Abstract

Object detection is an important part of the image processing system, especially for applications like Face detection, Visual search engine, counting, and Aerial Image analysis. Hence the performance of object detectors plays an important role in the functioning of such systems. With the advancements in the Deep learning field, Convolutional Neural Networks (CNN) are now the state of the art in object detection and classification.

This project focuses on the design and implementation of a computer vision application designed to utilize object detection algorithms to recognize objects in real-life scenarios, the main goal is to create a versatile Android application capable of accurately detecting and identifying objects from images or video streams captured by the device's camera, the project begins by exploring various object detection algorithms and their suitability for mobile platforms. Pre-trained models, such as YOLO (You Only Look Once), SSD (Single Shot Detector), and Faster R-CNN (Region-based Convolutional Neural Network), and after checking accuracy, speed, and memory requirements. After careful analysis, the most appropriate object detection model is selected and integrated into the Android application.

These algorithms are trained in COCO COCO (Common Objects in Context) is a widely used dataset for training and evaluating object detection, contains more than 200,000 images, These images cover 80 different object categories, including common everyday objects like people, animals, vehicles, and household items, which provide a robust base for object detection, and then use transfer learning on custom data set called Table equipment from roboflow to make them capable of identifying different objects in images such as book, bottle, earphone, glass, headphone, keyboard, laptop, mobile, mouse, pen, penstand.

Contents

1. Introduction
   1. Scope
2. S
3. S

List of Figures

List of Tables

Chapter 1

Introduction

With the evolution in Deep Learning (DL) [1] in the past few years, we are able to create complex ML models for detecting objects in images, As people are using their mobile phones to a larger extent, and also expect increasingly advanced performance from their mobile applications, the industry needs to adopt more advanced technologies to meet up to expectations. One such adaptation could be the use of ML algorithms for object detection.

ML is commonly divided into two phases namely the training and the inference phase. Training is the phase where a model, usually a neural network, is trained to behave in a certain way based on given datasets. This step can easily be carried out in the cloud and distributed to mobile devices, where the trained models can be used for inference on previously unknown data.

When applying more advanced technologies and algorithms in a mobile environment one of the challenges is the limited computational power of the mobile hardware. As inference is computationally expensive, it is crucial that operations are optimized for mobile devices, By using the mobile version of TensorFlow named TensorFlow lite [2].

* 1. Scope

The scope of the project is to develop an “Objects Detection Application” which is an Android mobile application that allows users to detect objects from a photo or real-time Video. It can be utilized for a wide range of applications, such as visual search, augmented reality, Coarse Classification, accessibility, image organization, and classification for a custom model. As a result, users will be able to engage with the digital world smartly and intuitively.

The following challenges have been identified.

* Computational cost time during the recall of the model, as it should be capable of running on a mobile device with limited computational power.
* Identification of multiple objects in a single frame, where some objects might be only partially visible, and others are overlapping.
* getting the suitable dataset
  1. Problem Statement

the task of identifying objects in an image in resource-limited computing environments is challenging as their limited computational power and memory constraints hinder the usage of complex deep-learning models, this thesis aims to examine if DL models can be used on mobile devices to outperform, the existing heuristic-based visual object detection algorithms in terms of recall performance. Furthermore, this thesis examines the constraints in delay and computational time in the inference phase for the use of DL models on mobile.

* 1. Approach

\*\*\*

Our object detection mobile application aims to provide real-time object detection capabilities on mobile devices. The approach we follow has three main phases

* general object detection

I begin by selecting a suitable object detection model that balances accuracy and computational efficiency for mobile devices. Pre-trained models, such as YOLO (You Only Look Once) [3], SSD (Single Shot Detector) [4], and Faster R-CNN (Region-based Convolutional Neural Network) [5], algorithms are trained in COCO (Common Objects in Context)[6] is a widely used dataset for training and evaluating object detection, contains more than 200,000 images, These images cover 80 different object categories, including common everyday objects like people, animals, vehicles, and household items.

* Train on custom dataset

because of the limitation of our dataset hence contains only 1400 images, by utilizing pre-trained models and their learned representations, transfer learning enables the transfer of knowledge and features from a source domain to a target domain, even with limited labeled data.

* Deployment using TensorFlow Lite

With TensorFlow Lite, models can be optimized and converted into a format that enables fast and low-latency inference, making it ideal for real-time applications and scenarios where computational resources are limited.

* 1. Outcome

\*\*

At the start of the project, the idea was to develop a basic object detection application by using the machine learning algorithm of TensorFlow API. There were two features of detection which are simple detection and mirror detection along with the results screen to follow the detection flow. In the results screen, a list of all detected objects will be displayed, and a user can click on any object name to make online shopping or research.

Chapter 2

Aims and Objectives

The aims and objectives of this project is to provide an enhanced experience for the users to detect objects from an image by using advanced machine learning algorithms. Unlike human vision, computers and mobiles can’t detect objects in a fraction of a second. Humans can recognise and detecting the objects we have been looking for, but to facilitate this need by artificial intelligence requires the usage of machine learning algorithms prepared especially for these types of tasks to detect the objects from an image, or a live camera feed.

I planned and used one of the most appropriate model based on the project requirements from the TensorFlow API. The single-shot detection (SSD) network was the ideal choice for a high-speed model which is capable of detecting video feeds at a high frame rate. The SSD network, as the name implies, determines all bounding box probabilities in one pass, making it a faster model [4]. The goal of this project is to fulfil the growing demand for precise and efficient object detection from images or live camera feeds. The ultimate result will be a better user experience, allowing consumers and organizations to employ artificial intelligence to quickly and reliably recognize and interact with objects. And on later stage, the detected objects can be referred to the online platforms where a user can search the similar products for reviews & purchases [5].

All aims and objectives listed below were achieved:

Aim 1: Provide an enhanced experience for the users to detect objects from an image by using advanced machine learning algorithms.

* Objective 1.1: Design & develop a mobile application with the help of Google machine learning algorithms.
* Objective 1.2: Functional and non-functional requirements fulfilment of objects detection application.

The aims and objectives were successfully achieved by designing and developing an interactive mobile application for detecting objects from images. A live feed camera input feature is being achieved by using pretrained dataset from TensorFlow to detect objects from live feed camera.

Aim 2: Resolve the ongoing demand for object detection from a photo with the help of Google machine learning algorithms.

* Objective 2.1: A complete comprehensive solution to detect objects from uploaded images.
* Objective 2.2: Machine learning algorithms testing and deployment inside a mobile application.
* Objective 2.3: An easy-to-use and user-friendly interface for app users.

The aims and objectives were successfully achieved by getting responses from the users that the application is easy to use. Moreover, uploading images from the mobile local storage and then being converted into a bitmap image feature for detection is achieved.

Aim 3: Fulfil the growing demand for precise and efficient object detection from images or live camera feeds. The ultimate result will be a better user experience, allowing consumers and organizations to employ artificial intelligence to quickly and reliably recognize and interact with objects. And on a later stage, the detected objects can be referred to the online platforms where a user can search the similar products for reviews & purchases.

* Objective 3.1: Detected objects filtration by a customised smart solution to categorise the objects.
* Objective 3.2: Compare the detected objects with online datasets and prepare the outputs.
* Objective 3.3: Find detected objects on e-commerce websites and recommend the users where they can purchase the items.

The goal here is to allow users to select the objects detected to be redirected to an e-commerce website for suggestions and purchase feature was successfully achieved. WebView intent to redirect users to amazon UK website and pass the object names in the URL was utilised.

Chapter 3

Background

this section provides an overview of the key concepts, technologies, and developments relevant to the field of object detection. It aims to provide a contextual understanding of the problem and the advancements made in the area of object detection.

* 1. Machine learning

machine learning is defined by Arthur Samuel as a” field of study that gives computers the ability to learn without being explicitly programmed. machine learning algorithms avoid us need for explicit programming by improving an internal model through data. This process is called training and the data used to train the model is often regarded as the model’s experience. As depicted in Figure 1, ml is divided into the subfields of supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [7].

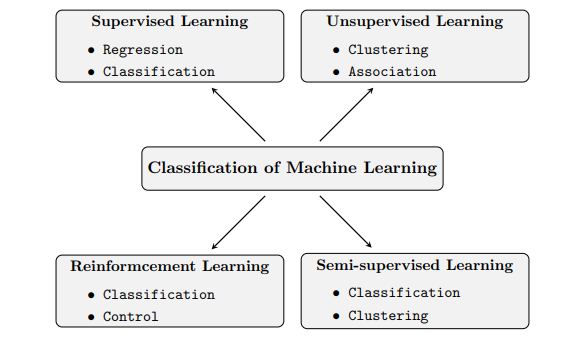


Figure 1: the subfields of machine learning

* 1. Artificial neural network

An artificial neural network (ANN) also called a neural network is the foundation of deep learning, As figures Figure 2 shows, the artificial neurons in an ANN are grouped in layers. There are three important types of layers

* The input layer
* the output layer
* an arbitrary number of hidden layers in between the input and output layer

Similar to neurons in human brains, nodes of different layers can be connected, In ANNs, the nodes exchange signals in the form of numbers. Each node outputs a number that is computed by applying a non-linear function to its inputs. The output signal can then be a new input for other nodes, or it can be part of the result returned by the output layer. The connections between nodes are also known as edges and typically carry a weight, With ANNs, the training process that is typical for all machine learning systems is the adjustment of these connection weights. The weights and other variables of the ANN are grouped under the term parameters

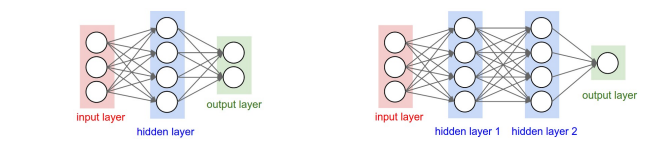


Figure 2: 2-layer ann and 3-layer ann

* 1. Deep learning

Deep learning is a subarea of machine learning. Deep learning is characterized by the use of and with many hidden layers. The more hidden layers a network has, the deeper it is.

The deeper a network is and the more nodes the network has per layer, the more complex the computations that the ann can successfully perform. As the number of layers and nodes grows, so does the number of parameters. Their large number is the reason deep learning requires extensive amounts of data to provide adequate results compared to other sub-disciplines of machine learning as shown in Figure 3. Networks of this genus have the ability to perform extraordinarily complex computations at the expense of a resource-intensive training process [8]

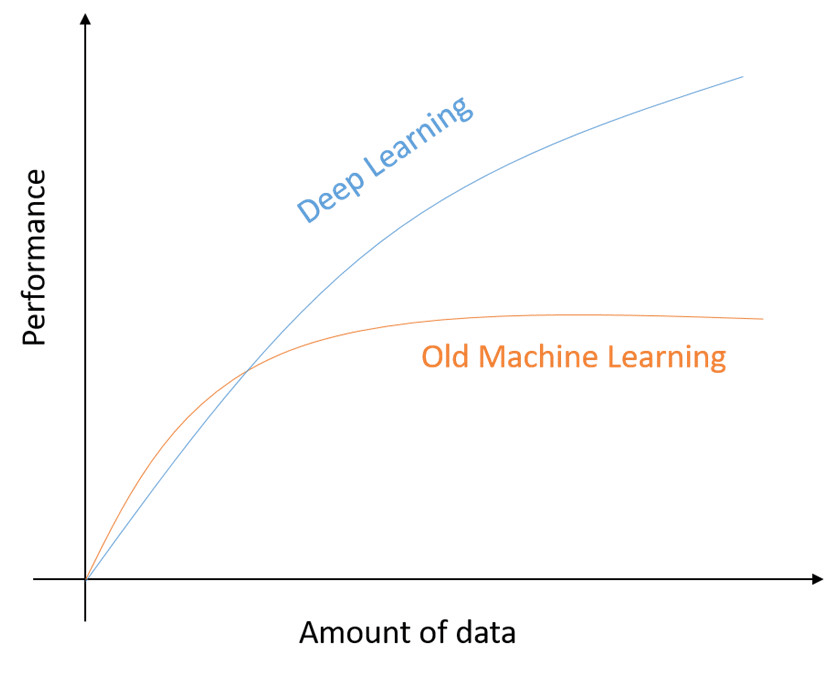


Figure 3: the amount of data vs performance in ML and DL

* 1. Computer Vision

Computer vision is a field of study that focuses on enabling computers to gain an understanding of visual data, such as images and videos, by extracting meaningful information and patterns they see in a similar way as that of humans. The main aim of computer vision is to generate relevant information from image and video data in order to deduce something about the world. It can be classified as a sub-field of artificial intelligence and machine learning. This is quite different from image processing, which involves manipulating or enhancing visual information and is not concerned about the contents of the image. Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision, and traffic automation [9] .

Today, machine learning is a necessary component of many computer vision algorithms. These algorithms are typically a combination of image processing and machine learning techniques. The major requirement of these algorithms is to handle large amounts of image/video data and to be able to perform computation in real-time for a wide range of applications. For example, real-time detection and tracking.

* 1. Object Detection

Object detection is one of the fundamental tasks in computer vision. Typically, object detection and recognition involve two steps: first, the potential location of each target object is localized; then, the objects are classified into different categories. Before the bloom of deep learning methods, object detection methods relied on manually designed features and designed classifiers based on how humans understand objects. In recent years, the field of object detection has dramatically advanced due to the success of deep learning, especially deep convolutional neural networks (CNN) [10]. Object detection has been widely used in many applications, such as autonomous driving, visual search, virtual reality (VR), and augmented reality (AR), etc.

* 1. Transfer Learning

It is rare to train a CNN from scratch, as it requires rarely large quantities of data to perform well, and training a CNN with such large amounts of data could take weeks on GPU clusters. As of this, it is a common practice to use pre-trained networks as an initialization or feature extractor for the task to be implemented. The task of reusing a pre-trained network is known as Transfer Learning and implies transferring knowledge from one domain to another. The base network is commonly trained on a dataset such as the ImageNet dataset, which holds 1.2 million images over 1000 categories. This network then serves as the base network used in transfer learning. When the pre-trained network is used as a feature extractor, the last fully connected layer in the base CNN is removed, and two new adaption layers are added to the network. During training, all layers but the new two last layers remain fixed, and the only weights that are adjusted are the weights of these two fully connected layers [11].

Chapter 4

Ethical Use of Data

Chapter 5

Design

In this chapter, our focus will be on the intricacies of the design of object detection applications. Understanding the significance of the design process becomes paramount as it lays the groundwork for the software's development.

* 1. System Architecture

the system architecture for the proposed system is shown in Figure 4.

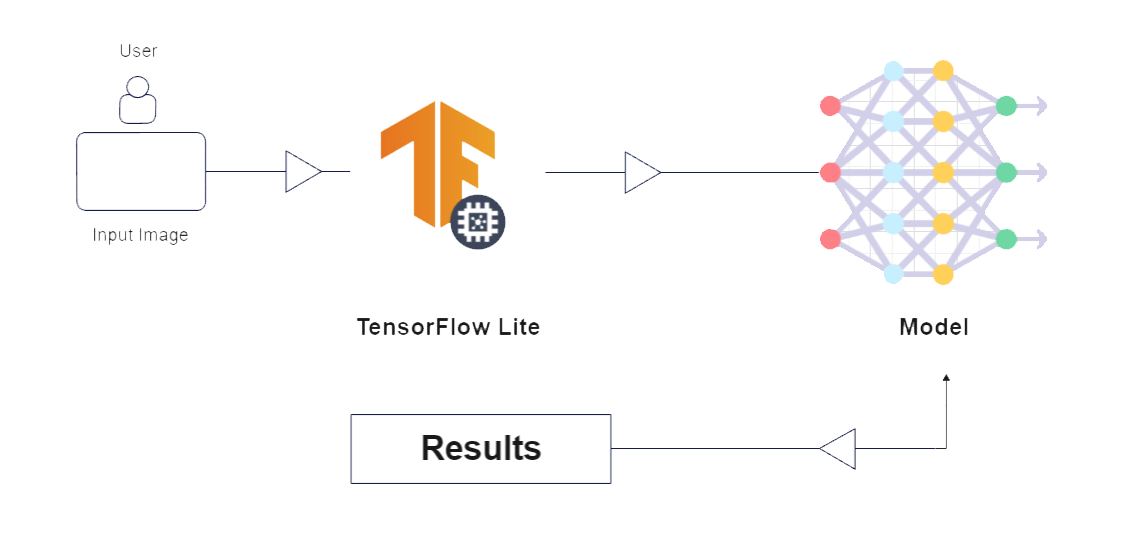


Figure 4:

The architecture of the object detection application using TensorFlow Lite involves three components working together to enable efficient and accurate object detection on mobile devices. The three main components are

**Application Interface**

It provides an intuitive user interface (UI) that enables users to interact with the application and initiate object detection tasks. The UI includes features such as selecting an image from the gallery or capturing real-time frames using the device's camera. The interface also displays the results of the object detection process, displaying a list of the detected objects.

**TensorFlow lite**

The TensorFlow Lite framework is the core of the object detection app. It is a lightweight version of the TensorFlow library specifically designed and optimized for limited resource environments such as mobile and embedded devices. TensorFlow Lite enables the efficient execution of Deep learning models, including object detection models, on resource-constrained devices like smartphones. The framework provides tools for model conversion, model deployment, and inference execution. It enables the app to load a pre-trained object detection model and perform real-time inference on the captured images or selected images from the user.

**Object Detection Model**

The object detection model is responsible for identifying and localizing objects in images. Typically, the model is pre-trained on large datasets and fine-tuned using techniques like transfer learning to achieve high accuracy and generalization. The model architecture, such as SSD (Single Shot MultiBox Detector) or YOLO (You Only Look Once), determines its ability to detect objects efficiently. The object detection model is loaded into TensorFlow Lite and executed using the provided inference engine to process input data and generate detection results.

The overall system architecture follows a client-server model, where the client is the mobile application running on the user's device, and the server component encompasses the TensorFlow Lite framework and the Model. The client sends image data captured or selected by the user to the server for object detection, and the server processes the input image using the loaded model to identify and locate objects of interest. The detected objects are then sent back to the client for visualization and display.

* 1. Object Detection Algorithms

**SSD (Single Shot Detector):**

SSD is a real-time object detection algorithm that stands out for its speed and accuracy. It takes a different approach compared to some other algorithms by predicting object bounding boxes and class probabilities at multiple scales within a single network. SSD is designed to efficiently handle objects of various sizes and aspect ratios in an image. This makes it well-suited for applications like video analysis, autonomous driving, and robotics, where real-time processing and the ability to detect objects of different sizes are essential.

* 1. User Interface Mockup

This is an Android Application consisting of several user interface screens. Each application screen has some different functionality from the other screen, and it differentiates the application interface from each other. The step-by-step user journey is covered via these screens where the overall UI mock-up design is as given below.

* **Welcome Screen**

The first screen is about Welcoming the users to our Object Detection Application. This is where the users will land and start detecting the objects after clicking on the “Start Detection” button.

Here I will apply my skills in application UI designs for a welcoming screen, then to proceed further with the complex tasks of the application.

* **Home Screen**

The second screen of my Object Detection application is “HOME”. It will be displaying two buttons for the different types of detections whatever user is looking for. Accordingly, by choosing “Upload Image” it will allow the users to upload an image from the mobile local storage, or a user can start the camera by which a live camera stream will be running and the application will start detecting the objects whatever comes in the preview by choosing “Mirror Detection.”

I am going to use the Google Machine Learning algorithm and Java & kotlin language for the objects detection. Regarding the “mirror detection”, I am still in the research phase about how to get it done and I kept it as optional until I finish the object detection from the uploaded image successfully

* **Photo Dialog**

The third screen of this object detection application is actually about the dialogue design in the Android application, in which it will give the freedom to the app user for choosing the image source. The user can either upload the image from photo gallery or take the photo by using the mobile camera.

In either case, the user will need to give permission for the file storage or for the camera access to the application. This is necessary because Android operating system needs user consent before using any of the default functionality of the Android mobile such as camera access, file storage etc.

* **Output Screen**

This is going to be the final screen of the application as all the results will be shown here. Once the objects has been detected by using the machine learning algorithms of the application, it will compile the results for the user. Therefore, the percentage of the accuracy will be displayed as well as multiple objects might be detected by using a single click.

For example, in the given sample layout design: Two fruits photo are uploaded by the user and it contains Banana & Apple, as the application will have the capability to detect multiple objects from the same photo, it will show the percentage of accuracy for each fruit.

Welcome Screen Home Screen

Figure 5: Figure 6:

Photo Dialog Output Result Screen

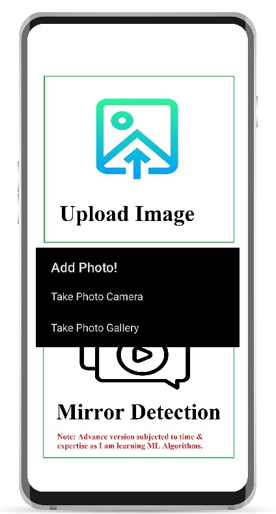
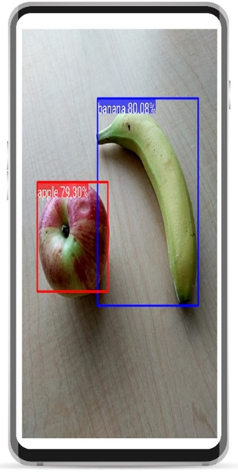
 

Figure 7 Figure 8:

* 1. Navigation Flow

in this section, we show the navigation flow for users within the system, Figure 9 shows the steps of the interaction of the user with the system, the user starts the navigation in the Home Screen which allows the user to choose between whether to use pre-trained or retrained model, then the user directed to Selection operation page which allows user to choose between detecting objects in upload image or real-time video using smartphone's camera, and then the detected objects is listed in list in results page

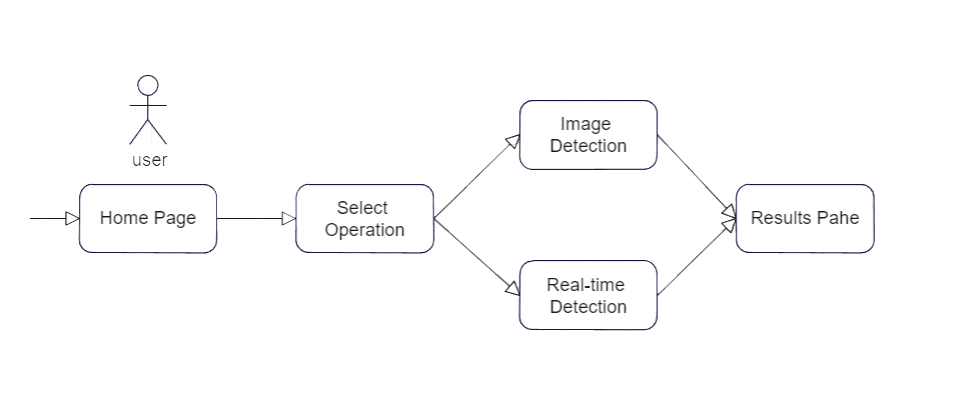


Figure 9 : Flow Diagram

Chapter 6

Implementation

This section focuses on the practical aspects of turning theoretical concepts and designs into tangible and functional products. It outlines the tools, technologies, methodologies, and steps taken to build and deploy the system.

* 1. General Object Detection

The initial phase of this project is to use one of the popular object detection frameworks to achieve trusted performance and allow users to identify common objects in images such as chairs, cups, cars, and people to achieve robust and accurate object detection

**SSD Model Trained on COCO Dataset:**

The Single Shot Detector (SSD) model trained on the Common Objects in Context (COCO) dataset forms one of the core components of object detection. COCO dataset contains over 330,000 images and more than 2.5 million object instances, which is a comprehensive dataset that encompasses a wide range of object classes, making it suitable for detecting a variety of common objects

* **Task Type Selection:**

Users can choose between whether to use General Object Detection “Mobile net v1” or Table Equipment Detection “Mobile net v2”. Selecting Object Detection invokes the call of the pre-trained model for identifying general objects such as cars, buses, vegetables, etc.

while choosing Table Equipment Detection triggers a custom-trained model specifically designed for Table Equipment objects such as laptops, phones, mice, etc

* 1. Table Equipment Detection

Chapter 7

Evaluation

this chapter outlines the performance evaluation of object detection models

* 1. Evaluation Methodology

the evaluation of systems has been done using different approaches such as a peer assessment to assess user satisfaction and gather feedback on the application’s usability and model performance using different metrics, Peer assessment involves users interacting with the application and providing feedback on their experience, interface design, and overall satisfaction.

///

the performance of the model was evaluated using various metrics such as inference time and mAP(mean average precision).

**inference time**

which refers to the amount of time that the machine learning model takes to produce the output, is a critical factor in real-time applications where predictions need to be generated quickly, it depends on various factors such as the complexity of the model architecture refers to how many layers the model has, the size of the input image, the computational resources available, and the efficiency of the deployment environment, Faster inference times are generally desirable as they enable more responsive and efficient use of object detection models in various applications, including autonomous vehicles, robotics, surveillance systems,

**mAP(mean average precision)**

Mean Average Precision (mAP) extends AP by the average of the Average Precision (AP) scores across multiple categories or classes. In object detection, there may be multiple objects or classes , and mAP provides an overall performance measure that considers the average precision across all classes , To calculate mAP, the precision and recall values are computed for every class. Then for every class, the average precision is calculated by interpolating the precision values at different recall levels. Finally, the mAP is obtained by averaging the average precision values across all classes,

mAP is valuable because it accounts for both the precision and recall of a model across various object classes, offering a comprehensive assessment of its object detection capabilities. Higher mAP values indicate better object detection performance, with values closer to 1.0 representing high accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Training Dataset | mAP | Inference time |
| SSD-Mobile Net v1 | COCO | 20.3 | 26 |
| SSD-Mobile Net V2 | COCO | 22.2 | 22 |
| SSD Mobile Net V2 | Table Equipment | 51.1 | 23 |

Table 1: The model performances

Chapter 8

Learning Points

Any software development project ends up with so many learning points and always helps the developer to gain more knowledge and sharpen their skills. Every project in the evolving field of software development has a distinct set of difficulties that encourage developers to broaden their horizons. The Object Detection application was not an easy task to do at the start and it took a lot of effort as well as time to complete. In order to create an Object Detection application, it was needed to take on challenging methods, keep updated with the most recent machine learning algorithms, and optimize the code for performance. It was a fun activity and a way of exploring the programming world but sometimes stressful due to coding bugs. These challenging encounters ultimately contribute to a developer's progress and knowledge in the dynamic field of programming.

I have learned the basics of Android application development and achieved hands-on experience in Kotlin programming language, building a solid base for my venture into software development. In the past few months, I have been immersed in a deep learning process, implementing advanced programming approaches to bring the functionalities of the object detection application to life. One of the toughest parts of this project was integrating the TensorFlow algorithm, an important part that required a lot of effort and time to resolve and integrate effortlessly. It offered countless opportunities for creative ideas as well as for developing critical software development skills like bug fixing and exception handling.

If I have to start this project again from scratch, I will use a different method to ensure a smoother transition. Before entering into the project, I would devote 2-4 weeks to completely learning and grasping the Kotlin programming language. This preventive step would enable me to manage possible bottleneck situations with ease and greater confidence as an application developer. During the development of the Object Detection program, I faced situations where I got stuck and spent sleepless nights battling with seemingly minor mistakes that unexpectedly impacted the functionality of the application. Mainly, these obstacles arose due to my position as a junior developer with limited proficiency in Kotlin. I am confident enough that with a strong foundation in Kotlin and comprehensive training, the same project could be completed much more smoothly and swiftly, eliminating the likelihood of such irritating blockages.

During the journey of development, I acquired some skills not only technical skills but also some soft skills such as Problem-solving skills and critical thinking, decision-making, project, and time management, as we progress towards the future of work that will be driven by artificial intelligence, experts predict that soft skills in software engineering will become even more critical if you take a step back and look at the software development from high-level view you'll realize that it really is about solving complex problems, as I said earlier that I faced situations where I got stuck and spent sleepless nights battling with seemingly minor mistakes that unexpectedly impacted the functionality of the application, because of these situations my Problem-solving skills and critical thinking skills has been improved, this project taught me the importance of adaptability and flexibility in research and development, also taught me how to Strom my brain to get the best decision in various situations

Chapter 9

Professional Issues

Chapter 10

Conclusion

in the conclusion chapter, we will discuss the aims of the project, summarize the main findings, and discuss potential directions for further work.

* 1. Recap of Aims

1.

* 1. Main Findings

The project's main findings outline the successful implementation of Object detection Android application using Kotlin programming language using one of the state-of-the-art detection algorithms such as SSD "Mobile net V1", this algorithm built a robust base for the objection system to identify a wide range of objects within images and retraining model such as Mobile net V2 on custom dataset such as Table equipment dataset this gives the model to identify specific range of objects.

it is critical to develop a friendly User-Interface for Android applications, and managing to achieve this aim represented a significant achievement, this UI provides a good experience to interact with system tools such as uploading images and seeing the results of detection.

* 1. Directions for Further Work

thanks to the succuss of the emplementation of the projetc, it achived substantial milestones, but there are several ehancements and explocations in the future, firstly imporve user-interface and user-experience by implementing intuitive controls, informative visual feedback during object detection, or integrating additional features, secondly fine-tuning existing alogthim or expolring more advance object detection algorthims to improve system metrics such as inference time and mAP, integrate application with cloud such as deploying application ob cloud servers such as AWS, GCP, Azure . Or leveraging distributed computing resources, This can enable more scalable and collaborative object detection scenarios, where multiple devices can contribute to a centralized detection system, continuously updata and expand our dataset with varity of examples to improve the performace of application and enable application to detect more objects .

Bibliography

1. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. url: http://www.deeplearningbook.org.
2. Google Inc. Tensorflow Lite. https://www.tensorflow.org/ mobile/tflite/. Accessed: 2018-02-07.
3. J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
4. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu: “SSD: Single Shot MultiBox Detector”, 2016; arXiv:1512.02325.
5. Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: Advances in neural information processing systems 28 (2015), pp. 91–99.
6. COCO: Common Objects in Context. Retrieved from https:// cocodataset.org/.
7. Tom M. Mitchell. Machine Learning. Ed. by Erick M. Munson. McGraw-Hill, Mar. 1997, p. xv. isbn: 0-07-042807-7.
8. lan Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. url: <http://www.deeplearningbook.org>.
9. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Las Vegas, NV, USA). IEEE Computer Society, Dec. 2016. isbn: 978-1-4673-8851-1.
10. R. Girshick, J. Donahue, J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. arXiv preprint arXiv:1311.2524, 2014
11. Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. “Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks”. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR ’14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1717– 1724. ISBN: 978-1-4799-5118-5. DOI: 10.1109/CVPR.2014.222.

Appendix A

UI screenshots

A screenshot of a phone

Description automatically generatedA screenshot of a website

Description automatically generatedA screenshot of a computer

Description automatically generatedA cell phone and a cup

Description automatically generatedA computer with a screen and a sign

Description automatically generated with medium confidenceA screenshot of a phone

Description automatically generated