latency with different kinds of Model Optimization Techniques while keeping the architecture lightweight and efficient. One more thing that can be improved for better performance is the implementation of Image Processing techniques in order to achieve accurate predictions.

ACKNOWLEDGEMENT

All praise and gratitude are due to Allah Almighty, the Most Gracious and the Most Merciful, whose countless blessings, guidance, and mercy enabled me to complete this research work successfully. Peace and blessings be upon the Holy Prophet Muhammad (PBUH), whose teachings continue to inspire the pursuit of knowledge and wisdom. I would like to express my sincere gratitude to my supervisor, Dr. Rao Muhammad Asif, for his invaluable guidance, continuous encouragement, constructive criticism, and unwavering support throughout this research. His expertise and insightful suggestions played a vital role in the successful completion of this thesis. I am also grateful to the faculty members and staff of the Department of Electrical Engineering, Superior University, for providing a supportive academic environment and necessary resources. I extend my appreciation to everyone who contributed directly or indirectly through technical assistance, valuable discussions, or moral support. A special note of thanks goes to my parents and family for their unconditional love, prayers, patience, and encouragement throughout my academic journey. Their sacrifices and belief in my abilities have been a constant source of motivation. I pray that Allah Almighty blesses them with good health, happiness, and prosperity. Finally, I would like to thank all those who contributed in any way to the successful completion of this thesis entitled "Lightweight CNN Deployment on ESP32-CAM for Real-Time Image Classification." Their support and goodwill will always be remembered.

CHAPTER 6 REFERENCE

[1] Y. LeCun and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[2] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618," *Genet. Program. Evolvable Mach.*, vol. 19, May 2017, doi: 10.1007/s10710-017-9314-z.

[3] C. Konar, D. Das, K. Jc, S. Singh, and others, "Edge-optimized threat detection with Florence v2 for autonomous surveillance in resource-constrained environments," *PeerJ Comput. Sci.*, vol. 12, p. e3579, 2026.

[4] G. Hinton, A. Krizhevsky, I. Sutskever, and Y. Rachmad, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, May 2012, doi: 10.1145/3065386.

[5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv 1409.1556*, May 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," May 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[7] S. Han, H. Mao, and W. Dally, "A Deep Neural Network Compression Pipeline: Pruning, Quantization, Huffman Encoding," May 2015, doi: 10.48550/arXiv.1510.00149.

[8] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, May 2017, doi: 10.1109/JPROC.2017.2761740.

[9] S. Iqbal, U. Mujtaba, M. Shafiq, D. N. Ahmed, and T. Amjad, "Multi-Level Deep CNN for Glaucoma Detection and Classification," in *2026 7th International Conference on Advancements in Computational Sciences (ICACS)*, 2026, pp. 1–6.

[10] A. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," May 2017, doi: 10.48550/arXiv.1704.04861.

[11] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," May 2016, doi: 10.48550/arXiv.1602.07360.

[12] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *ArXiv*, May 2016.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," May 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[14] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, pp. 303–338, May 2010, doi: 10.1007/s11263-009-0275-4.

[15] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet Things J.*, vol. 3, p. 1, May 2016, doi: 10.1109/JIOT.2016.2579198.

[16] J. Dean *et al.*, "Large Scale Distributed Deep Networks," *Adv. Neural Inf. Process. Syst.*, May 2012.

[17] G. Hinton, J. Dean, and O. Vinyals, "Distilling the Knowledge in a Neural Network," pp. 1–9, May 2014.

[18] R. Singh, D. Kumari, and R. Chandervanshi, "Lightweight Convolutional Neural Network with Residual Attention for Efficient Image Classification," May 2026.

[19] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," May 2019. doi: 10.48550/arXiv.1905.11946.

[20] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," May 2018. doi: 10.48550/arXiv.1812.00332.

[21] G. Litjens *et al.*, "A Survey on Deep Learning in Medical Image Analysis," *Med. Image Anal.*, vol. 42, May 2017, doi: 10.1016/j.media.2017.07.005.

[22] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," May 2019.

[23] U. Gupta *et al.*, "The Architectural Implications of Facebook's DNN-Based Personalized Recommendation," May 2020, pp. 488–501. doi: 10.1109/HPCA47549.2020.00047.

[24] N. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," May 2017, pp. 1–12. doi: 10.1145/3079856.3080246.

[25] W. Khan, E. Ahmed, I. Sahib, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems*, vol. 97, May 2019, doi: 10.1016/j.future.2019.02.050.

[26] V. Janapa Reddi *et al.*, "MLPerf Inference Benchmark," May 2020, pp. 446–459. doi: 10.1109/ISCA45697.2020.00045.

[27] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning," May 2016, doi: 10.48550/arXiv.1611.06440.

[28] J. Donahue *et al.*, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," *arXiv preprint*, vol. 32, May 2013.

[29] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges," *IEEE Signal Process. Mag.*, vol. 35, pp. 126–136, May 2018, doi: 10.1109/MSP.2017.2765695.

[30] B. Park, H. Lee, Y.-K. Kim, and S. Youm, "Progressive Pruning of Light Dehaze Networks for Static Scenes," *Applied Sciences*, vol. 14, p. 10820, May 2024, doi: 10.3390/app142310820.

[31] G. Hinton, A. Krizhevsky, and I. Sutskever, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, May 2012, doi: 10.1145/3065386.

[32] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," May 2016, doi: 10.48550/arXiv.1612.03144.

[33] R. Girshick, "Fast r-cnn," May 2015, doi: 10.1109/ICCV.2015.169.

[34] E. Strubell, A. Ganesh, and A. Mccallum, "Energy and Policy Considerations for Deep Learning in NLP," May 2019, pp. 3645–3650. doi: 10.18653/v1/P19-1355.

[35] C. Corches, A. I. Ilies, M. Daraban, and G. Chindris, "Concept and Challenges in Microcontroller Artificial Intelligence Development," in *IEEE International Symposium for Design and Technology of Electronics Packages, SIITME - Conference Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 270–273. doi: 10.1109/SIITME63973.2024.10814823.

[36] V. Sineglazov and V. Khotsyanovsky, "Camera Image Processing on ESP32 Microcontroller with Help of Convolutional Neural Network," *Electronics and Control Systems*, vol. 2, no. 72, pp. 26–31, Nov. 2022, doi: 10.18372/1990-5548.72.16939.

[37] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *ArXiv*, May 2016.

[38] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet Things J.*, vol. 3, p. 1, May 2016, doi: 10.1109/JIOT.2016.2579198.

[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, May 2015, doi: 10.1109/TPAMI.2016.2577031.

[40] R. Immonen and T. Hämäläinen, "Tiny Machine Learning for Resource-Constrained Microcontrollers," 2022, *Hindawi Limited*. doi: 10.1155/2022/7437023.

[41] H. Bourekouche, "TinyML for Low-Power Embedded Intelligent Systems: A Review." [Online]. Available: https://ssrn.com/abstract=5431335

[42] B. Zoph and Q. Le, "Neural Architecture Search with Reinforcement Learning," May 2016, doi: 10.48550/arXiv.1611.01578.

[43] G. Litjens *et al.*, "A Survey on Deep Learning in Medical Image Analysis," *Med. Image Anal.*, vol. 42, May 2017, doi: 10.1016/j.media.2017.07.005.

[44] M. Kanotra and R. Gupta, "Smart Surveillance System using Deep Learning," May 2023.

[45] C. Gupta and A. Farahat, "Deep Learning for Industrial AI: Challenges, New Methods and Best Practices," May 2020, pp. 3571–3572. doi: 10.1145/3394486.3406482.

[46] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A Survey of Accelerator Architectures for Deep Neural Networks," *Engineering*, vol. 6, May 2020, doi: 10.1016/j.eng.2020.01.007.

[47] L. Al Amin, M. Hossain, R. K. Das, M. Islam, S. Mukta, and A. Tabbakh, "Democratizing AI: A Comparative Study in Deep Learning Efficiency and Future Trends in Computational Processing," May 2026. doi: 10.48550/arXiv.2603.20920.

[48] Y. LeCun and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[49] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618," *Genet. Program. Evolvable Mach.*, vol. 19, May 2017, doi: 10.1007/s10710-017-9314-z.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: a Large-Scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition*, May 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[51] G. Hinton, A. Krizhevsky, I. Sutskever, and Y. Rachmad, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, May 2012, doi: 10.1145/3065386.

[52] Y. Lecun, P. Haffner, and L. Bottou, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, May 1998, doi: 10.1109/5.726791.

[53] M. Sreenivasulu, "Smart AI Waste Classification for Sustainable Management," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 14, pp. 6290–6294, May 2026, doi: 10.22214/ijraset.2026.79436.

[54] S. Naveen and M. R. Kounte, "Optimized Convolutional Neural Network at the IoT edge for image detection using pruning and quantization," *Multimed. Tools Appl.*, vol. 84, no. 9, pp. 5435–5455, Mar. 2025, doi: 10.1007/s11042-024-20523-1.

[55] Y.-H. Chang, F.-C. Wu, and H.-W. Lin, "Design and implementation of ESP32-based edge computing for object detection," *Sensors*, vol. 25, no. 6, p. 1656, 2025.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," May 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[57] C. Szegedy *et al.*, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, May 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[58] S. Han, H. Mao, and W. Dally, "A Deep Neural Network Compression Pipeline: Pruning, Quantization, Huffman Encoding," May 2015, doi: 10.48550/arXiv.1510.00149.

[59] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, May 2017, doi: 10.1109/JPROC.2017.2761740.

[60] V. M. Sineglazov and V. P. Khotsyanovsky, "CAMERA IMAGE PROCESSING ON ESP32 MICROCONTROLLER WITH HELP OF CONVOLUTIONAL NEURAL NETWORK.," *Electronics & Control Systems*, vol. 72, no. 2, 2022.

[47] L. Al Amin, M. Hossain, R. K. Das, M. Islam, S. Mukta, and A. Tabbakh, "Democratizing AI: A Comparative Study in Deep Learning Efficiency and Future Trends in Computational Processing," May 2026. doi: 10.48550/arXiv.2603.20920.

[48] Y. LeCun and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[49] J. Heaton, "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618," *Genet. Program. Evolvable Mach.*, vol. 19, May 2017, doi: 10.1007/s10710-017-9314-z.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: a Large-Scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition*, May 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[51] G. Hinton, A. Krizhevsky, I. Sutskever, and Y. Rachmad, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, May 2012, doi: 10.1145/3065386.

[52] Y. Lecun, P. Haffner, and L. Bottou, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, May 1998, doi: 10.1109/5.726791.

[53] M. Sreenivasulu, "Smart AI Waste Classification for Sustainable Management," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 14, pp. 6290–6294, May 2026, doi: 10.22214/ijraset.2026.79436.

[54] S. Naveen and M. R. Kounte, "Optimized Convolutional Neural Network at the IoT edge for image detection using pruning and quantization," *Multimed. Tools Appl.*, vol. 84, no. 9, pp. 5435–5455, Mar. 2025, doi: 10.1007/s11042-024-20523-1.

[55] Y.-H. Chang, F.-C. Wu, and H.-W. Lin, "Design and implementation of ESP32-based edge computing for object detection," *Sensors*, vol. 25, no. 6, p. 1656, 2025.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," May 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[57] C. Szegedy *et al.*, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, May 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[58] S. Han, H. Mao, and W. Dally, "A Deep Neural Network Compression Pipeline: Pruning, Quantization, Huffman Encoding," May 2015, doi: 10.48550/arXiv.1510.00149.

[59] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, May 2017, doi: 10.1109/JPROC.2017.2761740.

[60] V. M. Sineglazov and V. P. Khotsyanovsky, "CAMERA IMAGE PROCESSING ON ESP32 MICROCONTROLLER WITH HELP OF CONVOLUTIONAL NEURAL NETWORK.," *Electronics & Control Systems*, vol. 72, no. 2, 2022.

[61] R. Immonen and T. Hämäläinen, "Tiny machine learning for resource-constrained microcontrollers," *J. Sens.*, vol. 2022, no. 1, p. 7437023, 2022.

[62] A. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," May 2017, doi: 10.48550/arXiv.1704.04861.

[63] Y. Huang, T. Gong, S. Jang, F. Kawsar, and C. Min, "Energy characterization of tiny AI accelerator-equipped microcontrollers," in *Proceedings of the 2nd International Workshop on Human-Centered Sensing, Networking, and Multi-Device Systems*, 2024, pp. 1–6.

[64] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," May 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[65] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," May 2016, doi: 10.48550/arXiv.1602.07360.

[66] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," May 2018, pp. 6848–6856. doi: 10.1109/CVPR.2018.00716.

[67] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," May 2019. doi: 10.48550/arXiv.1905.11946.

[68] B. Park, H. Lee, Y.-K. Kim, and S. Youm, "Progressive Pruning of Light Dehaze Networks for Static Scenes," *Applied Sciences*, vol. 14, p. 10820, May 2024, doi: 10.3390/app142310820.

[69] B. Marvelous, A. John, and J. Loveth, "Developing AI techniques to create algorithms that minimize computational resources and energy consumption," May 2025.

[70] G. Hinton, J. Dean, and O. Vinyals, "Distilling the Knowledge in a Neural Network," pp. 1–9, May 2014.

[71] A. F. Rakib, R. Rahman, A. Al Razi, and A. S. M. T. Hasan, "A Lightweight Quantized CNN Model for Plant Disease Recognition," *Arab. J. Sci. Eng.*, vol. 49, no. 3, pp. 4097–4108, Mar. 2024, doi: 10.1007/s13369-023-08280-z.

[72] S. Han, H. Mao, and W. Dally, "A Deep Neural Network Compression Pipeline: Pruning, Quantization, Huffman Encoding," May 2015, doi: 10.48550/arXiv.1510.00149.

[73] G. Hinton, A. Krizhevsky, and I. Sutskever, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, May 2012, doi: 10.1145/3065386.

[74] Y. Yue *et al.*, "Automatic segmentation of esophageal gross tumor volume in 18FFDG PET/CT images via GloD-LoATUNet," *Comput. Methods Programs Biomed.*, vol. 229, p. 107266, May 2022, doi: 10.1016/j.cmpb.2022.107266.

[75] Y. Wang, X. Zhang, X. Hu, B. Zhang, and H. Su, "Dynamic Network Pruning with Interpretable Layerwise Channel Selection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 6299–6306, May 2020, doi: 10.1609/aaai.v34i04.6098.

[76] G. Mei and R. Li, "Convolutional Neural Network Analysis Framework based on Cyclic MCU Simulation Software Data: from the Perspective of Limit Testing," in *2022 6th International Conference on Trends in Electronics and Informatics, ICOEI 2022 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 1087–1090. doi: 10.1109/ICOEI53556.2022.9776861.

Ref No.159/SU/SIRC/Center Campus/July 15, 2026

PLAGIARISM CERTIFICATE

For MPhil and PhD

This is certified that the thesis entitled:**Lightweight CNN Deployment on Esp32-Cam for Real Time Image Classification.**

Student Name:**Abdul Hannan**

Reg /Roll No:**SU92-MSEEW-S24-001**

Submitted on date:**13.07.2026**

We have checked this document and found 5% plagiarism/similarity, 1% single source, Findings and Conclusion Similarity 9%, and 0% AI in the document. It is certified that this document meets the HEC criteria.

Supervisor Signature

Librarian

SIRC
SUPERIOR UNIVERSITY
Hassan Khalid
15-07-2026

Research Project / Thesis

Superior Information Resource Center

sana khan

cnn_on_esp_thesis_defense_2 (Findings & Conclusion)

M.PHIL

Document Details

Submission ID
trn:oid:::3618:145644189

Submission Date
Jul 15, 2026, 11:39 AM GMT+5

Download Date
Jul 15, 2026, 11:42 AM GMT+5

File Name
cnn_on_esp_thesis_defense_2 (Findings & Conclusion).docx

File Size
382.2 KB

9 Pages

2,008 Words

11,204 Characters

9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Small Matches (less than 8 words)

Match Groups

15 Not Cited or Quoted **9%**
Matches with neither in-text citation nor quotation marks

0 Missing Quotations **0%**
Matches that are still very similar to source material

0 Missing Citation **0%**
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted **0%**
Matches with in-text citation present, but no quotation marks

Top Sources

7% Internet sources
7% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 15 Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

7% Internet sources
7% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
mdpi-res.com		2%
2	Publication	
Poonam Nandal, Tapas Kumar, Meeta Singh, Mamta Dahiya. "Progressive Compu...		2%
3	Internet	
oulurepo.oulu.fi		<1%
4	Internet	
sciencedocbox.com		<1%
5	Internet	
www.ijraset.com		<1%
6	Internet	
ijsrem.com		<1%
7	Publication	
Iqrah, Jurdana Masuma. "Scalable Polar Sea Ice Classification Over Remote Sensi...		<1%
8	Publication	
S. Prasad Jones Christydass, Nurhayati Nurhayati, S. Kannadhasan. "Hybrid and A...		<1%
9	Internet	
www.jetir.org		<1%
10	Publication	
Sukhpreet Kaur, Amanpreet Kaur, Manish Kumar. "Recent Advances in Computati...		<1%

tayyab ali

cnn_on_esp_thesis_defense_2 (For Similarity)

M.PHIL

Document Details

Submission ID
trn:oid:::3618:145644064

Submission Date
Jul 15, 2026, 11:36 AM GMT+5

Download Date
Jul 15, 2026, 11:41 AM GMT+5

File Name
cnn_on_esp_thesis_defense_2 (For Similarity).docx

File Size
6.0 MB

40 Pages

13,615 Words

77,450 Characters

5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Small Matches (less than 8 words)

Match Groups

- 74 Not Cited or Quoted 5%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

3% Internet sources
4% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 74 Not Cited or Quoted 5%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 3% Internet sources
- 4% Publications
- 0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Publication	
Poonam Nandal, Tapas Kumar, Meeta Singh, Mamta Dahiya. "Progressive Compu...		<1%
2	Publication	
Paolo Ferro, Harinadh Vemanaboina, Chander Prakash. "Computational Techniqu...		<1%
3	Publication	
S.P. Jani, M. Adam Khan. "Applications of AI in Smart Technologies and Manufactu...		<1%
4	Publication	
Aryee, Elias Nii Ayitey. "From Packets to Pixels: A Comparative Analysis of Machin...		<1%
5	Publication	
Bhabani Prasanna Pattanaik, Dhirendra Kumar Shukla, Sambit Satpathy, Kamala...		<1%
6	Publication	
"Medical Image Computing and Computer-Assisted Intervention – MICCAI 2017", ...		<1%
7	Publication	
S. Prasad Jones Christydass, Nurhayati Nurhayati, S. Kannadhasan. "Hybrid and A...		<1%
8	Internet	
mdpi-res.com		<1%
9	Internet	
ijisae.org		<1%
10	Internet	
sciencedocbox.com		<1%