Leveraging Robust CNN Architectures for Real-Time Object Recognition from Conveyor belt

by

Nowrin Tasnim Moon
19101109
Sabiha Afrin Siddiqua
19101050
Shahana Parvin
19101037
Sidratul Muntaha
22241162
K.M. Mehedi Hassan
22241188

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

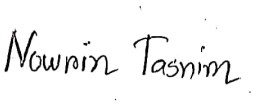
Department of Computer Science and Engineering
Brac University
January 2023

Declaration

It is hereby declared that

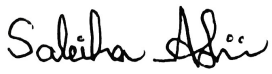
1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



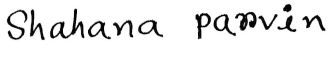
Nowrin Tasnim Moon

19101109



Sabiha Afrin Siddiqua

19101050

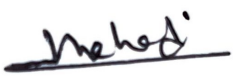


Shahana Parvin

19101037

Sidratul Muntaha

22241162



K.M. Mehedi Hassan

22241188

Approval

The thesis titled "Leveraging Robust CNN Architectures for Real-Time Object Recognition from Conveyor belt" submitted by

1. Nowrin Tasnim Moon (19101109)
2. Sabiha Afrin Siddiqua (19101050)
3. Shahana Parvin (19101037)
4. Sidratul Muntaha (22241162)
5. K.M. Mehedi Hassan (22241188)

Of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 19, 2023.

Examining Committee:

Supervisor:
(Member)

Moin Mostakim
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:

Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The findings presented in this thesis are the entire outcome of our extensive research and analysis. In order to accomplish the purpose of our thesis, we conducted research using a wide variety of websites, publications, and conference papers. Materials taken from external sources have been cited in the appropriate format.

Abstract

In the innovative era, the problem of recognizing undesirable objects and individuals on conveyor belts is addressed by various architectural or algorithmic approaches. Conveyor belts are those by which things go in a straight line for transportation, and it works as an object-carrying medium. Sometimes some unwanted objects go through the belt mistakenly, which can be very dangerous. Moreover, detection accuracy is much needed to avoid such deadly occurrences. Better accuracy can be achieved by performing detection in real-time. Furthermore, this upgraded system will assist individuals by lowering the danger of any accidents and provide a real-time example of an airport conveyor belt by detecting any unwanted moving objects with the help of a camera sensor and by applying different algorithms and methods of the neural network. Therefore, in this paper, we have implemented a few algorithms that comprise a customized Convolutional Neural Network, YOLOv5, YOLOv7, and Vision Transformer as well as some Transfer learning methods over a few pre-trained models such as VGG16, ResNet50, and MobileNetv2 to produce a better strategy on our customized dataset to boost the accuracy of recognition.

Keywords: Neural Network; Object Detection; Conveyor Belt; VGG16; ResNet; MobileNet; YOLOv5; YOLOv7; Convolutional Neural Network, CNN, Vision Transformer

Dedication

We devote this report to our respected faculties, whose direction, instruction, and assistance made it possible for us to complete it. This would not have been possible without their help and inspiration, and we are really grateful to them.

Acknowledgement

First and foremost, all praises go to the Almighty Allah, to whom we owe the completion of our thesis without substantial disruption.

After that, we would like to express our gratitude to our Supervisor, Moin Mostakim, and our Co-Supervisor, Rafeed Rahman, for their advice and unwavering support throughout this entire process. Additionally, we want to thank other faculty members, especially Tanvir Ahmed and Shoaib Ahmed Dipu for their help and support.

At the end, we would want to express our respect to our parents for their continuous support and prayers, which have enabled us to reach this point in our pursuit of graduating.

Table of Contents

Declaration i

Approval ii

Ethics Statement iv

Abstract v

Dedication vi

Acknowledgment vii

Table of Contents viii

Table of Contents viii

List of Figures x

List of Tables 1

1 Introduction 2
- 1.1 Introduction . . . 2
- 1.2 Problem Statement . . . 3
- 1.3 Research Objectives . . . 3
- 1.4 Research Contributions . . . 4
- 1.5 Research Outline . . . 4

2 Literature Review 6
- 2.1 Related Works . . . 6
- 2.2 Background Study . . . 13
  - 2.2.1 Convolutional Neural Network (CNN) . . . 13
  - 2.2.2 Faster R-CNN . . . 14
  - 2.2.3 ResNet50 . . . 14
  - 2.2.4 VGG16 . . . 16
  - 2.2.5 MobileNetv2 . . . 17
  - 2.2.6 YOLOv5 . . . 17
  - 2.2.7 YOLOv7 . . . 18
  - 2.2.8 Vision Transformer (ViT) . . . 19
  - 2.2.9 DCGAN . . . 19

3 Methodology **21**
3.1 Data Collection . . . 21
3.1.1 Train-Test Split . . . 22
3.2 Pre-Processing . . . 23
3.2.1 Image Pre-Processing . . . 23
3.2.2 Data Pre-Processing . . . 23

4 Model Implementation **25**
4.1 Convolutional Neural Network (CNN) . . . 25
4.2 Pre-Trained CNN models (VGG16, ResNet50, MobileNetv2) . . . 26
4.3 Vision Transformer (ViT) . . . 26
4.4 YOLOv5 . . . 28
4.5 YOLOv7 . . . 28

5 Result and Analysis **30**
5.1 Metrics . . . 30
5.2 Result Analysis . . . 30
5.2.1 YOLOv5 . . . 31
5.2.2 VGG16 . . . 33
5.2.3 ResNet50 . . . 34
5.2.4 MobileNetv2 . . . 35
5.2.5 YOLOv7 . . . 36
5.2.6 DCGAN . . . 38
5.2.7 CNN . . . 39
5.3 Limitation and Future Work . . . 42

6 Conclusion **43**

Bibliography **47**

List of Figures

2.1 Architecture of the Typical Convolutional Neural Network [24] 13
2.2 Architecture of Faster R-CNN [10] 15
2.3 Architecture of the VGG16 model [43] 16
2.4 Architecture of the YOLOv5 model [21] 18

3.1 Flowchart of Collecting Data 21
3.2 Subsets of the Dataset Partition 23

4.1 Mechanism of Convolutional Neural Network (CNN) [35] 25
4.2 Architechture of ViT [44] 27
4.3 Network Architecture of YOLOv7 [46] 29

5.1 Curve of Precision-Confidence and Precision-Recall of YOLOv5 model 31
5.2 Learning of Training Batch 32
5.3 Prediction of YOLOv5 model 32
5.4 Curve of Training-Validation Accuracy and Loss of VGG16 model . . 33
5.5 Prediction of VGG16 model 33
5.6 Curve of Training-Validation Accuracy and Loss of ResNet50 model . 34
5.7 Prediction of ResNet50 model 34
5.8 Curve of Training-Validation Accuracy and Loss of MobileNetv2 model 35
5.9 Prediction of MobileNetv2 model 35
5.10 Curve of Precision-Confidence and Precision-Recall of YOLOv7 model 36
5.11 Confusion Matrix of YOLOv7 model 36
5.12 Prediction of YOLOv7 model- part1 37
5.13 Prediction of YOLOv7 model- part2 37
5.14 Original Images 38
5.15 Generated Images 38
5.16 Curve of Training-Validation Accuracy of CNN model 39
5.17 Curve of Training-Validation Loss of CNN model 39
5.18 Prediction of CNN model- part1 40
5.19 Prediction of CNN model- part2 40
5.20 Real-time prediction through a video from an internet source and our surroundings 41

List of Tables

3.1 Image and Instance variances of different classes 22

5.1 Result Analysis of all the Models 41

Chapter 1

Introduction

1.1 Introduction

Any warehouse, airport, metro station, shopping mall, or heavy material industry cannot pass a day without a conveyor belt system. It is a vital feature of product transportation in all industries, including ports that we use on a daily basis. There are different types of Conveyor belts in different sizes. However, they all have the same goal which is to transport stuff. The main objective of a conveyor belt is to move something from one place to another, usually within a short distance, at a rapid frequency, and quickly. They are placed to regulate the movement of items or things across an industry. As a result, the workforce's work runtime and complexity will be reduced. For that, the package and baggage sorting system needs to be advanced to meet its demand. Moreover, it is essential to minimize hazards in the environment. Making workers' jobs easier by avoiding the handling and moving of possibly large and oversized objects can reduce concerns in the workplace.

To make the sorting system more convenient, they are already using the conveyor belt system, but sometimes many inconveniences might occur. Such as, small children get on the belt and get lost, someone can also drop any unwanted objects on the belt, any object which is not aligned to the center. Even the boarding time of other passengers might get delayed while foreign objects can create structural damage, sometimes packages get damaged in the process of sorting them leading to material loss, etc. In order to fix the issues, we need to detect the issue in real-time, where we will try to recognize if there are any unwanted objects or not. In the modern world, human detection has become more important for video surveillance applications, autonomous driving vehicles, and person recognition as these are becoming increasingly vital in computer vision. Detecting a human is very difficult because every person has a unique appearance and a wide range of poses. Even when the background is busy, there should be a reliable mechanism for feature extraction.

In this paper, we have used different types of algorithms which are CNN, YOLO (versions 5,7), Vision Transformer (ViT), and some transfer learning methods over pre-trained models (VGG16, ResNet50, MobileNetv2) to train our model and make the neural network efficient for improving the system by training-testing on our custom image dataset and finding out the result from all the images if it can recognize objects and the human body which is basically a toddler. Furthermore, it will give

a comparison of all the implemented models on our customized dataset for a better understanding of the efficiency of a model.

1.2 Problem Statement

Airports in every country, shopping malls, and big industries or warehouses use conveyor belts to ease their workload and send products from one place to another. In that case, any unwanted or harmful things may pass through the conveyor belt, which causes a defect in the belt or create problems for the surroundings such as amputations, blunt force trauma, crush injuries, degloving injuries [31]. However, accidents are proven deadly in many cases, as a human baby may go onto the belt while playing and be squashed through the rollers. According to a report, a two-year-old boy was carried on the belt, further, he was detected by the baggage handling system but suffered a severe arm injury [25]. In other news, we got to know that a five-month-old baby was crushed and died through the system [28]. Many researchers have already worked on improving conveyor belt detection accuracy, therefore, this study focuses on making a system that is more accurate and efficient. Here, we will try to improve the system to avoid such incidents by recognizing different objects, and their motion while our main focus will be to detect the human body to protect the toddlers who climb up on the conveyor belt unknowingly or accidentally. We will try to detect the human body and moving objects with the help of algorithms by recognizing different objects and their characteristics.

1.3 Research Objectives

A 2-year-old toddler was harmed after evading an airport desk and landing on a luggage conveyor belt [25]. Even if the conveyor belt is not running, a youngster getting on it will trigger it, since conveyor belts are often operated by pressure. There was also a cat that was only seconds away from being crushed in a trash divider belt until a worker noticed [30]. These incidents are very common and can bring anyone's life at risk. Thus, the study proposes an improved and more efficient system that can recognize humans, pets, and other objects in a conveyor belt in order to avoid potential health injury and material loss. Our primary focus will be :

- Creating an efficient belt system where no unfortunate events can lead to death and delay packed flights.
- Developing a framework that is more functional and efficient than existing systems
- Detecting objects in real-time with real-world datasets
- Having training data for the model to use and evaluating its accuracy
- Validation data will be used to see whether the data we want is accurate or not

- Using CNN, Vision Transformer, VGG16, ResNet50, MobileNet, YOLOv5, and YOLOv7 to detect the accuracy of the data to evaluate the performance
- If the accuracy is fulfilled, we will be ready to use the network and proceed to predict the test data

1.4 Research Contributions

In this study, we have implemented a few CNN architectures to apply on our custom dataset to provide detection services in real-time on a conveyor belt using a high-quality learning model. The following is a list of the paper's primary contributions:

1. The main contribution of our research is our own customized dataset which includes real-world images of 11 different classes.
2. We have taken real-world data for now but some objects might not be cleared on the belt, it can look like noise on camera. That's why we have tried to use DCGAN which can create fake images by learning from our given dataset as input. We will try to include these fake images in our dataset to enlarge our dataset.
3. For YOLOv7, it was the latest version released from YOLO versions until this study was conducted. We will try to upgrade the result by training on our custom dataset to predict more effectively.
4. We have trained ViT as it can handle image data with high resolution and fine-grained tasks for our research purpose as it is still less used in other research purposes.
5. For our customized CNN model, since we have a custom-built dataset for our research field, we could understand the result more effectively by custom CNN as it doesn't have any pre-trained weights. We customized the layers including flattened and dense layers to get better prediction results.

1.5 Research Outline

Here, this portion provides a comprehensive description of every chapter of our thesis paper. After completing this chapter's introduction, problem statement, and contributions section, we now list the other chapter of our thesis paper below:

- Chapter 2 conducts the Literature Review where we list relevant works that we have gathered from numerous articles that are related to our thesis work, as well as the Background Study which displays the models or architectures we utilized and read for our research.
- Chapter 3 covers the methodology where we portray the information of the dataset and the selection of the model.
- Chapter 4 represents the implementation of the models which we have used in our thesis.

- Chapter 5 provides the research results and analysis.
- Chapter 6 mentions the limitations of our research work and also future work.
- Chapter 7 concludes our thesis with the following strategies.

Chapter 2

Literature Review

2.1 Related Works

In this very paper [19], Ansari et al. mentioned using a CNN-based object detector and how it can sense human presence to detect social distance. When a problematic occurrence happens, the system sends out an alert so that security can take necessary measures in light of this information. The initial stage of any detection includes the localization of an item, followed by the categorization of the localized object in the next stage. Therefore, to enhance accuracy, this method utilizes the image edge length characteristics, as well as a classifier based on the k-means clustering technique, to minimize the overall complexity. Later on, another proposed system was developed, which was using the background extraction approach to extract moving objects. Previously, object recognition frameworks were used to locate objects inside an image. An image is segmented into a certain number of units or sectors using this method. However, it has a significant amount of operational cost and can create a lot of false alarms. With the use of the convolution process, CNN can obtain important characteristics, as its extensive data representation ability was vital in detecting objects. The system takes an image as input, allocates relevance to various items inside the image through different parameters, and recognizes each object. Two CNN-based sequential models are presented in this study to detect the presence of an individual within a picture. This work proposes two consecutive CNN-based models for detecting the presence of a human within a picture. The camera orientation is changed using actual measurement while keeping our system in mind. Substantial trials were carried out using CNN-based object detectors. Experiments show that CNN-based object detection algorithms exceed others in terms of accuracy.

This paper [1] focuses on Range Imaging Technology, which enhances the capabilities of existing 3D imaging devices like laser scanners and stereo vision. For each pixel of a two-dimensional sensor array, range cameras capture the size and position of objects in real time by combining amplitude, intensity, and precise spectrum (i.e. distance) information. A high-resolution image is required to capture detailed specifics of an item, yet current 3D range cameras only have limited X-Y resolution. In particular, the sensor's 3D range information is used to extract meaningful information about things traveling on a conveyor belt. There are two possibilities: The dimensions of boxes traveling on a conveyor belt are measured first, followed by

the recognition and classification of other objects. The eigenvalues of the Laplace operator have also been used to recognize objects in both 2D and 3D pictures. In this work, eigenvalues of the Laplacian are used for object recognition. For the experiment part, 6 objects were taken and positioned into 10 orientations, and measurements were taken and fourteen eigenvalues were kept from the 600 computed. To separate each object from the rest, six binary support vector machines were taken and a Gaussian kernel was used for the test. From the results of the first and second tests, they got an average success rate of 91.2% which is a good percentage to start off. Eigenvalues are averaged for 3,5 and 8 frames and the result for the 8th frame is 98.3% which is very much appreciated. All the tests had significant results above 90%. So, time-of-flight (ToF) sensors for identifying objects on a conveyor belt can be used, but future studies will strengthen the topic by covering more loopholes.

In this research, [8] Wendi et al. mainly focused on detecting an object localized in a street area based on deep neural networks. They implemented a transfer learning approach that can convert a network of the convolutional to street object identification in the accuracy of detecting objects and minimize the time it takes to train a convolutional network. In this paper, they showed two ways to convert the learning to deep learning networks. The first way is to tweak the convolutional network's structure to generate a feature that can be used to detect street objects. Furthermore, the CNN model employed in this article is the Faster R-CNN, which is made up of an Inception-ResNet-v2 network. There are primarily two components in the Faster R-CNN model. The first one is a fully deep convolutional network that propagates domains, and the second one is a Faster R-CNN detector that employs the regions proposed. Although the inception-ResNet-V2 network slows detection time, it enhances the precision of the Faster R-CNN model. The second way of fine-tuning the network parameter is by training it with a custom self-created dataset that contains information about street objects. They discovered that using a Faster R-CNN on a self-created dataset can boost the accuracy of detecting objects. They demonstrated that fine-tuning a Faster R-CNN model produces greater outcomes than the original network. The accuracy rate of this fine-tuned Faster R-CNN model is close to 100% on the other hand normal Faster R-CNN is less than this.

Dinama et al. [11] were primarily interested in developing a security system capable of creating and converting tracing movement and human position on the footage as additional information using a human detection and tracking algorithm. To track observed humans, channel and spatial correlation filters are utilized in tracking algorithms. They used mainly two datasets for footage. One is the PETS2009 dataset, while the other is the PSVR2019 dataset. For deep learning, CNN used mainly two kinds of network architectures, ResNet101 and RPN. This study used a modified version of the PASCAL VOC [42] dataset to train the suggested algorithm. They utilized OpenCV's library [33] to detect and track video footage. However, just the CSRT algorithm was employed in this research.

Liu et al. introduce faster R-CNN which is more efficient and works with Region Proposal Network [7]. R-CNN to Fast R-CNN and finally Faster R-CNN, it has gone through all the experimental phases to reach the level of object detection. At first, R-CNN collects thousands of data from top to bottom in an image using the

selective search algorithm. The features extracted from each fraction are used as input for Support Vector Machine (SVM), to classify the details. R-CNN flaws include numerous phases of training, which are costly, time-consuming, and take up disk space. Secondly, Fast R-CNN extracts thousands of regions from top to bottom in the frame using the selective search algorithm. The region of interest (ROI) pooling layer ensures the pool size is dependent on the input size, which means that the output is always of the same size. The said layer is added to the last layer of the convolutional layer. Fast R-CNN sends the extracted features and regions into the training network, and the trained data is sent to the layer so that it can ensure optimum performance. Faster R-CNN incorporates embeddings and extraction of features in a network. It uses the intended area effectively, using the convolution network and sharing it with the object detection network. Caffe's deep learning tool was utilized to train the network in this research. The ROI pooling layer extracts the feature and inserts it into a complete connection for classification. By researching the Faster R-CNN of several networks, the training set and the corresponding classification findings of mean average precision are achieved. Various neural networks have been introduced in this research to achieve classification by applying the Faster R-CNN.

Here [9], the authors mainly mentioned the novel approach for object detection and tracking by applying the Convolutional Neural Network. Moving object detection is executed by applying TensorFlow API which is an open-source platform that examines the image and returns the object's location while the detected object tracking is handled by the CNN algorithm. They basically discovered two challenges when object detection and tracking happened, where to track the object detection in the occlusion conditions is one of the huge challenges and the other challenge is because of variations in the scenes for robust object detection. Moreover, due to these challenges, the authors mainly focused on eliminating the challenges by being able to detect the object in different occlusion and illumination. CNN is used in image classification and recognition to improve substantial performance, as it is a machine-learning algorithm that uses spatial information in a picture to learn complicated properties automatically. This proposed algorithm has sensitivity, specificity, and accuracy parameters that are used in the quantitative analysis. Their study concluded that the proposed approach showed an accuracy of 90.88% on self-generated image sequences and reached a sensitivity of 92.14%, a specificity of 91.24%

This paper [13] shows that human life can be protected at nighttime in dangerous places like a sea beach with the help of infrared images and CNN, deep learning. To make this successful they have used CCTV cameras, infrared images from the cameras, conventional computer vision to detect humans, etc. Originally the image is in the infrared format, then the background subtraction mask is used for capturing the temporal image information. Then it's passed through the neural network as the network is trained on the binary classification task using the cross entropy method, and the result of the network is presented in a 2D array of confidence scores for each pixel. The results show better performance compared to the baseline methods.

In the paper [16] a novel human detection method which is mainly an improvisation of the framework- Mask R-CNN is proposed to solve the issues of detecting humans

in complicated scenarios by integrating the forwarding research findings of object recognition using deep learning while combining the ResNet and feature pyramid network (FPN) to figure out the aspects of the picture and after that, to proofread the pixels the authors have used RoIAlign and fine-grained Slic. From the research, it was proven that the framework of improved Mask R-CNN works better than the algorithm R-CNN, as the value of mAP and AR is higher in the upgraded framework. RoI and region proposal network (RPN) was introduced to make R-CNN stronger with a faster approach, but it made it difficult to achieve real-time observation. It is helpful for segmentation, target detection, and the point for detection. In the improved Mask R-CNN system, certain improvisations are done where a fine-grained slice is added to predict the impact of the network. From bottom to top, the mask R-CNN FPN creates four series of detail maps using a CNN. The top-down and parallel association techniques have been used to merge four sets of various characteristic maps to overcome the issue that the characteristics of various portions of CNN vary substantially. Fine-grained slice creates five-angled directions to create superpixel segmentation. So, to conclude, the proposed method increases the ability to detect human bodies and the new fine-grain layer improves local characteristics.

In the latest projects [17] with RGB-Depth cameras, wearable sensors were combined to create a 3D skeletal model in an identification system. However, these devices are limited to different lighting and range. In order to get around this limitation, the suggested scheme uses real-time video patterns to analyze individual interactions in both inside and outside settings. The suggested method uses frame-wise dislocation to develop human identification and motion tracking, and authentication is based on a skeleton model with neural network architecture. The adaptive feature can translate optical data from the surrounding environment into a core issue. Absolute position, joint displacement, orientation, and integrated information are used to divide 3-dimensional skeleton-based human models into four categories. Human activity recognition using the structural model currently has several restrictions. For enhanced quality using development methods, most systems include data production. The technology used here is designed to be a reliable 3D human skeletal system. The results of the execution are presented in this part using one dataset which was gathered from a variety of sources, including YouTube, Google videos, largely movies, and a modest public database. It shows the accuracy of human recognition findings throughout several sessions. Real-time data is critical in many applications and is required for efficient human motion detection, monitoring, and action recognition systems. The method is constructed in this research utilizing a deep neural network framework to achieve high-precision identification of movement patterns in both indoor and outdoor environments. The experimental findings revealed that all methods are highly dependent on various backgrounds, sensor fusion, and lighting changes.

This paper [18] shows how to improve noise included with the object and the coarse-grained detection process with a Deep Convolutional Neural Network. It is hard to detect real-time objects due to the large field of view and high resolution and other things like complex background, and image noise with these it gets more complex and becomes computing heavy. To solve the problem, the proposed method is Deep CNN. Here, in the coarse-grained section, filtering the photo then detecting move-

ment and then applying mathematical morphology, and finally extracting the region where the movement can be detected. After that, in the fine-grained section, that region extraction goes through Deep CNN and finds the result with less computing power. The experiments are performed on an NVIDIA GeForce GTX 1080 Ti where different scenarios are given and a SimitMoving Dataset is performed on it. Here, they compared the gaussian mixture modeling (GMM) data clustering algorithm with the K-nearest Neighbor (KNN) supervised machine learning algorithm for cross-grained detection.

In this article, [22] the author investigates how to improve object detection performance using the original and upgraded version of YOLOv5. They used the acquired data for two very different the basic YOLOv5 model and the upgraded YOLOv5 model to find the primary indications. Fast R-CNN, despite its success, has poor learning and execution performance because of its distinct candidate area generating module. The computing speed is more than doubled by these processes. As a result, this replacement is more effective and quick. They took note of the Convo layer, which is the primary layer of the existing main YOLOv5, and then updated it. In order to train the model and improve object search accuracy, class selection and data collecting are crucial. Moreover, in this study, both YOLOv5 models ran tests using an identical set of data. In this paper, they employ CIoU to check the comparison of the loss functions of both YOLOv5 models in order to address this issue. The 3360 training, validation, and testing photos were used to train both YOLOv5 models. The original YOLOv5 model took the longest to run. Here, they observe the display of the object detection comparison results of the two models. After conducting training and assessment, they discovered that the YOLOv5 model was the most effective. The best-performing YOLOv5 model gave the prediction weight. This outcome demonstrates the efficacy of our strategy in accurately forecasting experiments carried out in various situations. The function loss between the models differs slightly. A chart showing the function loss values of the two models shows that YOLOv5 is effective yet slow.

The authors of the article [20] presented a neuroimaging strategy using Convolutional Neural Network (CNN) with transfer learning approaches for improving the detection of a cerebral hemorrhage on CT scan data. They obtained the dataset from the physio next repository, which consists of 82 patients' head CT scan images, for this purpose. They took 2500 images and eliminated the rest since they were scans of bones, and they only needed the brain CT scan images to train the model. They also separated the images into two groups. The images that do not have hemorrhage are in one group, and the ones that do have hemorrhage are in the other. This dataset's training, validation, and testing ratios are 7:2:1.For better results and accuracy, they performed data preprocessing through image resizing and data augmentation. To solve the problem of sparse data, they used four geometric transformations: RandomFlip, RandomZoom, RandomCrop, and RandomRotation in data augmentation. After adjusting the dataset, the authors tested it through seven CNN models, six of which were pre-trained and one of which was their suggested 11-layered CNN model, to compare accuracy and precision. Traditional CNN has three levels, but they expanded it to eleven by adding layers to their proposed model. Out of these eleven, four convolutional layers were employed in each, and

a max pooling layer was added to each. The last three layers, which include the flattening layer, dense layer, and activation function, are fully connected layers. In the pre-trained model, they notably used ResNet50, VGG16, InceptionV3 [37], EfficientNetB6 [32], InceptionResNetV2 [36], and DenseNet121 . They got the highest accuracy rate in EfficientNetB6 which is approximately 96% which is higher than any other model in their research.

In the study [23], the author improved the accuracy of the pest detection system by incorporating machine learning algorithms like Exception [45], InceptionV3 [37], and support vector machines (SVM) [41]. 1500 images make up their 1500-image raw dataset. Before preprocessing the datasets, they eliminated unnecessary objects. They used normalization and scaling during the data preprocessing phase. In all, there were 5 different types of images. Then, for the SVM model, they divided it into 'Training'and 'Testing', and for the other two models, they added 'validation'. SVM can accurately recognize a small section of a large object, thus they first adjusted the kernel to linear and polynomial functions to compare their efficiency. As a result of their finding that polynomials performed better, they focused on the polynomial function for subsequent processing. When the learning rate is lowered for the Exception model, performance is slower and the validation loss is reduced, but the inverse is applicable when the learning rate goes up. The InceptionV3 was executed in the same method, and the outcome was the same. Following that, both models experienced hyperparameter tuning, with the same validation accuracy for both models. Both models, though, experienced validation loss. The final SVM result shows that this model has an accuracy rate of about 73% in identifying pests, compared to an accuracy rate of about 50% for the other two models.

In this research paper, it is mentioned that the Transportation Security Organization (TSA) supervises the security of traveling open within the United States [12]. One of the foremost obvious capacities of the TSA is the security screening of travelers and their individual assets for potential dangers. Hand-searching each passenger's bag would be both time-consuming and meddling, so X-ray scanners are conveyed. At the safety stations, the TSA utilizes an armada of X-ray scanners, such as the Rapiscan 620DV, so Transportation Security Officers (TSOs) can assess carry-on belongings. The TSA is curious about evaluating the possibility of sending calculations that can consequently highlight objects intrigued by TSOs. Most profound learning strategies require to need a sizable training collection of marked cases for the attainment of great execution. For question discovery, this implies information comprising both pictures and bounding boxes with lesson names. A large data collection effort was necessary to train and evaluate object detection models. The system's final prototype has shown a lot of potentials and could eventually be used by the TSA. Here in order to obtain the data, they follow the Rapiscan 620DV X-ray screening framework which is outlined for flying and high-security applications. The scanner produces two sees through the near-orthogonal introduction of the fan-shaped bars from X-rays sources. In this segment, we see how to utilize pre-processed RGB coloration ordinarily appeared to TSOs. They are also scanning and labeling the captures. Various examinations were conducted using a single bag to house many individual risks, with a small amount of harmless material being exchanged between each scan. In order to enable our models to develop invariance

to precise placement within the tunnel, each item was additionally scanned in a variety of positions. Living creature workers manually tagged each image, giving each name a combination of the risk group category and the designs of the bounding box. Every container was said to be as small as was physically possible in each view while yet holding the entire question. This description contained the handle if there existed any, which applied to items like blunts and sharps. Since they construct, examine, and classify each training sample themselves, CNN is particularly crucial in the context of autonomous danger identification at TSA checkpoints. They can drastically reduce labor expenses and man-hours by using pre-trained connections. Moreover, a two-step procedure is used by faster R-CNN to create forecasts. They trained models with pre-trained parameters from the MSCOCO Image Identification Competition as well as did preliminary processing on every image by subtracting the channel means from the MSCOCO dataset [38] for every pixel. All Faster R-CNN models in the CNN framework utilized one batch size, while the SSD framework used other batches. They have looked into the application of cutting-edge methods for the difficult task of danger identification in luggage at airport safety stations. Initially, they gathered a large quantity of data by manually putting together numerous luggage and containers to represent typical traffic. These hid a huge range of dangers. They acquired X-ray scans by scanning each sack, and then annotated both search perspectives. Furthermore, using the data that had been gathered, they trained a number of contemporary object detection algorithms, investigating various settings and tailoring them for the work in mind. The outcomes of testing the model using verification and experimental data that was held out have been reported. Lastly, they also mentioned that there are Faster R-CNN versions that can function on equipment that is readily obtainable in the marketplace while still effectively identifying threats.

Sangwan et al. provided a method for detecting threat objects utilizing CNN-based techniques, such as YOLOv2 and Faster region-based CNN (FRCNN) [14]. The database of this report was developed in two phases. In the first phase, the creator makes use of X-ray image modeling, which leads to an increase in the object count. In the second phase, they made use of an image augmentation approach to include variability in the dataset that was produced following image modeling. In x-ray image modeling, initial images were obtained from the GRIMA X-ray detector. After that, they increased the database to a limited set of X-ray images. The first dataset for X-ray image augmentation was obtained from the GDXray database. As the amount of luggage expanded in the preceding phase, they found new threat objects in the augmentation steps. In this step, the image transformation method was utilized, which comprises operations such as rotation, flipping, zooming, etc. Afterward, they annotated each image. After that, they implemented YOLOv2 and FRCNN for detection. In FRCNN, RPN is used to build a predefined set of regions for image classification in order to generate a region. In addition, object boundaries and object class scores are predicted for each point. In YOLOv2, the images were resized to 13 by 13 to boost the chance of object recognition. They then consecutively performed image classification, localization, thresholding, and non-max suppression. Ultimately, they discovered that the FRCNN gives higher threat detection outcomes with a 98.4% accuracy rate.

2.2 Background Study

2.2.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a Deep Learning technique that can take in an image as input, assign priority to different parts of the image using learned weights and biases, and then use those weights and biases to identify and differentiate between different objects and parts of the image [6]. In this paper, we review various key CNN issues and outline the impact that adjusting the parameters can have on its overall efficiency. The convolution layer is the most crucial part of it because it uses up so much of the network's processing time. For problems related to machine learning, CNN fares quite effectively. In particular, results from image-related activities, including the greatest photo interpretation information gathering (ImageNet), computer vision, and natural language processing (NLP), proved absolutely remarkable. The effectiveness of a network can also be affected by the number of phases it contains. Furthermore, training and testing the network takes more time as the amount of parts increases. Among the many uses for Neural Networks, image identification is where Convolutional Neural Networks have found the most widespread use. Its built-in convolutional layer reduces images' high dimensionality without any loss of quality. They find applications in fields as diverse as image and video recognition, recommendation engines, image classification and segmentation, medical image analysis, NLP, BCI, linguistics, finance, and more. Convolutional neural networks (CNNs) are adapted multilayer perceptrons. CNN uses a grid topology approach to process the data. A two-dimensional grid is used for holding images. Next, the convolution method is used to isolate distinct aspects of the input. CNN can be applied on any 2D and 3D array of data. The convolutional neural network (CNN) is constructed from three kinds of layers: fully connected layers, pooling layers, and convolutional-like non-linearity layers. CNN outperforms feed-forward networks due to its adoption of features like parameter sharing and dimensionality reduction. Parameter sharing in CNN reduces the total amount of variables, which in turn reduces the total quantity of calculations required. To answer visual inquiries or to caption pictures, CNN networks take in pictures as input and provide answers in natural speech.

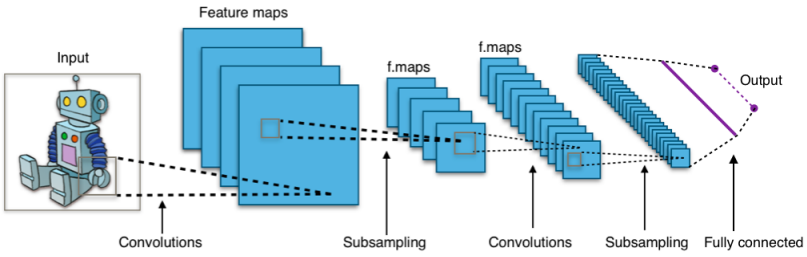


Figure 2.1: Architecture of the Typical Convolutional Neural Network [24]

Pooling is used to reduce the difficulty of deeper layers by down-sampling the data. In terms of image processing, it's analogous to decreasing the resolution. Filter counts are unaffected by pooling. Maximum-pooling methods are extremely common and efficient. The image is segmented into small rectangular parts, and the greatest value within each region is returned. In addition, non-uniform filters and stages can

benefit from pooling to maximize their efficiency. Again, the non-linearity might be used to modify or halt the output. By adding this layer, we may control the volume and/or quality of the output. Then, One major drawback of the fully connected layer is the high computational complexity involved in determining its many parameters from training data. We, therefore, make an effort to lessen the number of vertices and paths. In order to restore functionality after dropping a node or connection, the dropout method might be employed. For example, LeNet [26] and AlexNet [27] built a deep and broad network while keeping the computing complexity the same.

2.2.2 Faster R-CNN

Two modules comprise the single-staged model for object detection known as Faster R-CNN. That means a single-stage model called faster R-CNN is trained from the beginning to the conclusion. The first module is a deep fully convolutional network that proposes regions, while the second module is a Fast R-CNN detector [4]. Because it has a regional proposal network, this newly updated model is better than Fast R-CNN. Fully convolutional RPN generates proposals of various scales and aspect ratios. It instructs the other module where to focus using the terminology of neural network 'attention' process. Faster R-CNN researchers created a training technique for both modules. Additionally, due to RPN being Faster, R-CNN may be trained from beginning to end to perform a particular detection task. Similar to how Fast R-CNN analyses images, RPN also employs a convolutional layer. As these two modules share the same convolutional layer, this new model combines these two networks. It is possible to establish a single network for Faster R-CNN by simultaneously training RPN and Fast R-CNN. The working sequence of Faster R-CNN is :

- Regional proposals are generated by the RPN.
- Using the ROI Pooling layer, a fixed-length feature vector is derived from every image region proposal.
- Then the other module is then applied to classify the extracted feature vectors.
- The provided data includes the bounding boxes and class scores for the locations The RPN and Fast R-CNN were trained in three ways in this study. The first is alternate training, followed by approximate joint training, and finally non-approximate joint training.

The researchers did experiments on PASCAL VOC and MS COCO. Given that every sample from a single image is connected, faster R-CNN has demonstrated that the network requires a long time to attain convergence. In conclusion, it is claimed that the RPN is cost-free and that it increases the accuracy of object identification and the quality of region proposals objects.

2.2.3 ResNet50

ResNet50 is developed from ResNet. The primary issue with deep neural networks is that they cannot generalize well to new data, making them inefficient models. This

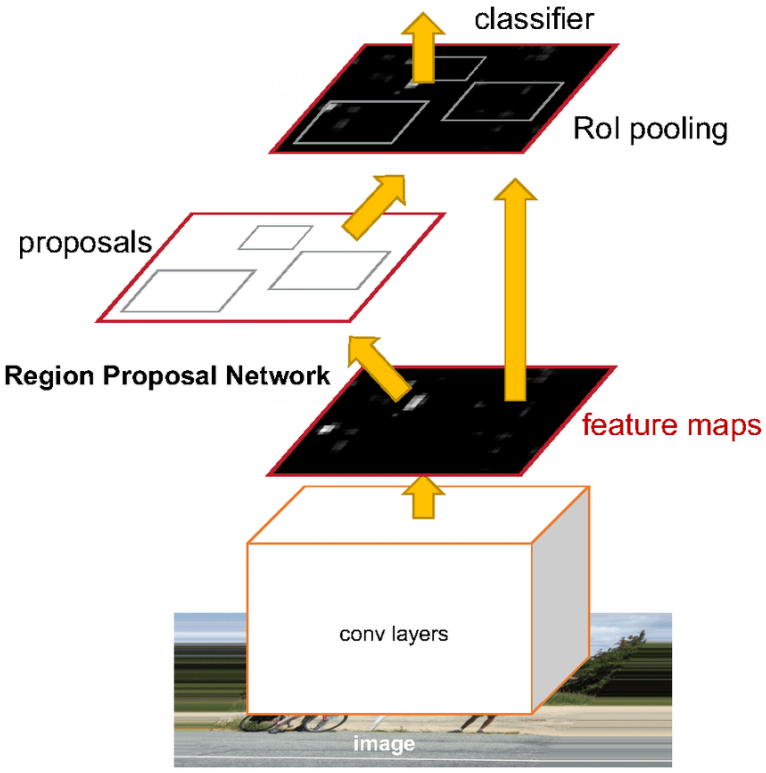


Figure 2.2: Architecture of Faster R-CNN [10]

degrading issue makes it difficult to optimize some systems. Therefore, Deep Residual Learning is discussed in this study for the purpose of mitigating the degradation issue. ResNet's primary innovation is residual learning. Furthermore, it has been demonstrated that these Residual networks are simple to train and more accurate than others. Two convolutional layer blocks comprise residual learning. Each layer contains the same amount of filters. The final layer's output is added to the first.

In this study, the authors used the VGGnet, a 33-filter, 2-layer plain network model, as their baseline model [5]. These follow 2 factors. The output feature map size and the number of layers in the first are the same and to preserve the time complexity per layer, the second involves reducing the size of the feature map and increasing the number of filters. A 100-way fully linked layer and a global average pooling layer concluded this network. Additionally, the shortcut connections are added to the simple network to convert it into the residual network.

The photos were scaled into 256*480 at the implementation phase. After that, data augmentation was used. Before activation and after each convolution, normalization was carried out. Additionally, the network was trained by a mini-batch of 256 using stochastic gradient descent. They trained the CNN residual network of 18 and 34 layers as well as the plain network for evaluation. When the depth is increased from 18 to 34 layers, the error for plain networks rises, however, it reduces in ResNet with an increase in depth and the error was lower than for plain networks. The primary structural difference in ResNet50 is the use of a stack of three layers rather than two. Therefore, a 3-layer bottleneck block was used in place of each layer block in ResNet34. However, ResNet50's accuracy is greater than ResNet-34's.

2.2.4 VGG16

The Convolutional Neural Network model is used as the basis for the generation of the VGG16 model [2]. Despite the fact that several CNN models have been built, the accuracy rate of such models is not quite as great as this one. Because of this, the VGG16 model was generated from the CNN model. This is because the CNN model has been shown to be superior to other CNN models, as evidenced by the fact that the author of this model demonstrated a more accurate ConvNet architecture, which resulted in improved accuracy in ILSVRC classification and localization and was applicable to all image recognition datasets. In addition to that, they discovered exceptional performance. The developer of this model carried out an evaluation using 3×3 convolution filters, which revealed a notable improvement in comparison to the prior-art configurations. They increased the depth to between 16 and 19 layers, which resulted in roughly 1138 trainable parameters.

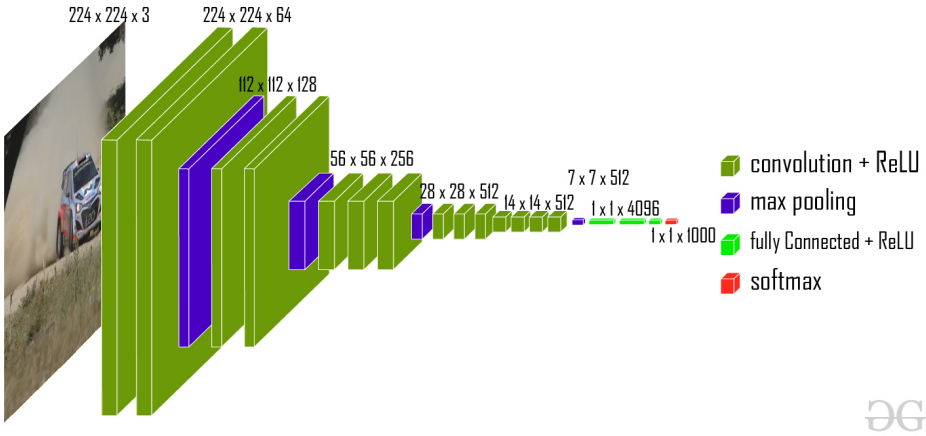


Figure 2.3: Architecture of the VGG16 model [43]

The '16' in VGG-16 models refers to the 16 layers with weights. There are 21 layers total, including 13 convolutional layers, 5 Max Pooling layers, and 3 Dense layers. The input of the convolutional layer conv1 is a fixed 224*224 RGB picture. Applying this layer to an image results in a 33-pixels-wide filter. However, it also makes use of 11 Conv filters in one of the configurations. 1 pixel is the convolutional stride. The padding for 33 convolutional layers is 1 pixel. Five max-pooling layers perform spatial pooling. Every 2 by 2-pixel window has a max-pooling operation using stride 2. Throughout the entire structure, the Maxpooling layer and the convolutional layer are constantly organized. The first three fully-connected (FC) layers follow a series of convolutional layers. The soft-max layer is the last one.
Despite the fact that VGG-16 training takes a long time and the weights are substantial, this method is simple to execute. The performance of this model is superior to that of the ILSVRc-2012 and 2013 models. The VGG16 architecture accomplishes the best result (7.0% test error), exceeding a single GoogleNet by 0.9%. In this model, it was demonstrated that representation depth is beneficial for classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be accomplished using a conventional ConvNet architecture with significantly

higher depth. Additionally, it was shown that this performance can be achieved with a significantly higher number of hidden layers. In addition to that, this model is suitable for a variety of datasets.

2.2.5 MobileNetv2

It is a CNN architecture that functions effectively on mobile devices. It is designed to reduce the number of steps and memory while maintaining the same precision. MobileNetv2 consists primarily of two parts [10]. The inverted residual block is the first, followed by the Bottleneck residual block. These two are regarded as the innovative layer module of this architecture. It is simple to implement and permits any model to operate in numerous performance modes.

The MobileNetv2 system consists of 53 convolution layers and 1 AvgPool layer in total. In this architecture's convolution layers, two types of layers are applied. 1×1 convolution and 3×3 depthwise convolution.There are two different forms of block: stride 1 and stride 2. Each block contains three Convolution layers. The first layer is expansion, the second is depthwise convolution, and the third is the convolution layer applied with Relu6. The activation function is performed by Relu6. Its primary function is to enhance performance. Bottleneck Residual block is primarily a residual connection, but if stride 2 is applied for depthwise convolution, it loses the residual connection.

This study demonstrates that linear bottlenecks work better without Relu6 activation and that when a shortcut is placed between bottlenecks, the results are improved. The author of this report focuses mostly on three MobileNetv2 experiments. MobileNetv2 exceeded MobileNetv1 and ShuffleNet in ImageNet classification with comparable model size and computational cost. Again, it exceeds ShuffleNet and NasNet when a width multiplier is applied. The second experiment involves the detection of objects using the COCO dataset. MobileNetv2 exceeds state-of-the-art real-time detectors on the COCO dataset in terms of the accuracy and complexity of the model in this experiment. The final experiment is Semantics Segmentation. It is used as a feature extractor with Deeplabv3 in this instance.

2.2.6 YOLOv5

Using the PyTorch framework, Glenn Jocher released YOLOv5 after the YOLOv4. Moreover, YOLOv5 is the most recent and compact version of prior YOLO algorithms, and it uses the PyTorch framework rather than the Darknet framework [22] For smooth transitions from training to deployment, YOLOv5 now fully integrates TensorFlow, Keras, TFLite, and TF. js model export. Like other single-stage object detectors, YOLOv5 relies on three essential parts. Model Backbone's main purpose is to take important features from an image and then use those features to predict future outcomes.

CSPDarknet serves as the backbone, PANet serves as the neck, and YOLO Layer serves as the head, making a total of three parts. The issue of non-overlapping

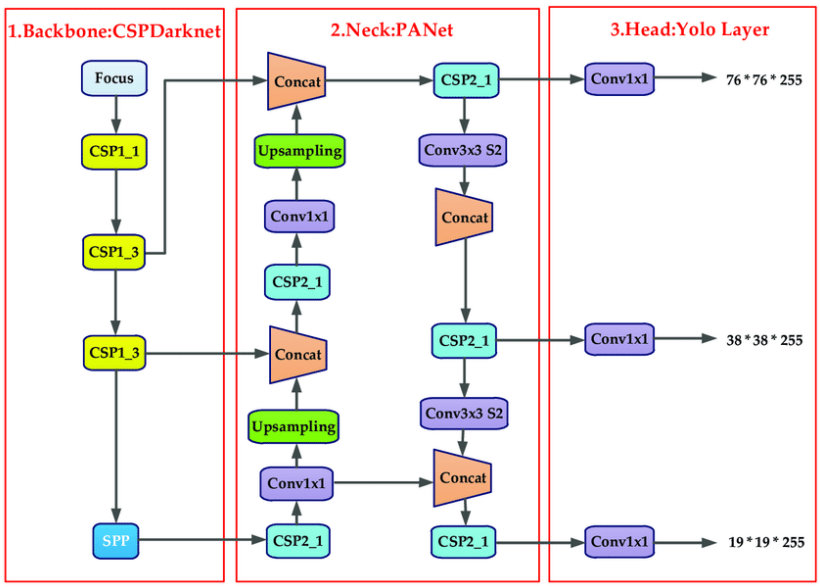


Figure 2.4: Architecture of the YOLOv5 model [21]

bounding boxes is effectively resolved by YOLOv5's usage of GIoU as the loss function. Screening many target frames with the weighted NMS operation at the target detection prediction result processing step allows for the best target frame to be selected. Our model can be initialized using weights from a pre-trained COCO model by providing the model's name in the 'weights'option. Pre-trained on the COCO dataset, YOLOv5 is a library of object recognition structures and algorithms Based on YOLOv5, research has developed SF(Small-Fast)-YOLOv5, an improved YOLOv5 method for small object recognition that not only considerably reduces the number of parameters and calculated amount of the model but also performs better than the original version in the small. Therefore, We have used YOLOv5s for our research purpose. YOLOv5 takes inputs from URLs, Filenames, PIL, OpenCV [33], NumPy [39], and PyTorch [40], and delivers detections in torch, pandas, and JSON formats. YOLOv5 trained on MS COCO dataset [38]. It was discovered that YOLOv5 performs better in terms of accuracy than YOLOv4 and YOLOv3. YOLOv3 had a superior recognition speed than YOLOv4 and YOLOv5, while YOLOv4 and YOLOv5 had the same recognition speed.

2.2.7 YOLOv7

The YOLO family's newest and most sophisticated object detection is known as YOLOv7. When it comes to real-time object detection, YOLOv7 is currently leading the pack. The official YOLOv7 boasts amazing rate and precision in comparison with the preceding iterations [34]. All real-time object detectors are outperformed by YOLOv7 in both speed and accuracy. Using Microsoft's COCO dataset, YOLOv7 weights are trained without the use of pre-trained weights.

YOLOv7 brings about a number of architectural advancements that boost effectiveness and precision. For the same reason that Scaled YOLOv4 did not, YOLOv7 does not employ pre-trained backbones from ImageNet. Instead, the complete COCO

dataset is used to train the models. The similarity is to be expected as Scaled YOLOv4 as both were written by the same people. The architecture is derived from YOLOv4, Scaled YOLOv4, and YOLO-R. In order to create a new and improved YOLOv7, additional tests were conducted using these models as a base. According to the YOLOv7 paper, the best model achieved 56.8% Average Precision, the highest score of any object detector currently in use. A number of parameters are down 40% and computation time is down 50% in YOLOv7 compared to the benchmark models. YOLOv7-tiny is 127 FPS quicker and 10.7% more accurate on AP than YOLOv5-N. When compared to the equivalent YOLOv5-L with 99 FPS, version YOLOv7-X gets 114 FPS inference speed, but YOLOv7 also achieves a higher accuracy (higher AP by 3.9%).

2.2.8 Vision Transformer (ViT)

Vision transformer was developed so that computer neural networks (CNN) would no longer be necessary for the task of image classification and this transformer is particularly effective in performing this particular task [15]. In addition, in comparison to CNN, Vision Transformer achieves superior results when it is pre-trained on substantial amounts of data and then applied to a number of benchmarks that assess image recognition performance on images of varying sizes.

The Vision Transformer's developer emulated the original Transformer. The transformer encoder alternates self-attention and MLP blocks. The transformer encoder output represents the image. The Transformer encoder output is connected to an MLP classification head for pre-training and fine-tuning. MLPs have one hidden layer during pre-training and one during fine-tuning.

Pretrained on a large dataset, the ViT was fine-tuned downstream. Researchers tested new methods on ResNet, Vision Transformer, and hybrid. The testing revealed that ViT calculates more quickly. The position embeddings also allow the model to represent patch distance. They also tested the transformer model to Big Transfer and Noisy Student image classification standards. The study configured ViT based on BERT and updated ResNet by replacing Batch Normalization with Group Normalization and using standardized convolutions to increase transfer learning. Self-supervised pre-training boosted accuracy by 2% compared to training from scratch, according to an early study on self-supervised training ViT.

2.2.9 DCGAN

Deep Convolutional Generative Adversarial Networks (DCGAN) was presented as a strong class for unsupervised learning, and they can make any model stable for training. In the Generative Adversarial Network (GAN) generator and discriminator, convolutional and convolutional transpose layers are implemented. GAN is made up of a generator and a discriminator [3]. The generator generates artificial or fake images that resemble training images. The discriminator examines an image and the output to determine whether the image is authentic or fake. During

training, the generator makes increasingly realistic fake images that mislead the discriminator into believing they are real, while the discriminator seeks to get more effective at detecting and categorizing whether an image is authentic or fake. In the DCGAN architecture, we found that it -

- Replace any pooling layers with stride convolutions (discriminator) and fractional-stride convolutions in the DCGAN architecture. We found that this was the most effective way to train the network (generator).
- Implementing batchnorm in both the generator and the discriminator at the same time.
- Eliminating hidden layers that are fully connected in order to create deeper architectures
- The Rectified Linear Unit (ReLU) activation is being used for each of the layers, except for the output, which is using tanh.
- Activating all layers of the discriminator with leaky ReLU using that activation.

Chapter 3

Methodology

3.1 Data Collection

It is recommended to use a complete public dataset for research initiatives. We may build a reference point using those datasets' better samples and reported model performances. However, because of the limited resources and lack of complete datasets in this field, we had to develop a unique dataset for our research. We require examples that were derived from the relevant domains for real-world issues. We have developed a dataset with a few attributes that are connected to conveyor belt object detection for real-world problems.

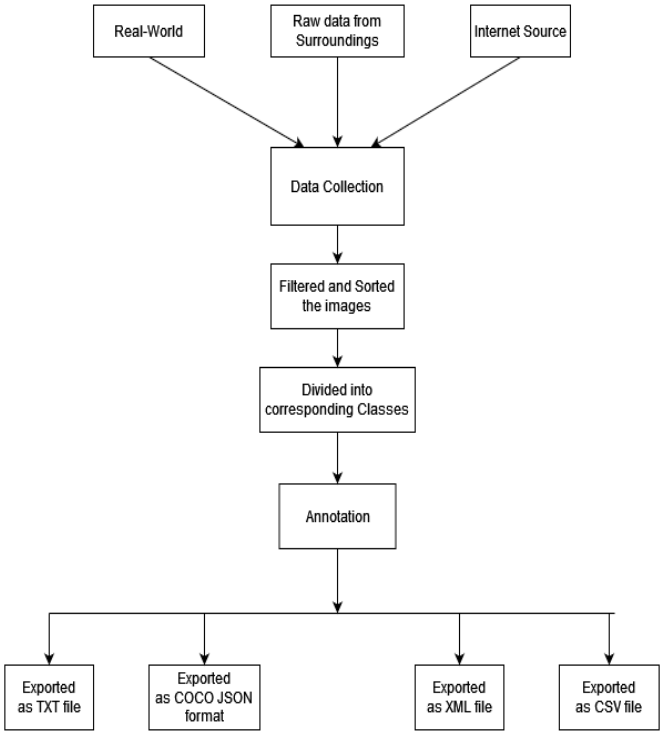


Figure 3.1: Flowchart of Collecting Data

Since we mainly focused on object classification and detection on conveyor belts, we have collected a number of images of different classes which can possibly get on the belt, such as starting from illegal objects, electronic items, defected and non-

Class Name	Image Number	Instance Number
baby	769	894
bottle	519	1097
box	506	1101
defect	394	464
electronics	508	761
illegal	525	1015
luggage	503	737
medicine	501	1851
metal	502	1046
pet	508	709
sport	506	1022
All	5741	10697

Table 3.1: Image and Instance variances of different classes

defected luggage, metal items, etc. to even toddlers who can playfully get on the belt. We have collected our data from three different sides, real-world data which is airport environment focused, raw data from our surroundings which is similar objects that can be found on the airport conveyor belts, and data from the internet. After collecting all the images of different classes, we annotated and labeled all images using makesense.ai and created the dataset to be used for the models to train.

3.1.1 Train-Test Split

Datasets are commonly divided into two categories: training and test data. A part of a dataset used to instruct a machine learning algorithm is referred to as a training set. Contrarily, a test set is the portion of the dataset used to evaluate a machine learning model. The dataset is often divided into a 70:30 ratio, which implies that you may either use 70% of the data for training the model and the remaining 30% for testing and validating. On the other hand, the validation dataset, in contrast to the test dataset, is not used to develop the algorithms but rather to examine and evaluate the performance of final models in a neutral way. The data is given in the following structure:

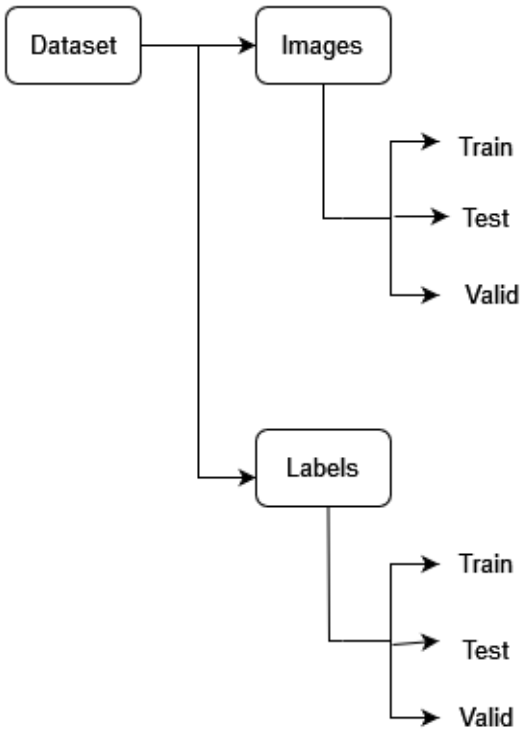


Figure 3.2: Subsets of the Dataset Partition

3.2 Pre-Processing

3.2.1 Image Pre-Processing

Our images have been saved in jpg, jpeg, and png formats to be used for further image analysis. All images have been scaled down as per the model's requirement. However, we have taken image sizes 340*340 for the YOLO model, 128*128 for Convolutional Neural Network (CNN) model, and 32*32 for Vision Transformer (ViT) model. We have developed our models in a cloud environment (google collaboratory), hence we have mounted Google Drive to access the dataset. After that, we divided each image by 255 for normalization after converting it to a NumPy array of the float32 data type. The images were then stored in an array to further continue with the algorithms.

3.2.2 Data Pre-Processing

Real-world data often lacks particular attribute values and thus data preprocessing is much needed as it helps to filter and organize raw data so that machine learning models can use it right away. Our dataset was compiled from several sources, which were then properly integrated to form a dataset. In order to load the raw image dataset into our various models, we have annotated and labeled them based on their classes. Every structured dataset consists of several columns that combine numerical and categorical information, and this is why label encoding was needed. The null and missing values were handled by machine-learning algorithms automatically. On the other hand, each data from the collection was resized followed by different augmented

steps such as flip, blur, shear, zoom, etc to make the dataset more effective. The dataset is formatted differently for each model. For instance, for Faster R-CNN, the dataset was exported with COCO JSON format. On the other hand, for YOLOv5 and YOLOv7, the dataset was formatted in TXT format. For further pre-trained models such as VGG16, ResNet50, and MobileNet, only images in the jpg, png, and jpeg formats were used.

Chapter 4

Model Implementation

4.1 Convolutional Neural Network (CNN)

CNN model requires both feature extraction and classification steps to function. We had to import all the necessary libraries and packages such as TensorFlow, Keras, NumPy, pandas, Matplotlib, etc on a cloud environment like Google Colab. After a thorough data pre-processing, we advanced to implement layers like Conv2D for feature extraction and MaxPooling2D for downsampling the image. We have taken four Conv2D layers starting from the filter of 32x32 followed by four Maxpooling layers each. To further enhance the model's performance, both during training and validation, we have implemented a BatchNormalization layer which can help to regularize the model that can reduce overfitting. This was followed by the addition of the Flatten() layer, which reduces the multi-dimensional output of the convolutional layers into a one-dimensional array so that it can be given to the model's dense() layer, also known as a fully connected layer. Since we have 11 separate multi-classes, the dense() layer requires a parameter value of 11, hence the softmax function is applied as the activation function in this case. The learning process was then set up on the model using the compile function before the model was trained. This includes a metrics parameter for the accuracy as well as the specifiers 'adam' for the optimizer and 'categorical_crossentropy' for the loss function.

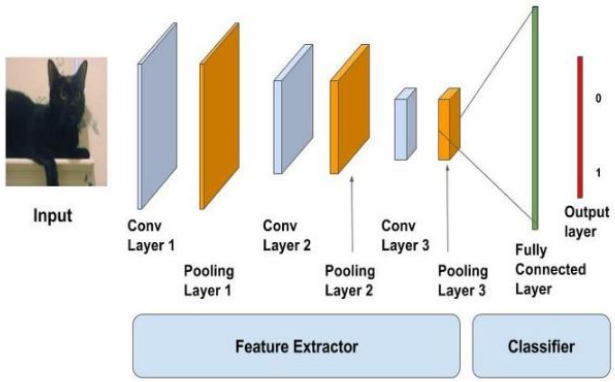


Figure 4.1: Mechanism of Convolutional Neural Network (CNN) [35]

The model was then trained using a fit function with 50 epochs across our custom

training and test datasets. It took a while to conclude the training as the Conv2D layers were taken from scratch and the dataset was relatively customized with a fair size.

4.2 Pre-Trained CNN models (VGG16, ResNet50, MobileNetv2)

In order to identify the objects, we had to look for patterns in the pixels of the images. This occurs in the network's convolutional layers, which are able to recognize those patterns as factors.

We have implemented VGG16 where the input matrix size was 150 by 150 after importing all the necessary libraries and packages. We have split the dataset into a train and test of 80:20. We have loaded the VGG model where 'ImageNet' was the pre-trained weight. As it is transfer learning, we used include_top = False to remove the classification layers which were already trained over their pre-trained dataset to reduce the training process. Instead of using a large number of parameters, our VGG16 was implemented with convolution layers of a 3x3 filter with a stride 1 and a max pool layer of a 2x2 filter with a stride 2. The dense layer parameter had the number of folders as a number of classes we had in our dataset. We have used ImageDataGenerator to generate the train and test dataset to train the model and label the classes according to the folder of the images. We trained the model with 20 epochs and got an accuracy of 87.7% and a validation accuracy of 70.5%. We have applied graphs to better understand the training and validation accuracy and loss. We have used the predict() function to predict classes of a sample image corresponding to their labels.

Similarly, we have also implemented ResNet50 and MobileNetv2 by using their pre-trained weights on our custom dataset. For ResNet50, we have taken a Conv2D layer with a filter of 1024 and a Dropout layer of 20% to avoid overfeeding. We have used a GlobalAveragePooling2D and a dense layer with 11 as parameters for our total class number and 'softmax' as an activation function for the output layer. After training the model for 20 epochs, we have a training accuracy of 90.5% meanwhile a validation accuracy of 73%. As for MobileNetv2, after implementing the model with a different pre-trained weight in a similar manner, we have gotten a training accuracy of 92% and a validation accuracy of 86%, which is the highest among all the pre-trained models we have trained over our custom dataset. As these are pre-trained models, we got better accuracy than our custom-built CNN model which was built from scratch by adding convolution layers.

4.3 Vision Transformer (ViT)

For setting up our environment, we used Google Colaboratory. Next, we implemented the required libraries and software in a virtual environment built on a Python version 3.9 foundation. In this case, Python is being utilized as the programming language, which means the application is written in Python and then run in a Google

Colaboratory Notebook. The libraries that we have used to execute the program: NumPy, TensorFlow, Keras, TensorFlow_Addons, Torch, Torchvision, OS, Matplotlib.pyplot, cv2, tqdm.

The vision transformer uses both attention and transformer mechanisms. This is how the architecture functions:

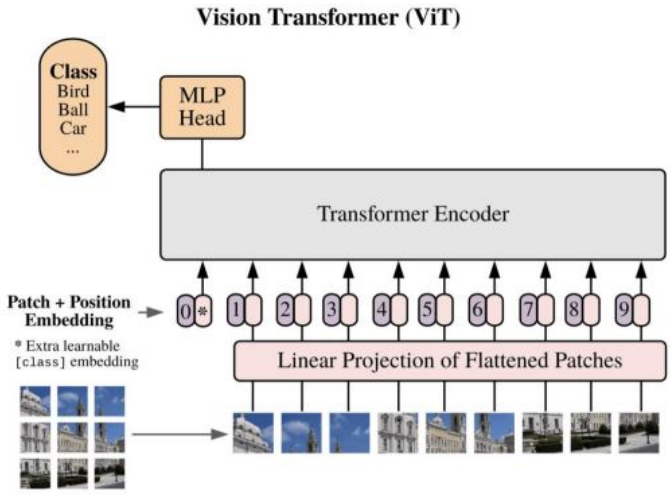


Figure 4.2: Architechture of ViT [44]

The next step is to obtain rich vector embeddings, which constitute the ultimate core output of the transformer model including the vision transformer. The only difference is how we preprocess the image before we input it into the transformer. After that, we process these patches into linear projection layers to get out initial patch embeddings. And we sum the patch embeddings to position embedding. Here the image patches work as tokens. In this transformer, there are several attention layers to reduce the computation of the image and we utilize these patches. Here, One of the formulas is

Attention (Q, K, V) = softmax $(QK^T)/(\sqrt{d_k})\ V$

The Tesla T4 GPU for Tensorflow was utilized because we are working with an image dataset. For our transformer model, we used a 32*32*3 size image as input. Then, we split it into 3352/1652 train test values.

The ViT model consists of multiple Transformer blocks, which utilize the layers. The patch sequence was subjected to a self-attention approach called the MultiHeadAttention layer. The [batch size, num patches, projection dim] tensor formed by the Transformer blocks is processed using a classifier head with softmax to provide the class probabilities output. Without this layer in ViT we have some other layers to implement the whole model.

At first, for Hyperparameters, here we have our learning rate weight decay that is used for adamw optimizer input size of images as 72*72 it uses the super-resolution technique to have kernel filter that makes the pixel to upsample image and patch

size of 6 where our image sample pf 72*72 images are divided into 6 samples of the original image. Then we have the transformer layers of 8.

Secondly, for Data augmentation, we are using this technique to prevent data overfitting. Here we have a normalization layer for normalizing the pixel value of our dataset. We also resize the image from 32*32 to upto 72*72 images. Then comes the multilayer perceptron where we override the dense layer to add a skip connection to know how many hidden nodes are used to transform the dimension so that they are compatible with each other with the output of the skip connection. After that, we mentioned that Patch creation helps to overwrite Keras layer objects for implementing the patches layer. In our case, we used 72*72 images and then transformed them into 6*6 patches. We transformed our patches so that they are compatible to be added to each other.

4.4 YOLOv5

We have used the default python environment setup for our YOLOv5. The python version we used is 3.10.9. The libraries and software we utilized are

OpenCV-Python (4.1.1), PyYAML (5.3.1), torch (1.11.0+cu113) and so on.

We found that YOLOv5's model trained more quickly on our image dataset, therefore we adopted it. This YOLOv5 model trains quickly and has a lower penalty for detection than YOLOv5nano since it is less sophisticated than other models. Additionally, it performs better in terms of real-time detection. There are a total of 213 layers and, 7225885 parameters in this version of YOLOv5s.

Due to the fact that we are employing our own dataset, we had to generate our own data and define our classes that were written in a YAML file. The YOLOv5s model was then trained for 300 epochs, and we used the best result weight file for real-time detection and other predictions.

4.5 YOLOv7

Like YOLOv5, here we also relied on the standard python installation for our YOLOv7. And Python 3.10.9 is the version that we utilized.

Next, we used some libraries and software named:
torch(1.11.0+cu113), OpenCV-Python (4.1.1), PyYAML (5.3.1) etc.

As we are working with an image dataset and using YOLOv7 so we used our local environment. Our local environment configuration is the same as the YOLOV5 configuration.

For our detection system, we used YOLOv7's default weights model because it is more straightforward to train on the images we are working with. Additionally, this YOLOv7 model performs better for real-time detection which means it is more effective for real-time detection and is less complex than other YOLO variants. This version of YOLOv7 has 415 layers, 37255890 parameters, 37255890 gradients, and 105.3 GFLOPS.

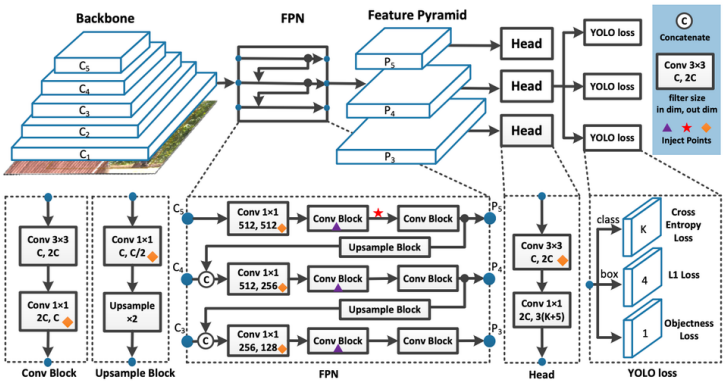


Figure 4.3: Network Architecture of YOLOv7 [46]

Since we are creating our own dataset, we need to generate our own data. The dataset.yaml file contains the information about the classes we desired to apply. A weight file with the best training results from 80 epochs of YOLOv7 model training was selected for the detection phase.

Chapter 5

Result and Analysis

5.1 Metrics

In this section, we mentioned the formulas for a few metrics:

For Precision, the formula is TP / (TP + FP),

For Accuracy, the formula is : TP + TN / (TP + FN + TN + FP),

For Recall, the formula is TP / (TP + FN),

For F1 score, the formula is : 2 * precision * recall / precision * recall

$$\Rightarrow TP/(TP) + ((FP + FN))$$

Here, True Positive is denoted as TP,
False Positive is denoted as FP,
True Negative is denoted as TN and
False Negative is denoted as FN.

5.2 Result Analysis

In the beginning, we simply used the architectures for our research such as VGG16, ResNet50, and MobileNetv2 to get accuracy. We have a total of 11 classes and compare the training and validation accuracy of the models after 20 epochs of training with 4598 train images and 1143 test images.

5.2.1 YOLOv5

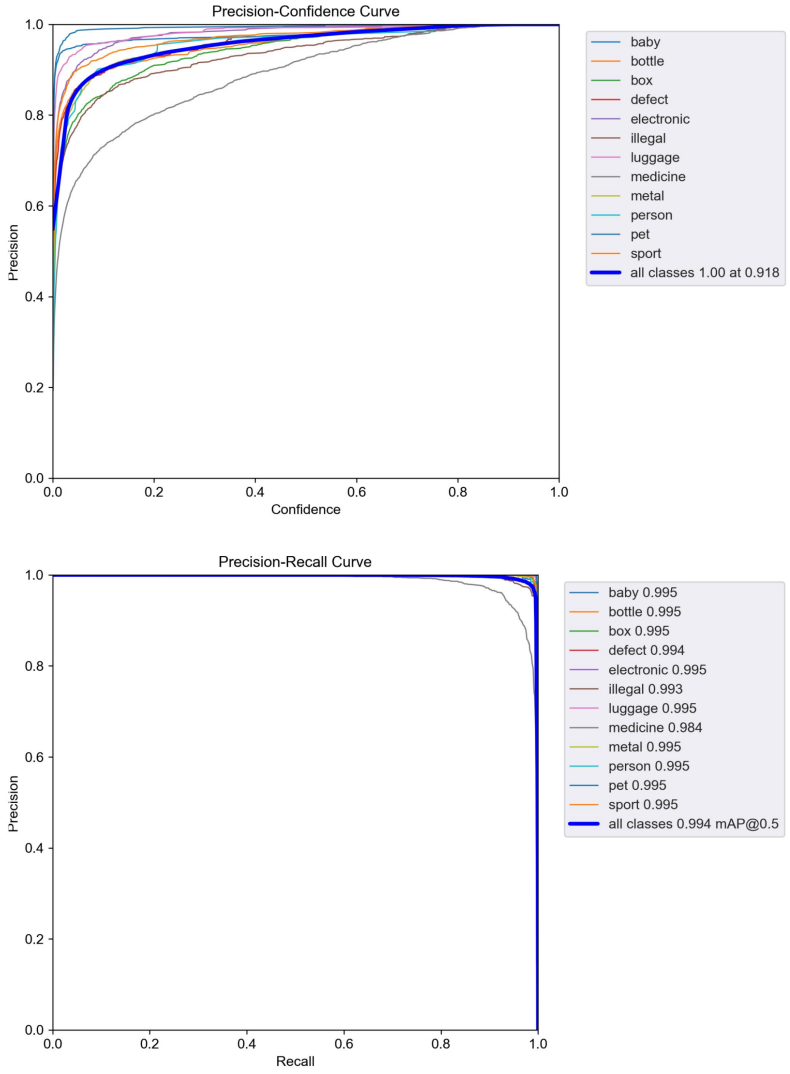


Figure 5.1: Curve of Precision-Confidence and Precision-Recall of YOLOv5 model

After 300 epochs of training, we can see in FIG 5.1 that the precision confidence curve is close to perfect for most of the classes. For all classes, we are getting precision confidence of an average of 91.8%

We can see in FIG 5.1 that the precision-recall curve is almost perfect for all classes. And the model can correctly identify objects in an image at different levels of confidence with an average of 99.4% accuracy.

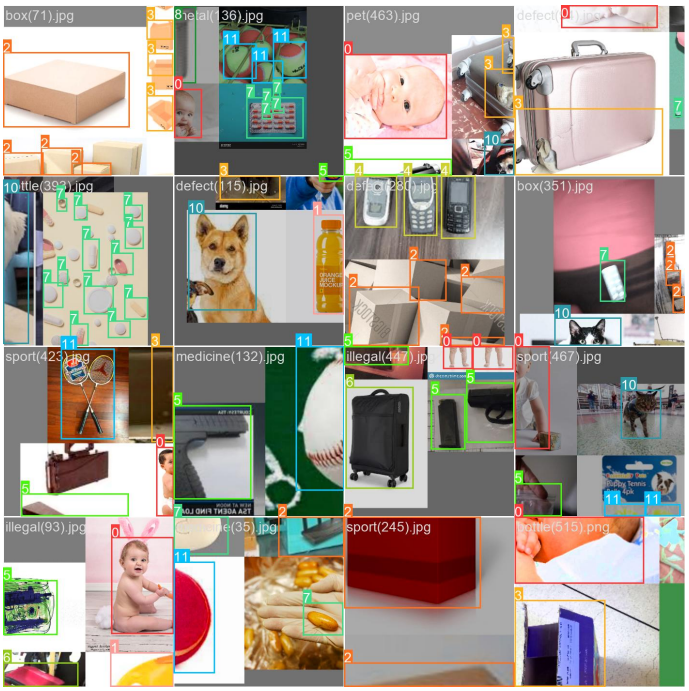


Figure 5.2: Learning of Training Batch

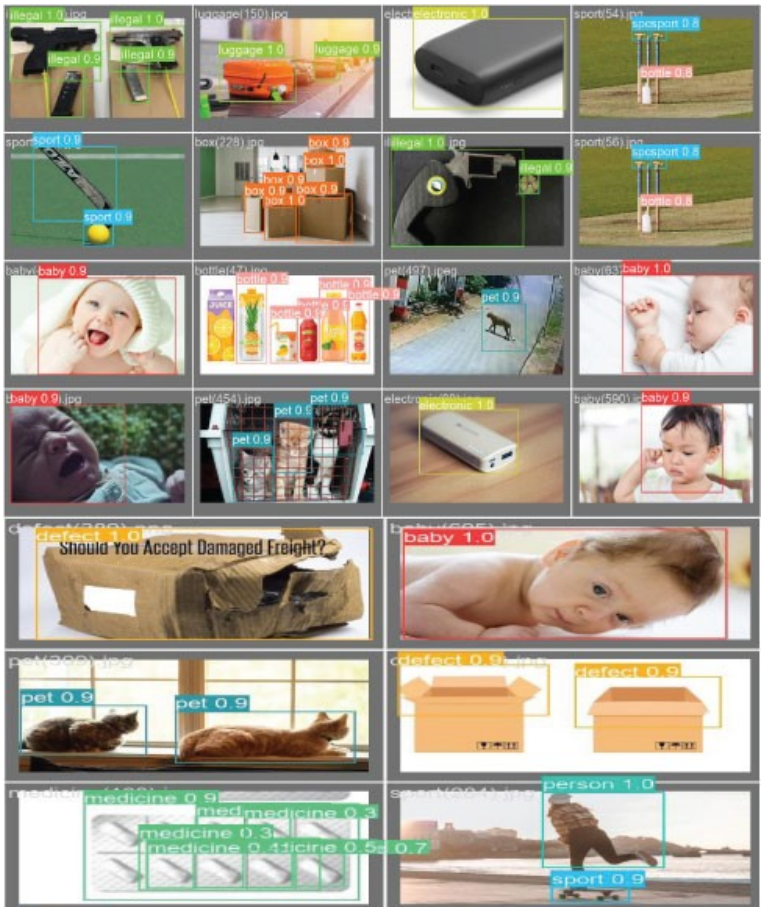


Figure 5.3: Prediction of YOLOv5 model

5.2.2 VGG16

Under 20 epochs, we obtained a training accuracy of 87% while a validation accuracy of 70% with 4598 train images and 1143 test images.

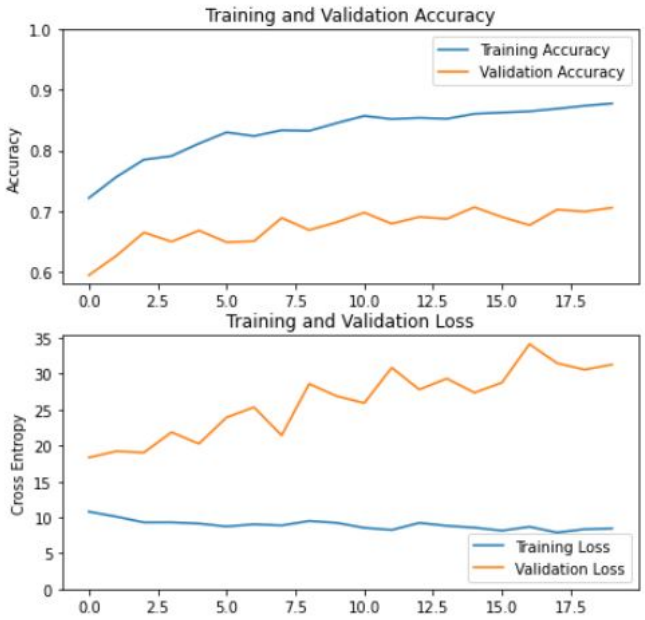


Figure 5.4: Curve of Training-Validation Accuracy and Loss of VGG16 model

We may observe random images from the test dataset where the model predicted the class by learning the image's feature.

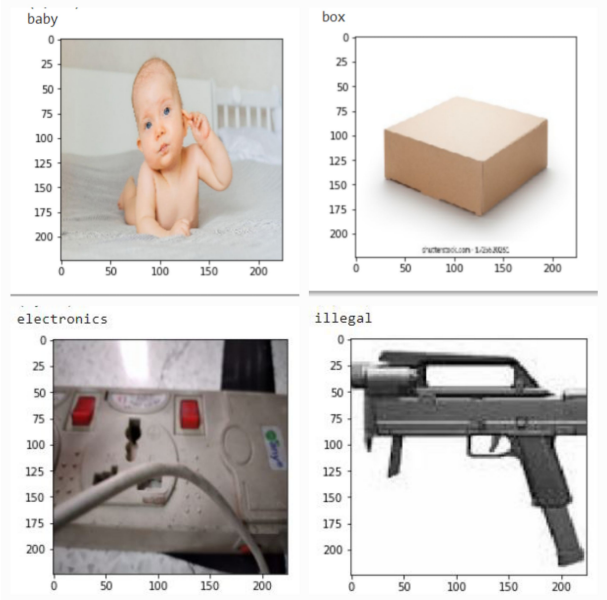


Figure 5.5: Prediction of VGG16 model

5.2.3 ResNet50

Under the similar train, test split, we obtained a training accuracy of 90% and a validation accuracy of 72%. As we can see, the model slightly outperformed the prior model, VGG16.

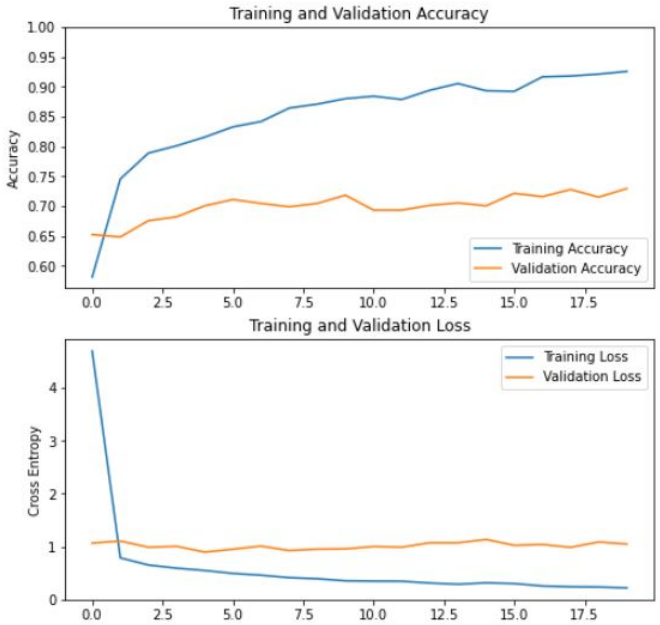


Figure 5.6: Curve of Training-Validation Accuracy and Loss of ResNet50 model

As we can see from the prediction result, we have randomly given some test sample paths to our model while it predicted the corresponding classes.

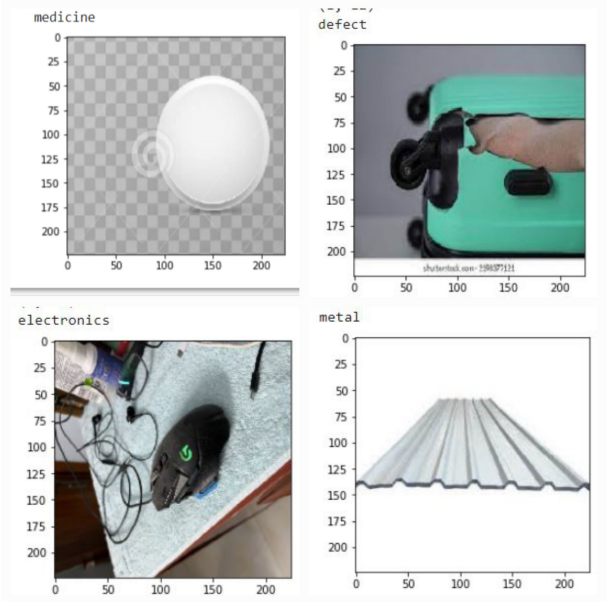


Figure 5.7: Prediction of ResNet50 model

5.2.4 MobileNetv2

Under the same train, test split, we obtained an accuracy of 92% and a validation accuracy of 86%. As we can see, the model performed considerably better than the previous models, VGG16 and ResNet50, due to the fact that it is a compact model meant to function on devices with limited computational power.

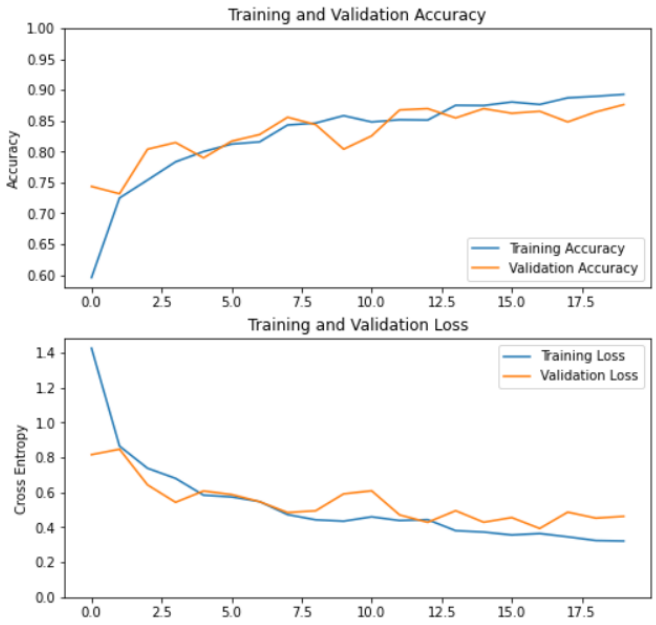


Figure 5.8: Curve of Training-Validation Accuracy and Loss of MobileNetv2 model

Similarly, we have predicted and classified sample data from our test dataset according to the model's feature extraction and training.

Figure 5.9: Prediction of MobileNetv2 model

5.2.5 YOLOv7

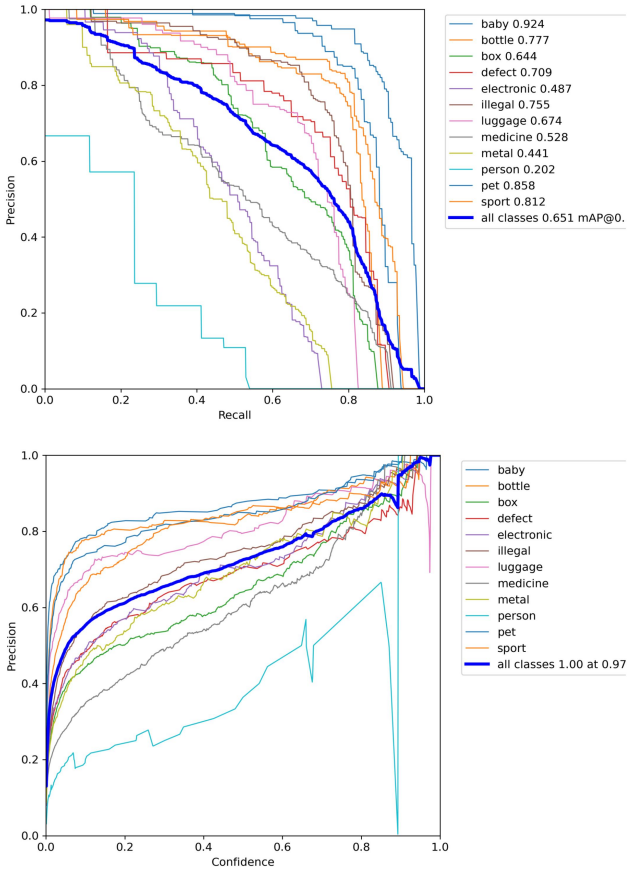


Figure 5.10: Curve of Precision-Confidence and Precision-Recall of YOLOv7 model

In FIG 5.10 we can see that after 80 epochs, the precision confidence of 97% would be more perfect if the model were trained for longer epochs. Moreover. we can see here the precision-recall curve has less confidence than YOLOv5 when it's compared with 300 epochs of YOLOv5. but it's actually better than YOLOv5 with the same epochs number. It averages around 65%.
Here it shows after 80 epochs top 3 classes that were predicted correctly are sports, pet, and baby classes which have predicted percentages over 78%.

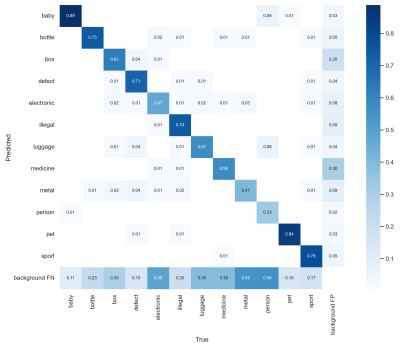


Figure 5.11: Confusion Matrix of YOLOv7 model

Figure 5.12: Prediction of YOLOv7 model- part1

Figure 5.13: Prediction of YOLOv7 model- part2

5.2.6 DCGAN

For DCGAN we have used 92 images and generated around 8000 images. But for the sample, we have 4 images and a few outputs that are generated as fake images.

Figure 5.14: Original Images

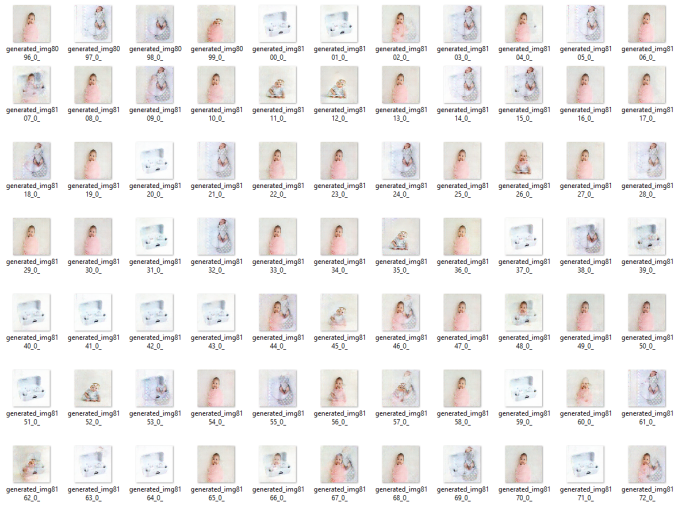


Figure 5.15: Generated Images

5.2.7 CNN

As CNN was trained from scratch, our validation accuracy was around 58% while training accuracy was around 98%. Due to training from scratch with fewer epochs over our custom dataset, and some of the images having multiple classes to train from, we can visualize the reason why validation accuracy was dropped.

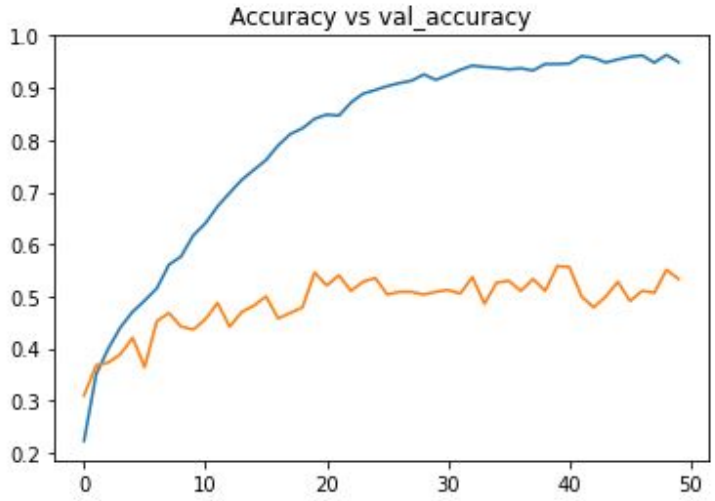


Figure 5.16: Curve of Training-Validation Accuracy of CNN model

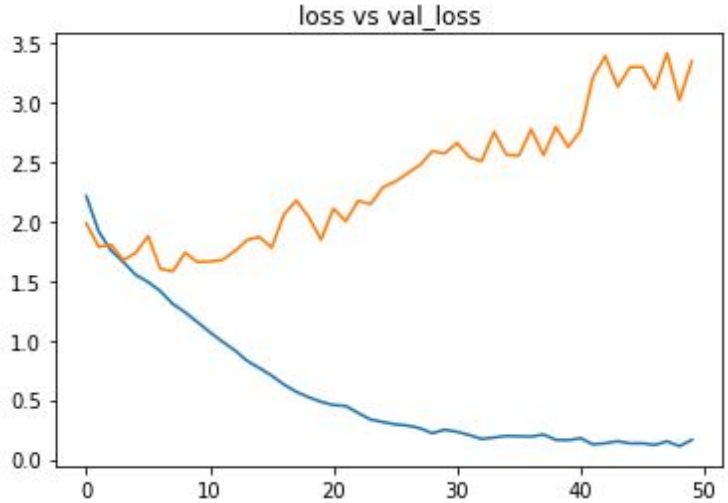


Figure 5.17: Curve of Training-Validation Loss of CNN model

Even though the validation accuracy was low, we had a high training accuracy which could predict almost all the classes accordingly of the given samples.

Figure 5.18: Prediction of CNN model- part1

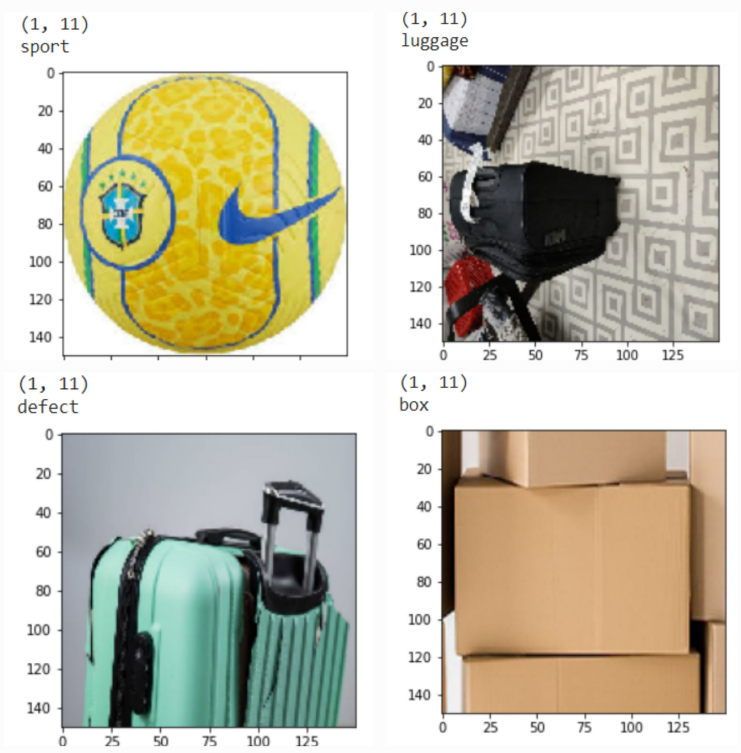


Figure 5.19: Prediction of CNN model- part2

Models	Epoch	Train_Accuracy	valid_Accuracy
MobileNetv2	20	92	86
ResNet50	20	90	72
VGG16	20	87	70
YOLOv5	300	99	87.7
YOLOv7	80	65	43
ViT	100	56.54	87.65 (top 5)
CNN	50	98	58

Table 5.1: Result Analysis of all the Models

After implementing various models, we can see that we got the highest validation accuracy with MobileNet with the same custom dataset and 20 training epochs among all the pre-trained models. On the other hand, We got the highest training accuracy with an almost accurate prediction rate with a custom CNN model.

We have applied our YOLOv5 model which had the highest validation accuracy over two videos for real-time detection where the first video is from an internet source related to the airport conveyor belt [29] and another one is from our own surroundings.

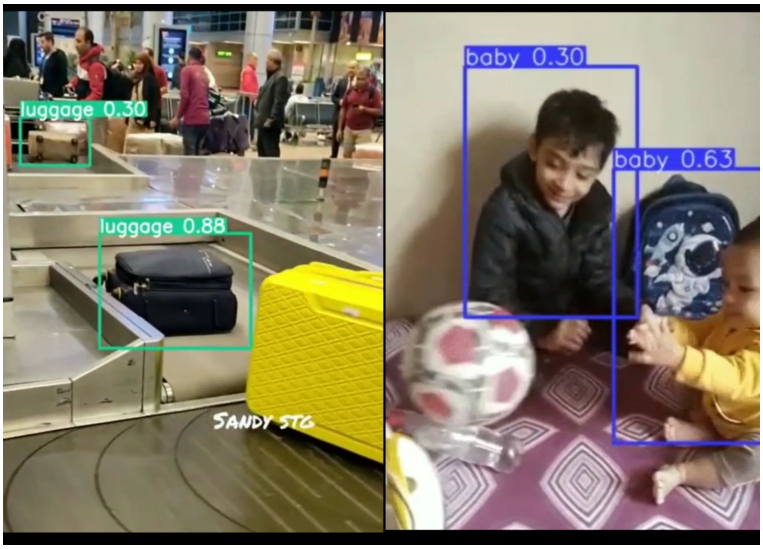


Figure 5.20: Real-time prediction through a video from an internet source and our surroundings

5.3 Limitation and Future Work

We have analyzed a wide range of reports about object detection and classification since the very beginning of this thesis. In order to achieve the proper accuracy, we tried to train our model using various pre-trained models with machine learning algorithms. However, due to a few constraints, we were unable to make it practically flawless.

- Our model mainly focuses on predicting classes given random image data. As we took the labels by training dataset subfolders, multi-class single images might affect the training process.
- Among many multi-formatted images, PNG will be dropped due to SRGB mismatch and train the rest of the images from the dataset.
- For a few models, such as YOLO, it requires high image sizes as smaller ones can not extract the feature details. Our dataset had mixed sizes and resolution images.
- The training process was comparatively slow due to using a cloud environment (Google Colaboratory). We frequently got disconnected from the model training and had to restart it.
- When we were training on a local system, we encountered various limitations related to hardware configuration and library versions. So we had to limit the batch size for some models such as YOLOv7.
- As the model training was taking a lot of time, we could not train it for a large number of epochs. For example, it is recommended to train for 500 epochs for real-time detection for good accuracy. However, we could only train for 300 epochs as it was taking a long time.

In the future, we will try to refine the dataset more by increasing the class numbers and image varieties. We would also try to use high-end GPUs with larger VRAM so that we can overcome these limitations by increasing the image/batch sizes and epochs. Furthermore, as we have implemented the Vision Transformer model in our study, we would like to work with some other transformer models such as-DeiT, DETR, T2T-ViT, and PVT. Lastly, we would like to use real images from the airport conveyor belt to train our models.

Chapter 6

Conclusion

Real-time object detection is essential for enhancing the security system for daily living. Therefore, the primary objective of our research was to enhance detection accuracy by optimizing the neural network system. Moving objects like human detection and other safety object detection has become more crucial in the current world for purposes like surveillance footage, self-driving vehicles, and human identification since these are becoming more and more significant in machine learning. In this study, we have applied a few versions of YOLO, some pre-trained models, a vision transformer, and a custom CNN model to train our model and optimize the neural network efficiently for system improvement. We have applied the algorithms for training-testing on our own custom dataset and determined the result in real-time from each of the training images and videos from surroundings to see if it is able to recognize the human body, which is essentially a toddler, and by detecting other objects that could possibly get on an airport conveyor belt. Although our custom CNN model had a lower validation accuracy due to single images having multi-class with a higher training accuracy, it could recognize almost all the given testing images. By recognizing objects, this study will assist us in ensuring adequate safety concerns on the airport conveyor belt.

Bibliography

[1] O. Arif, M. Marshall, W. Daley, *et al.*, "Tracking and classifying objects on a conveyor belt using time-of-flight camera," Jun. 2010. DOI: 10.22260/ISARC2010/0022.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, Sep. 2014.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[6] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[7] B. Liu, W. Zhao, and Q. Sun, "Study of object detection based on faster r-cnn," in *2017 Chinese Automation Congress (CAC)*, 2017, pp. 6233–6236. DOI: 10.1109/CAC.2017.8243900.

[8] W. Cai, J. Li, Z. Xie, T. Zhao, and K. LU, "Street object detection based on faster r-cnn," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 9500–9503. DOI: 10.23919/ChiCC.2018.8482613.

[9] S. Mane and S. Mangale, "Moving object detection and tracking using convolutional neural networks," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2018, pp. 1809–1813. DOI: 10.1109/ICCONS.2018.8662921.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," Jun. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[11] D. M. Dinama, Q. A'yun, A. D. Syahroni, I. Adji Sulistijono, and A. Risnumawan, "Human detection and tracking on surveillance video footage using convolutional neural networks," in *2019 International Electronics Symposium (IES)*, 2019, pp. 534–538. DOI: 10.1109/ELECSYM.2019.8901603.

[12] K. J. Liang, J. B. Sigman, G. P. Spell, *et al.*, "Toward automatic threat recognition for airport x-ray baggage screening with deep convolutional object detection," *arXiv preprint arXiv:1912.06329*, 2019.

[13] J. Park, J. Chen, Y. Cho, D. Kang, and B. Son, "Cnn-based person detection using infrared images for night-time intrusion warning systems," *Sensors (Basel, Switzerland)*, vol. 20, Dec. 2019. DOI: 10.3390/s20010034.

[14] D. Sangwan and D. Jain, "An evaluation of deep learning based object detection strategies for threat object detection in baggage security imagery," *Pattern Recognition Letters*, vol. 120, Apr. 2019. DOI: 10.1016/j.patrec.2019.01.014.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[16] Y. Wang, J. Wu, and H. Li, "Human detection based on improved mask r-cnn," *Journal of Physics: Conference Series*, vol. 1575, p. 012 067, Jun. 2020. DOI: 10.1088/1742-6596/1575/1/012067.

[17] S. Win and T. L. L. Thein, "Real-time human motion detection, tracking and activity recognition with skeletal model," in *2020 IEEE Conference on Computer Applications(ICCA)*, 2020, pp. 1–5. DOI: 10.1109/ICCA49400.2020.9022822.

[18] H. Zhu, X. Yan, H. Tang, Y. Chang, B. Li, and X. Yuan, "Moving object detection with deep cnns," *IEEE Access*, vol. 8, pp. 29 729–29 741, 2020. DOI: 10.1109/ACCESS.2020.2972562.

[19] M. Ansari and D. Singh, "Monitoring social distancing through human detection for preventing/reducing covid spread," *International Journal of Information Technology*, vol. 13, pp. 1255–1264, Apr. 2021. DOI: 10.1007/s41870-021-00658-2.

[20] A. I. Rahman, S. Bhuiyan, Z. H. Reza, J. Zaheen, and T. A. N. Khan, "Detection of intracranial hemorrhage on ct scan images using convolutional neural network," Ph.D. dissertation, Brac University, 2021.

[21] S. Han, X. Dong, X. Hao, and S. Miao, "Extracting objects' spatial–temporal information based on surveillance videos and the digital surface model," *ISPRS International Journal of Geo-Information*, vol. 11, p. 103, Feb. 2022. DOI: 10.3390/ijgi11020103.

[22] H.-K. Jung and G.-S. Choi, "Improved yolov5: Efficient object detection using drone images under various conditions," *Applied Sciences*, vol. 12, p. 7255, Jul. 2022. DOI: 10.3390/app12147255.

[23] P. B. Prithvi, F. Zahin, and S. S. Anny, "Pest detection system using machine learning techniques," Ph.D. dissertation, Brac University, 2022.

[24] Wikipedia contributors, *Convolutional neural network — Wikipedia, the free encyclopedia*, [Online; accessed 16-January-2023], 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=1131414814.

[25] *2-year-old injured after taking ride on atlanta airport conveyor belt - abc news*, https://abcnews.go.com/US/year-injured-taking-ride-atlanta-airport-conveyor-belt/story?id=64546347, (Accessed on 01/17/2023).

[26] *401 unauthorized*, https://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf, (Accessed on 01/17/2023).

[27] *Alexnet — pytorch*, https://pytorch.org/hub/pytorch_vision_alexnet/, (Accessed on 01/21/2023).

[28] *Baby dies in spain in airport conveyor belt accident — cbc news*, https://www.cbc.ca/news/world/baby-dies-in-spain-in-airport-conveyor-belt-accident-1.1860732, (Accessed on 01/17/2023).

[29] *Baggage collection at airport baggage conveyor belt baggage handling system airport experience - youtube*, https://www.youtube.com/shorts/gQz_kiF2Ndo, (Accessed on 01/21/2023).

[30] *Cat is saved from being crushed in a garbage machine — daily mail online*, https://www.dailymail.co.uk/news/article-9081993/Cat-saved-crushed-garbage-machine.html, (Accessed on 01/17/2023).

[31] *Conveyor belt accidents – larry pitt*, https://larrypitt.com/aop/workers-compensation/workplace-accidents/conveyor-belt/, (Accessed on 01/17/2023).

[32] *Efficientnet b0 to b7*, https://keras.io/api/applications/efficientnet/, (Accessed on 01/23/2023).

[33] *Home - opencv*, https://opencv.org/, (Accessed on 01/23/2023).

[34] *How to train and use a custom yolov7 model*, https://blog.paperspace.com/yolov7/, (Accessed on 01/21/2023).

[35] *Image classification using cnns in keras — learnopencv*, https://learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/, (Accessed on 01/17/2023).

[36] *Inceptionresnetv2*, https://keras.io/api/applications/inceptionresnetv2/, (Accessed on 01/23/2023).

[37] *Inceptionv3*, https://keras.io/api/applications/inceptionv3/, (Accessed on 01/23/2023).

[38] *Ms coco dataset: Using it in your computer vision projects*, https://datagen.tech/guides/image-datasets/ms-coco-dataset-using-it-in-your-computer-vision-projects/, (Accessed on 01/23/2023).

[39] *Numpy*, https://numpy.org/, (Accessed on 01/23/2023).

[40] *Pytorch*, https://pytorch.org/, (Accessed on 01/23/2023).

[41] *Support vector machine — introduction to machine learning algorithms — by rohith gandhi — towards data science*, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47, (Accessed on 01/23/2023).

[42] *Understanding pascal voc dataset — engineering education (enged) program — section*, https://www.section.io/engineering-education/understanding-pascal-voc-dataset/, (Accessed on 01/23/2023).

[43] *Vgg-16 — cnn model - geeksforgeeks*, https://www.geeksforgeeks.org/vgg-16-cnn-model/, (Accessed on 01/22/2023).

[44] *Vision transformer explained — papers with code*, https://paperswithcode.com/method/vision-transformer, (Accessed on 01/17/2023).

[45] *Xception*, https://keras.io/api/applications/xception/, (Accessed on 01/23/2023).

[46] *Yolov7 - a breakdown of how it works*, https://blog.roboflow.com/yolov7-breakdown/, (Accessed on 01/23/2023).