Deep Learning-based Marker Classification and Detection for Augmented Reality in Ethiopian High School Biology Education

Yared Tekalegn Negash

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June 2023

Adama, Ethiopia

Deep Learning-based Marker Classification and Detection for Augmented Reality in Ethiopian High School Biology Education

Yared Tekalegn Negash

Advisor: Dr. Bahiru Shifaw

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June 2023

Adama, Ethiopia

DECLARATION

I hereby declare that this Master thesis entitled "**Deep Learning-based Marker Classification and Detection for Augmented Reality in Ethiopian High School Biology Education**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Yared Tekalegn Negash _____ _____

Name of Student Signature Date

RECOMMENDATION OF ADVISORS/SUPERVISORS

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Deep Learning-based Marker Classification and Detection for Augmented Reality in Ethiopian High School Biology Education**" prepared under my guidance by Yared Tekalegn Negash submitted in partial fulfillment of the requirements for the degree of Master's of Science in Computer Science and Engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Bahiru Shifaw	_____	_____
Major Advisor/Supervisor	Signature	Date

APPROVAL PAGE

I, the advisors of the thesis entitled "**Deep Learning-based Marker Classification and Detection for Augmented Reality in Ethiopian High School Biology Education**" and developed by Yared Tekalegn Negash, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Bahiru Shifaw	_____	_____
Major Advisor/Supervisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by Yared Tekalegn Negash have read and evaluated the thesis entitled "**Deep Learning-based Marker Classification and Detection for Augmented Reality in Ethiopian High School Biology Education**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master's of Science in Computer Science and Engineering.

_____	_____	_____
Chair Person	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGMENT

I am, now and forevermore, indebted to the eternal and unerring selfless hands of God cleansed with pure love and grace. There is not a glimpse of an instant from any moment of my life that could aggregate to make me the person I am today without the aid and presence of God. Lord, you have always been here along with your mother, I can’t thank you both enough!

I would like to pass on my deepest gratitude to my thesis advisor Dr. Bahiru Shifaw, who has been patient and a presence of positive encouragement entailing wise guidance. It has been an honor having a wonderful academic companion in the entirety of this thesis work.

One of the numerous gifts that God has blessed me with is my family and most of all my parents. The relentless push and goodwill advice on any of my personal, spiritual, and academic life from my father, Tekalegn Negash, has been nothing but an arsenal for every dare of life. The rinsing love, the playful banter, and the uninterrupted attention from my mother, Gudaye Tafesse, have instilled in me a polite and hopeful future. The sacrificial and backbreaking life these two people have bravely endured for the good of several individuals including me and my brothers, is something I will hold dear till the end of my time. Thank you, Indaya-Baya.

I would also hate not to mention other individuals who have been nothing but a buoyant presence in these two years of my life. These individuals include my SIG mates, classmates, and friends I have known since I was an undergraduate student. I would love to extend a special thank you to Mebatsion Sahle for her support. May God be with them in their adventures to come.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF SAMPLE CODES

LIST OF ACRONYMS AND ABBREVIATIONS

AR	Augmented Reality
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CSE	Computer Science and Engineering
CSPDarknet53	Cross-Stage Partial DarkNet53
CV	Computer Vision
DenseNet	Dense Network
DL	Deep Learning
E-ELAN	Extended Efficient Layer Aggregation Network
emoFBVP	emotion Face, Body gesture, voice, and physiological signs
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
IoU	Intersection Over Union
JPEG	Joint Pictures Expert Group
NASA	National Aeronautics and Space Administration
NasNet	Neural Architecture Search Network
PNG	Portable Network Graphics
RCNN	Region-Based Convolutional Neural Network
ResNet	Residual Network
QR	Quick Response
SeNet	Squeeze and Excitation Network
SSD	Single Shot Detector
SVM	Support Vector Machine
U. S	United States
VR	Virtual Reality
YOLO	You Only Look Once

LIST OF NOTATIONS AND SYMBOLS

Notation	Meaning
$\mathbb{D}$	Domain
$\mathfrak{X}$	Feature Space
$\mathbb{P}^X$	Marginal Probability
$\mathbb{T}$	Task at hand
$\mathcal{Y}$	Label Space
$f(\cdot)$	Function
$\mathbb{D}_s$	Source Domain
$\mathbb{T}_s$	Learning Task from source
$\mathbb{D}_t$	Targeted Domain
$\mathbb{T}_t$	Learning Task from target
M	Number of classes
N	Number of rows.
σ	Softmax
$\vec{z}$	Input vector
e^{z_i}	Exponential function of the input vector
k	Amount of classes
e^{z_j}	Exponential function for the output vector

ABSTRACT

In recent decades, the significance of Augmented reality (AR) in its application for several fields of study has been taken into consideration. The same could also be stated for Deep Learning (DL). The combined effort of these two concepts has been deemed crucially important in medicine, education, the military, marketing, games, entertainment, sports, etc. Applying DL in augmented reality marker detection on educational books is the focus area of this research. This study aims in finding an efficient deep-learning solution suited for marker classification and detection of augmented reality based on Ethiopian Educational books. Using a pre-trained model is the best option for marker classification due to the low amount of data available. The pre-trained models used as feature extractors for classification include MobileNetv2 and MobileNetv3. The training on detection has been done with the YOLOv7 detection algorithm. Secondary school books published by the Ministry of Education (MOE) are chosen for this study, specifically biology from grades 9 to 12 since it is a subject that has ample abstracted figures that require more explanation and demonstration than a simple description with words. The dataset gathered from each book is by choosing at most 20 figures as markers and building the dataset from there, by keeping several instances of the environment in mind, and to help enrich the dataset, a synthetic dataset is used. The overall dataset size consists of over 17,000 images created using the book figures for the training. The proceeding process has been preprocessing and feature extraction then was followed by the deep learning process which is to train the enhanced data. For classification, MobileNetv3Large has performed better with 96.21% accuracy than the rest of the pre-trained models used as feature extractors. The best model for detection has been used to create an AR system that overlays information on top of the detected markers. In comparison, we have achieved a detection with better performance and higher inference speed by using the proposed architecture than the previous works.

Keywords: *Augmented Reality, Marker Detection, Marker Classification, Deep Learning, Synthetic Data*

CHAPTER ONE

1 INTRODUCTION

1.1 Background of the Study

Augmented Reality (AR), to define it bluntly, is an added reality. To define AR more specifically and clearly, the individual experiencing an Augmented reality is not disconnected from the real world instead what they see is an additional simulated information on top of the real world (Slater et al., 2020). Hence, why we defined it as added reality. The best and simple example of AR is a sticker on our smartphone camera, it is a layer of information added as soon as it recognizes our face.

Mainly there are two types of AR, Marker-Based AR, and Marker-Less AR. Marker-Based AR works by making use of an identifiable marker or token, it can be a simple QR Code, a water bottle, or a figure on a book. Once the system has identified the marker it can then proceed to overlay information on top of the identified marker. Marker-Less AR as the name suggests doesn't require a marker to identify a surface. It makes use of a built-in sensor and camera information of the device to get a better scope of the environment and it performs the AR experience on top of the identified surface.

AR is not a novice idea in fact, the research in AR has been around for over 50 years now (Billinghurst et al., 2014). The first purposeful AR was introduced in 1992 in the Armstrong laboratory of the U.S Air Force in making use of virtual fixtures (Louis b Rosenberg, 1992.). Since then, AR has seen its way through every possible array of domains both commercially and non-commercially. From entrepreneurs to scientists and researchers to fashion designers and other professionals are diving into the art and science of Augmented Reality. AR is used in gaming, marketing, entertainment, and several other application domains.

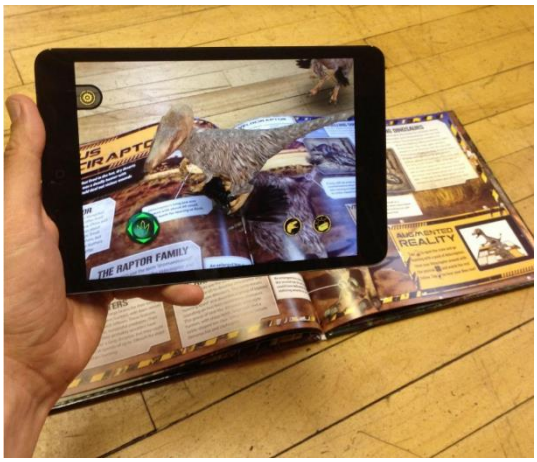
Most AR applications apply computer vision techniques to achieve marker detection tasks. AR can convert an abstracted text or object on paper to a visually descriptive graphical object by over-imposing it onto a real-world object which clarifies the concept of the text or object. Once the AR is well aware of the object in the real world it can now overlay information on top of the object. The information augmented on top of the object is known as Augogram. An Augogram is a computer-generated information used by AR. And the art of developing an

Augogram is known as Augography (Ramamoorthy & Kiran, 2021). However, Augogram is not necessarily the only information over-imposed on the object as we've mentioned earlier, normal videos, and images can also be an option.

Deep learning doesn't need any introduction as it is an integral part of every technology being developed in recent years. There is a lack of research around education that makes use of AR and DL. Those that have performed research on this area all try to attempt to solve problems such as training skills, teaching, and guiding students by using these two techniques. All of the research use either marker-based or marker-less AR. Alternatively, researchers try to train the data by training the marker posed as the object for detection, which will be discussed in the proceedings topics.

Augmented reality has been integrated with deep learning as the emergence of both began and researchers started to realize their combined impact. Augmented reality provides real-time interaction between the individual and the digital world while keeping the interactivity smooth and enjoyable. Augmented reality can benefit from the advantages of deep learning in designing intelligent, user-focused, user-friendly, informative, and interactive systems (R.B. Ravi Varma et al., 2021).

In this study, we have selected secondary education books published by the Ministry of Education (MOE) specifically from grades 9-12. The reason behind this is that students in high school are more subjected to concepts that require more activities, tasks, and practical approaches that require using laboratory experiments, demonstrations, and simulations (Daba & Anbesaw, 2016). Ethiopian secondary schools are foundationally missing such resources and their implementations as we will discuss in detail in the coming sections. This study will identify and propose a solution for the problem by focusing on AR as the main tool and the enhanced capability of deep learning will be used to make the solution more effective.

a. 

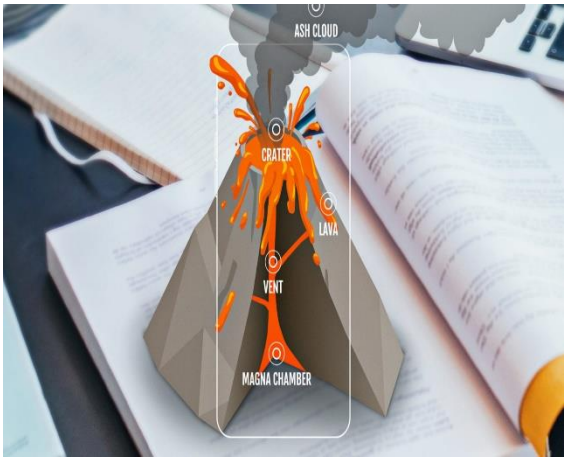
b. 

Figure 1.1 Augmented Reality in Education
a. (Education, 2015), b. (Blippar, 2020)

1.2 Motivation of the Study

The impact of augmented reality is being magnified in recent years, especially with the birth of machine learning. In Ethiopia, there are very limited application areas that are making the necessary leap assuring the use of this solution-rich domain, especially in education. Students can have a deeper and more meaningful understanding of a certain topic if they were approached with graphically detailed animations and relevant videos. This will open up their imagination and guide their creativity innumerably, they wouldn't have to be forced to memorize every word in the book. It is bothering that little or no research is being made on this area. This is what motivated this study, to make the first mark of progress in research of amalgamating the concepts of augmented reality and deep learning for Ethiopian Education.

1.3 Statement of the problem

It is recommended to have a practical and simulated environment in education along with theoretical feedback as it has a favorable impact on students which will help them in their academic endeavors and initiate them to enroll in fields of natural science (Jabeen & Afzal, 2020). The trend of applying heavy measures to teaching theoretical concepts while disregarding practical methods has been an increasing habit in developing countries which limits the students' scope of knowledge and motivation (le Huy and Nguyen, 2020). The chalk-and-talk method as it is infamously known is an unattainable burden to students which wrings out an unproductive and dismissive individual to a country and the world.

In Ethiopian schools, the implementation of education aided with practical study is very weak (Bekalo et al., 2000). Ample researches are available on secondary schools that identify the lack of laboratories and the practical implementation of teaching. For instance, a study on schools of Borena zone, Mekele, Wolaita, Jimma, and Afar regions depicted the absence of laboratories in some schools also the lack of equipment, organization and making use of them, and lack of time allotted for laboratory sessions were observed in several other schools in the regions with added notes that students are losing hope in science-related subjects and no efforts were being made by the school and regional institutes to remedy the problem (Muleta Daba et al., 2016), (Bekele et al., 2010), (Gogile Zengele & Alemayehu, 2016), (Daba & Anbesaw, 2016), and (Tesfamariam et al., 2014). All suggest that students are losing interest in STEM field subjects with no immediate available solution, which is a strain on the development of the country.

Out of the few available tools in CSE to solve the problem, we propose AR. AR can serve as, if you will, a digital laboratory; since its capability of simulating and demonstrating concepts to students prove its similar purpose to a real-life practical purpose (M. Wang et al., 2018). Additionally, students can get access to knowledge where it is too dangerous to interact with such as chemicals, and also things that are difficult to just visualize such as microorganisms and outer space (M. Wang et al., 2018). Furthermore, AR can help schools from buying both expensive and inexpensive laboratory equipment and also complex apparatus that could more or less be available easily with the help of AR (Bower et al., 2014).

Furthermore, there is an immeasurable lack of research that has been performed to use this mode (AR and DL) of technology on educational books here in Ethiopia and abroad. Research involving marker classification is not being implemented which has the advantage of providing an easier and cheaper path to detection. However, on the detection front, there are researches available but for example, the previous work has made use of YOLOv4 which uses a backbone architecture with a high number of parameters and has lower accuracy than other available architectures. This research will try to cover the detection along the classification of markers front so it can serve as a baseline for other researchers that want to develop AR for students using figures on books as markers.

1.4 Research Questions

During this research endeavor an effort will be made to give a viable answer for the following questions.

RQ1. What are the different techniques of acquiring data specified for the augmented reality learning model from educational books?

RQ2. What are the improvements observed by using deep learning in custom data, as opposed to using computer vision techniques?

RQ3. What hyperparameter improves the performance of the classification models?

1.5 Objective

1.5.1 General Objective

The main objective of this study is to use an efficient and more appropriate pre-trained deep learning model for marker classification and develop an AR marker-detecting model based on Ethiopian Secondary School Biology Books.

1.5.2 Specific Objective

The specific objectives of this study are as follows:

- To design a custom network for marker classification using pre-trained models as feature extractors.
- To collect and label the data from educational books and create synthetic data.
- To outline a preprocessing method that improves the quality of the data.
- To train the proposed model of classification and detection with the acquired data set.
- To analyze and identify the performance of the trained model.
- To supplement and modify the model with correct regularization techniques.
- To create an augmented reality experience.

1.6 Scope of the study

The focus of this research is to use a deep learning pre-trained model to efficiently classify marker figures of educational books. It also considers detecting specified images even under varying environmental conditions such as illumination change, orientation change, blurriness, and change in the source of the figures. All this is done by making use of deep learning

approaches to classify and detect marker targets. Sampled candidates are selected to be target markers for the classification and detection from the specified books.

1.7 Limitation

The constraints revolving around this thesis work are given as follows.

- There is no available dataset publicly that could be garnered for the study.
- Due to the high number of classes required for the model, it is difficult to gather data that can fit every class that is rich enough to make the model more efficient.
- The time resource available will not allow more figures to be chosen from each book therefore the data will be limited considering the vast number of figures per book.
- The lack of well-documented available resources and libraries has limited us from creating an immersive AR using the DL model. Instead, a two-dimensional image is overlayed on a planar view of the marker detected.
- Even though there is research available on the general domain, there isn't enough research for this specific study.

1.8 Contribution and Significance of the Study

1.8.1 Contribution of the Study

The following are lists of the contribution this study can provide

- Provide a working method to develop a deep learning-based marker classification and detection model that can be used to create an Augmented Reality on books that are designed for Ethiopian Biology Educational Books.
- Collection and use of real data from Ethiopian high school biology books considering several instances of the environment such as illumination, orientation, etc. which provides a working method of acquiring datasets for similar research.
- Use of MobileNetv2 and MobileNetv3 as feature extractors and design of a custom network to yield good performance results in marker classification. While contrasting the deep learning pre-trained models in the task of classifying the markers collected from the books and identifying the one which provides a more efficient and accurate result.

- Use of state-of-the-art detection algorithm YOLOv7 and compare its performance with YOLOv4, an algorithm used in the previous works.
- Creation of a 2D augmented reality after markers have been detected.
- Analyzing previous works of literature on the same topic to demonstrate the gaps and provide a solution.

1.8.2 Significance of the study

So far there are several methods of AR used in education such as Discovery-Based Learning, Object Modeling, Game Based Learning, and last but not least AR books which will be the area of our focus (Majeed & Ali, 2020). Studies have shown applying AR in books has an escalating positive impact on students. Some of the measured ones include increased students' motivation to learn, improved student participation, and signified high levels of fulfillment. The only improvement identified in the increased attention of students was recorded in this type of AR (Majeed & Ali, 2020). This identifies that the enhanced education system for students that combines teaching techniques with augmented reality can immensely help the educational progress of the country. That is students have the capability of studying their books with an application that allows them to grasp the concept by only pointing their phone camera at the page or the image in the study, and seeing a rendered animation or tutorial videos or even a reference on the topic can immeasurably help them understand quickly. It also helps them save time by making them refrain from researching resources they have little or no idea about.

The following are the benefits of AR in books:

• **Enhanced engagement** – Augmented reality brings books to life with visuals, sounds, and interactive content that encourages readers to become fully immersed in the topic at hand. It's a great way for students to learn about the world around them in interesting and exciting ways.

• **Increased accessibility** – Textbooks can now easily be converted to AR-enabled digital versions. This allows students who may have difficulty accessing physical copies of the same materials as their peers.

• **Improved retention** – Studies have shown that students who use augmented reality for educational purposes tend to retain information for longer periods than those without AR

access. They are also more likely to recall this information later on, as augmented reality encourages deeper learning by helping learners visualize concepts more easily and accurately.

• **Enhanced creativity** – With AR, students will have an enhanced imagination ability to how things work, which will allow them to have a sharp perspective in solving problems. By allowing readers to customize stories in a completely new way, this form of storytelling expands the traditional boundaries of publishing.

Deep learning can boost the application performance of Augmented Reality by allowing the target markers to be trained by vast amounts of data which also applies efficient graphics processing. The problems of object detection that come due to illumination, orientation, and several available means of the objects can be solved with the aid of deep learning (Lampropoulos et al., 2020). Our deep learning-guided model development can aid the birth of such systems to be available for the public with minimum effort.

1.9 Operational Definition

Marker

Any physical object that is utilized to instantiate the display of virtual information in an Augmented Reality (AR). A marker is an image with two-dimensional (2D) and three-dimensional (3D) characteristics that can distinctively be identified by an AR system.

The characteristics of a marker include:

- The marker should be unique and identifiable
- The location of a marker should allow the AR to recognize it easily
- The size of the marker should be compatible with the AR.

Examples of a marker include QR Codes, Buildings, Logos, Magazines, etc.

1.10 Organization of the Thesis

This work of research is a juxtaposition of seven chapters, as identified below.

Chapter One: This chapter will identify the introductory concepts of the study with provides clarity on what will be achieved, how will it be achieved, and the purpose of the study.

Chapter Two: This chapter introduces the core conceptual and theoretical areas of the research as well as the reviewed literature.

Chapter Three: In this chapter the methodology used in achieving the aims of the research are provided, which includes data collection, preprocessing, tools used for design, templates, and platforms used for performing tasks, and evaluation metrics.

Chapter Four: This section justifies how the solutions proposed for the experiment combat the problem.

Chapter Five: Concerned with the implementation of the research it contains the appropriate actions taken to achieve the results.

Chapter Six: The results observed are discussed here in this section of the document, with appropriate descriptions, graphs, and tables to demonstrate the outcome.

Chapter Seven: Here the problem, method, and result of the research are inscribed as a conclusion and identifies works to be done in the future.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 Introduction

Augmented Reality and books together are being implemented at schools in different countries of the world, especially in developed countries. It is intended to help students understand topics that are too difficult to grasp the ideas from and too complicated to describe and teach with words and text. This method has shown significant results in improving students' learning capabilities with several other benefits, not to mention that it has made it easier for teachers to instruct students with little or no hassle (Alzahrani, 2020). Researchers are now incorporating AR with education even with the help of deep learning. This approach is being practiced and it is still in the development process. However, very few studies have managed to bring the concepts to books to identify the practical problem it solves along with their inevitable technical problem. The learning process that has been implemented in the studies is used with pre-trained models due to a lack of rich data. One of the biggest challenges in training an AR marker is that each class is available only in one form, there is only one marker available, and deep learning demands the dataset to be available in large quantities (Le et al., 2020). This problem has been remedied to a degree by making use of synthetic data. The marker in the class is taken and imposed over other random images as a background which are acquired from different sources on the internet. The next step is to train these new images with deep-learning algorithms.

Deep learning algorithms will provide a solution for detecting these markers with enhanced feature extraction and classification capabilities. Out of the most used and highly praised deep learning algorithms CNN is one of them which is complementary for face detection, computer vision, object detection, speech processing pattern recognition, and several other deep learning services(Alzubaidi et al., 2021). CNN highly depends on the careful selection of hyper-parameter to achieve better performance, if not well-regulated and properly chosen the outcome of the training will yield a very poor result (Alzubaidi et al., 2021). Models such as MobileNetV2 and other major deep-learning models have been built with the help of CNN.

Standard book figures are used as markers in the implementation of the training process to produce the learning model. CNN is composed of several layers each of which has its own unique set of values and weights. These layers are the input layer, several hidden layers, and finally output layers. Hidden layers lie in between the input and output layers. Each layer is built using entities of what's called, in deep learning, neurons, they are mimics of biological neurons. The set of values and weights for each neuron will be chosen randomly and updated through the training process called backpropagation (Kriegeskorte & Golan, 2019).

Even though little research is being performed on this area, they are finding ways to maneuver around this problem and attempting to find a solution. Since the research requires an ample amount of data, dataset-creation methods are also being considered in that research.

2.1.1 Marker-based and Marker-less Augmented reality

Marker-based AR works by using inalterable distinctive tokens usually around the object to be detected. The computer vision (CV) algorithm can recognize this marker and make necessary calculations to make a positional relationship between the marker and the camera. The computer-generated information can then be augmented on top of the object while in the background the CV algorithm constantly checks the position of the camera from the object being detected by keeping an eye on the marker (Oufqir Zainab and El Abderrahmani, 2020).

Marker-less AR works with no marker as a target token. To achieve the job of detecting the target image, it is guided by hardware equipment such as sensors and other means to detect the environment and its surrounding. Once the system is well aware of the surrounding and has been identified, the necessary information will be overlaid on top of the surroundings while the algorithm keeps track of the change in sensor information to make the positional change of the overlayed information (Oufqir Zainab and El Abderrahmani, 2020).

2.2 Deep Learning

Deep learning is a field of computer science that consists of algorithms that can help build neural network models which are large using large datasets as input and therefore can make precise decisions (Kelleher, 2019). This is all possible with the array of current technological advancements such as Graphical Processing Unit (GPU), the easy access to make large image

datasets due to the high quality and accessible cameras, and powerful computers with big processors has allowed deep learning to reach its prominence stage.

Deep learning is applied in different application areas where Artificial Intelligence is required to provide better results. Among the fields where DL is used are natural language processing, speech recognition, and medical applications. The application area where DL shines even better is in Computer Vision. Under this application area, DL is used in image segmentation, face recognition, object detection, image semantic segmentation, marker detection, video object segmentation, and several others (Dong et al., 2021). Taigman et al., 2014 developed a model for face recognition using face modeling in 3D and applied transformation using piecewise to produce attributes with smaller dimensions and conjured a 27% reduction of error compared to the current models. Zhao et al., 2019 surveyed object detection methods by applying deep learning and segmentation methods. Rahnemoonfar & Sheppard, 2017 introduced a crop yield estimation project making use of Inception-ResNet Architecture and applying this model to synthetic data they were able to create the yield estimation system. Synthetic images were used for training and real images for testing which has yielded a high accuracy with much less annotated real images for training. Ranganathan et al., 2016 wrote a paper on emotion recognition systems by applying Deep Belief Network on emoFBVP a dataset that is multimodal recordings of facial expressions, vocal expressions, and other alternate features and contains 23 alternate emotions in both audio and video format.

Deep learning has the trend of taking data or arrays of data as an input which then guides that data through a series of layers each equipped with their algorithmic properties to yield an output of classification. Deep Learning also has the extra ability of feature extracting without the need for direct supervision from humans which is one of the expertise of computer vision. Computer vision-based traditional algorithms have come a long way to provide marker detection ability for Augmented Reality related applications. Yet now with the maturity of deep learning, it would be wise to consider the possibility of utilizing DL algorithms for marker detection.

The traditional Computer vision method to detect markers is reliable with high accuracy but with major limitations. The first limitation is that markers are identified in different situations but have difficulty identifying differences in chrominance or color (Siltanen, 2012). The other

limitation is that the algorithm should be able to locate and identify the pose of the camera by detecting the marker but the process in which the marker is identified requires a capable device with high-end processing power. The marker detection process can be classified into five major steps with the presupposition goal of finding the outline of markers then identifying the location corners of markers and having an affirmative idea that the marker has been identified and deciphered.

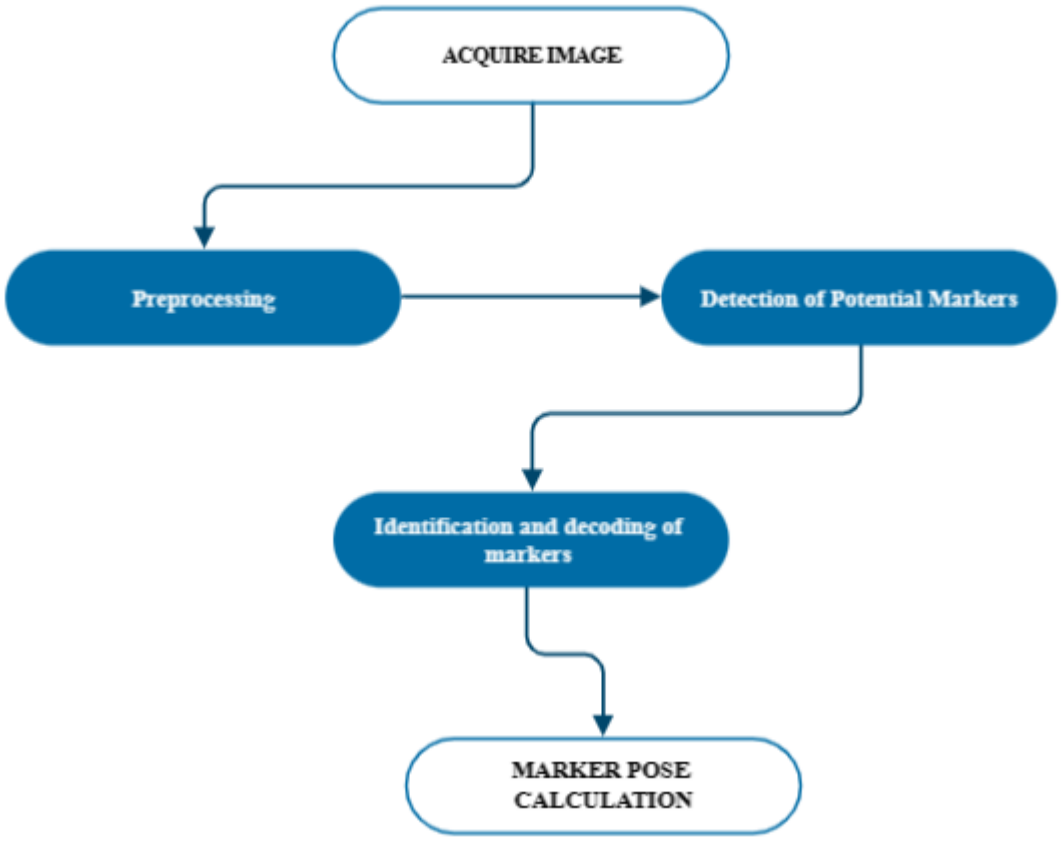


Figure 2.1. Traditional CV Process of Marker Detection

Acquiring the image with its intensity, performing preprocessing which involves correcting distortion, detection of lines, and detection of markers, then in the major process follows filtering possible markers from non-markers, the proceeding step is to identify then decode markers with the available templates, finally computing the pose of the marker to identify the location are the steps taken to identify and decipher markers using Computer vision algorithm (Siltanen, 2012). The figure below depicts this process which detects the square marker which has a similar appearance to a QR code by distilling it from the pen and the circle object.

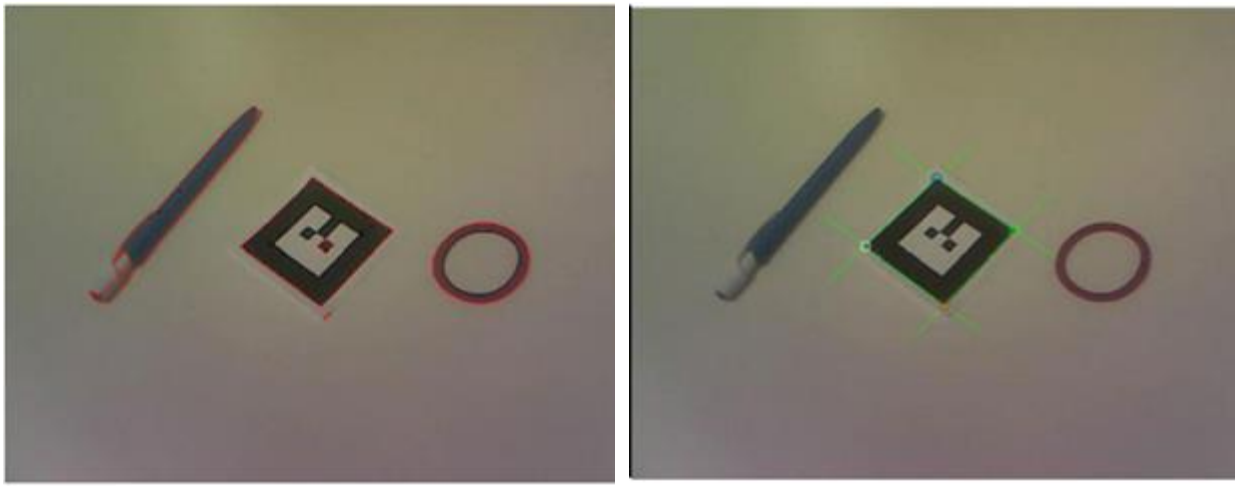


Figure 2.2. Marker Detection using the CV algorithm (Siltanen, 2012)

In the case of deep learning, however, the feature extraction process is done along with the training process of the model. From the given data, deep learning will undergo a process of identification of patterns in each given class of markers to automatically compute the definitive description of each marker (O'Mahony et al., 2020). This makes DL much more robust than traditional CV algorithms; it allows the arduous struggle of feature extraction and the low accuracy output of classifying features with shallow architecture to be much easier and more precise.

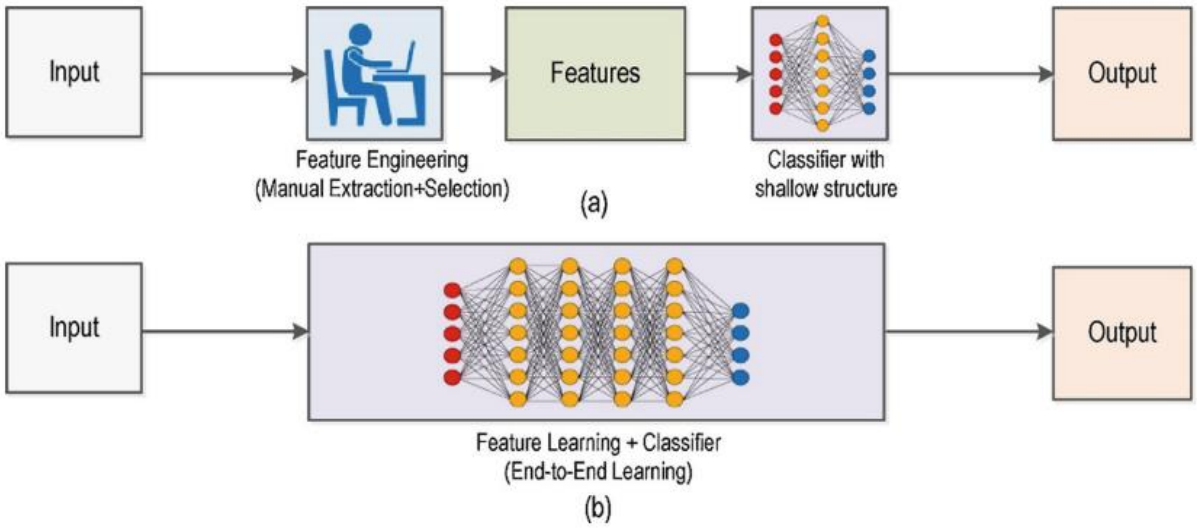


Figure 2.3. (a) traditional CV workflow, (b) DL workflow (O'Mahony et al., 2020)

2.2.1 Synthetic Data

Synthetic data is a method of generating datasets that are imitations of real ones. It is supplementary when there is very little data in hand and it poses an alternative option for law-protected data. Since deep learning is the most efficient when tooled with large amounts of data it will be necessary to have a rich and clean dataset to squeeze better performance from the algorithm, but when it comes to the unfortunate alternative option which is lack of data then synthetic data will be the handy option. Synthetic data could be of various types among them the first is synthetic composite imagery which is to inject a new image on the target image as a background. The other type could be virtual synthetic data which is to create complete synthetic data without using real data(Man & Chahl, 2022).

Synthetic data can be generated using a deep learning algorithm known as Generative Adversarial Network (GAN) and its variations such as BigGAN, LOGAN, CycleGAN, StyleGAN, etc. (Vega-Márquez et al., 2020). The other method of creating synthetic data is by manual generation, this is where human supervision comes in, the composite image is created one by one. This is a very arduous and time-consuming method and poses a threat to the number of data to be generated (Man & Chahl, 2022).

Training with synthetic datasets however creates a domain gap when the model is tested on real data and synthetic data (Alkhalifah et al., 2022). The model performs well on synthetic data but not on real data as the synthetic dataset lacks to hold all the available features on real data. Solutions that can mitigate this problem have been proposed by several studies. Among the solutions, one of them is domain randomization, which is to add or take several variations of the same data as much as possible (Tremblay et al., 2018). An alternate solution is also available namely structured domain randomization which considers the effect of the structure and context of the scene to minimize the domain gap created by synthetic dataset (Prakash et al., 2018).

Figure 2.4. Domain randomization on a car image. (Tremblay et al., 2018)

Almost all research performed so far that encompass marker detection with deep learning have all used synthetic data to create diversity in their dataset. Granted they have also used data augmentation along with alternating the illumination and blurriness behavior of the markers. By doing this they have been able to generate great results in their training process.

2.3 Transfer Learning for Feature Extraction

Transfer learning is a learning mechanism by which the algorithm draws out existing knowledge from another case to make the learning performance better for a target case (Yang et al., 2020). It helps utilize the defined knowledge for applications with problems of data scarcity and for time-sensitive cases. It provides aid for applications where labeling is costly and gathering data is time-consuming and troublesome. This transfer process involves successfully passing the knowledge to the new case without losing the accuracy and performance of the pre-existing model. Transfer learning can be classified into a few categories, feature extraction is one of them. A common feature will be mined from the new data to be trained for a custom model.

To define transfer learning more rigidly we need to be familiar with basic concepts like domain and task. Let's assign the domain as $\mathbb{D}$ which holds two components feature space $\mathfrak{X}$ and marginal probability $\mathbb{P}^X$ and each input x $\in \mathfrak{X}$. Two different domains may contain different feature spaces and different marginal probabilities(Yang et al., 2020). Assuming we have $\mathbb{D} = \{ \mathfrak{X}\ \mathbb{P}^X \}$ then task $\mathbb{T}$ has label space $\mathcal{Y}$ and a function $f(\cdot)$ which can be represented as $\mathbb{T} = \{ \mathcal{Y}, f(\cdot) \}$. $f(\cdot)$}is a function used for prediction on new instances(Yang et al., 2020).

Now that we have defined the necessary components, we can come to a more general definition of transfer learning as follows.

> "Given a source domain $\mathbb{D}_s$ and learning task $\mathbb{T}_s$, a target domain $\mathbb{D}_t$ and learning task $\mathbb{T}_t$, transfer learning aims to help improve the learning of the target prediction function $f_t(\cdot)$ for the target domain using knowledge in $\mathbb{D}_s$ and $\mathbb{T}_s$, where $\mathbb{D}_s \neq \mathbb{D}_t$ or $\mathbb{T}_s \neq \mathbb{T}_t$." (Yang et al., 2020)

Currently, there are several pre-trained deep learning models available that are very efficient to be used for transfer learning. Some of the most profound models include EfficientNet, ResNet, DenseNet, Inception, MobileNet, AmoebaNet, NasNet, and SeNet. These models are trained with large datasets and with well-designed algorithms which makes them reliable for transferring knowledge to medium or small cases of training. MobileNet is the model used for transfer learning compatible with mobile phone CPU, which makes it suitable for real-time applications for mobile phones.

2.4 YOLO Object Detection Algorithm

Yolo (You Only Look Once) is an object detection algorithm that has integrated the combined ability of feature extraction, chosen box extraction and classification of objects approaches to a neural network (Liu et al., 2018). YOLO has revolutionized the object detection problem by outperforming prior algorithms. YOLO has several versions and it is continually being improved by researchers from different organizations.

The first YOLO version was introduced in 2016 by Redmon et al. and they considered the concept of detection as a regression problem that revolves around bounding boxes with class probabilities. The image is divided into an $N \times N$ grid and then the probability for each grid is calculated based on the given bounding box B along with the confidence and class probabilities C. This information is then encoded as a tensor with the following attribute:

$$S \times S \times (B * 5 + C)$$

The main driving force for YOLO detection architecture is Convolutional Neural Network (CNN). The initial layer is responsible for extracting features while the fully connected layer is assigned to predict the output result.

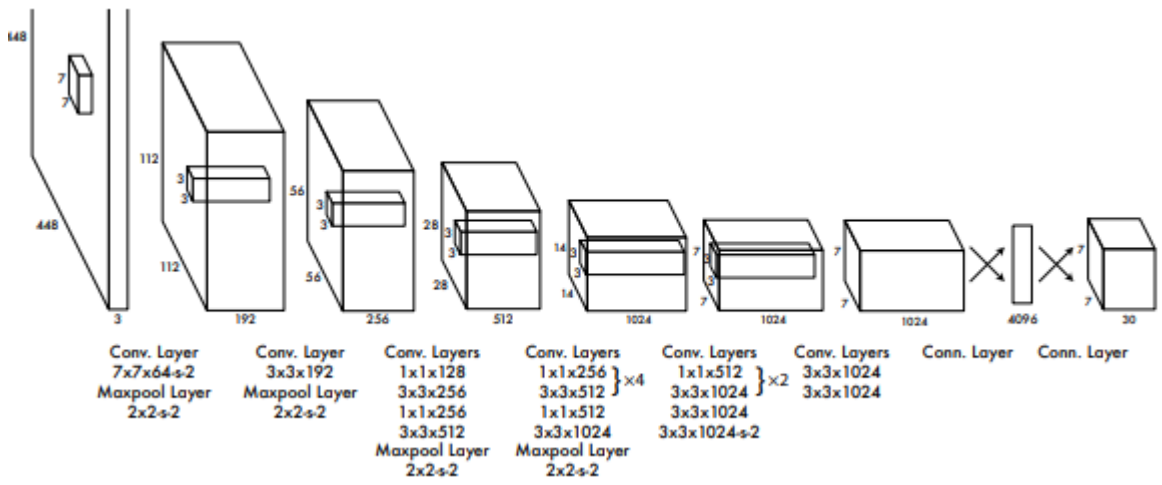


Figure 2.5. YOLO Architecture.

2.5 Related Works

Le et al., (2020) attempted to incorporate the aid of Machine Learning into the detection of markers by depending heavily on synthetic datasets. They made use of YOLOv4 to train their dataset. The dataset was created synthetically using playing cards as markers, data augmentation and illumination change are also applied to create a rich dataset. Multiple markers on the same scene, scaling the markers concerning the scene, rotation, and random location of markers on the scene were the basic formats used to create the dataset. 2000 background images were scrapped from Google to overlay the markers. The training class contains 6 classes and 13 classes reduced from 52 by clustering the cards with their ranks. For comparison purposes, they trained the model with the Kaggle dataset along with their dataset created synthetically. They compared the results on both datasets and found that the Kaggle dataset created overfitting while theirs provided a satisfactory result.

Le et al., (2021) introduced a system that integrates machine learning with AR to detect and track augmented reality marker targets. The research was done on four image targets: trading/business cards, posters, children's educational books, and food advertisements. The dataset is collected synthetically since very little available data exists. The dataset is generated by enforcing the image on different backgrounds, and different light illusions, from various views and orientations. The authors used a Convolutional Neural Network model called YOLOv3 and they used different versions of v3 to train their data for 80 iterations with 80% of the dataset, the remaining was used for validation. The trained model's performance is

tested under three different lighting scenarios which are normal indoor light, dim light with a black background, and direct artificial light or high-contrast lighting.

Estrada et al., (2022) also designed and developed a system with AR and DL that can be applied in lab training for engineering students. They also used the Convolutional Neural Network algorithm to build their model. Initially, the model was trained with 643 images collected, from different orientations and sides, with each image having a 4032 x 3024 size. They used TensorFlow records to store the dataset in binary format which makes the model efficient and occupies less space. Making use of a CNN pre-trained model, MobileNet-SSDv2 they made the model for detecting lab equipment efficiently. They also used an additional model RCNN and compared the result with MobileNet-SSDv2 and the performance of the latter has proved significant. The efficiency of the model was tested in real-time using high-end smartphones which are Samsung S21 Ultra and One Plus 6T with RAM capacity of 12 GB and 6 GB respectively.

Dash et al., (2018) developed a marker-based AR for kids with the sole purpose of creating suitable learning environments. It is done on the 26 letters of the English alphabet, by designing each alphabet on a board that has the required marker on top of it. CNN is the main tool in building this system for kids. The process by which this development proceeds is by first identifying the marker in view, then the system identifies the marker, it follows by calculating the marker's pose, it then simulates a 3D image on top of the identified marker. Here CNN is applied to guide the marker detection. Another algorithm that is used to identify the marker and also used to compare it with CNN is the support vector machine (SVM) classifier. The dataset for marker detection was implemented here also using the synthetic marker dataset creation method. By considering the 26 letters they've managed to create 104 classes and 350 samples by which the dataset was a total of 36,400 images. Then the dataset is fed to the CNN classifier that trains and builds a model for the input using 75% of the dataset input, the remaining is used for validation and testing. The result showed the markers trained by CNN presented more satisfying results than the ones trained with SVM.

Alhalabi et al., (2021) tried in providing engineering students with a hand-written circuit diagram solver using neural networks and augmented reality. It takes a static image of a circuit diagram drawn by hand and identifies the problem with the model created by the neural

network and augments the answer along with the digital diagram version of the drawn circuit. Here the neural network used much like the others is CNN. However, CNN didn't seem to perform well, as the input is dynamic and could change very rapidly depending on the user input which makes the algorithm underperform when the object exhibits a spatial change in position. Instead, what they found necessary is a different neural network called Capsule Network Classifier. It contains neurons that work together to identify a target feature by considering other factors like thickness, size, skewness, etc. This approach was necessary because as mentioned the input is dynamic by nature and it should consider other factors onto consideration and has proved this with the admitted result. They used over 800 images for the training after labeling them. The specific classes include resistors, inductors, capacitors, voltage sources, and current sources.

Lindner et al., (2019) developed an AR system that will help students visualize the distance of the moon to earth. They used AR developing tools such as Vuforia and Unity to create the application. The application included distance simulation along with tide simulation of the earth to the moon as a part of the astronaut's perspective. Moon position along with its distortion, the moon shadow has been processed by considering two markers. A video is displayed on top of one of the markers while the other serves as an identifier for a 3D model. The videos and 3D models have been extracted from NASA experiments and other aeronautic institutes as a source. Sensor technology and UI design have been implemented to create a more immersive AR for students. However, distance calibration has been a challenging aspect of the development process but the authors have promised a solution for the future.

Arslan et al., (2020) developed an AR application for biology education. It is also made by making use of the Unity AR application developing environment. However, it is not made by making use of markers instead they used a library that will allow the reality to appear after the surrounding surface has been identified. This library makes use of camera position and sensor information to perceive the surrounding and identify a flat surface or a suitable surface. Once this surface has been identified the information or biological objects such as the anatomy of frogs, the human respiratory system, and other animations are overlayed. This system is tested on high-end smartphone devices.

The previous research works have found a way to solve the problems at hand by making the most out of AR and DL. The results have shown to be promising, but that's not to say there are no weaknesses and gaps in the research. Dash et al., 2018 have tried marker-based AR and have found promising results, however, the marker target used is user-designed and imposed over the image which makes the sample very restrictive. Since a pre-defined unchangeable marker has to be set for every image, the trained model can only detect those markers and not the image which makes it less dynamic. In the other works, Le et al., 2021 and Dash et al., 2018, tried using synthetic data for training which can be a great way of generating datasets with less cost of resources and is also able to solve illumination problems. Depending on synthetic data entirely on creating dataset however exhibits lower performance and accuracy on real target images or real-world environments (Vanherle et al., 2022). Estrada et al., 2022 performed the training with much less data and the objects to be detected can be trained with transfer learning to achieve high performance. Even though Le et al., 2021 produced an output that can track the images in different illuminations and different orientations, they relied on devices that have high image processing ability (iPhone X) not on low-end devices. Additionally, they didn't manage to cover blurriness and different sources of the same image. Le et al., 2020, uses the YOLOv4 which uses CSPDarkNet53 as a backbone architecture. However, YOLOv4 has its drawbacks of having a higher number of parameters and the architectural design requires less capability of improving the performance continuously without losing the current knowledge.

2.6 Summary of Related Works

The table below depicts the summarized version of the reviewed literature in the previous section.

Table 2.1. Related works summary

Related Studies	**Architecture (Method)**	**Dataset**	**Limitation**
Le et al., (2020) Machine learning with synthetic data–a new way to learn and classify the pictorial augmented reality markers in real-time	YOLOv4	Kaggle dataset and created a synthetic dataset using 2000 images scrapped from Google.	➢ Depended entirely on synthetic datasets and data augmentation. ➢ Doesn't provide clear steps taken in the creation of a synthetic dataset. ➢ The detection algorithm (YOLOv4) uses too many parameters.
Le et al., (2021) Augmented reality and machine learning incorporation using yolov3 and ARkit	YOLOv3	Created a synthetic dataset with four target markers.	➢ Depended entirely on synthetic datasets and data augmentation. ➢ Older version of YOLO
Estrada et al., (2022) Deep-Learning-Incorporated Augmented Reality Application for Engineering Lab Training	MobileNet-SSDv2	Created data using laboratory equipment as markers.	➢ Uses high-quality images affecting the training and testing time. ➢ Low amount of data in the dataset. ➢ Designed for, and tested with high-end devices.
Dash et al., (2018) Designing of marker-based augmented reality learning	CNN and SVM	Created a synthetic dataset based on the 26	➢ Pre-designed markers are imposed over the image which makes the sample very

environment for kids using Convolutional Neural Network architecture		alphabetic markers.	restrictive. ➢ Depended entirely on a synthetic dataset.
Alhalabi et al., (2021) Smartphone handwritten circuits solver using augmented reality and capsule deep networks for engineering education	Capsule Network Classifier	Collected circuit diagrams with different strokes and sizes and rotations.	➢ Capsule Network Classifier has low performance than CNN. ➢ A small amount of real data. Could've been improved using synthetic data and data augmentation techniques. ➢ The object localization algorithm doesn't allow certain changes in the drawings.
Lindner et al., (2019) Augmented Reality Applications as digital experiments for Education – An Example in the Earth-Moon System	Computer Vision	Marker data is used by uploading on Vuforia Engine	➢ Distance calibration from camera to marker is weak ➢ Use computer vision techniques that don't allow the dynamic choice of markers.
Arslan et al., (2020) Development of Augmented Reality Application for Biology Education	Computer Vision	Marker data is used by uploading on Vuforia Engine	➢ Makes use of Marker-less AR which requires high computation capability. ➢ Uses computer vision techniques.

CHAPTER THREE

3 METHODOLOGY

3.1 Chapter Overview

Here the methodology highlight used on markers from the books using deep learning will be illustrated as an overview. In the coming sections, the necessary steps taken to collect the data and create the synthetic dataset will be presented along with the evaluation metrics used to analyze the performance of the models.

The figure below shows the overall abstracted demonstration of what is performed and the steps taken in this research. The steps involve the collection and creation of synthetic data, which is followed by pre-processing of the gathered and created data then proceeds the step which involves training the dataset identified in steps 1 and 2, then finally evaluating the results achieved. To finish it off and see where it could be applied, we built an AR system based on the findings achieved in the marker detection domain.

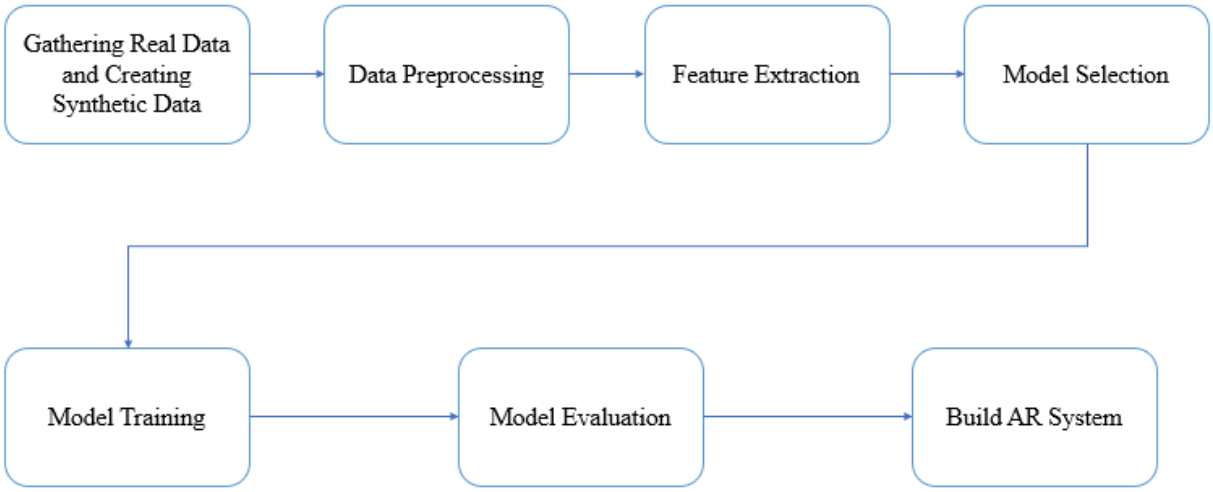


Figure 3.1. Research Methodology

3.2 Dataset Collection

Deep learning among other machine learning algorithms is highly dependent on the availability of data. The more the data the better the algorithmic performance which produces an accurate and well-trained model that can be put to use on preferred scopes. This research focuses its data to be on Ethiopian educational books specifically on Grade 9, Grade 10, Grade

11, and Grade 12 books on a specific subject which is Biology. And no such marker data is available publicly for researchers to make the study. So, the dataset was collected from scratch based on the following formats.

To make the number of classes more manageable and to maintain less fluctuation among subjects the selected figures from each book were set to be not more than 20, for later calculation let's assign this value with variable (BM). Each figure that is collected from these books as a marker is taken by considering copied (C) and original (O) versions of the books and also from a digital screen or monitor (M). From the mentioned sources again, the marker figures were taken by interchanging between the environmental instances which are Normal(N), Blurred (B), and Illumination change (I). From each of these environments the marker figures were taken from 4 different orientations (ORI) which are 0º, 90º, 180º, and 270º, five marker figures were taken by changing positional distance under these four conditions. With these in mind, the approximated total number of marker figures gathered is

5 * (N, B, I) * (C, O, M) * (ORI) *(number of books) * (BM)= 5 * 3 * 3 * 4 * 4 * 20=14,400.

Out of the entire data (including the synthetic data) collected 80% were used for training, 10% were used for validation, and the remaining 10% for testing. Data were mainly gathered from the books' different available sources including but not limited to:

- Libraries
- Schools and
- Individuals who own the necessary resource.

In the selection process of markers from the figures of the books, one high school teacher along with his students have been involved and another isolated student was also involved in the process. They have chosen markers for the research; those chosen markers have been used as input in the data collection process by considering the above-mentioned procedures and were used for the deep learning process.

3.3 Evaluation Methods

The evaluation method in general is used as a guide to analyze and test the integrity and optimality of the proposed model as well as its quality. Evaluation of a model is done through several other metrics divided into accuracy and loss metrics. Under the accuracy metrics we

have training accuracy and validation accuracy, within the metrics of loss we have training loss and validation loss. These metrics will help identify the performance of the model which indirectly suggests making readjustments on the hyperparameters of the nutrient detection model to increase and decrease the mentioned metrics. The analysis can be further simplified by plotting a graphical representation of the results.

3.3.1 Training Accuracy and Validation Accuracy

The process of measuring the capability of the model in classifying the given image, from the training dataset, precisely while performing the training is done by the training accuracy metric. The training data set is an input within the total dataset among other sets. This can be illustrated by drawing the resulting curve for better depiction.

The process of measuring the capability of the model in classifying the given image, from the validation dataset, precisely while performing the training is done by the validation accuracy metric. The validation dataset serves as an input within the total dataset among other sets however, the validation dataset is only accessed after each epoch of the training process is done. The validation dataset will help test and measure the performance of the model on these datasets while the training process is in progress. This is well noticed by plotting the curve.

3.3.2 Training Loss and Validation Loss

The measured value of how effectively the model has learned from the training dataset by disregarding generalization as a factor is the training loss metric. So, faults of the model are detected and evaluated by the training loss. The training loss is determined after each batch has completed its cycle and is numerically calculated by taking the summation of faults for each set. Much like the others the curve of training loss can be plotted for further analysis.

Based on the input validation set the performance of the deep neural network model is evaluated by using the validation loss metric. In general, the dataset labeled as a validation set is an isolated section of the dataset to evaluate the working ability of the model. Much like training loss, the validation loss is also determined after each batch has completed its cycle and is numerically enumerated by taking the summation of faults for each validation dataset.

3.3.3 Precision, Recall, and Mean-Average-Precision

These three-evaluation metrics have been used to measure the performance of detection algorithm output. Precision is the value identified by taking the ratio between the number of true positives to the total number of true positives and false positives. Recall is the value identified by taking the ratio between the number of true positives to the total number of true positives and false negatives. mAP@0.5 is also used which is the mAP computed at an intersection over a union (IoU) threshold of 0.5. IoU is the ratio between the intersection of the boxes to their union.

3.4 Development Tools

For this thesis effort to emerge with a successful and fruitful result, the availability of research instruments is imperative. From IDEs to libraries, designing tools, frameworks, programming languages, and platforms are necessary to finish the work with high precision and efficient resource utilization. The main tools that will be used in the course of this research are discussed below.

3.4.1 Design Tools

Design tools are appliances used for designing creative, attractive, resourceful diagrams, interpretations, and design ideas. Enterprise Architect will be used for this purpose as it is a freely available impressive graphic-related design option used for making several charts, diagrams, etc. However, other available web applications like are also used when necessary.

3.4.2 Hardware Tools

This research is very much dependent on hardware devices. The most common hardware we'll be making use of are the following:

No	Hardware tools	Description
1.	GPU	For convenient and faster training.
2.	Hard disk	Used as storage for the dataset.

Table 3.1. Hardware tools

3.4.3 Software Tools

The software used here is for implementing the training, testing results, debugging codes, maintaining output, loading the necessary libraries for intended purposes, and simulating results. In the table below are the software tools that will be used in the lifetime of this research.

No	Software tools	Description
1.	Python	A powerful tool for any machine learning-related project. The ample presence of libraries integrated with this high-level programming language preserves resources.
2.	Anaconda	A platform that allows installing the latest version of Python along with its machine learning functionalities.
3.	Jupyter Notebook	The IDE accompanies the features of anaconda which associates it with Python in turn.
4.	TensorFlow	An open-source library used for optimum performance in machine learning experiments best for its nominal, numerical model processing approaches.
5.	Keras	Keras is a TensorFlow library dominantly used for its simplicity, ease of maintenance, and accuracy purposes.
6	Matplotlib	A Python library based on object-oriented programming, which is also an excellent tool to map and embed plots of the training results will immensely help in analyzing the progress of the model.
7	OpenCV	This will allow us to create augmented reality along with the accepted and best-developed model.

Table 3.2. Software Tools

CHAPTER FOUR

4 PROPOSED SOLUTION FOR CLASSIFICATION AND DETECTION

4.1 Chapter Overview

In this chapter the solution put forward for marker detection for AR will be outlined in detail with the required and necessary diagrams and demonstrations. The chapter encompasses going over the collected dataset and then describes the pre-processing stage taken which then proceeds to the models used for feature extraction and training the collected dataset, finally, the activation function and the loss function and generally the architecture of the solutions is explored in detail.

4.2 Dataset

The figures that serve as markers for AR have been collected from the books in the format outlined in the previous chapter. 20 images per book have been selected which means 80 marker figures have been chosen. The tables below demonstrate the figures collected from all four books. The data collected without counting the synthetic data sums up to 14,473. The tables below show the number of markers chosen from each book along with the number of data collected from each book.

From the total data gathered of the markers which is 14,473 and after the addition of synthetic data, 80% from that total dataset is used for training purposes and the remaining 20% is divided into two 10% of the total data gathered and is intended to be used for both validating and testing the performance of the trained model. The images of the markers gathered are large and need further image processing techniques that will be covered in the next section.

Table 4.1. Summary of the Real Data Collected

Grade	**№ of Markers**	**№ of Data Collected**
9	20	3617
10	20	3638
11	20	3610
12	20	3608
Total	**80**	**14473**

Table 4.2 Figures chosen from the books

Grade 9 Book	Grade 10 Book	Grade 11 Book	Grade 12 Book
Grade 9 Figure 1.1	Grade 10 Activity 3.14	Grade 11 Figure 1.13	Grade 12 Figure 1.1
Grade 9 Figure 2.2	Grade 10 Activity 3.15	Grade 11 Figure 1.18	Grade 12 Figure 1.5
Grade 9 Figure 2.5	Grade 10 Figure 1.8	Grade 11 Figure 1.19	Grade 12 Figure 1.13
Grade 9 Figure 2.9	Grade 10 Figure 1.10	Grade 11 Figure 1.35	Grade 12 Figure 1.17
Grade 9 Figure 2.10	Grade 10 Figure 1.13	Grade 11 Figure 2.12	Grade 12 Figure 1.18
Grade 9 Figure 2.11	Grade 10 Figure 2.3	Grade 11 Figure 2.15	Grade 12 Figure 2.20
Grade 9 Figure 2.13	Grade 10 Figure 2.12	Grade 11 Figure 2.25	Grade 12 Figure 2.32
Grade 9 Figure 2.15	Grade 10 Figure 3.2	Grade 11 Figure 2.27	Grade 12 Figure 2.48
Grade 9 Figure 2.19	Grade 10 Figure 3.5	Grade 11 Figure 2.37	Grade 12 Figure 3.1
Grade 9 Figure 2.29	Grade 10 Figure 3.19	Grade 11 Figure 2.40	Grade 12 Figure 3.20
Grade 9 Figure 3.4	Grade 10 Figure 3.27	Grade 11 Figure 3.2	Grade 12 Figure 3.26
Grade 9 Figure 3.19	Grade 10 Figure 3.62	Grade 11 Figure 3.11	Grade 12 Figure 3.33
Grade 9 Figure 3.20	Grade 10 Figure 3.65	Grade 11 Figure 3.13	Grade 12 Figure 4.7
Grade 9 Figure 3.22	Grade 10 Figure 4.1	Grade 11 Figure 3.20	Grade 12 Figure 4.8
Grade 9 Figure 3.25	Grade 10 Figure 4.3	Grade 11 Figure 4.1	Grade 12 Figure 4.18
Grade 9 Figure 3.29	Grade 10 Figure 4.16	Grade 11 Figure 4.10	Grade 12 Figure 4.24
Grade 9 Figure 3.31	Grade 10 Figure 4.21	Grade 11 Figure 4.14	Grade 12 Figure 5.8
Grade 9 Figure 3.33	Grade 10 Figure 4.22	Grade 11 Figure 4.16	Grade 12 Figure 5.10
Grade 9 Figure 3.34	Grade 10 Figure 5.4	Grade 11 Figure 4.22	Grade 12 Figure 5.17
Grade 9 Figure 3.51	Grade 10 Figure 5.8	Grade 11 Figure 4.41	Grade 12 Figure 5.23

4.3 Pre-processing

After the necessary data has been gathered what follows is pre-processing the data, so that it will be performance-efficient and time efficient. Since the marker size is fairly large it would be necessary to compress the markers while preserving the detail and the quality of.

Pre-processing is necessary to detect the markers as it will make the training and validation process more robust and error-free. The dimensions of the markers are as well big which makes the neural network process several pixels for each marker making the training time longer, so this needs to be remedied. Third-party software and libraries were used to make all the pre-processing jobs easier.

To summarize even though there isn't a high level of pre-processing needed for marker detection it was to use compressing the images to smaller disk size and resizing the image dimensions.

4.3.1 Synthetic Data Creation

As mentioned earlier, since the data collected might not be enough to make the necessary training that yields an efficient model, other alternative options were required to be explored. Synthetic data was an option that has been used for such tasks in research with similar problems and has shown a promising result (Estrada et al., (2022), Le et al., (2021), Le et al., (2020)). Synthetic data was created using the marker images collected and overlaying them over different background images. The background images are taken randomly from places that are conceived as study areas such as libraries and laboratories in our research.

4.3.2 Compression

As stated, the gathered images of the markers are fairly large on disk size and we need to minimize the effect of each image on the total. To perform this, third-party software is used to compress the size of the images down to 65% of the original size. This software is great because it doesn't allow quality loss and is very dynamic in its choices, allowing users to compress images in custom mode.

4.3.3 Resizing Dimension

The dimension of the data acquired was 3120 x 4160, which is not the optimum dimension to train with and can require more effort from the algorithm. This will create complications like increasing the time of training. In the case of both MobileNetV2 and MobileNetV3, the marker images are resized to dimensions 590 x 445. For the YOLO training, the image is reduced to 640 x 640 as it is required by the algorithm.

4.3.4 Data Augmentation

Data augmentation is a technique used to create a diverse dataset when the need for one is necessary, in the case of deep learning for example. Data augmentation techniques such as flipping, cropping, rotation, and noise injection implemented on different datasets have been able to bring better performance in training a deep learning model (Shorten & Khoshgoftaar, 2019).

Data augmentation in this research is done in two ways. The first is by not doing the augmentation with the help of software or any form of library but instead, it was done during the data collection stage. The reason for this is that data augmentation is preferable when a fair amount of diversified data is available in the dataset. So, the training won't be affected by the drawbacks of data augmentation which are:

- **Data Bias**: if the original data contains data bias, then creating a dataset by augmenting that data will make all the new data to be biased which hinders the performance of the model (Shorten & Khoshgoftaar, 2019).
- **Combining data warping and oversampling techniques:** the lack of a solution to the problem of a dataset consisting of weak diversity concerning the testing data. Augmentation techniques work in the presupposition that training data and testing data are from the same source (Shorten & Khoshgoftaar, 2019).

Now this problem won't have a great effect if the dataset is already filled with diversified data but if the entire dataset depends on augmented data, it is not wise to augment the few images collected. This is the case in this research, so to render this problem clean we propose augmenting the data when acquiring the data so that each marker image will be independent of the effects of other data in the dataset. Markers were collected by considering traditional transformation augmentation methods which are rotation, illumination, and blurriness as well as different sources of the books.

The second form of data augmentation was done on the synthetic data created, and the augmentation is done using a Python library. Once the synthetic data has been synthesized each image is augmented by considering horizontal flip, zoom, rotation, and illumination changes. This method will increase the number of data to be used for training.

4.4 Proposed Method for Classification and Detection

There are several pre-trained models available that can be used to train custom data with, which can then be applied for different required purposes. Feature extraction is one of the main reasons why these pre-trained models are used, as they are built to perform such tasks with high precision. The other reason is, it's very difficult to gather a large amount of data which is required when working with deep learning (Nakasi et al., 2020). Using pre-trained models saves the extra hard work by allowing the training to be done with little or no cost of a decrease in accuracy. To add to the list of reasons, deep learning requires large-scale calculation, variables, and parameters to be used for training (Younis et al., 2020). The training can be done with as few calculations, variables, and parameters used by making use of pre-trained models.

For the classification purpose, we have chosen two pre-trained models MobileNetV2 and MobileNetV3. These architectures are used because they are compatible with mobile phone CPUs (Howard et al., 2019). They allow training to be performed with fewer parameters and less computation time. MobileNetV3 has several advanced advantages over version two. For instance, version three has built-in pre-processing techniques meanwhile version two requires the external preprocessing method to be called. There are other critical advantages that version three has over version two that we will discuss in the coming sections.

The proposed method for the detection aspect is to use the reliable and fast detection algorithm YOLO (You Only Look Once). This algorithm from its inception in 2016 has shown a new and improved method of detecting objects. The algorithm dependent on labeled datasets built around bounded boxes has shown several significant results throughout the years of development and updates. It is designed to be trained using conventional datasets like ImageNet, and COCO or it allows the training algorithm to be trained with custom datasets developed by the customization of the developer. YOLO has had several versions from the moment it was introduced.

4.5 Classification

4.5.1 MobileNetV2

Sandler et al., 2018, from Google, introduced a new version of MobileNetV1 which provides an improvement over multiple tasks and performs better on different benchmark datasets as well as different models. This model can be used for object detection with the help of SSD (Single Shot Detector) and other better architectures. This model uses the algorithmic advantage of Convolutional Neural Network (CNN) which is used in several pre-trained models. The main added feature of this version over the previous one is the inverted residual in between thin linear bottle-neck layers. In addition, non-linearities were discarded in the layers with narrow sizes to induce better algorithmic performance. Convolution is used to convert features to smaller representations of dimensions after they have been enlarged to a higher dimension and isolated with a small weight depth-wise convolution.

The development of this model has made use of Depth Wise Separable Convolutions which consists of the two splits from the full convolutional operator. They are used to perform lightweight filtering by using only one convolutional filter per input channel (Sandler et al., 2018). MobileNetV2 is known to use 3×3 depth-wise separable convolutions, at the cost of a very insignificant accuracy loss. This concept was introduced in MobileNetV1. The other shift made in this version is linear bottlenecks, which are a better solution to nonlinear layers as they can perform better without damaging too much information. The final major enhancement is reversed residuals which are inverted from the normal residuals based on the thickness of each block. This allows the bottlenecks to be connected, which this model highly depends on, instead of the layers with a high number of channels.

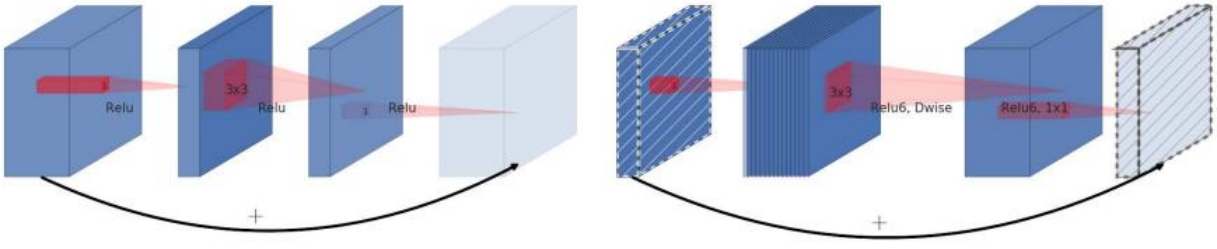


Figure 4.1. (a) residual block (b) inverted residual block (Sandler et al., 2018)

The architecture of MobileNetV2 as discussed earlier relies on bottlenecks and contains fully connected layers with 32 filters, then 19 residual bottleneck layers. ReLU6 is utilized for its reliable benefits and a kernel size of 3×3 since it is the default size for current networks.

MobileNetV2 has applied an efficient design on the architecture compared to the previous version and also similar architecture like ShuffleNet. The better side of version three can be seen when the comparison between the architectures is observed based on channel/memory. As shown in the table below MobileNetV2 shows a smaller memory allocation in Kilobit.

The convolutional block for MobileNetV2 depicted in the above table has been displayed in the above figure. This has shown significant results compared to other networks such as NasNet, ShuffleNet, and MobileNetV1. The result of MobileNetV1 has been tested on the COCO dataset.

4.5.2 MobileNetV3

Howard et al., 2019, from Google, introduced MobileNetV3, which is designed to well adapt to mobile phone CPUs by combining the efforts of network architecture search (NAS) and NetAdapt algorithm. The development process relied on extracting the effects of automated search algorithms along with network design to produce the new and enhanced version. The research has explored and achieved compatible search methods, identification of non-linearities complementary to the mobile environment, novel and better network design, and lastly a well-designed segmentation decoder. Two versions were introduced here which are MobileNetV3Large and MobileNetV3Small, both of which are found to be much better than the performance of version two. This new version is better than the previous one in both accuracy and comparable latency.

MobileNetV1 introduced depth-wise separable convolutions as an alternative to traditional convolutional layers by putting a barrier between spatial filtering and feature generation mechanism. MobileNetV2 brought the idea to use linear bottleneck and the reversed residual structure to get the best out of the layer structure which is discussed in detail in the previous subsection. Another architecture known as MnasNet introduces squeeze and excitation injected in the bottleneck architecture of MobileNetV2 to produce an efficient model. MobileNetV3 makes use of all these features to yield an efficient model. The squeeze and

excitation method is introduced in the residual layer. Non-linearity which was mostly avoided in version two is explored in this version(Howard et al., 2019).

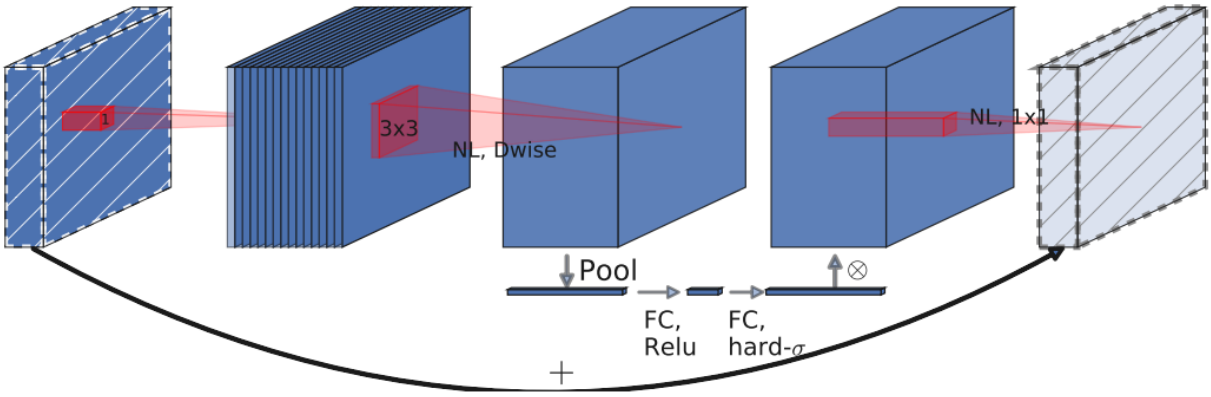


Figure 4.2. MobileNetV2 with Squeeze and Excitation (Howard et al., 2019)

Squeeze and excitation are included in the 5th to 7th block and the 12th to 16th block. The efficient last-stage architecture of MobileNetV3 has dropped three costly layers from its constituents.

This architecture version has been tested for detection on the COCO dataset and has shown an improved result over the previous second version. The result of latency and accuracy caused by non-linearity is measured with all architectures using three smartphone devices namely Google Pixel 1, Pixel 2, and Pixel 3. The impact of non-linearity and other components has shown a better performance of V3 over the previous versions. The result is summarized in the following table and figure.

The effect of using h-swish as non-linearity has shown better improvement over ReLU. MobileNetV3 has also allowed the addition of a segmentation head on top of it. This segmentation decoder is called LR-ASPP. The question that is not answered in this model is the inquiry of how to mix the automated search techniques with human intuition which has been proposed as a future research work. As demonstrated in the above figure V3 is much better than the effects of V2, this is seen in both versions of V3 small and large exceeding the performance of V2 in both accuracy and latency.

4.5.3 Proposed Architecture for classification

Since transfer learning is the proposed solution to classify markers, the design should consider the use of the pre-trained model but only a certain portion of it. We propose to use the knowledge of both pre-trained models to train our custom neural network architecture. The neural network that we propose to train on contains four dense layers with differing kernel sizes and dimensions.

The network will take the markers as input which then passes through the base model, which is MobileNetV2 or MobileNetV3. The output of the base model will go through four dense layers. The first one has 128 neurons; the second layer has 64 neurons. Each layer has an activation function of ReLU and the last one is equipped with a softmax activation function. Global average pooling is used as the pooling algorithm to be applied to the output of the pre-trained model.

The main idea behind this architecture is that both MobileNetV2 and MobileNetV3 are models used for classification mainly however, if they are introduced to another architecture known as SSD (Single Shot Detector) they can be used for object detection. The purpose of this research is to understand how well these pre-trained models perform in classifying the markers. Since the classification capability is used for detection, it can suggest how well the models perform in that regard.

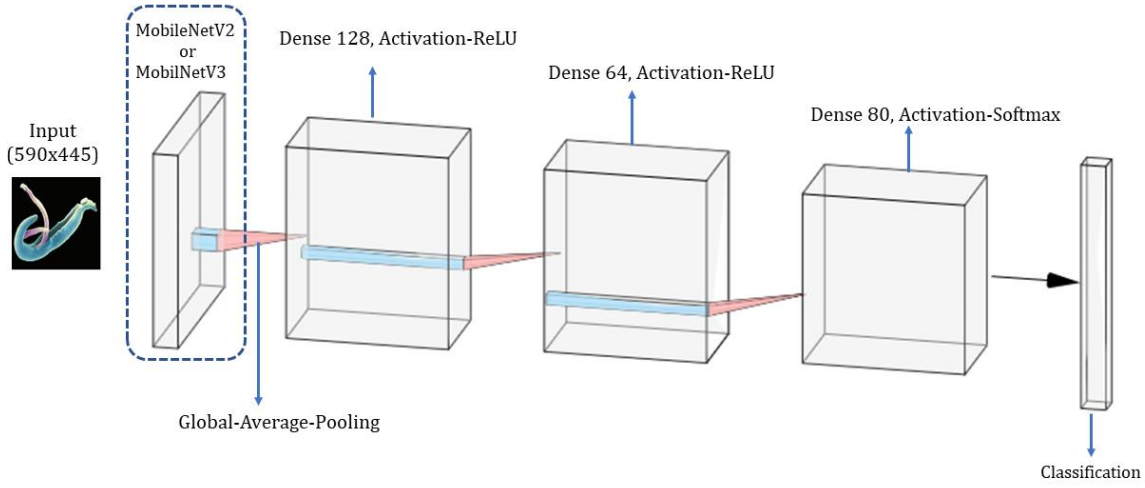


Figure 4.3 Proposed Architecture for both MobileNetV2 and MobileNetV3

4.6 Detection

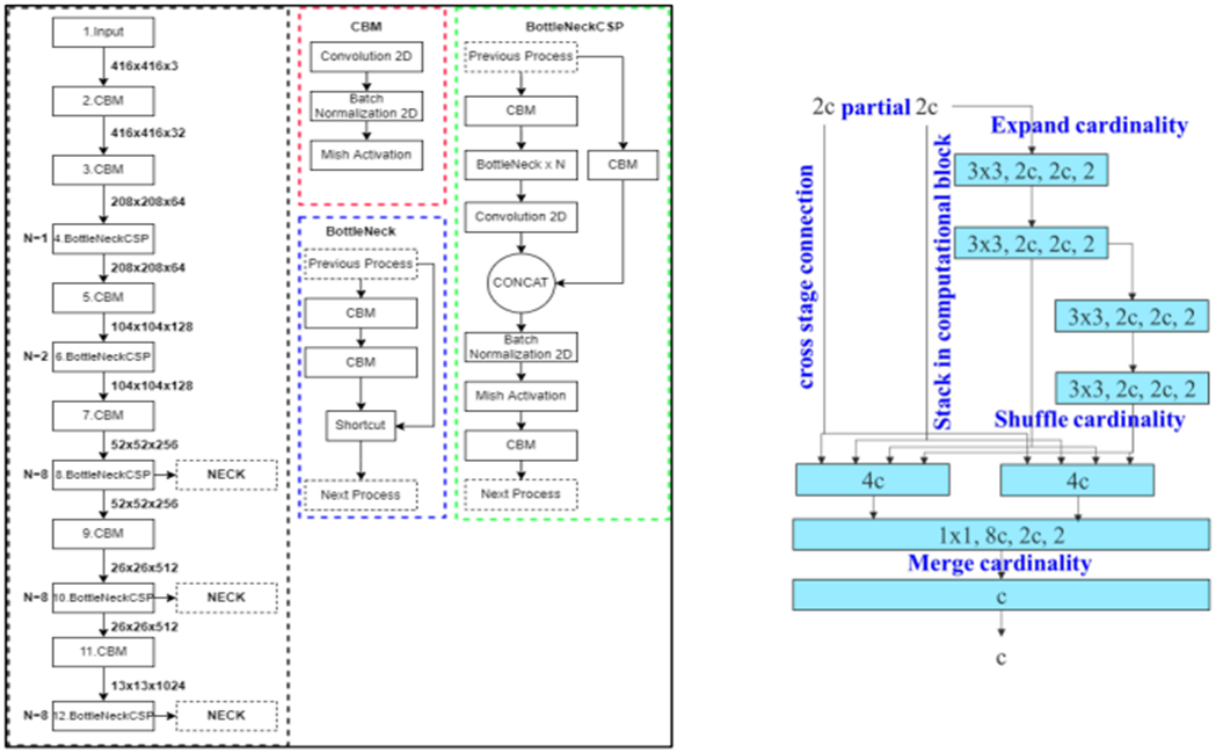
4.6.1 Improvements on YOLOv4

Since the introduction of this detection algorithm in 2016, it has seen several modified and improved versions throughout the years. Real-time object detection has been a lot easier with this algorithm as its approaches are easy to implement on public datasets or custom datasets. YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, and YOLOv6 are the most prominent versions of the algorithm. Although, each version has seen minor modifications which means there are sub-versions for each of the versions and some other independent versions.

The tiny version of YOLOv4 was the algorithm implemented in the paper used for detecting markers (Le et al., 2020). This version uses CSPDarknet53 as a backbone, on the head YOLOv3 is implemented to come up with the overall architecture. There are technical methods used in this version to get a better-performing algorithm such as a bag of freebies and a bag of specials. These methods include several other minor adjustments for instance, in creating new data augmentation techniques and designing new connections and networks (Bochkovskiy et al., 2020).

YOLOv7 released in 2022 has proven its performance with exceeding results compared to its YOLOv4 (C.-Y. Wang et al., 2022). It introduces a new backbone architecture known as an Extended Efficient Layer Aggregation Network (E-ELAN) that is lightweight and also version 7 has introduced other methods of increasing accuracy and reducing latency for detection. YOLOv7 has reduced the number of parameters from YOLOv4-tiny by 39% and the computational complexity by 49%. Making version 7 YOLO to be the best choice for detection with high accuracy and low latency compared to previous versions.

The figure below illustrates the architectural design of CSPDarkNet53 used as a backbone for YOLOv4 and E-ELAN used as a backbone for YOLOv7. As it is demonstrated in the figure CSPDarknet53 is a fairly larger architecture than E-ELAN. This is how version 7 has found a way to minimize the number of parameters by a large percentage and reduced the computational time of version 4. Other factors have also played a significant role to come up with this result such as better algorithm design by making use of model re-parameterization techniques.



I II

Figure 4.4 I. CSPDarkNet53 *(Mahasin & Dewi, 2022)* II. E-ELAN (Wang et al., 2022)

However, YOLOv4 has seen an improvement by other versions other than version 7. The following is the summary of the improvement made by each version.

- **YOLOv5** made the change by introducing optimization on hyperparameters and tracking on integrated experiments.
- **YOLOv6** released in 2022 aided by improving the single state detection for a better result.

From the mentioned YOLO versions v7 is more robust and performs better than the previous versions. The seventh version can achieve this result by introducing instance segmentation, pose estimation, and anchor-free detection head. Model re-parameterization along with dynamic label assignments are current and infant concepts introduced in the detection world. This has brought new issues to be addressed, which is exactly what v7 has been able to solve. As for model re-parameterization, the concept of gradient propagation path is used to propose a solution. For the dynamic label assignment problem, a new and well-developed method of label assignment known as coarse-to-fine lead guided label assignment. The addition of

trainable bag-of-freebies is another issue addressed to increase accuracy and at the same time not alter the inference cost. For better utilization of parameters, new methods have been used called extend and compound scaling. This method has decreased by 40% parameters and 50% computation while increasing detection accuracy and inference speed (C.-Y. Wang et al., 2022).

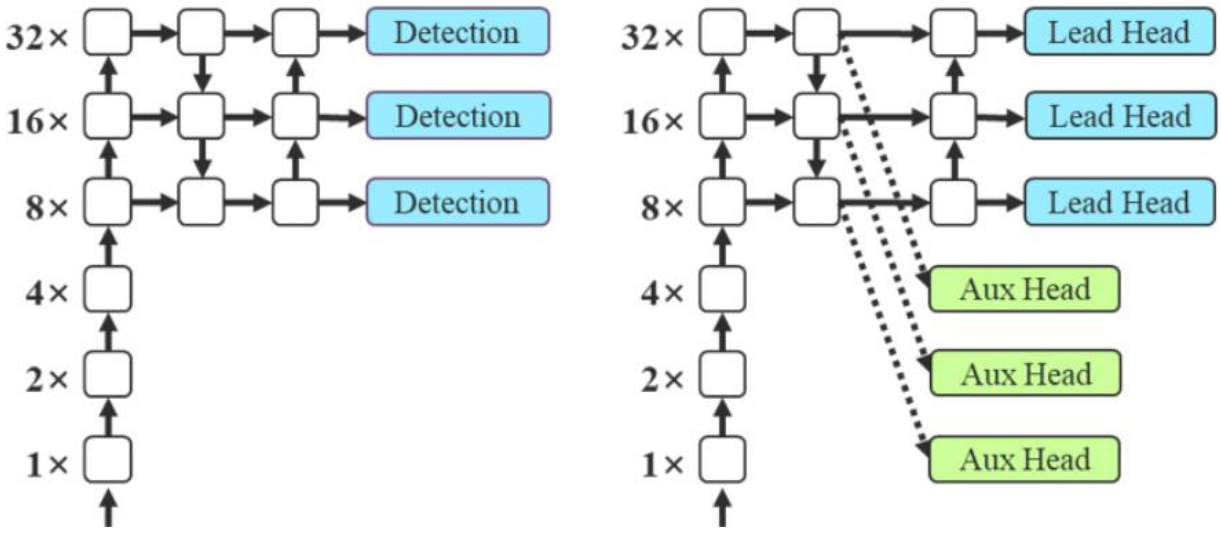


Figure 4.5. (a) Normal Model (b) With auxiliary head(C.-Y. Wang et al., 2022)

The label assigner used is revised with the help of lead head prediction along with the ground truth to produce the labels in the lead head of training and auxiliary head at the same time.

4.7 Activation and Loss Function

4.7.1 Activation function

The activation function is responsible to identify the value of the output given the input. There are several activation functions like the sigmoid function, ReLU function, softmax are among the activation functions. For the training purpose, we chose ReLU and Softmax as activation functions.

ReLU: is a piece-wise function in that if the input is positive the output will be the input will be taken directly otherwise it will be zero.

$$ReLU(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Softmax: used in multi-class classification problems returns the prediction probability that belongs to the assigned class.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Where:

σ = Softmax

$\vec{z}$ = input vector

e^{z_i} = exponential function of the input vector

k = amount of classes

e^{z_j} = exponential function for the output vector

4.7.2 Loss function

Loss functions are used to minimize the distance between actual and predicted value in turn reducing the error. Mean Square Error, Binary Cross Entropy, and Categorical Cross Entropy are examples of Loss functions.

For our training purpose, Categorical Cross Entropy is chosen.

Categorical Cross Entropy: it is used in multi-class training which takes the same number of outputs as the classes passing through the final layer.

$$logloss = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log\left(p_{ij}\right)$$

Where:

M- number of classes

N- number of rows.

CHAPTER FIVE

5 IMPLEMENTATION OF CLASSIFICATION AND DETECTION

5.1 Chapter Overview

This chapter will include the experimental procedure taken to apply the proposed solution. The environment installment of the software and hardware components is covered. The implementation procedure undertaken with code snippets and detailed explanations is presented in this chapter.

5.2 Working Environment

The working environment is comprised of both software and hardware environments that have been applied for the implementation of the study.

5.2.1 Hardware Components

In the performed research two sets of hardware devices, one (desktop) is used most prominently for training and occasionally used for documentation preparation while the other (Laptop) is used for testing and documentation preparation.

- ✓ Desktop used for training
  - Operating System: Windows 11 Pro version 21H2
  - Processor: Intel® Core (TM) i7-8700U CPU @ 3.20GHz 3.19 GHz.
  - External Graphics Processing Unit (GPU): GeForce RTX 2070 Super 8GB RAM.
  - Internal Graphics Processing Unit (GPU): Intel® UHD Graphics 630.
  - Hard Disk: Configured with 1TB disk size.
  - Installed Memory: Physical Memory (RAM) 8GB (7.86 GB usable)
  - System Type: 64-bit operating system, x64-based processor.
- ✓ Laptop for detection inference and documentation preparation
  - Operating System: Windows 11 Pro version 21H2
  - Processor: Intel® Core (TM) i5-6200U CPU @ 2.30GHz 2.40 GHz.
  - Graphics Processing Unit (GPU): Intel® HD Graphics 520

- Hard Disk: Configured with 1TB disk size.
- Installed Memory: 4.00 GB (3.89 GB usable)
- System Type: 64-bit operating system, x64-based processor

5.2.2 Software Components

a) The software environment for a training device

Software information about the desktop computer that was used for training and documentation preparation is given below.

- OS: Windows 11
- System model: HP 290 G3MT Business PC (desktop)
- GPU driver: latest version of NVIDIA GPU driver
- CUDA toolkit

b) The software environment for inference device

Software information about the laptop computer that was used for the document preparation process and object detection inference process.

- OS: Windows 11 Pro version 21H2
- System model: Dell Vostro 3559

c) Software Dependencies and Libraries

The following are the programming tools, libraries, dependencies, and IDEs used for this study.

JupyterNotebook: is a multipurpose and easy-to-use interactive web-based integrated development environment. It can be used in several application areas such as machine learning, computer vision, AI, and deep learning tasks, and is integrated with Python.

Python: is a programming language that engulfs the libraries and IDEs used in this study. The Python version used is 3.7 for MobileNetV2 and V3 and 3.6 for the YOLOv7 training process.

PyTorch: is a library under the umbrella of a component called torch and has methods and features that allow computer vision and machine learning applications to take advantage of the library.

TensorFlow: is a well-designed library that was created by Google and entertains distributed numerical computation which then allows the training and execution of big neural networks

efficiently by dividing the calculation across several multi-GPU servers. It also allows other Machine Learning Applications to be implemented by it. The latest TensorFlow version 2.12 is used in this study.

Keras: A Python library implemented as a high-level deep learning API and is implemented to train and execute neural networks. It also provides support for several backend DL computations.

OpenCV: a Python library that utilizes image processing algorithms to be used in the training and testing processes. Its capability to access and read cameras and process videos was handy for the success of this study.

5.3 Training Procedure

The training process is done in a controlled manner, to make things manageable and easy to maintain. The following order shows the outline of the procedure undertaken in the study.

- ✓ Data collection
- ✓ Synthesizing synthetic data
- ✓ Minor pre-processing
- ✓ Configure the GPU and prepare the dataset for training
- ✓ Undertake the process of implementing the proposed solution.

The methods used and how they are implemented are described in the coming sections.

5.4 Implementation of Synthetic Data and Data Augmentation

5.4.1 Synthetic Data

Synthetic data and data augmentation techniques are used to create rich and diversified data when there is a lack of data to be acquired especially for object detection, semantic segmentation, and pose estimation research areas (Schraml, 2019). They serve as the best alternatives for deep learning training that require a large amount of data as input. Here both are used together for the best result. Synthetic data is created by manually overlaying the marker images onto a new background image. Generating synthetic data has been done in two main ways. The first one is done by overlaying each of the 80 different markers on separate

background images. The second is done by overlaying 5 marker figures on one background image. The figures below depict this process.

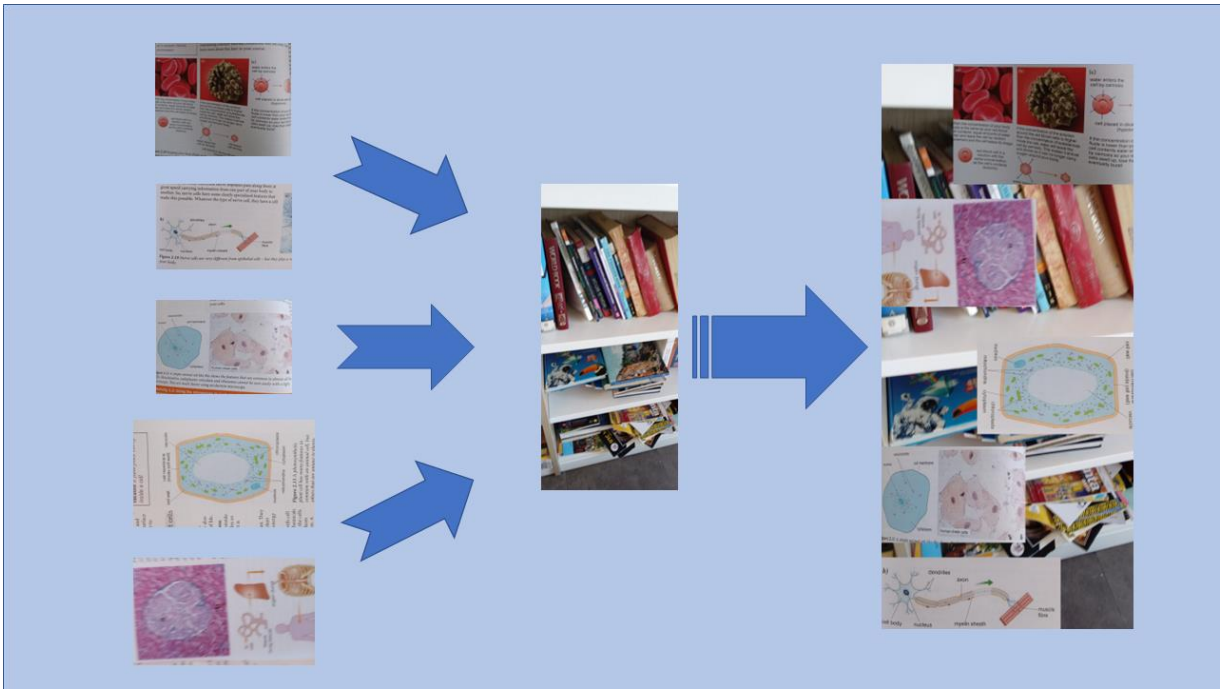


Figure 5.1. Synthetic Data Sample, Type one

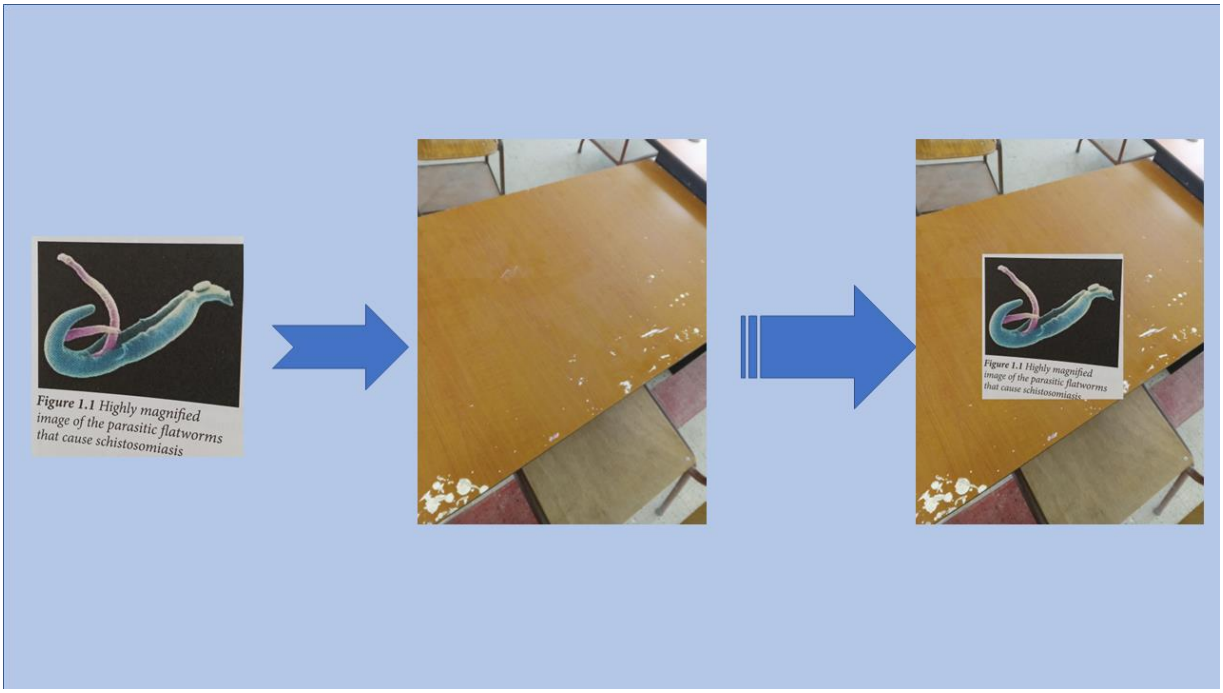


Figure 5.2. Synthetic Data Sample, Type two

This method of creation of synthetic data has been suggested by researches and it has shown a positive impact on diversifying the data and increasing the DL training performance (Man & Chahl, 2022). The background images have been collected from different study areas from both libraries and laboratories in Addis Ababa and Adama. The marker images are taken from the collected real data. A software tool is used to combine both the marker image with the background image.

5.4.2 Data Augmentation

Proceeding with the creation of synthetic data, data augmentation follows by augmenting the created synthetic data. The augmentation process was implemented by altering the rotation, zoom, and brightness range and by making a few horizontal flips.

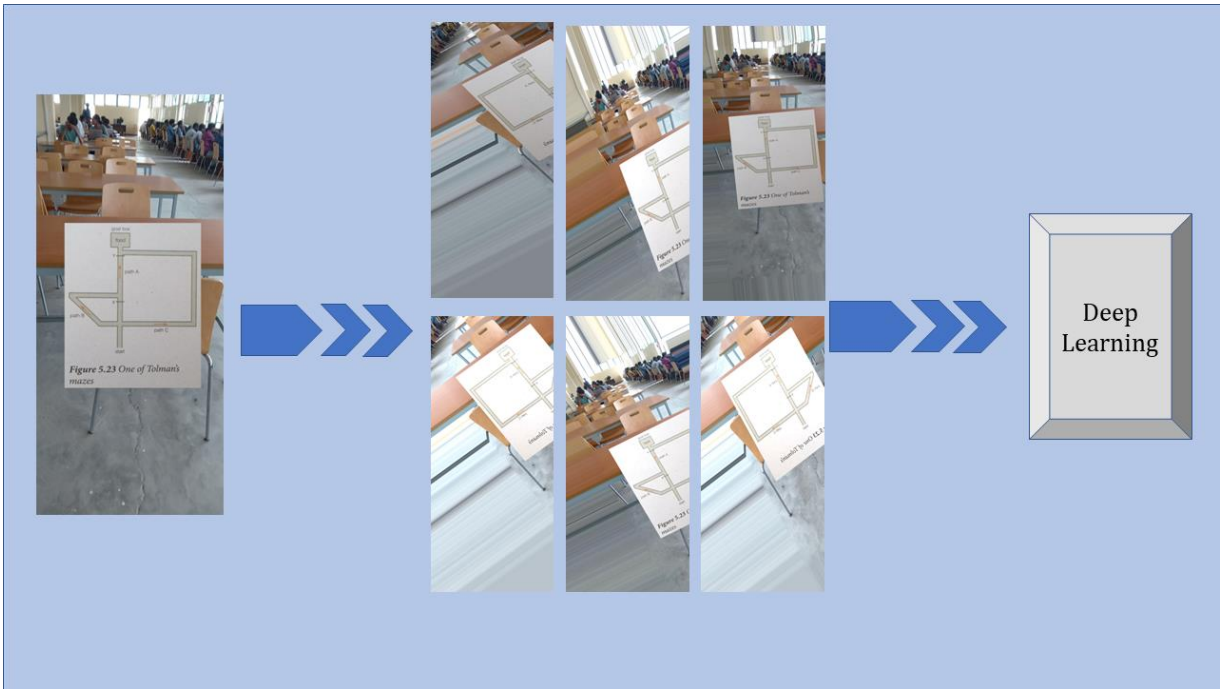


Figure 5.3 Augmentation on Type two Synthetic Data

```
Data_gen = ImageDataGenerator(
        rotation_range = 40,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = True,
        brightness_range = (0.5, 1.5))
```

Snippet Code 5.1. Data Augmentation Code

Using synthetic data and augmented data together a total of 3,379 images were generated to be added to the whole sum of the dataset. In total, the whole dataset is now around 17,852 marker figures.

5.5 Classification Implementation

MobileNetV2 and MobileNetV3 are used for classification as feature extractors. They are used to extract features from the markers of all 80 classes. Then that information is fed to a custom network designed to train the extracted features using the pre-trained models.

5.5.1 Marker image resize and Loading data

Resizing the marker images will allow the training process more efficient and will limit the time taken to train the dataset. Originally the dimensions of the images were quite big and difficult to work with but for the training purpose resizing the images without losing the important details in the marker image. In training with MobileNetV2, the preprocessing has to be done with an external library call. However, in MobileNetV3 the preprocessing is automated with the model and doesn't require additional preprocessing libraries to do the job. For both models, however, marker images are resized to 590 X 445. The training data is collected with a batch size of 32 and the validation data is collected with a batch size of 20.

```
train_path='D:\Dataset\Train'
valid_path='D:\Dataset\Validation'
batch_size_train=32
batch_size_valid=20

image_height=590
image_width=445

train_generator=ImageDataGenerator(preprocessing_function=preprocess_input).
flow_from_directory(
    train_path,
    target_size=(image_height,image_width),
    batch_size=batch_size_train)
valid_generator=ImageDataGenerator(preprocessing_function=preprocess_input).
flow_from_directory(
    valid_path,
    target_size=(image_height,image_width),
    batch_size=batch_size_valid)
```

Snippet Code 5.2 Data loading and Preprocessing MobileNetV2

As mentioned, the preprocessing for version three is not necessary as it has built-in preprocessing capability. The above code will be the same for version three except for some minor changes in generator variables, and the altered code is demonstrated as follows.

```
train_generator=ImageDataGenerator().flow_from_directory(
    train_path,
    target_size=(image_height,image_width),
    batch_size=batch_size_train)
valid_generator=ImageDataGenerator().flow_from_directory(
    valid_path,
    target_size=(image_height,image_width),
    batch_size=batch_size_valid)
```

Snippet Code 5.3. Data Loading MobileNetV3

5.5.2 Optimizer

```
optimizer=Adam(learning_rate=.00001)
```

Snippet Code 5.4 Optimizer with Adam

The optimizer used is Adam with a learning rate with an initial value of 0.00001.

5.5.3 Learning rate

Since deep learning networks use the stochastic gradient descent optimization algorithm, the learning rate is the responsible hyperparameter which maintains the amount of change to be made on the model as a reply to the approximated error every time the learned weights are changed. The learning rate is the most fundamental aspect of training a deep neural network and the most challenging one to choose. If the learning rate amount chosen is too low the training time and the computation made on each learned weight on every iteration will be complex and slow, meanwhile choosing a high value for learning rate will result in the training process having a rapid and unbalanced lifetime. The back-propagation method in stochastic gradient descent calculates the current error gradient of the model by using features from the training data which then corrects the weights by considering the error computed. The learning rate is responsible for identifying how much of those weights are changed during the training process. The learning rate can fluctuate between the values of 0 and 1 as it is an adjustable value. In the experiments made in this research two learning rates have been used which are 0.0001 and 0.00001.

5.5.4 Defined Model

The model-defining process has been executed at the baseline of the pre-trained model. Feature extraction ability has been taken from the pre-trained models while building a custom layer at the end to help with the specific prediction required for the output. The pre-trained models will allow the knowledge to be transferred serving as the output for the custom layers. The custom neural layers will take the knowledge of the pre-trained model and give predictive output concerning the classes of the dataset.

As suggested the experiments were made using transfer learning from the models MobileNetV2 and MobileNetV3. Three main dense layers have been used sequentially after the output of the pre-trained models, the fourth dense layer serves as the predictive and final layer. Global average pooling is used in between the outputs of the pre-trained layers and the custom layers to minimize an imbalance.

The pre-trained model output will serve as the input for the custom model. This doesn't mean that all the parameters of the pre-trained model will be trainable which will only make the training process slow and time-consuming. Since the final or custom layers are part of the entire neural network that holds the key summaries of our objective, those will be used as trainable parameters. The final layers are more specific to our problem and that is why they are the main parts of the training process while taking the knowledge of the pre-trained model.

```
base_model=MobileNetV3Small(weights='imagenet',include_top=False)

y=base_model.output
y=GlobalAveragePooling2D()(y)
y=Dense(128,activation='relu')(y)
y=Dense(64,activation='relu')(y)
preds=Dense(80,activation='softmax')(y)


model=Model(inputs=base_model.input,outputs=preds)
```

Snippet Code 5.5 Model Definition

As depicted from the snippet code the base model variable is set to be the pre-trained model. It is invoked with weights of 'image net' which is the standard weight for such purpose. The top layer is not included because the output is not done based on the output or prediction of the

pre-trained model instead it is our custom output. The base model variable on the snippet code is for the MobileNetV3Small model. The rest of the pre-trained models are created similarly.

```
base_model=MobileNetV2(weights='imagenet',include_top=False)
```

Snippet Code 5.6 Base Model for MobileNetV2

```
base_model=tf.keras.applications.MobileNetV3Large(weights='imagenet',include
_top=False)
```

Snippet Code 5.7 Base Model for MobileNetV3Large

The variable x will be used to hold the custom neural network. The custom neural network will take the output of the base model as an input as stated in the previous chapter. Global average pooling of 2D will be applied to the input. A dense layer with neuron sizes of 512, 256, and 128 will follow with the activation function of ReLu. Finally, the prediction layer of size 80 which is the number of classes used and an activation function of Softmax. The custom model is then created by taking the input of the base model which are the marker images and the output will be the prediction layer. The model is then trained by compiling it with the loss function of categorical cross entropy since there are several classes to be trained. The optimizer used as stated is Adam optimizer and the evaluation metrics are set to accuracy. With this in mind, the trainable parameters are 830,416 while the total parameters are 3,088,400.

5.5.5 Hyper-Parameters

Hyper-parameters are the values responsible for controlling the process of training and identifying the instances of the model by which the training algorithm uses for learning. They are very important for the training process to produce good results. When choosing hyper-parameters for the learning process it should be done with prior analysis and care.

Learning rate as mentioned in the previous sections, the training has been alternated between 0.0001 and 0.00001. The epoch size for training the model was done with 20 epochs. The optimizer used is Adam for its enhanced optimization capability with faster computational time, furthermore needs fewer parameters for fine-tuning the model.

The table below shows a compact form of the hyper-parameters used initially while training. These values however are changed for experimental purposes to get a better-performing and more efficient model.

Table 5.1. Hyper-Parameters used for MobileNetV2 and MobileNetV3

No.	Hyper-Parameters	Types and Values
1.	Learning rate	0.0001
2.	Epoch	20
3.	Optimizer	Adam
4.	Batch size	30 (train) 20 (validation)

5.6 Detection Implementation

The algorithm for detection used is YOLO version 7. This algorithm comes with well-organized training, testing, and detecting algorithms. The required aspect when working with this algorithm is to prepare a labeled dataset. Our custom dataset has been prepared and labeled with the required format. The model is trained using the fetched yolo weights, then the detection is made using the prepared model from the yolo weights.

The annotation process has been done using a labeling tool, which is developed using Python as the primary language The data gathered must be labeled in Yolo labeling format using the bounding box labeling method. It should be in a .txt file with the same name as the labeled image name. Each label file contains the annotation of the corresponding image file using four numbers as follows:

```
0 0.477404 0.485577 0.491346 0.390865
```

The first number indicates the object class, and the following second and third numbers are the x-y coordinate of the center of the bounding box of the object being labeled. The final two numbers are the width and the height of the bounding box.

5.6.1 Custom dataset configuration

After the custom-labeled dataset has been prepared the dataset directory and the class names along with the number of classes should be specifically identified in a .txt file. Then this txt file is added to another file with YAML format. For the training to fetch the correct information and proceed with the task, the mentioned steps are important.

```
# train and val data
train: ./custom dataset/Train.txt
val: ./custom dataset/Validation.txt


# number of classes
nc: 80

# class names
names: [ 'Grade 9 Figure 1.1'…]
```

Snippet Code 5.8 Snippet code for a custom dataset

5.6.2 Training

The training as mentioned will be performed with the pre-trained Yolo weights to create our custom weights. The training is customized to fit 80 classes so the batch size has to be reduced. This is because the GPU used was not able to perform the training with large batch size. The image size is converted to 640 x 640 for training purposes. The learning rate used fluctuates between the initial 0.01 to the final 0.1. The following snippet code shows the call to instantiate the training process. The 'workers' line code shows the number of CPU cores to be utilized per CPU.

```
!python train.py --workers 8 --epochs 10 --device 0 --batch-size 8 --data
data/custom.yaml --img 640 640 --cfg cfg/training/yolov7_custom.yaml --
weights yolov7.pt --name yolov7x --hyp data/hyp.scratch.p5.yaml
```

Snippet Code 5.9. A Command for Training

5.6.3 Hyper-Parameters

The hyper-parameters used for the training process of YOLO are depicted in the table below.

Table 5.2. Hyper-Parameters for Yolo

No.	Hyper-Parameters	Types and Values
1.	Learning rate	0.01
2.	Epoch	10
3.	Batch size	8
4.	Weights	YoloV7
5.	Workers	8

5.6.4 2D Augmented Reality Implementation

The AR implementation has been applied after the detection model has been successfully developed. The best weight produced by the training is used for inference. Making use of the YOLOv7 inference code, we added our custom segment of code to implement 2D augmentation on the detected markers. The images are two-dimensional but have the effect of appearing as if they are three-dimensional. Video is also augmented on top of the detected marker. This method uses libraries mostly OpenCV in Python but other libraries were also implemented for incorporating the detection model.

The figure below demonstrates the design for the AR development process. The marker is detected with installed cameras on the device. The device then uses the capability of the detection model to identify the marker and present content. The contents that are provided for the student include videos, text content, and 3D model content. The text content involves a simple definition or identifying a reference to the link presented with the barcode.

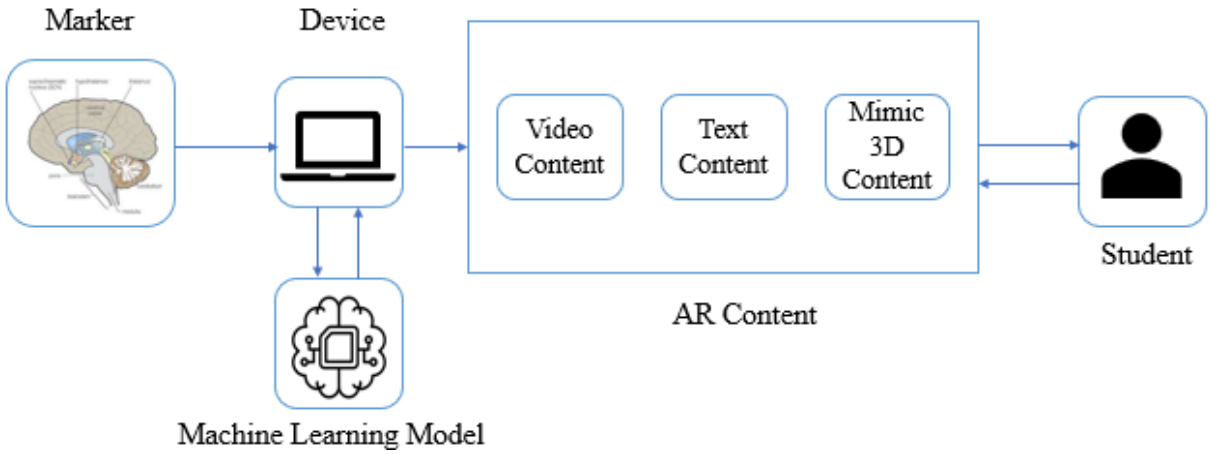


Figure 5.4 AR implementation

Image information and Text overlay: The model will detect and return the location value of the detected marker. This value is used to overlay the image by tracking the marker and text at the bottom of the screen.

Video Overlay: Each frame taken from the input video will be played frame by frame by using methods designed to take each frame as floats to be displayed. The frame of the video is overlayed on the marker using the location value given by the detection model.

CHAPTER SIX

6 RESULTS AND DISCUSSION

6.1 Chapter Overview

This chapter is concerned with the depiction of the result identified based on the implementation of the proposed solution in the previous chapters. This section will hold detailed descriptions of the result with the necessary evidence shown with graphs and plotted images that would help envision the experiments performed in this research.

6.2 Result Discussion for Classification

The pre-trained models have been performed on different experiments to get a better result by altering the hyperparameters. The training was performed for all models using the most powerful libraries TensorFlow and Keras. To plot the figures shown below we used matplotlib. The trained models have been trained with accuracy metrics. Several instances have been changed to get better performance from all models. The pre-trained models used for feature extraction specifically are three MobileNetV2, MobileNetV3Small, and MobileNetV3Large. The Accuracy graphs along with their Loss graphs have been illustrated below for each experiment.

6.2.1 MobileNetV2

The first classification training was done with MobileNetV2 pre-trained model with batch size 32 for training and 20 for validation data and a learning rate of 0.00001. The custom model consists of two hidden layers, the first is 128 neuron size dense layer and another 64-neuron size dense layer after passing through the global average pooling layer. The training process was completed on 50 epochs. The layers have a ReLU activation function except for the output layer with a softmax activation function with 80 neurons, the same size as the number of the classes. The performance of the model has shown an 85.82% accuracy and 1.0465 loss. The maximum validation accuracy observed at the last epoch is 84.06% along with the minimum validation accuracy of 3.81%. The maximum loss along with the validation loss was observed at the beginning of the training which is 4.39, and 4.36 respectively. Although this model has a pretty fair result it could be better and can be enhanced for further and more

efficient results. It requires fine-tuning process on the model by changing some technical factors.

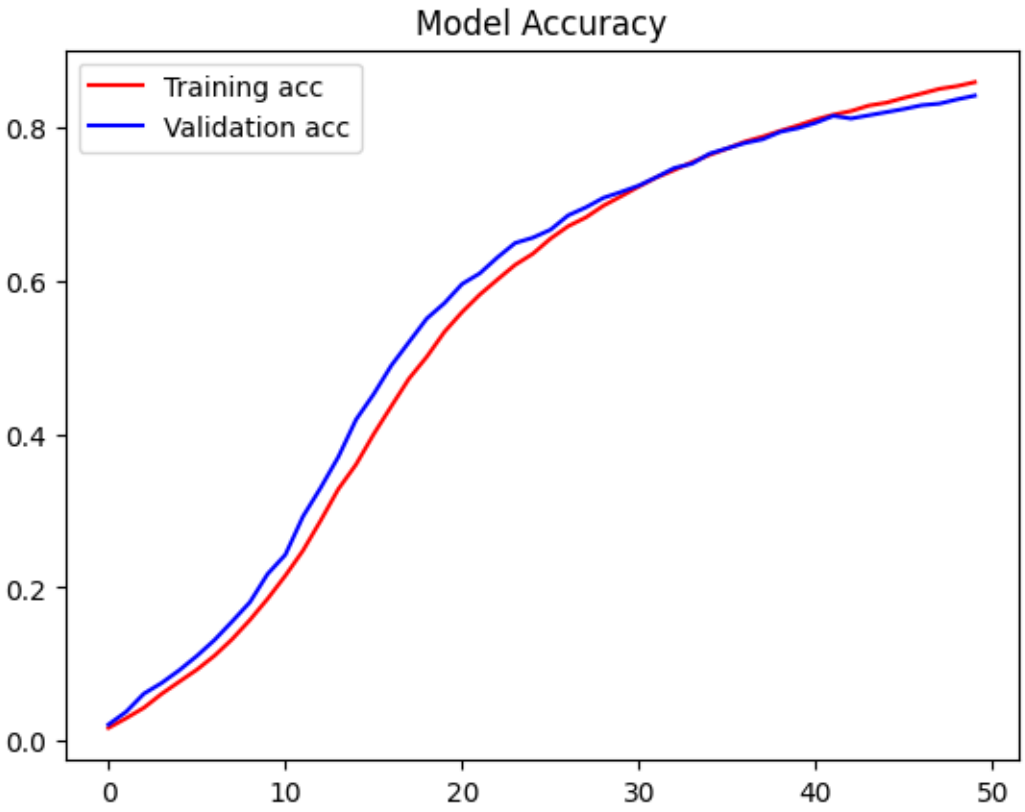


Figure 6.1 MobileNetV2 (Dense 128 x 64) Accuracy

The second experiment was performed again with MobileNetV2 pre-trained model with some changes. The training was done by adding two dense layers with neuron sizes of 512 and 256. The last dense layer with 64 neurons has been dropped. The training process was done in 20 epochs. The layers have a ReLU activation function while the output layers have a softmax activation function. The performance of the model is 88.09% accuracy and a loss of 0.8557. The maximum validation accuracy value is 87.83%. The model has shown a minimum accuracy at epoch 1 of 2.37% and a validation accuracy of 5.7%. The minimum validation loss for this model is 0.8783. The maximum loss and validation loss recorded was on epoch 1 which is 4.3622 and 4.3032 respectively. The alteration made in this experiment has shown an increase in performance for MobileNetV2.

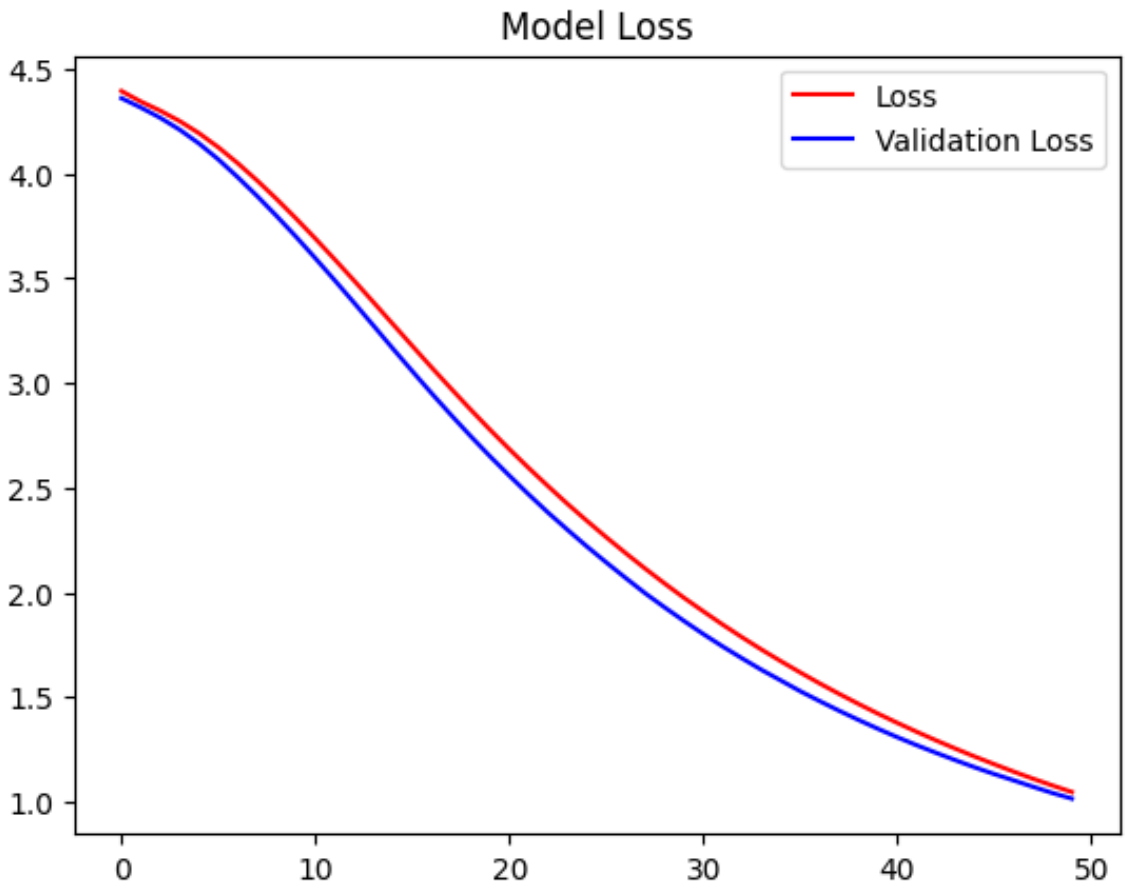


Figure 6.2 MobileNetV2 (Dense 128 x 64) Loss

The figures are depicted with accuracy on the y-axis and epochs on the x-axis. Both of the above figures show the model accuracy and model loss respectively for the first experiment. The graphs show that there is a linear increase and decrease in both accuracy and loss respectively which is an indication for a better generalization.

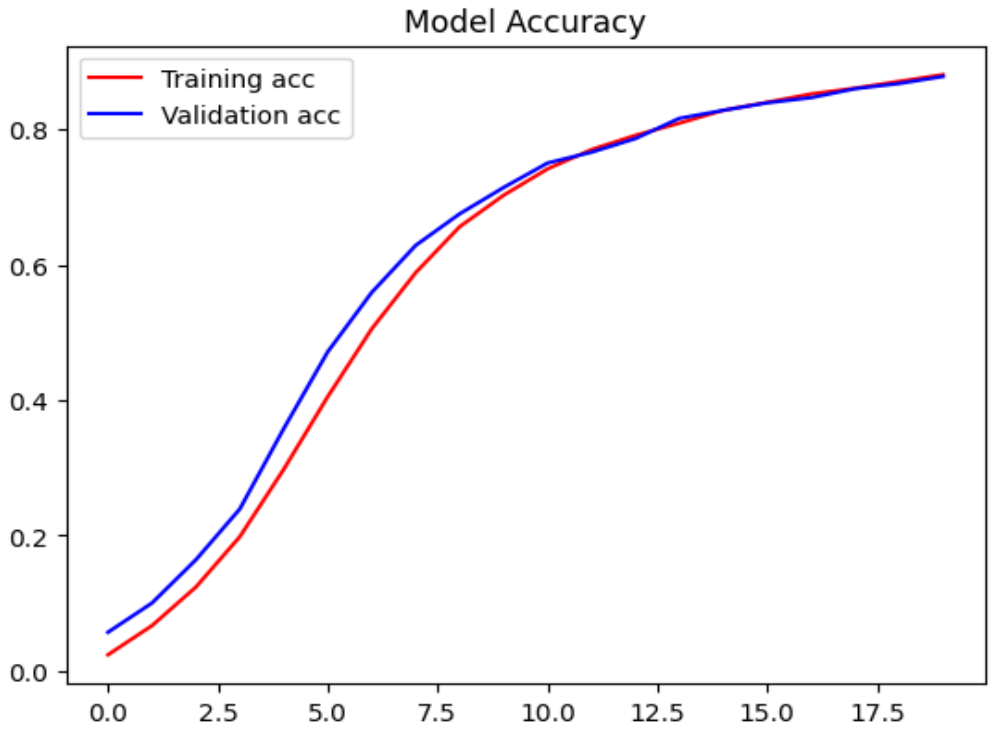


Figure 6.3. MobileNetV2 (Dense 512 x 256 x 128) Accuracy

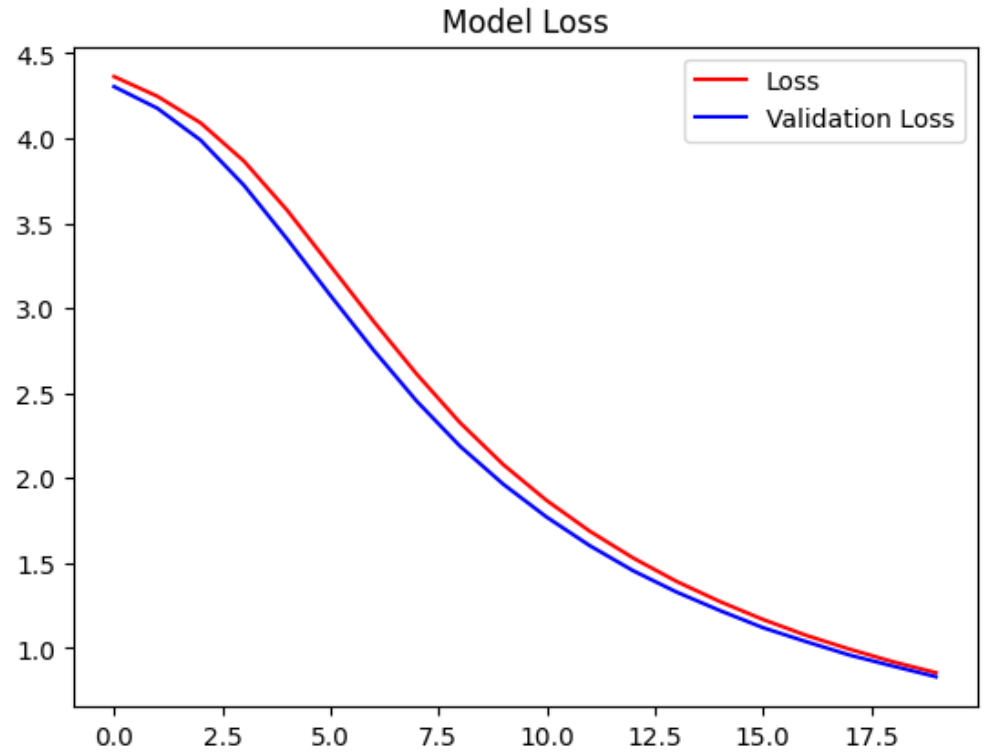


Figure 6.4. MobileNetV2 (Dense 512 x 256 x 128) Loss

6.2.2 MobileNetV3Small

The third experiment is done on MobileNetV3Small by allowing the training process to proceed with the initial parameters of the better-performing experiment with MobileNetV2 to compare the result of both pre-trained models. The training was done by allowing for a batch size of 32 for training and 20 for validation data and a learning rate of 0.00001. The model consists of three custom layers that will help our custom prediction instead of the prediction of the pre-trained model. The training process was done in 20 epochs. The input custom layers have a ReLU activation function while the output layer has a softmax activation function as it is an activation function utilized better for prediction. A global average pooling is used in between the output of the pre-trained model and the input of the custom model. This will the necessary knowledge to be filtered for the custom model creating a more robust and precise model result.

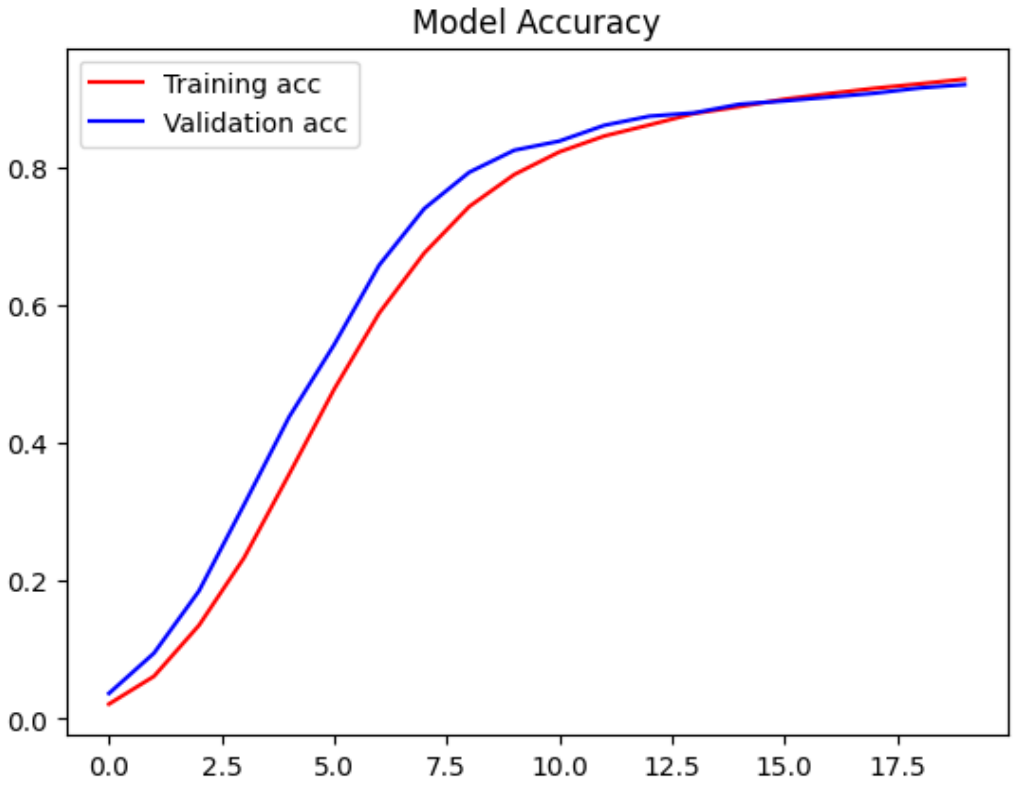


Figure 6.5 MobileNetV3Small (Dense 512 x 256 x 128) Accuracy

According to the documentation, MobileNetV3Small has better accuracy specifically 4.6% more accuracy than MobileNetV2 while reducing latency by a fair margin. This has been put

to the test for the custom dataset created and the result identified is quite similar to the theory. The next experiment has shown an improvement over its fair rival of the second experiment. In this experiment the model has produced an accuracy of 92.86% and the loss recorded here is 0.5621. Maximum validation accuracy of 92.07% and minimum validation loss is 0.5214 this result is recorded at the 20th and final epoch. The minimum accuracy recorded is on the first epoch 2.14% and the validation accuracy of 3.69%. The maximum loss and validation loss in this trained model are 4.3566 and 4.2941 respectively. As stated, the experiment has proven that MobileNetV3Small has better accuracy than MobileNetV2 by an increase of 4.77 percent. The result was recorded by training both pre-trained models with the same parameters. However, the trainable parameters for both models are different in amount with MobileNetV2 having a higher number. The trainable parameters amount for MobileNetV2 is 830,416 meanwhile the amount for MobileNetV3Small is 699,344 of the total parameters. V3small has been able to perform this better with an even less trainable parameter for the same configuration made, making it more accurate and less time-consuming than version 2.

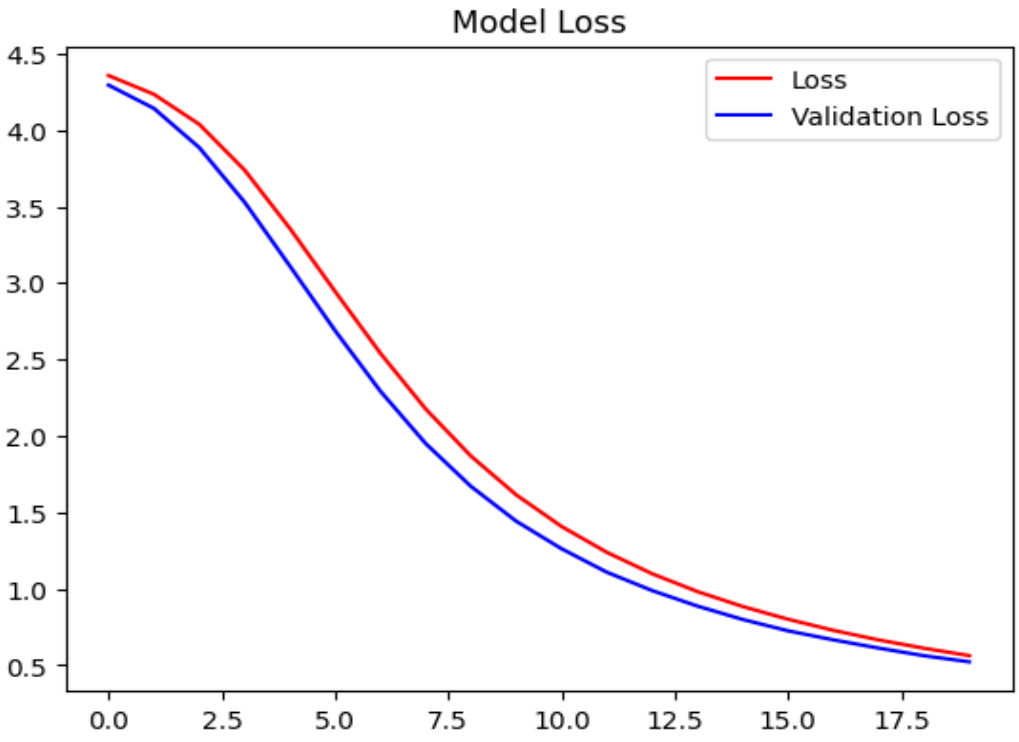


Figure 6.6 MobileNetV3Small (Dense 512 x 256 x 128) Loss

6.2.3 MobileNetV3Large

The next experiment done on these pre-trained models was done with the aid of MobileNetV3Large. This pre-trained model is more efficient than its prior versions of the same model. Again, for measurement purposes, this model was trained based on the better-performing architecture much like the other ones. This will help us identify which model has been able to produce a better custom-trained model. This model was trained with a 0.00001 learning rate, and it is trained on 20 epochs. The batch size, the architecture of the custom neural network, the activation functions, the loss functions, and other parameters are the same as the second experiment and the third experiment. It turns out this version has better accuracy and latency rate than version 2 of the pre-trained model. This version is also better than version 3 small in terms of the accuracy identified on the custom dataset.

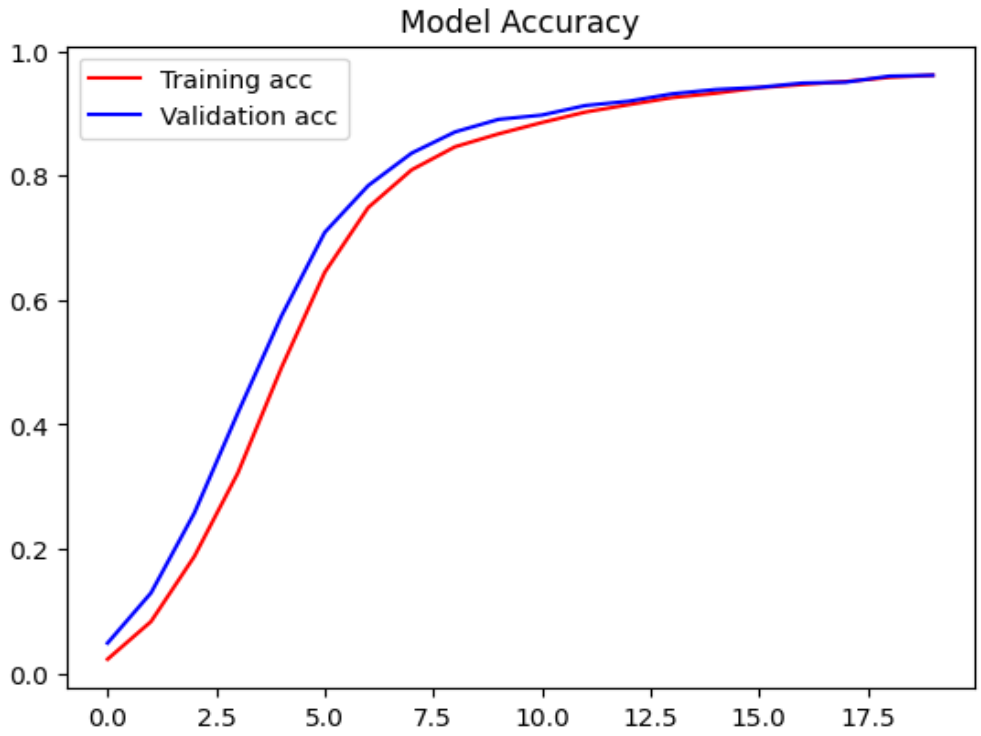


Figure 6.7 MobileNetV3Large (Dense 512 x 256 x 128) Accuracy

This experiment has produced a fairly better result than MobsileNetV3Small and an even greater result than MobileNetV2. The accuracy of this model is 96.10% and it has seen a loss of 0.3229. A maximum validation accuracy of 96.11% and minimum validation loss of 0.3048 has been recorded in this experiment. The minimum accuracy recorded on the first epoch is

2.29% and the validation accuracy of 4.87%. The maximum loss recorded is 4.3474 and the validation loss is 4.2474. This is a comparably much more proficient model than the prior models. The next figures demonstrate this clearly where validation accuracy has kept pace even to the last epoch to be greater than normal accuracy, unlike the previous models. Enhancement on this model is almost redundant as the achieved result has shown a convincing generalization on the dataset provided.

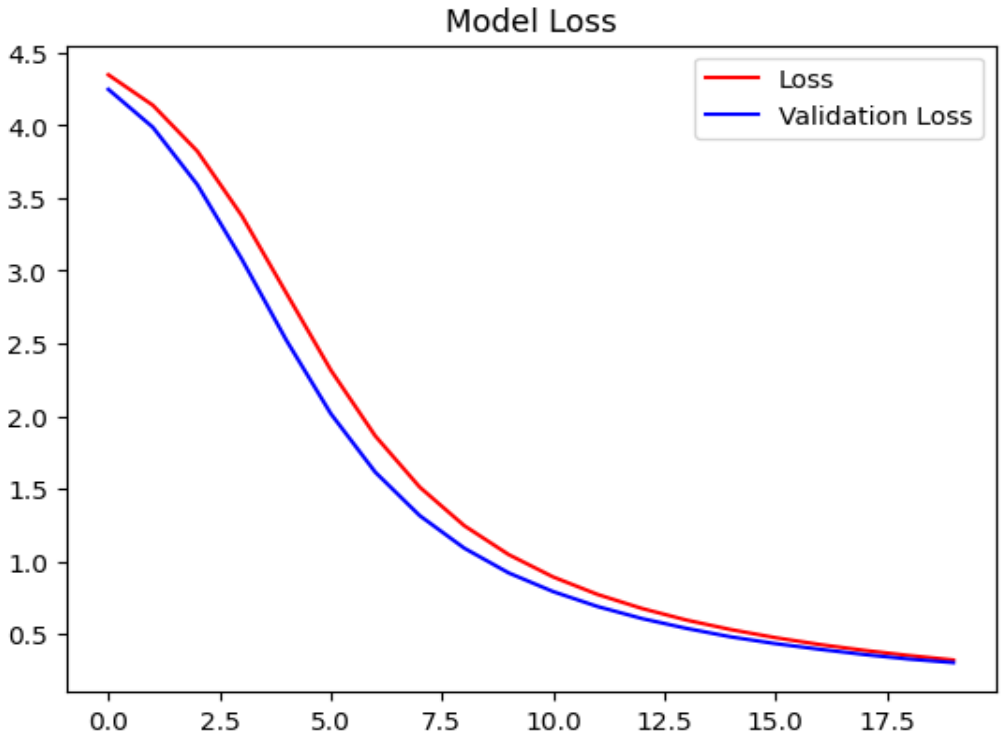


Figure 6.8 MobileNetV3Large (Dense 512 x 256 x 128) Loss

The final classification training was performed in an attempt to improve the result of the MobileNetV3Large pre-trained model with batch size 32 for training and 20 for validation data and a learning rate of 0.00001. The model network is customized and consists of two hidden layers one of 128 neuron size dense layer and another 64-neuron size dense layer after passing through the global average pooling layer. The training process was completed on 50 epochs. The output layer with softmax activation function with 80 neurons, the same size as the number of the classes. The performance of the model has shown a 96.21% accuracy and 0.31 loss. The maximum validation accuracy observed at the last epoch is 94.94% along with

the minimum validation accuracy of 2.84%. The maximum loss along with the validation loss was observed at the beginning of the training and the values are 4.4, and 4.34 respectively.

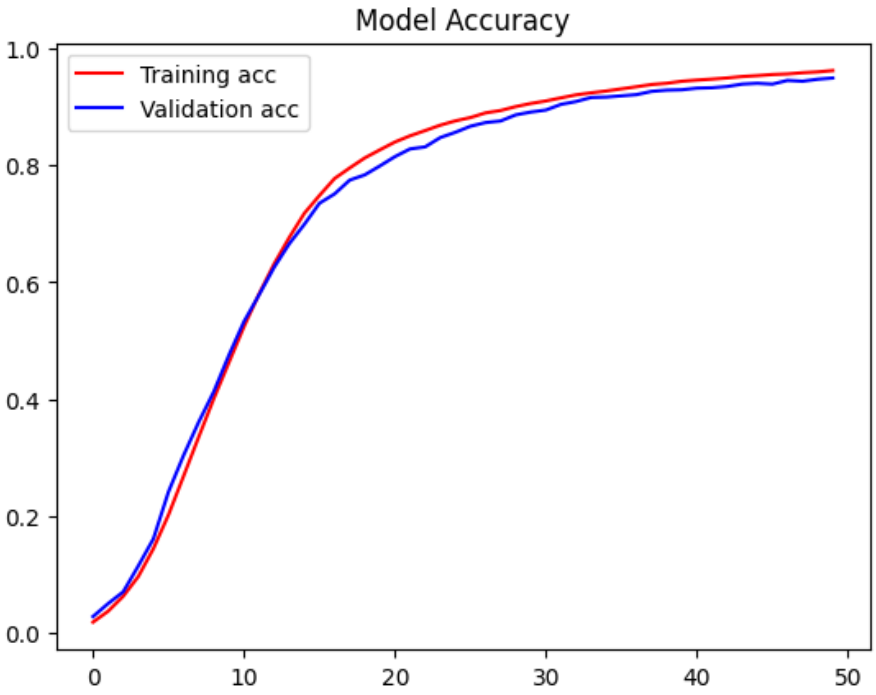


Figure 6.9 MobileNetV3Large (Dense 128 x 64) Accuracy

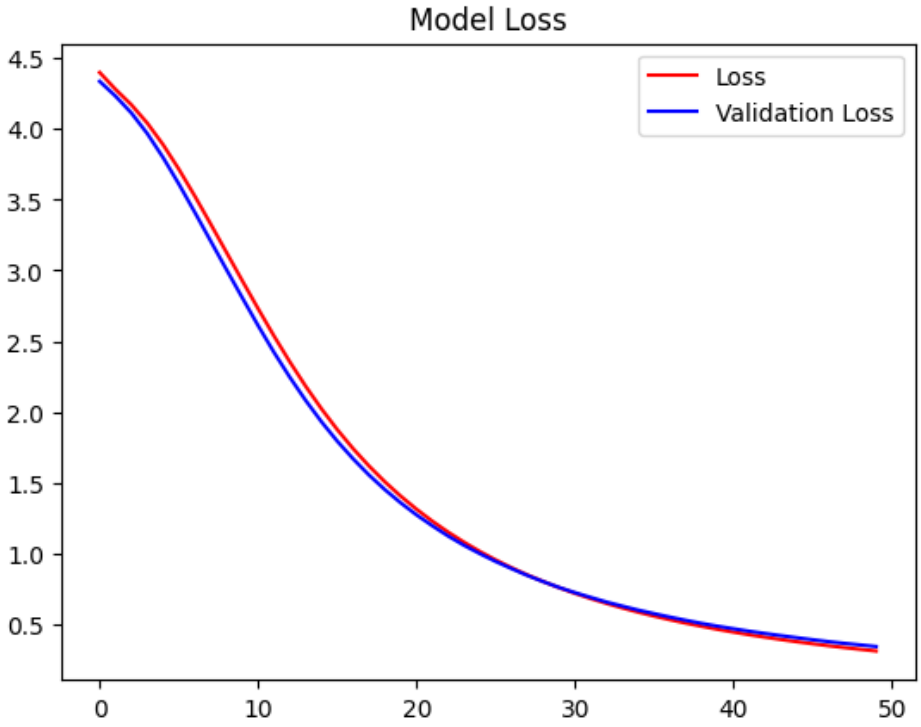


Figure 6.10 MobileNetV3Large (Dense 128 x 64) Loss

A summary of all the Mobile Net pre-trained models on the markers is given below in the table.

Table 6.1 Measured Accuracy of Experiments

Experiment	**Pre-trained Model**	**Model Accuracy (%)**
1.	MobileNetV2	85.82
2.	MobileNetV2 (Enhanced)	88.09
3.	MobileNetV3 Small	92.86
4.	MobileNetV3 Large	96.10
5.	MobileNetV3 Large (Enhanced)	96.21

The models trained in the proposed architecture have shown satisfying results for the given marker figures in all four books. By alternating values of a few parameters, by dropping and adding layers an even better performance has been able to be extracted from the training process. MobileNetV3 having the upper hand on the performance result of the training over MobileNetV2 has shown an exceeding result on both accuracy and loss of the model trained. The experiments have shown varying results as shown in the bar graphs below, this is due to the enhancement of the parameters made to each architecture while training. The highest accuracy recorded with equal configurations is MobileNetV3Large with 96.21% accuracy which is much more favorable than the rest of the models. The minimum loss recorded is also held by the same model with 0.31. Table 10 and Table 11 show this result numerically in detail to get a better performance scope of the models produced for each experiment.

Table 6.2 Measured Loss of Experiments

Experiment	**Pre-trained Model**	**Model Loss**
1.	MobileNetV2	1.05
2.	MobileNetV2 (Enhanced)	0.86
3.	MobileNetV3 Small	0.56
4.	MobileNetV3 Large	0.32
5.	MobileNetV3 Large (Enhanced)	0.31

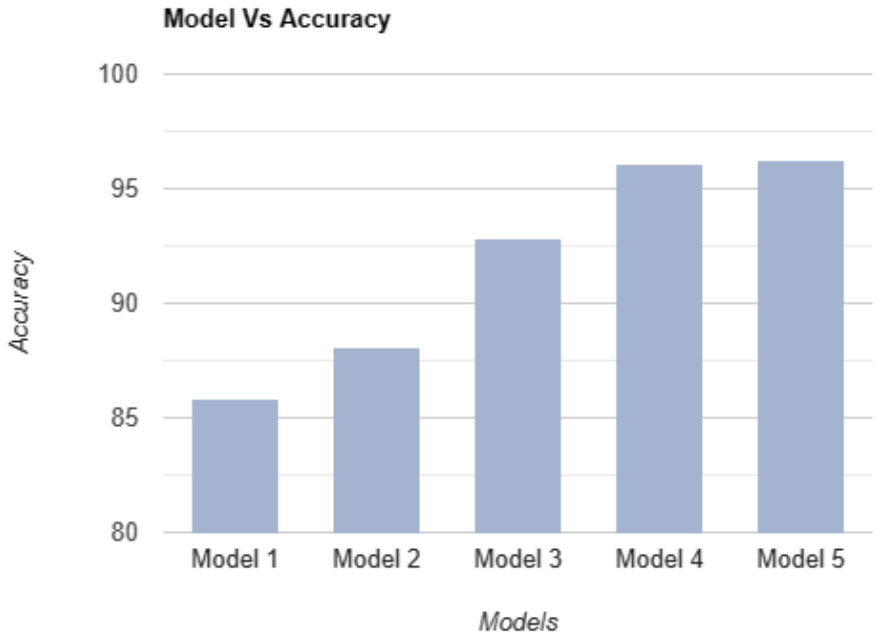


Figure 6.11 Accuracy of Models

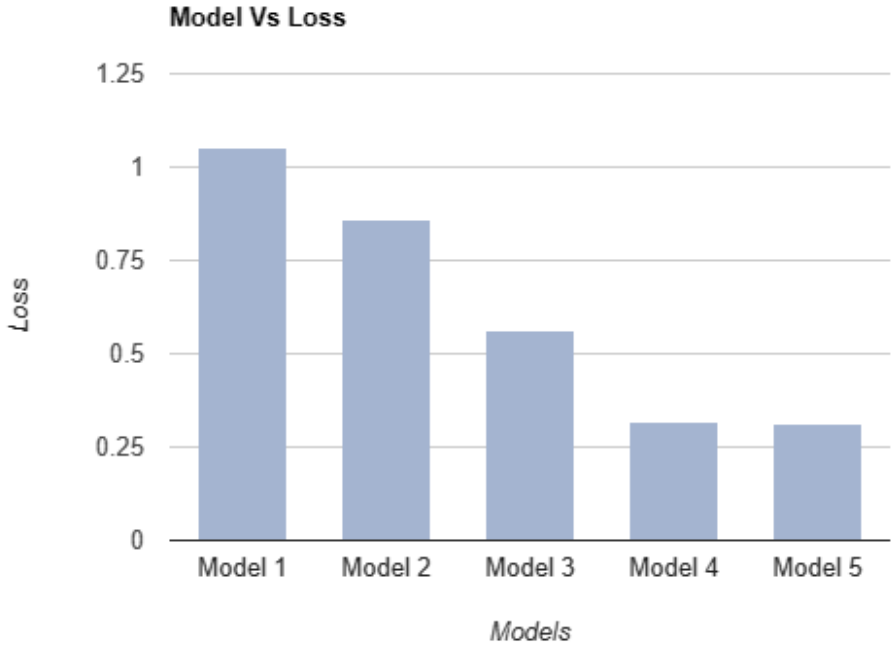


Figure 6.12 Loss of Models

6.3 Result discussion for Detection

As discussed in the previous chapters, YOLO is a very powerful and precise algorithm used for object detection. To detect markers in the books for augmented reality version 7 of Yolo has been used. This version has improved the result of detection by introducing instance segmentation, pose estimation, and anchor-free detection head. YOLOv7 has shown a higher detection rate than the previous pre-trained models. Its enhanced algorithm for object detection has made it incomparable with the other tested models. However, for comparison purposes, we have also trained our dataset with YOLOv4.

The training process for both YOLOv4 and YOLOv7 has gone through 30 epochs while simultaneously training and testing the batches being fetched. Although the batch size was assigned to the value 8 due to memory shortage reason, it was more than enough for the algorithm to run the training properly. The training proceeded while producing the training weights on each epoch and was able to extract the best weight from them. The precision yielded in the training of YOLOv7 is 98.65%. The minimum precision is recorded at epoch 1 which is 79.78%. The recall of the model is 99.83%. The minimum recall observed at epoch 3 is 73.2%. The mAP is 99.68%. The minimum mean average precision is recorded at epoch 1 which is 81.81%.

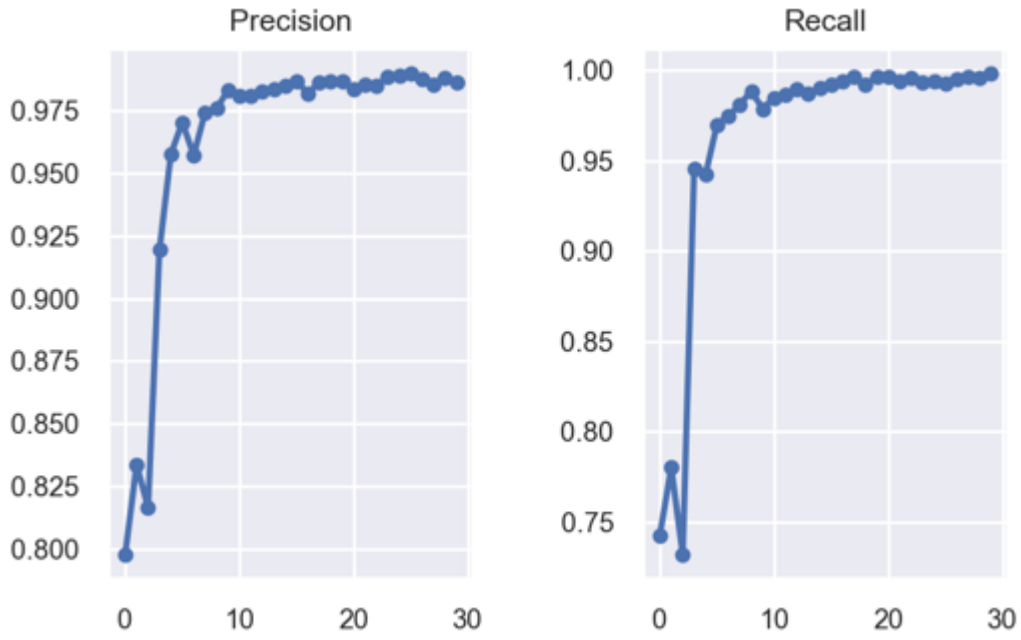


Figure 6.13 YOLOv7 Precision and Recall

Figure 6.14 YOLOv7 mAP@0.5

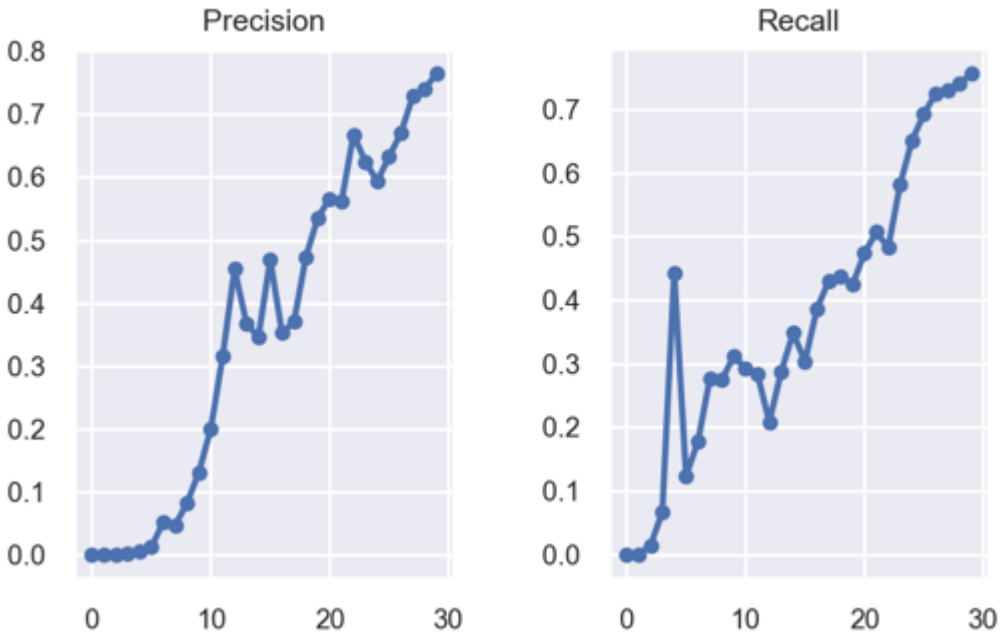


Figure 6.15 YOLOv4 Precision and Recall

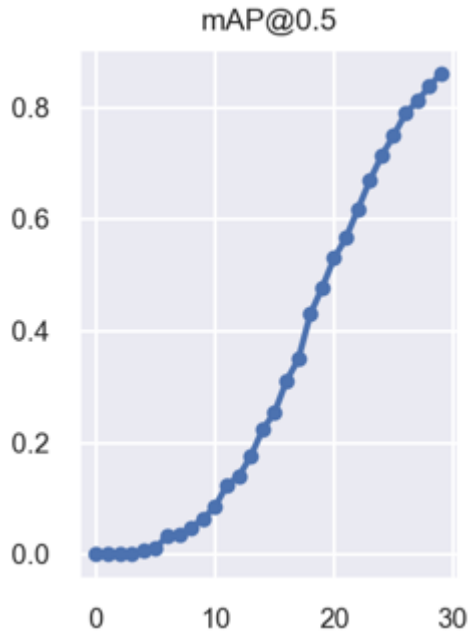


Figure 6.16 YOLOv4 mAP@0.5

In the case of YOLOv4, the training has shown a precision of 76.52% while the recall and mean average performance is 75.66% and 86.04% respectively. The table below demonstrates the performance results of both algorithms using precision, recall, and mean-average precision as evaluation metrics.

Table 6.3 Performance Results of YOLOv4 and YOLOv7

Models	**Precision**	**Recall**	**mAP@0.5**
YOLOv4	76.52%	75.66%	86.04%
YOLOv7	98.65%	99.83%	99.68%

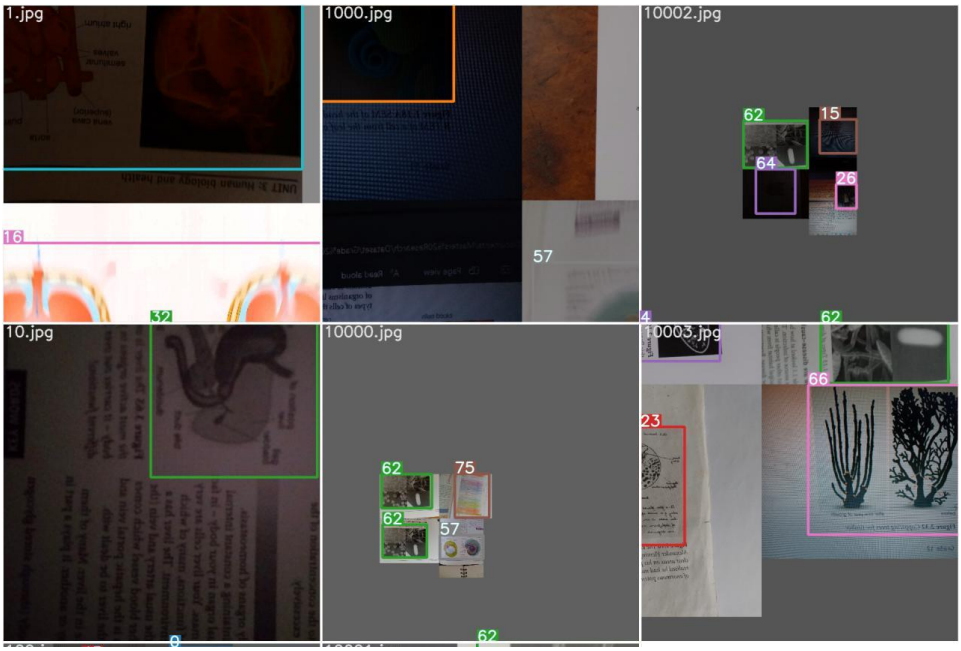


Figure 6.17. First train batch of YOLOv7

The above figure demonstrates the training batch generated while training with YOLO. This batch is generated on the first epoch. The below figure shows the first test batch for testing the weights of the trained model.

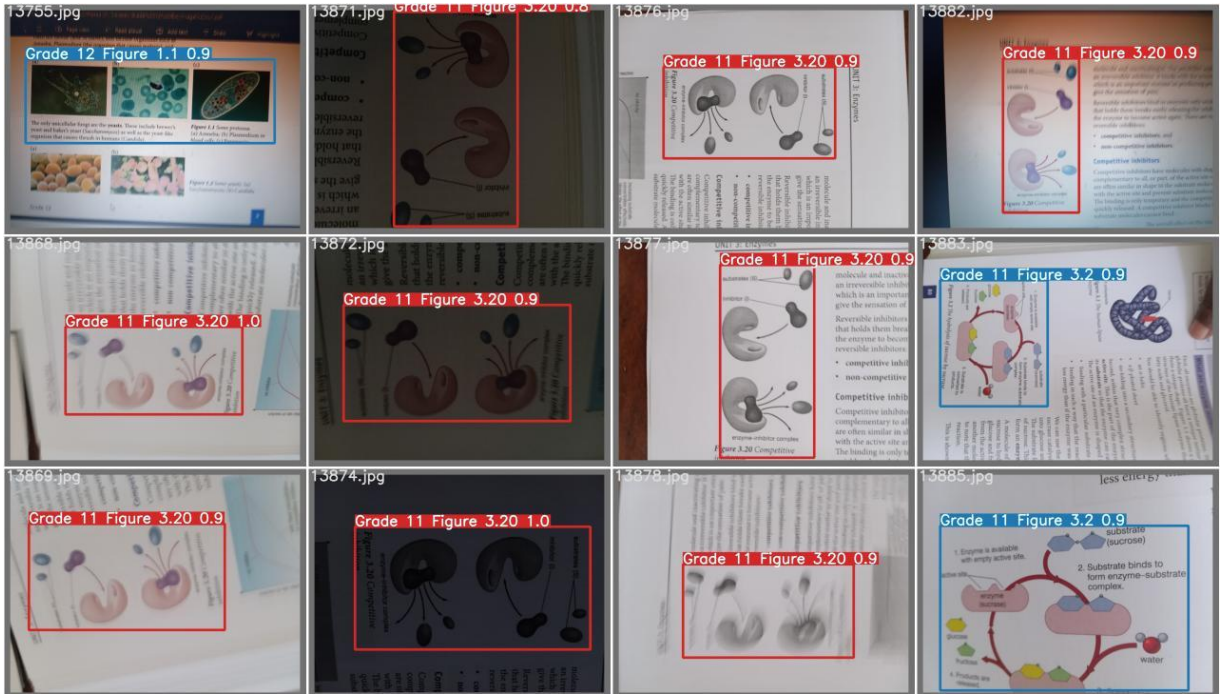


Figure 6.18. First test batch with a prediction of YOLOv7

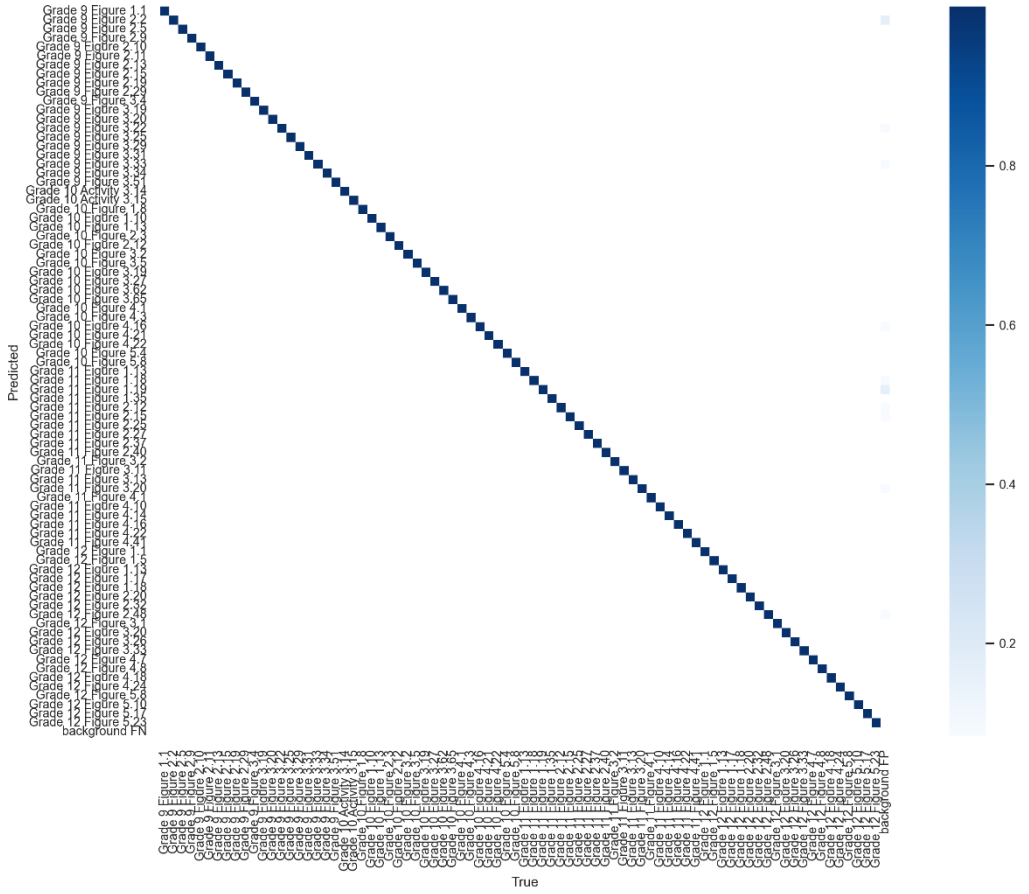


Figure 6.19 Confusion Matrix of YOLOv7

The confusion matrix is somehow blurred due to the high number of classes but it is seen that each class is above the color grade of 0.8 on the scale of the legend. This shows that YOLO has been able to detect all classes with ease and precision.

6.3.1 Performance Comparison between YOLOv4 and YOLOv7

Our custom dataset has been trained with both the architectures of YOLOv4 and YOLOv7. YOLOv7 has shown higher performance and lower latency than YOLOv4. The accuracy is measured with precision, recall, and mean-average precision. Latency is measured with frames per second (FPS) recorded in the inference stage. The following table demonstrates the performance comparison of both models.

Table 6.4 Performance Comparison between YOLOv4 and YOLOv7 on our dataset

Metric	YOLOv4	YOLOv7
mAP@0.5	0.86	0.99
Inference speed (FPS)	20	35
Parameters	64.4M	36.9M
Backbone network	CSPDarknet53	E-ELAN
Advantages	Considers more complex features	Faster and lighter weight
Disadvantages	A slower and larger model	Considers fewer complex features

6.4 Research Question Discussion

This research has been able to answer the questions raised in chapter one. A detailed discussion of the answers is given below.

Response for Q1: Acquiring the marker targets from the educational books has been the challenging aspect of this research. Every image including augmenting and contrasting the marker images was done by hand, not with the aid of software tools. This is because every image that will be acquired has to be unique and not synthetically made using software tools. Each marker figure has been acquired by considering versions of the books, orientation, blurriness, and lighting conditions of the environment. After all the images have been acquired in this manner, a manual synthetic data was created in which the acquired images are overlayed on top of different background images. The background images were gathered from different student libraries, and laboratories in both Addis Ababa and Adama City. This last step has been undertaken using manual labor and was unsuccessful in creating a wide variety of marker images due to time and other resource constraints.

Response for Q2: In the context of the books chosen for markers, deep learning has shown an immense improvement over traditional computer vision technique. This improvement is

especially observed in the copied version of the books, which most students have access to due to its cheap price. This can be illustrated in the figure below. Both figures are from the grade 12 book Figure 2.20.

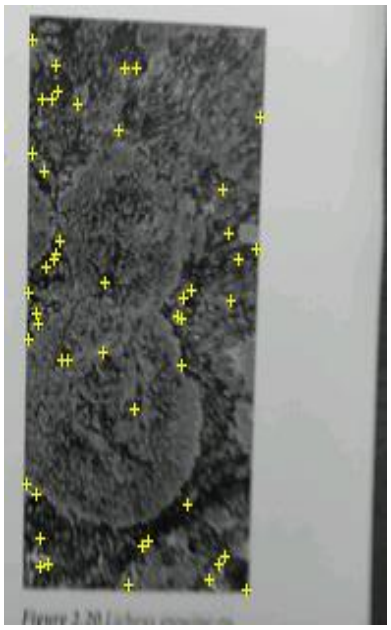
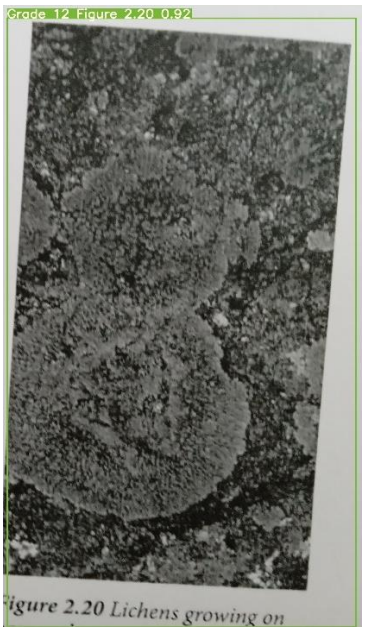
a.  b. 

Figure 6.20 Detection Comparison of CV (a) and DL (b)

The image on the left shows the feature extracted from the figure using the computer vision technique. It has shown a confidence rate of 20%. Meanwhile, the image on the right is the detection of the figure with the help of deep learning techniques. It has shown a confidence rate of 92%. Although this image is distorted and colorless the deep learning model has identified the marker from the image accurately. This shows that even with the impressive detection capabilities of the computer vision method there are certain scenarios, in the context of figures from the Ethiopian biology book, where its performance drops in the detection task.

Response for Q3: This research has been able to use three pre-trained models namely MobileNetV2, MobileNetV3Small, and MobileNetV3Large for classification. Among the Mobile Net pre-trained models MobileNetV3Large has shown better performance than its prior versions. By altering a few parameters such as epoch number, batch size, and adding and dropping hidden layers the result of the previous versions has been altered. Increasing the learning rate even by a small margin has shown a poor performance that led to an overfitting of the model even with a dropout layer added to decrease the effects of overfitting.

CHAPTER SEVEN

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

In the entirety of this research, we have been able to use the transfer learning method of deep learning for marker detection by using figures from educational biology books as markers. MobileNet is one of the pre-trained models utilized for this research since it is more compatible with Mobile CPUs by the augmented reality application could be developed on. The dataset was acquired from four Ethiopian high school biology books. Twenty figures from each book were selected as markers meaning 80 classes. Since creating a diverse dataset is very difficult for markers the dataset was collected while considering orientation, lighting, and blurriness conditions to create a richer dataset. The synthetic dataset is also a method used to create a more diverse dataset. This collected dataset was then trained using three pre-trained models as feature extractors which are MobileNetV2, MobileNetV3Small, and MobileNetV3Large. Around five experiments have been performed to enhance and increase the performance of each model. The experiments have shown MobileNetV3Large has shown better performance than its prior version with 96.21% accuracy and a loss of 0.31. Additionally, the dataset was trained using the YOLOv7 object detection algorithm. The YOLOv7 algorithm has shown a performance of high accuracy and better inference speed than the algorithm used in the previous works. Other research although attempted to achieve marker detection with other kinds of datasets it has not been done on figures of books and it has been laborious acquiring images and label all the marker images properly.

7.2 Future Work

Even though this research has covered most of the basic and difficult aspects of the work, there are still territories to be covered. In the future, it could be overseen that there are many works to be accomplished that this research hasn't been able to cover due to several reasons. This work covers only books from high schools, this could be increased by including lower educational books. It also only covers books on biology; this could be broadened to other STEM subjects like Chemistry.

Since there is no research performed on marker detection in the context of Ethiopian high school biology education, the scope of this research has been limited to covering marker detection and building a model for a two-dimensional augmented reality system. Works in the future can include building an augmented reality application by considering the domain experts, teachers, and students. Although students and teachers have been involved in this research on the data selection process and not in the development process, future works can take the finding of this research to build an AR system by involving them. By making use of the models created one can design and develop an augmented reality application that works on the selected figures of the books. Developing three-dimensional augmented reality inference was not the main focus of this research because of the lack of available libraries for yolov7, that would allow the detection model to be incorporated into the application development process. Lack of available time is also another factor for this exclusion. However, in the future, there is great promise for developers that want to incorporate their machine-learning model with augmented reality.

SPECIAL ACKNOWLEDGMENTS

This research study was funded by Adama Science and Technology University under the grant number:

ASTU/SM-R/850/23

Adama, Ethiopia

REFERENCES

Alhalabi, M., Ghazal, M., Haneefa, F., Yousaf, J., & El-Baz, A. (2021). Smartphone handwritten circuits solver using augmented reality and capsule deep networks for engineering education. *Education Sciences*, *11*(11). https://doi.org/10.3390/educsci11110661

Alkhalifah, T., Wang, H., & Ovcharenko, O. (2022). MLReal: Bridging the gap between training on synthetic data and real data applications in machine learning. *Artificial Intelligence in Geosciences*, *3*, 101–114. https://doi.org/10.1016/J.AIIG.2022.09.002

Alzahrani, N. M. (2020). Augmented Reality: A Systematic Review of Its Benefits and Challenges in E-learning Contexts. *Applied Sciences*, *10*(16). https://doi.org/10.3390/app10165660

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

Arslan, R., Kofoğlu, M., & Dargut, C. (2020). Development of augmented reality application for biology education. *Journal of Turkish Science Education*, *17*(1), 62–72.

Bekalo, S., Welford, G., Alasdair, M., & Welford, G. (2000). Practical activity in Ethiopian secondary physical sciences: implications for policy and practice of the match between the intended and implemented curriculum. *Research Papers in Education*, *15*(2), 185–212. http://www.tandf.co.uk/journals

Bekele, A., Melesse, K., Hunde, A. B., & Melesse Tegegne, K. (2010). *Qualitative Exploration on the Qualitative Exploration on the Application of Student-centered Learning in Mathematics and Natural Sciences: The case of Selected General Secondary Schools in Jimma, Ethiopia*.

Billinghurst, M., Clark, A., & Lee, G. (2014). A survey of augmented reality. In *Foundations and Trends in Human-Computer Interaction* (Vol. 8, Issues 2–3, pp. 73–272). Now Publishers Inc. https://doi.org/10.1561/1100000049

Blippar. (2020). *BRINGING THE AUGMENTED REALITY EDUCATION TO YOUR HOME FOR FREE*. https://www.blippar.com/blog/2020/04/07/bringing-the-augmented-reality-education-to-your-home-for-free

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*.

Bower, M., Howe, C., McCredie, N., Robinson, A., & Grover, D. (2014). Augmented Reality in education - cases, places and potentials. In *Educational Media International* (Vol. 51, Issue 1, pp. 1–15). https://doi.org/10.1080/09523987.2014.889400

Daba, T. M., & Anbesaw, M. S. (2016). CORRESPONDENCE Tolessa Muleta Daba Factors Affecting Implementation of Practical Activities in Science education in Some Selected Secondary and Preparatory Schools of Afar Region, North East Ethiopia OPEN ACCESS. In *INTERNATIONAL JOURNAL OF ENVIRONMENTAL & SCIENCE EDUCATION* (Vol. 11, Issue 12).

Dash, A. K., Behera, S. K., Dogra, D. P., & Roy, P. P. (2018). Designing of marker-based augmented reality learning environment for kids using convolutional neural network architecture. *Displays*, *55*, 46–54. https://doi.org/10.1016/j.displa.2018.10.003

Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, *40*, 100379.

Education, E. (2015). *Meetup: Augmented Reality in Education*. https://medium.com/emerge-edtech-insights/meetup-augmented-reality-in-education-e78039fdf80b

Estrada, J., Paheding, S., Yang, X., & Niyaz, Q. (2022). Deep-Learning-Incorporated Augmented Reality Application for Engineering Lab Training. *Applied Sciences (Switzerland)*, *12*(10). https://doi.org/10.3390/app12105159

Gogile Zengele, A., & Alemayehu, B. (2016). *The Status of Secondary School Science Laboratory Activities for Quality Education in Case of Wolaita Zone, Southern Ethiopia.* (Vol. 7, Issue 31). Online. www.iiste.org

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., & others. (2019). Searching for mobilenetv3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314–1324.

Jabeen, F., & Afzal, M. T. (2020). Effect of Simulated Chemistry Practicals On Students' Performance At Secondary School Level. *Journal of Education and Educational Development*, *7*(1), 119. https://doi.org/10.22555/joeed.v7i1.2600

Kelleher, J. D. (2019). *Deep learning*. MIT press.

Kriegeskorte, N., & Golan, T. (2019). Neural network models and deep learning. *Current Biology*, *29*(7), R231–R236. https://doi.org/10.1016/J.CUB.2019.02.034

Lampropoulos, G., Keramopoulos, E., & Diamantaras, K. (2020). Enhancing the functionality of augmented reality using deep learning, semantic web and knowledge graphs: A review. *Visual Informatics*, *4*(1), 32–42. https://doi.org/10.1016/j.visinf.2020.01.001

Le, H., Nguyen, M., & Yan, W. Q. (2020). Machine learning with synthetic data–a new way to learn and classify the pictorial augmented reality markers in real-time. *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 1–6.

Le, H., Nguyen, M., Yan, W. Q., & Nguyen, H. (2021). Augmented reality and machine learning incorporation using yolov3 and arkit. *Applied Sciences (Switzerland)*, *11*(13). https://doi.org/10.3390/app11136006

le Huy and Nguyen, M. (2020). An Online Platform for Enhancing Learning Experiences with Web-Based Augmented Reality and Pictorial Bar Code. In V. Geroimenko (Ed.), *Augmented Reality in Education: A New Technology for Teaching and Learning* (pp. 45–57). Springer International Publishing. https://doi.org/10.1007/978-3-030-42156-4_3

Lindner, C., Rienow, A., & Jürgens, C. (2019). Augmented Reality applications as digital experiments for education–An example in the Earth-Moon System. *Acta Astronautica*, *161*, 66–74.

Liu, C., Tao, Y., Liang, J., Li, K., & Chen, Y. (2018). Object detection based on YOLO network. *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 799–803.

Louis B. Rosenberg. (1992). *The Use of Virtual Fixtures as Perceptual Overlays to Enhance Operator Performance in Remote Environments*.

Mahasin, M., & Dewi, I. A. (2022). Comparison of CSPDarkNet53, CSPResNeXt-50, and EfficientNet-B0 Backbones on YOLO V4 as Object Detector. *International Journal of Engineering, Science and Information Technology*, *2*(3), 64–72.

Majeed, Z. H., & Ali, H. A. (2020). A review of augmented reality in educational applications. In *International Journal of Advanced Technology and Engineering Exploration* (Vol. 7, Issue 62, pp. 20–27). Accent Social and Welfare Society. https://doi.org/10.19101/IJATEE.2019.650068

Man, K., & Chahl, J. (2022). A Review of Synthetic Image Data and Its Use in Computer Vision. *Journal of Imaging*, *8*(11), 310.

Muleta Daba, T., Anbassa, B., Kere Oda, B., & Degefa, I. (2016). *Educational Research and Reviews Status of biology laboratory and practical activities in some selected secondary and preparatory schools of Borena zone, South Ethiopia*. *11*(17), 1709–1718. https://doi.org/10.5897/ERR2016.2946

Nakasi, R., Mwebaze, E., Zawedde, A., Tusubira, J., Akera, B., & Maiga, G. (2020). A new approach for microscopic diagnosis of malaria parasites in thick blood smears using pre-trained deep learning models. *SN Applied Sciences*, *2*, 1–7.

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., & Walsh, J. (2020). Deep learning vs. traditional computer vision. *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*, 128–144.

Oufqir Zainab and El Abderrahmani, A. and S. K. (2020). From Marker to Markerless in Augmented Reality. In S. C. and S. H. Bhateja Vikrant and Satapathy (Ed.), *Embedded Systems and Artificial Intelligence* (pp. 599–612). Springer Singapore.

Prakash, A., Boochoon, S., Brophy, M., Acuna, D., Cameracci, E., State, G., Shapira, O., & Birchfield, S. (2018). *Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data*. http://arxiv.org/abs/1810.10093

Rahnemoonfar, M., & Sheppard, C. (2017). Deep count: fruit counting based on deep simulated learning. *Sensors*, *17*(4), 905.

Ramamoorthy, G., & Kiran, M. (2021). Smart Library Using Augmented Reality. In *PAIDEUMA JOURNAL: Vol. XIV* (Issue 5).

Ranganathan, H., Chakraborty, S., & Panchanathan, S. (2016). Multimodal emotion recognition using deep learning architectures. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1–9.

R.B. Ravi Varma, I.M. Umesh, & R.S. Upendra. (2021). Augmented Reality and Deep Learning in e-learning – A New Approach. *International Journal of Applied Engineering Research ISSN 0973-4562 Volume 16, Number 9*, 749–751.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

Schraml, D. (2019). Physically based synthetic image generation for machine learning: a review of pertinent literature. *Photonics and Education in Measurement Science 2019*, *11144*, 108–120.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 1–48.

Siltanen, S. (2012). *Theory and applications of marker-based augmented reality: Licentiate thesis*.

Slater, M., Gonzalez-Liencres, C., Haggard, P., Vinkers, C., Gregory-Clarke, R., Jelley, S., Watson, Z., Breen, G., Schwarz, R., Steptoe, W., Szostak, D., Halan, S., Fox, D., & Silver, J. (2020). The Ethics of Realism in Virtual and Augmented Reality. *Frontiers in Virtual Reality*, *1*. https://doi.org/10.3389/frvir.2020.00001

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1701–1708. https://doi.org/10.1109/CVPR.2014.220

Tesfamariam, G., Lykknes, A., & Kvittingen, L. (2014). SMALL-SCALE CHEMISTRY FOR A HANDS-ON APPROACH TO CHEMISTRY PRACTICAL WORK IN SECONDARY SCHOOLS: EXPERIENCES FROM ETHIOPIA. *AJCE*, *4*(3).

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., & Birchfield, S. (2018). *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*. http://arxiv.org/abs/1804.06516

Vanherle, B., Moonen, S., van Reeth, F., & Michiels, N. (2022). *Analysis of Training Object Detection Models with Synthetic Data*. http://arxiv.org/abs/2211.16066

Vega-Márquez, B., Rubio-Escudero, C., Riquelme, J. C., & Nepomuceno-Chamorro, I. (2020). Creation of Synthetic Data with Conditional Generative Adversarial Networks. *Advances in Intelligent Systems and Computing*, *950*, 231–240. https://doi.org/10.1007/978-3-030-20055-8_22

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *ArXiv Preprint ArXiv:2207.02696*.

Wang, M., Callaghan, V., Bernhardt, J., White, K., & Peña-Rios, A. (2018). Augmented reality in education and training: pedagogical approaches and illustrative case studies. *Journal of Ambient Intelligence and Humanized Computing*, *9*(5), 1391–1402. https://doi.org/10.1007/s12652-017-0547-8

WongKinYiu. (2022). YOLOv7. In *GitHub repository*. GitHub. https://github.com/WongKinYiu/yolov7

Yang, Q., Zhang, Y., Dai, W., & Pan, S. J. (2020). *Transfer learning*. Cambridge University Press.

Younis, A., Shixin, L., Jn, S., & Hai, Z. (2020). Real-time object detection using pre-trained deep learning models MobileNet-SSD. *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, 44–48.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865

APPENDICES

Appendix A: Sample Code

Data Augmentation

```
from keras.preprocessing.image import ImageDataGenerator
from skimage import io
import numpy as np
import os
from PIL import Image
generated_data = ImageDataGenerator(
        rotation_range = 40,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = True,
        brightness_range = (0.5, 1.5))
image_source = r'D:/synthetic/syn/New folder/syn/16/'
width = 1856
height=4160
Input_data = []
input_images = os.listdir(image_source)
for i, name in enumerate(input_images):
    if (name.split('.')[1] == 'jpg'):
        input_image = io.imread(image_source + name)
        input_image = Image.fromarray(image, 'RGB')
        input_image = input_image.resize((width,height))
        Input_data.append(np.array(input_image))
x = np.array(Input_data)
i = 0
for batch in generated_data.flow(x, batch_size=16,
                          save_to_dir= r'D:/synthetic/syn/New
folder/syn/16/',
                          save_prefix='dr',
                          save_format='jpg'):
    i += 1
    if i > 50:
        break
```

Webcam Inference

```
import argparse
import time
from pathlib import Path
import cv2 as cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random
from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow,
non_max_suppression, apply_classifier, \
    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier,
time_synchronized, TracedModel
import numpy as np
import webbrowser
logo = cv2.imread('lifecycle.png')
size = 300
logo = cv2.resize(logo, (size, size))
logo1 = cv2.imread('eye.png')
logo1 = cv2.resize(logo1, (350, size))
heightt,widthh,channelss=logo.shape
heightt1,widthh1,channelss1=logo1.shape
qr=cv2.imread('qr1.png')
qr=cv2.resize(qr,(80,80))
qr1=cv2.imread('qr2.png')
qr1=cv2.resize(qr1,(80,80))
img2gray = cv2.cvtColor(logo, cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 1, 255, cv2.THRESH_BINARY)
img2gray1 = cv2.cvtColor(logo1, cv2.COLOR_BGR2GRAY)
ret1, mask1 = cv2.threshold(img2gray1, 1, 255, cv2.THRESH_BINARY)
img2gray2 = cv2.cvtColor(qr, cv2.COLOR_BGR2GRAY)
ret_qr, mask_qr = cv2.threshold(img2gray2, 1, 255, cv2.THRESH_BINARY)
img2gray3 = cv2.cvtColor(qr1, cv2.COLOR_BGR2GRAY)
ret_qr1, mask_qr1 = cv2.threshold(img2gray3, 1, 255, cv2.THRESH_BINARY)
def mouse_click(event, x, y, flags, param):
        if event == cv2.EVENT_LBUTTONDOWN and param==0:
                webbrowser.open('https://www.niaid.nih.gov/diseases-
conditions/schistosomiasis-
bilharzia#:~:text=Schistosomiasis%2C%20also%20known%20as%20bilharzia,japoni
cum.',new=2)
        elif event == cv2.EVENT_LBUTTONDOWN and param==1:
        webbrowser.open('https://stokes.byu.edu/teaching_resources/resolve.
html',new=2)
def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz, trace = opt.source,
opt.weights, opt.view_img, opt.save_txt, opt.img_size, not opt.no_trace
    save_img = not opt.nosave and not source.endswith('.txt')  # save
```

```
webcam     =     source.isnumeric()     or     source.endswith('.txt')     or
source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://', 'https://'))

    # Directories
    save_dir    =    Path(increment_path(Path(opt.project)    /    opt.name,
exist_ok=opt.exist_ok))  # increment run
    (save_dir  /  'labels'  if  save_txt  else  save_dir).mkdir(parents=True,
exist_ok=True)  # make dir

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(imgsz, s=stride)  # check img_size

    if trace:
        model = TracedModel(model, device, opt.img_size)

    if half:
        model.half()  # to FP16

    # Second-stage classifier
    classify = False
    if classify:
        modelc = load_classifier(name='resnet101', n=2)  # initialize
        modelc.load_state_dict(torch.load('weights/resnet101.pt',
map_location=device)['model']).to(device).eval()

    # Set Dataloader
    vid_path, vid_writer = None, None
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image size
inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride)

    # Get names and colors
    names = model.module.names if hasattr(model, 'module') else model.names
    colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

    # Run inference
    if device.type != 'cpu':
        model(torch.zeros(1,                    3,                    imgsz,
imgsz).to(device).type_as(next(model.parameters())))  # run once
    old_img_w = old_img_h = imgsz
    old_img_b = 1
```

```
t0 = time.time()
    for path, img, im0s, vid_cap in dataset:
        img = torch.from_numpy(img).to(device)
        img = img.half() if half else img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        # Warmup
        if device.type != 'cpu' and (old_img_b != img.shape[0] or old_img_h
!= img.shape[2] or old_img_w != img.shape[3]):
            old_img_b = img.shape[0]
            old_img_h = img.shape[2]
            old_img_w = img.shape[3]
            for i in range(3):
                model(img, augment=opt.augment)[0]

        # Inference
        t1 = time_synchronized()
        with torch.no_grad():    # Calculating gradients would cause a GPU
memory leak
            pred = model(img, augment=opt.augment)[0]
        t2 = time_synchronized()

        # Apply NMS
        pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres,
classes=opt.classes, agnostic=opt.agnostic_nms)
        t3 = time_synchronized()

        # Apply Classifier
        if classify:
            pred = apply_classifier(pred, modelc, img, im0s)

        # Process detections
        for i, det in enumerate(pred):  # detections per image
            if webcam:  # batch_size >= 1
                p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(),
dataset.count
            else:
                p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame',
0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # img.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode
== 'image' else f'_{frame}')  # img.txt
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain
whwh
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4],
im0.shape).round()
```

```
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) /
gn).view(-1).tolist()  # normalized xywh
                        line = (cls, *xywh, conf) if opt.save_conf else
(cls, *xywh)  # label format
                        with open(txt_path + '.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line +
'\n')
                    if save_img or view_img:  # Add bbox to image
                        label = f'{names[int(cls)]} {conf:.2f}'
                        plot_one_box(xyxy, im0, label=label,
color=colors[int(cls)], line_thickness=1)
            print(f'{s}Done. ({(1E3 * (t2 - t1)):.1f}ms) Inference, ({(1E3 *
(t3 - t2)):.1f}ms) NMS')
            if view_img:
                ty=""
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    ty+=f"{n} {names[int(c)]}{'s' * (n > 1)}, "
                if ty=="1 Grade 9 Figure 1.1, ":
                    roi1=im0[400:480,560:640]
                    roi1 [np.where(mask_qr)]=0
                    roi1+=qr
                    #result=cv2.addWeighted(roi,1,logo,0.5,0)
                    check_height=int(xyxy[1])+heightt
                    check_width=int(xyxy[0])+widthh
                    roi=[]
                    if check_height<480 and check_width<640:
roi=im0[int(xyxy[1]):check_height,int(xyxy[0]):check_width]
                            roi [np.where(mask)]=0
                            roi+=logo
                    cv2.putText(im0,"Reference
Link:",(300,450),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,255),1)
elif ty=="1 Grade 9 Figure 2.2, ":
                    roi4=im0[400:480,560:640]
                    roi4 [np.where(mask_qr1)]=0
                    roi4+=qr1
                    check_height1=int(xyxy[1])+heightt1
                    check_width1=int(xyxy[0])+widthh1
                    roi2=[]
                    if check_height1<480 and check_width1<640:
roi2=im0[int(xyxy[1]):check_height1,int(xyxy[0]):check_width1]
                            roi2 [np.where(mask1)]=0
                            roi2+=logo1
                    cv2.putText(im0,"Reference
Link:",(300,450),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,255),1)
                    param=1
                else:
                    param=9
```

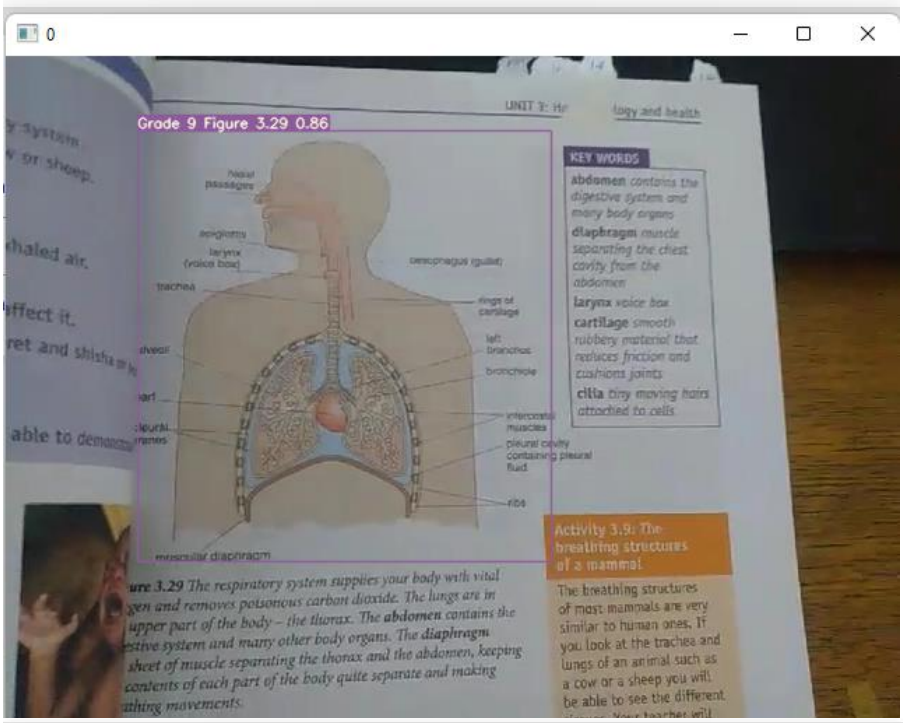
```
cv2.imshow(str(p), im0)
                cv2.setMouseCallback(str(p),mouse_click,param)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                    print(f" The image with the result is saved in:
{save_path}")
                else:  # 'video' or 'stream'
                    if vid_path != save_path:  # new video
                        vid_path = save_path
                        if isinstance(vid_writer, cv2.VideoWriter):
                            vid_writer.release()  # release previous video
writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                            save_path += '.mp4'
                        vid_writer       =       cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer.write(im0)

    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to
{save_dir / 'labels'}" if save_txt else ''
        #print(f"Results saved to {save_dir}{s}")

    print(f'Done. ({time.time() - t0:.3f}s)')


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights',          nargs='+',          type=str,
default='yolov7.pt', help='model.pt path(s)')
    parser.add_argument('--source',  type=str,  default='inference/images',
help='source')  # file/folder, 0 for webcam
    parser.add_argument('--img-size', type=int, default=640, help='inference
size (pixels)')
    parser.add_argument('--conf-thres',      type=float,      default=0.25,
help='object confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU
threshold for NMS')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or
0,1,2,3 or cpu')
    parser.add_argument('--view-img',   action='store_true',   help='display
results')
    parser.add_argument('--save-txt',    action='store_true',    help='save
results to *.txt')
```

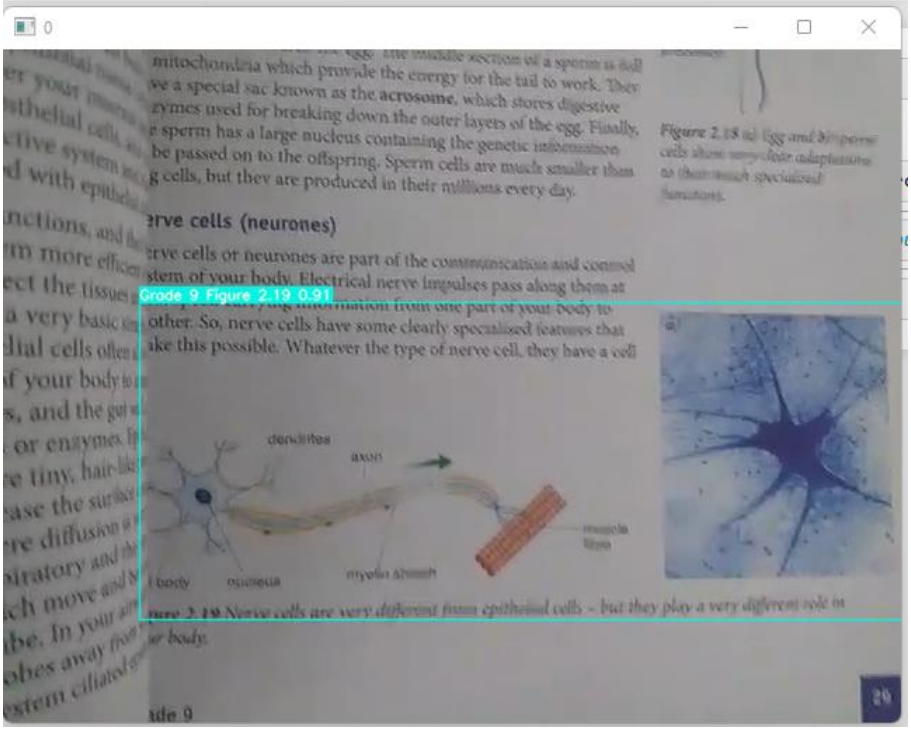
```
parser.add_argument('--save-conf', action='store_true', help='save
confidences in --save-txt labels')
    parser.add_argument('--nosave', action='store_true', help='do not save
images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by
class: --class 0, or --class 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-
agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented
inference')
    parser.add_argument('--update', action='store_true', help='update all
models')
    parser.add_argument('--project', default='runs/detect', help='save
results to project/name')
    parser.add_argument('--name', default='exp', help='save results to
project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing
project/name ok, do not increment')
    parser.add_argument('--no-trace', action='store_true', help='don`t trace
model')
    opt = parser.parse_args()
    print(opt)
    #check_requirements(exclude=('pycocotools', 'thop'))

    with torch.no_grad():
        if opt.update:  # update all models (to fix SourceChangeWarning)
            for opt.weights in ['yolov7.pt']:
                detect()
                strip_optimizer(opt.weights)
        else:
            detect()
```

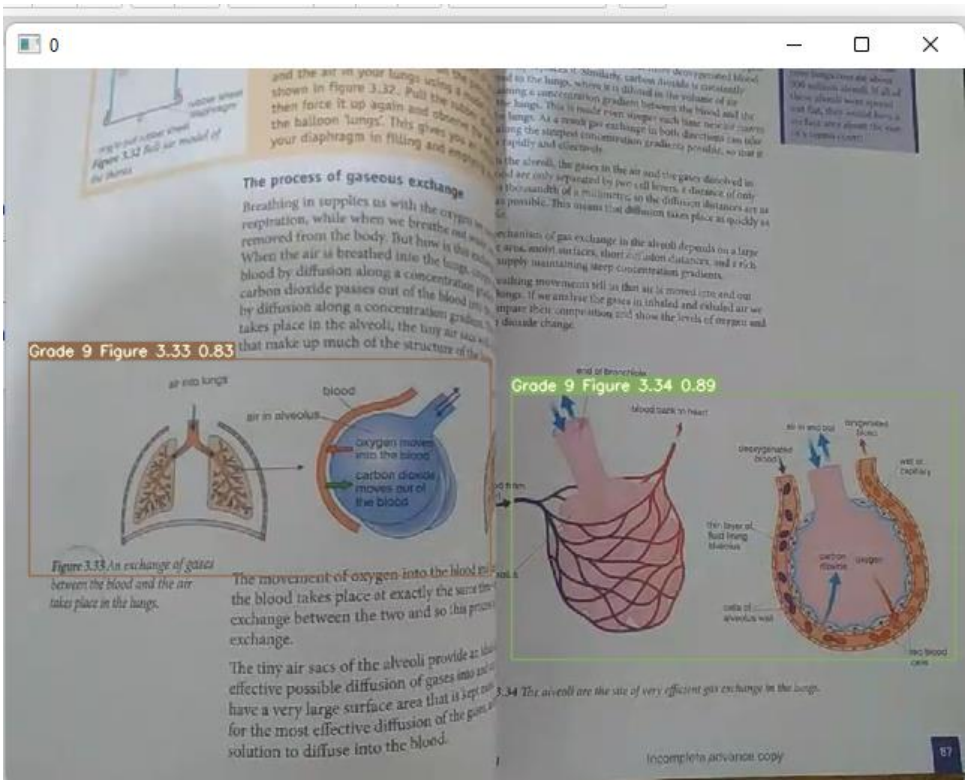
This code is referred from and edited from the official YOLOv7 GitHub repo (WongKinYiu, 2022).

Appendix B: Marker Detection and Augmented Reality Implementation Result

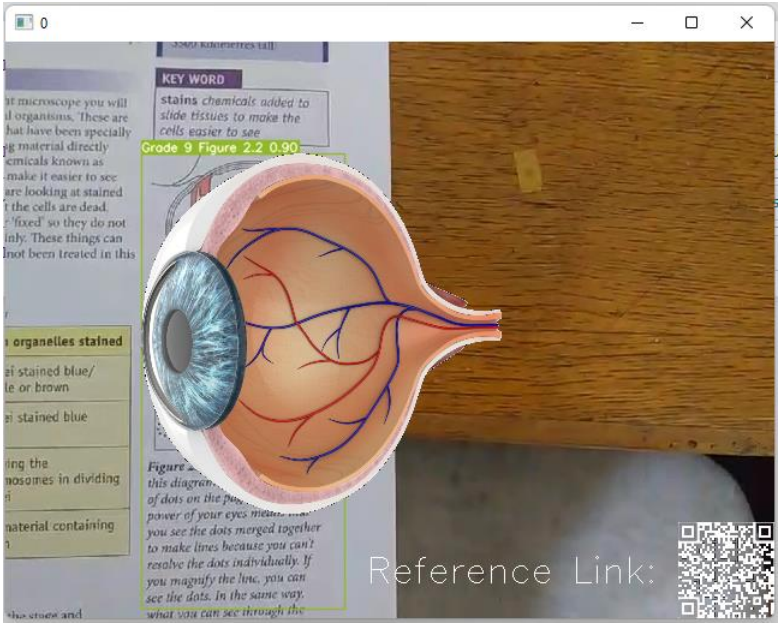
Marker Detection







Augmented Reality Implementation



X