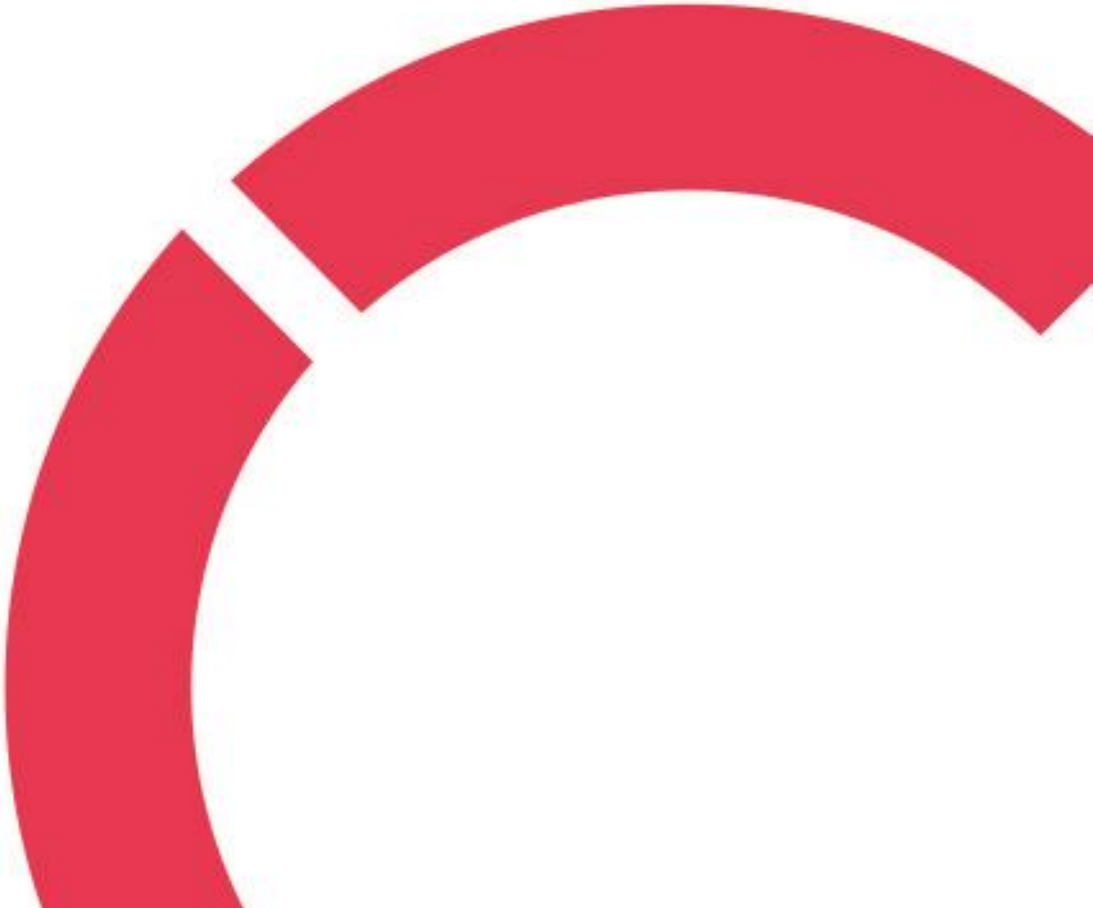
Ningyang Sun

THE DESIGN AND IMPLEMENTATION OF MASK WEARING RECOGNITION SYSTEM BASED ON YOLO V5

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Information Technology
October 2022



ABSTRACT

Centria University of Applied Sciences	**Date** October 022	**Author** Ningyang Sun
Degree programme Information Technology		
Name of thesis THE DESIGN AND IMPLEMENTATION OF MASK WEARING RECOGNITION SYSTEM BASED ON YOLO V5		
Centria supervisor Kauko Kolehmainen		**Pages** 37+3
Instructor representing commissioning institution or company Kauko Kolehmainen		

With the rapid development of deep learning and computer vision, related technologies and equipment have been widely used in various fields. And the most basic part of computer vision, as well as the most important step in image understanding, is target detection. Target detection can select the specified object in a large number of captured images. This helps researchers confirm more information about the target object, and further locate and classify the object.

Based on the nature and characteristics of target detection, combined with the current situation of the epidemic, computer vision should be more applied to epidemic prevention. According to multiple studies, in addition to COVID, there are a number of seasonal diseases that can be reduced by wearing masks. Target detection can be used to judge the situation of people wearing masks and help the staff in public places to judge the current environment and further processing in time. At present, such tools have not been widely used in the market, so the main direction of this system is to design a practical, lightweight and universal mask recognition system.

The system is designed to detect the wearing of masks in complex scenarios, which can help employees in crowded public places judge the situation of people wearing masks. he system uses Pytorch as a framework, uses YOLO V5 target detection algorithm to classify the video or image after feature extraction, which can make more accurate judgment, and uses PyQt to make a simple window that presents the results to the user.

Key words
Mask recognition, Objection detection, OpenCV, PyQt, Pytorch, YOLO V5s

CONCEPT DEFINITIONS

YOLO You Only Look Once

2D Filtering Two-dimensional Filtering

MATLAB Matrix and Laboratory

API Application Programming Interface

GPU Graphics Processing Unit

CNN Convolutional Neural Network

R-CNN Region Convolutional Neural Network

PANET Path Aggregation Network for Instance Segmentation

RF Random Forest

IPP Integrated Performance Primitives

SIFT Scale-invariant Feature Transform

HOG Histogram of Oriented Gradient

SVM Support Vector Mac

GT Ground Truth

ANN Artificial Neural Networks

FAIR Facebook Artificial Intelligence Research

SPP Spatial Pooling Layer

CIOU Complete Intersection Over Union

DIOU Distance Intersection Over Union

FPN Feature Pyramid Network

CSP Cross Stage Partial

OpenCV	Open Source Computer Vision Library
GUI	Graphical User Interface
UNIX	Uniplexed Information and Computing Service
COCO	Microsoft Common Objects in Context
mAP	mean of Average Precision
BN	Batch_Norm

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

FIGURES

PICTURES

1 INTRODUCTION

With the development of modernization, the pace of people's life has accelerated, and images have gradually become the carrier of effective and rapid communication. Images are not limited by language factors and can better express and convey people's information. Compared with text, images can convey more information. However, while providing convenience to people's life, this feature also brings new problems. Extracting effective information from tedious and large number of images has become a key research topic in the near future.

In recent years, due to the wide application of image processing in various scenes, image recognition has also become a research hotspot. People try to apply image recognition technology to a variety of fields. For example, in animal husbandry, people recognize the growth status of livestock through images. In industry, people recognize the quality of products through images. In the service industry, people recognize license plates, brands and so on through images. In addition, image recognition is also widely used in many other fields. In conclusion, image is an indispensable factor recently. Using traditional methods or deep learning methods to recognize images and extract color feature information is of great significance.

Image recognition is also a key problem in machine vision. Image recognition is the process of studying certain features in images and identifying and classifying objects in images, and is an important branch of pattern recognition. Under normal circumstances, an image recognition system includes multiple processes, such as image acquisition, preprocessing, feature extraction, classifier design and classification decision-making (Deng 2021, 2427). In the image preprocessing, it is needed to use cutting, denoising, feature extraction and other methods. Afterward, the classifier is used to classify the same features into one class to realize grouping. Timely acquisition of accurate and complete feature information is the key to reflecting user needs and serving the entire industry. At present, image processing has become a hot topic, and face recognition technology has been widely used in image processing. Face recognition can use cameras to collect information, detect it, and quickly identify the wearing of masks. Therefore, this technology is of great significance for epidemic control, and investment in this area is also very necessary and valuable.

In the face of fast-moving epidemics including monkeypox and new pneumonia virus, people's life has been seriously disrupted. Unlike the common flu, this epidemic can be spread through human saliva and

has a strong ability to infect others. Wearing a mask can effectively stop the spread of this virus. Masks can be recognized directly from the source, isolate human saliva and airflow from the first barrier. It had been found that wearing face masks is 96% effective to stop the spread of the virus (Adusumalli, Kalyani, Sri, Pratapteja & Rao 2021, 1304). In crowded places such as shopping malls, restaurants, and subways, there are often situations where someone does not wear a mask. This is a major safety hazard to one's own life and others. It is particularly important and urgent to have a system that can identify whether people are wearing masks through the collection and identification of digital images, and make corresponding early warnings and alarms. With the development of computer vision technology, image recognition provides a new way to solve this problem.

2 RESEARCH STATUS

Machine learning addresses the question of how to build computers that improve automatically through experience. It is one of today's most rapidly growing technical fields, lying at the intersection of computer science and statistics, and at the core of artificial intelligence and data science. (Jordan, Mitchell 2015, 255-260) Machine vision is an important branch of machine learning. Machine vision technology provides image-based inspection and analysis for demanding applications. With the use of software, sensors, cameras, and robot guidance, integrated systems can be realized. (Moru and Borro 2020, 105-123)

2.1 Machine Learning

Machine learning is used to teach machines how to handle the data more efficiently. Sometimes after viewing the data, we cannot interpret the extract information from the data. In that case, we apply machine learning. With the abundance of datasets available, the demand for machine learning is in rise. Many industries apply machine learning to extract relevant data. The purpose of machine learning is to learn from the data. Many studies have been done on how to make machines learn by themselves without being explicitly programmed. Many mathematicians and programmers apply several approaches to find the solution of this problem which are having huge data sets. (Mahesh 2018, 381-386)

2.1.1 Research Status of Traditional Machine Learning

Machine learning methods have been used comprehensively in many application areas like speech recognition, health care, business forecasting and agriculture. The research directions of traditional machine learning mainly include decision tree, random forest, artificial neural network, Bayesian learning . Decision tree is a common approach to machine learning. Since the end of the 20th century, this method has been continuously improved and developed vigorously. Random forest is a classification and prediction method using multiple tree classifiers. Over the past decade, the research of random forest algorithm has developed rapidly, and has been applied in many fields such as bioinformatics, ecology, medicine, genetics, remote sensing geography. Artificial Neural Networks is an algorithm with nonlinear

adaptive information processing ability, which can overcome the defects of traditional artificial intelligence methods in intuition, such as pattern, speech recognition and unstructured information processing. Bayesian learning is an early research direction of machine learning. After the joint efforts of many statisticians, Bayesian statistics was gradually established after the 1950s and became an important part of statistics. (Rani, Kotwal, Manhas, Sharma & Sharma 2021, 1803-1810)

2.1.2 Research Status of Machine Learning in Big Data Environment

Digital datasets have been rapidly growing in size and complexity, and the large volume of daily generated data exceeds the boundary of normal processing capabilities. And they are forcing people to take advanced computational infrastructures which are able to tackle parallel and distributed processing. Efficient mining of such massive data amounts is an extremely challenging practice which requires developing more sophisticated platforms to get extensive big data analysis accurately done in a timely fashion. The value of big data is mainly reflected in the diversion of data and the information processing ability of data. Many existing machine learning methods are based on memory theory. When big data cannot be loaded into computer memory, many algorithms cannot be processed. Therefore, new machine learning algorithms should be proposed to meet the needs of big data processing. Machine learning algorithms in big data environment can ignore the importance of learning results according to certain performance standards. Using distributed and parallel computing to implement the divide and conquer strategy can avoid the interference caused by noisy data and redundancy, reduce the storage cost, and improve the operation efficiency of the learning algorithm. (Nti, Quarcoo, Aning & Fosu 2022, 81-82)

2.1.3 Classification of Machine Learning

Over the five decades, machine learning can be classified according to the different aspects of emphasis. For example, classification based on learning strategies, classification based on learning methods, classification based on data forms and classification based on learning objectives. The method based on learning strategy includes two types, one is machine learning which simulates human brain and another is machine learning which directly adopts mathematical methods. Machine learning which simulates human brain includes symbol learning and neural network learning. Machine learning using mathematical methods includes mainly statistical machine learning. The three elements of statistical machine learning are model, strategy and algorithm. Classification based on learning methods includes inductive learning, analogy learning, deductive learning and analytical learning. Classification based on data form

includes structured learning and unstructured learning. Classification based on learning objectives includes concept learning, rule learning, function learning, category learning and Bayesian network learning. Based on these different classifications, developers continue to optimize methods according to their different characteristics. (Kotsiantis, Zaharakis & Pintelas 2006, 159-167)

2.2 Machine Vision

The surge of deep learning over the last years is to a great extent due to the strides it has enabled in the field of computer vision. Deep learning has fuelled great strides in a variety of computer vision problems, such as object detection, motion tracking, action recognition, human pose estimation, and semantic segmentation. Three of the most important types of deep learning models with respect to their applicability in visual understanding is Convolutional Neural Networks, the "Boltzmann family" including Deep Belief Networks and Deep Boltzmann Machines and Stacked Autoencoders. These three key categories of deep learning for computer vision, have been employed to achieve significant performance rates in a variety of visual understanding tasks, such as object detection, face recognition, action and activity recognition, human pose estimation, image retrieval, and semantic segmentation. However, each category has distinct advantages and disadvantages. CNNs have the unique capability of feature learning, that is, of automatically learning features based on the given dataset. CNNs are also invariant to transformations, which is a great asset for certain computer vision applications. On the other hand, they heavily rely on the existence of labelled data, in contrast to DBNs/DBMs and SdAs, which can work in an unsupervised fashion. Of the models investigated, both CNNs and DBNs/DBMs are computationally demanding when it comes to training, whereas SdAs can be trained in real time under certain circumstances. The focus of machine vision is image processing. Image processing mainly includes image preprocessing and image feature analysis. Now the image preprocessing and feature analysis has been integrated into the model, which is very fast and easy to use. Thus, the following only briefly analyses the steps and research status of these two steps. (Voulodimos, Doulamis, Doulamis & Protopapadakis 2017, 1-14)

2.2.1 Image Preprocessing

Affected by lighting, imaging distance, imaging angle, and noise, the quality of the obtained image can sometimes be degraded. The data preprocessing step performed before training a deep learning model

involves changing the existing data to data suitable for the learning algorithm. The main goal of image preprocessing is to eliminate irrelevant information in the image and restore useful information to enhance the detectability of its information. At the same time, image preprocessing simplifies the data as much as possible to obtain simpler data and more comprehensive results. Therefore, the image must be preprocessed before pattern recognition. Image preprocessing includes image scaling, grayscale processing, and smoothing images. (Yun, Oh, Shin 2021, 123)

Image scaling is to transform the size of an image according to a certain ratio. This transformation can change the spatial relationship between pixels in an image. Because images are usually represented by pixels at integer positions, it is possible to map to multiple points by the position of a point. Therefore, the scaled image is unable to accurately find the corresponding points all the time and must be approximated. For image processing, the scaling factors reflect the complexity of the texture. For approximate processing methods, interpolation is usually used. In the interpolation algorithm, it includes nearest neighbor interpolation and linear interpolation. Nearest neighbor interpolation is the simplest interpolation method. It directly selects the pixel value closest to the mapped position and assigns the pixel value to it. This method is simple, fast, and the grayscale fidelity is good. But the difference is large, and the visual characteristics are poor. Linear interpolation can be transformed with a matrix. Assume that the scaling ratio of the X-axis direction of the image is f(x), and the scaling ratio of the Y-axis direction is f(y). Then a point (x0, y0) in the original image corresponds to (x1, y1) in the new image, and the conversion between them can be represented by a matrix:

$$\begin{Bmatrix} X_1 \\ Y_1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{Bmatrix} \begin{Bmatrix} X_0 \\ Y_0 \\ 1 \end{Bmatrix}$$

The expression for the inverse operation is:

$$\begin{Bmatrix} X_0 \\ Y_0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 1/f_x & 0 & 0 \\ 0 & 1/f_y & 0 \\ 0 & 0 & 1 \end{Bmatrix} \begin{Bmatrix} X_1 \\ Y_1 \\ 1 \end{Bmatrix}$$ (Zhang, Fan, Bao, Liu & Zhang 2018, 3796-3784)

The image processing system must able to handle grayscale images, which are usually obtained in real time applications. In image processing, processing three channels of data can be complicated. Therefore, the image needs grayscale processing at first. (Dinesh and Vairavel 2014, 1-2)

Image grayscale is the process of converting a color image to a grayscale image. Since the three elements R, G, and B mix different colors in each pixel, and each element has 256 values, there are more than 16 million rotations (256 × 256 × 256) per pixel. Grayscale images are special color images in which each pixel is 256 pixels in size and all elements R, G, and B are equal. Usually, many color image formats are converted to grayscale images in order to improve the efficiency of subsequent image processing. Like color images, gray images can also represent graphic detail features. (He 2022)

Binarization refers to converting a grayscale image with sharp sensitivity into a binary image with only two-dimensional grayscale values by choosing appropriate boundaries. After such a change, although the background color is removed, the changed grayscale image can still represent the general characteristics of the image. The difference between the distance and the back is used, and the segmentation threshold is used to combine the back and back of different grayscales as a picture, the best threshold is selected to distinguish each pixel in the target and background areas, and then a binary image can be formed, and the target to be detected can be obtained. (He 2022)

At present, the principle of image smoothing technology is to reduce the noise of the image. There are many reasons for noise, including internal and external noise. It may also be caused by external current or working noise of the system. Electronic system, measuring instrument, climate can cause noise. In order to improve the quality of image, image enhancement technology is widely used in order to obtain images conducive to analysis and recognition. Image smoothing is to filter the noise inside the image while preserving the original information as much as possible. The basic principle of image smoothing is to compare the pixel differences between the pixels in the image and the surrounding pixels. If the pixel difference value is large, the pixel point with large difference value will be processed, and the pixel point in the image will be adjusted to the approximate value of the surrounding pixel point pixel value. Image smoothing methods include mean filtering, box filtering, Gaussian filtering, median filtering, bilateral filtering and 2D filtering. (Park, Jeong, Park, Kim, Lee, Mo, Jang & Park 2020, 1111-1113)

2.2.2 Feature Extraction

Feature extraction describes the relevant shape information contained in a pattern so that the task of classifying the pattern is made easy by a formal procedure. In pattern recognition and in image processing, feature extraction is a special form of dimensionality reduction. The main goal of feature extraction is to obtain the most relevant information from the original data and represent that information

in a lower dimensionality space. When the input data to an algorithm is too large to be processed and it is suspected to be redundant then the input data will be transformed into a reduced representation set of features. Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input Pattern recognition is an emerging field of research in the area of image processing. It has been used in many applications such as character recognition, document verification, reading bank deposit slips, extracting information from cheques, applications for credit cards, health insurance, loan, tax forms, data entry, postal address reading, check sorting, tax reading, script recognition etc. (Kumar and Bhatia 2021, 5-12)

Feature plays a very important role in the area of image processing. Before getting features, various image pre-processing techniques like binarization, thresholding, resizing, normalization etc. are applied on the sampled image. After that, feature extraction techniques are applied to get features that will be useful in classifying and recognition of images. Feature extraction techniques are helpful in various image processing applications like character recognition. As features define the behaviour of an image, they show its place in terms of storage taken, efficiency in classification and obviously in time consumption also. (Kumar and Bhatia 2021, 5-12)

2.3 Object Detective

There are several basic tasks in the field of machine vision: image classification, object detection, instance segmentation and semantic segmentation. Target inspection originated late. Around the beginning of this century, due to the return of deep learning to the public's vision, the idea of target detection was proposed. Object detection as the most basic task in computer vision, has attracted extensive attention in recent ten years. As shown in the figure below, it takes 2014 as a milestone, object detection can be roughly divided into two periods in the past two decades. Before 2014 was the traditional target detection period, followed by the deep learning-based target detection period. (Zou, Shi, Guo & Ye 2019)

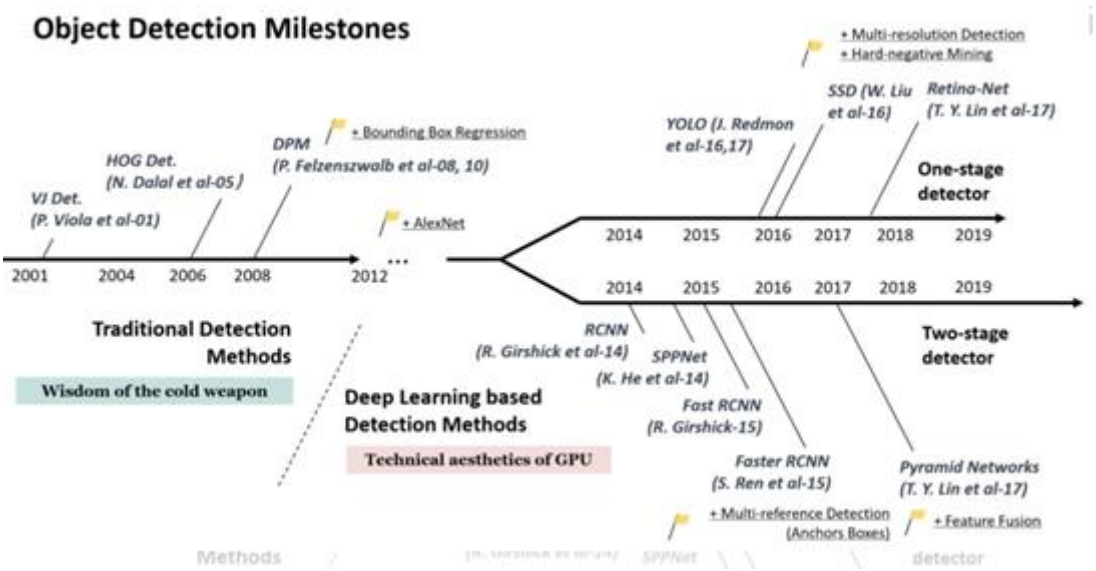


FIGURE 1 Object Detection Development (adapt from Zou, Shi, Guo & Ye 2019)

2.3.1 Traditional Target Detection Algorithms

The traditional target detection process consists of three steps, region selection, feature extraction and classification. Firstly, the whole image is traversed by the strategy of sliding window, so as to select regions. Then, the areas may contain objects is carried out feature extraction. The commonly used algorithms in this stage include Haar, SIFT, HOG. The last step is classification, detection and classification of extracted features, the main algorithms are SVM, Adaboost. In the multi-target tracking technology, the traditional multi-target tracking technology associates the measurement with the track, which not only has a large amount of calculation, but also has poor algorithm performance. (Liu, Deng, Ma, Du, Xiong, Hu, Zhang & Ji 2021)

FIGURE 2. Difference between object detection and other task (adapt from Liu, Deng, Ma, Du, Xiong, Hu, Zhang & Ji 2021)

2.3.2 Target Detection Algorithm Based On Deep Learning

The traditional target detection algorithm only applies to the situation with notable features and simple background. But in practical application, the background is complex, and the target to be detected is also changeable, it is difficult to complete the target detection through general abstract features. However, deep learning can extract rich features of the same target to complete the detection of the target. Object detection algorithms based on deep learning are mainly divided into Two categories: One stage and Two stage. One stage method unifies the process of location recognition and classification. Instead of using region proposal, features are extracted directly from the neural network to predict object classification and position. Common One stage algorithms include OverFeat, YOLO, SSD and RetinaNet. The other is two-stage method, which is also called region-based method because of its two-stage processing of images. Two stages firstly generate Region proposal, and then classify samples through convolutional neural network. CNN is based learner because it is demonstrated to extract the local visual features and they are used in the recognition algorithms. Classical algorithms in this mode include R-CNN, Fast R-CNN, Faster R-CNN and Mask-RCNN. By comparing the accuracy and speed of the two algorithms, one stage has an advantage in speed, while the Two stage is superior in accuracy. But over time, both methods have improved greatly in areas where they were weak, and both can maintain speed and accuracy in this area. (Mane & Mangale 2018, 1809-1810)

The process of One-stage target detection algorithm is clear. As the name suggests, One-stage can get the classification probability and coordinate through one process. As shown below, input the image first and generate output through CNN before decoding. CNN is trained with millions of images of different classes. Then, the corresponding detection frame is regenerated. However, when performing the task of target detection, there will be different losses. Here the loss function can be used to measure the deviation between the predicted value and the actual value. In addition, the One-stage target detection algorithm still has other problems, such as the inequality of positive and negative. These problems are also being improved. (Krizhevsky, Sutskever & Hinton 2012, 85-90)

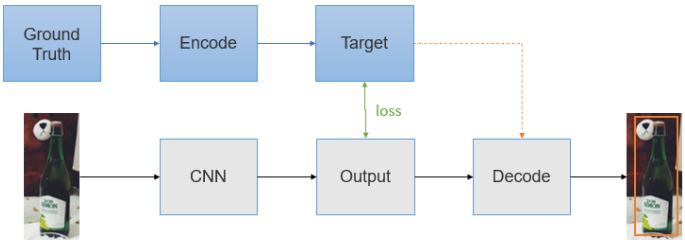


FIGURE 3. One stage basic process

Two-stage mainly completes the object detection process through convolutional neural network. What it extracts is CNN convolution feature. When training the network, there are two main parts. First, it generates a series of candidate boxes as samples by the algorithm, and then the samples are classified through the convolutional neural network. As shown in the Figure 4 below, first input an image and extract the depth features of the image through the convolutional neural network. Then, candidate regions are generated by RPN network, and the classification of regions is also completed. The image is divided into two different categories, background and target, and the target location is preliminarily predicted. Next, the Roi pooling layer is used to accurately and correct the location in the candidate areas, which can be understood as a matting operation. Then, the candidate target obtained by matting is corresponding to feature region on the feature map. Then through a full connection layer, the corresponding eigenvectors are obtained. Finally, two branches, classification and regression, are used to determine the candidate target category and target location. The location (x, y) is the coordinates of the top left vertex. And, w and h are the length and width of the detection frame. However, RPN generates multi-scale proposals by sliding a fixed set of filters over a fixed set of convolutional feature maps with the same resolution. This results in an inconsistency between the sizes of objects and filter receptive fields, as the scales of objects are variable, yet the sizes of filter receptive fields are fixed. In practice, it leads to compromised results with particularly poor detection performance for small-size objects. To exploit the diverging characteristics of instances from various scales, the more recent Faster-RCNN addresses the issue with a multi-scale region proposal network (RPN), which achieves excellent object detection performance. (Zhang, Li & Hu 2017, 2-4)

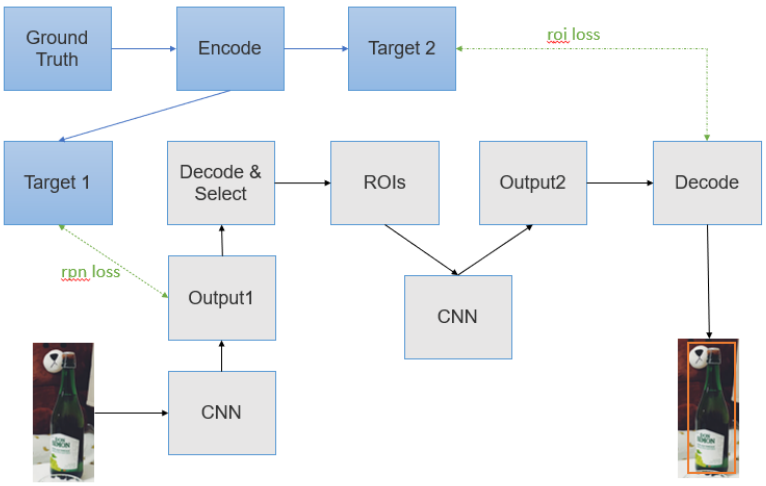


FIGURE 4. Two stage basic process

2.4 The Market For Mask Detection Systems

Amid current pandemic, Covid-19 has made people realize the importance of face masks and people need to understand the crucial effects of not wearing one, now more than ever. Right now, there are no face mask detectors installed at the crowded places So there is still a lot of demand for these kinds of systems. Some of the open source software uses OpenCV plus TenserFlow or YOLO v3. Because of the coexistence of viruses and humans is a long-term issue, and public places are needed to protect peoples' safety, so it is necessary to develop a long-term and available software. (Sakshi, Yadav, Gupta & Kumar 2021, 212-216)

3 TECHNOLOGY INTRODUCTION

The technologies that need to be used in the system include image preprocessing, machine learning, model training, target detection, image acquisition and display. With the development of machine learning and computer vision, the frameworks and models on the market tend to be developed. Among them, You Only Look Once is a viral and widely used algorithm. YOLO is famous for its object detection characteristic. After investigation, most of the technologies used in mask recognition system can be realized by YO-LO V5. (Jiang, Ergu, Liu, Cai & Ma 2022, 1066-1073)

3.1 OpenCV

With a growing and large market for computer vision and no standard API, computer vision software developed three questions today. One is that the code is slow, unstable, too independent and incompatible with other libraries. Second, that will need a variety of commercial tools, such as Halcon, MATLAB+Simulink. Third, hardware relies on AD hoc solutions, like video surveillance, manufacturing control systems, medical equipment. Third, the solution relies heavily on hardware, such as video surveillance, manufacturing control systems, and medical equipment. A standard API will simplify the development of computer vision programs and solutions. OpenCV aims to be such a standard API. OpenCV is a library of programming functions for real time computer vision. The library has more than 2500 optimized common algorithms in image processing and machine vision. (Lv, Wang, Shen 2013, 718)

OpenCV is dedicated to real world real-time applications. OpenCV has improved its execution speed by optimizing the code witten by C code, and can process faster by purchasing the Intel Integrated Performance Primitives multimedia function library. OpenCV based library makes systems easy to create due to the large amount of inbuilt functions of various image processing tasks like edge detection and feature tracking. (Shrivastava 2013, 947). OpenCV is a cross platform computer vision and machine learning software library based on apache2.0 license, which can run on Linux, Windows, Android and Mac OS operating systems. It is composed of a series of C functions and a small number of C++ classes. At the same time, it provides interfaces with Python, Ruby, MATLAB and other languages, and realizes many general algorithms in image processing and computer vision. It is lightweight and efficient. (Bradski and Kaehler 2008, 1-15)

3.2 PyTorch

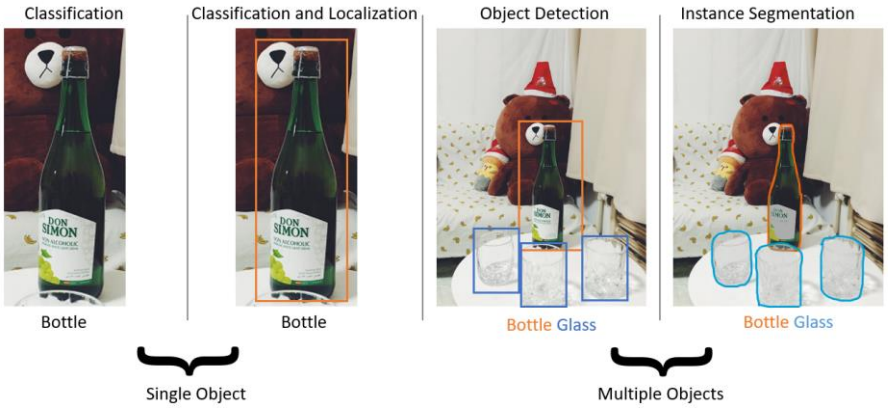
PyTorch is an open source Python machine learning library based on Torch for applications such as natural language processing. PyTorch is a machine learning framework released by Facebook in 2016. It covers two crucial issues when it comes to implementation of predictive models - tensor computations and neural network modelling (Nejkovic, Radenkovic, Petrovic 2021, 250). PyTorch is a library for Python programs that encourages deep learning programs. With this receptiveness and convenience found in (Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras. PyTorch makes it useful in developing deep neural networks. It has an expansive scope and is applied for various applications. As Python is for programming, PyTorch is both a magnificent prologue to profound learning just as an instrument usable in proficient real-world applications. (Imambi, Prakash & Kanagachidambaresan 2021, 87-104)

There is a small performance difference based on the type of artificial neural network libraries in CNN binary image classification. The accuracy of Keras is 78%. The accuracy of Pytorch is 70.8%. And the accuracy of MXnet is 68% (Kim, Wimmer & Kim 2022, 59).

3.3 Object Detection

Object detection, also called object extraction, is one of the four tasks of image recognition in computer vision. Object detection algorithms use a wide range of image processing applications for extracting the object's desired portion. It is commonly used in applications such as image retrieval, security, medical field, and military defense department. Object detection needs to be distinguished from image classification, targe location and target segmentation, which means to separate a variable number of targets from the background in a single image or a sequence of images. Because of the different appearance, shape and posture of various objects, and the interference factors such as illumination and occlusion during imaging, object detection is always the most challenging problem in the field of computer vision. At present, the applied target detection methods mainly include feature extraction based on parameter measurement, feature extraction based on non-parametric measurement, feature extraction based on Karhumen-Loeve expansion, wavelet transform feature extraction. In the application of computer vision,

target detection is a very important link, which directly determines the performance of subsequent recognition and tracking. At present, object extraction is widely used. For example, it is used to extract face features and fingerprints in computer vision, and to extract feature points and lines for image matching and 3D modeling in photogrammetry and remote sensing. (Raghunandan, Mohana, Raghav & Aradhya 2018, 563-567)



PICTURE 1. Difference between object detection and other task

3.4 YOLO

YOLO, short for "You Only Look Once," is an algorithm that uses neural networks to provide real-time object detection. It applies a single CNN to the entire image, divides the image into grids, and predicts the class probabilities and bounding boxes of each grid. For each grid, the CNN predicts a bounding box and a probability corresponding to each category. Each bounding box can be described using four descriptors, the center, height, width of the bounding box, and the class to which the object belongs. In addition, the algorithm can also predict the probability of objects in the bounding box. If the center of an object falls on a grid cell, the grid cell is responsible for detecting the object. There will be multiple bounding boxes per grid. When training, only one bounding box per object is expected. Therefore, a box is assigned to be responsible for the prediction object according to the overlap degree between box and ground truth box. Finally, the Non-Max Suppression method is applied to the objects of each class to filter out the bounding boxes whose confidence degree is less than the threshold. An image prediction is given. YOLO is very fast. YOLO considers the frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

It is 1000 times faster than R-CNN and 100 times faster than Fast R-CNN. It is capable of handling live video streams with a latency of less than 25 milliseconds. It is more than twice as accurate as previous real-time systems. Just as important, YOLO follows the practice of end-to-end deep learning. (Redmon, Divvala, Girshick & Farhadi 2016, 779-783)

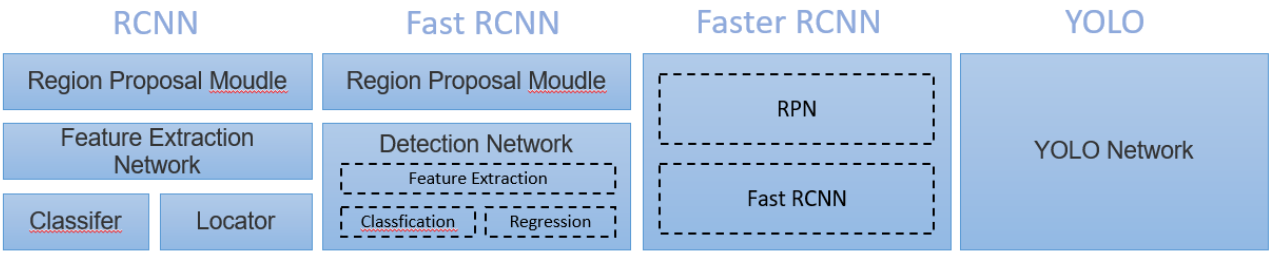
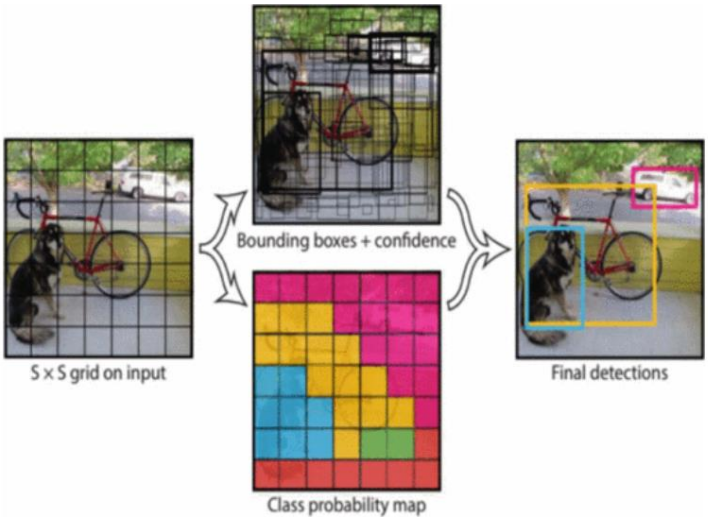


FIGURE 5. Difference between YOLO and CNN Series

3.4.1 The History Of YOLO

YOLO V1 is the beginning of the YOLO series, and its core idea is to treat target detection as a single regression task. It first divides the image into SxS grids. When the center of object real frame falls on the grid, the anchor frame of the grid is responsible for detecting the object. YOLO V1 uses the Darknet framework and ImageNet-1000 dataset to train the model. It distributes the given picture to a grid of S×S cells. For every cell in the network, it computes confidence for 'n' bounding boxes. The predicted result is encoded into a tensor as S×S×(n×5+p). Here, input image is divided into total S×S sub-images. In n×5, five represents the detection of five attributes for each bounding box, which are height, weight, confidence score, and centre coordinates (x, y) of detected objects. Here, 'p' represents the probability of class. YOLO V1 also has many limitations. Therefore, the use of YOLO V1 is restricted to some extent. Limitations of YOLO version1 are based on the closeness of the objects in the picture. If the objects appear as a cluster, they could not find the small objects. If the dimensions of the object are different from the image used in training data, then this architecture found difficulty in the localization and detection of objects. The primary concern is to locate objects in a given picture due to the error of localization. YOLO V1 fails sometimes. (Adarsh, Rathi & Kumar 2020, 687-689)



PICTURE 2. S×S grid representation of YOLO (Adarsh, Rathi & Kumar 2020)

YOLO V2 supersedes YOLO by offering a great balance between running time and accuracy. For better accuracy, YOLO V2 introduces batch normalization, which helps to enhance 2 percent in map by attaching it into each layer of convolution. High resolution classifier is used to operate thoroughly by modifying its filters for giving a more extensive understanding of network time to work excellently. When it comes to the prophecy of bounding boxes, then by eliminating entirely connected layers to anchor boxes makes decrement of map value by 0.3, but recall value is incremented by 7%. Hence, it gives more potential to detect the day-to-day objects. Fine grained features help to identify tiny objects by reshaping the layers employing a modified approach called pass through. As we have achieved accuracy now, our goal is to maintain a balance of running time with it. For this YOLO V2 uses darknet 19, although it could use Google net, but that reduces accuracy by 2%. For learning to locate targets from any visual, YOLO 9000 is employed across nine thousand categories with a 9418 node WordTree. It is certainly progress for concluding the localization and distribution. (Adarsh et al 2020, 689-690)

YOLO V3 is further optimized on YOLO V2. YOLOv3 uses a feature extractor network called Darknet 53 who is a hybrid variant of the network used in YOLOv2 Darknet-19. In addition, the optimization of YOLO V3 includes the use of multi-scale prediction, cross-scale feature fusion, and the clustering of 9 different scales of anchors on the COCO data set, 3 for each scale. YOLO V3 has the advantages of fast reasoning speed, low background error detection rate and strong versatility. However, YOLO V3 also has disadvantages. The recall rate is low. The positioning accuracy is poor. For small objects and the objects close to or blocked, the detection ability is relatively weak. On February 21, 2020, Joseph Redmon, the founder of the YOLO series of algorithms, announced on his personal Twitter account that he

would stop all CV research because his open-source algorithms have been used in military and privacy issues. This caused a great test to his morality. (Redmon & Farhadi 2017, 7263-7264)

Since then, others began to take over the improvement of the YOLO series algorithm. After that, the more famous algorithms are YOLO V4 developed by Alexey bochkovskiy and YOLO V5. YOLO V4 has made many innovations on the basis YOLO V3. For example, the input is enhanced by mosaic data. The YOLOv4 model uses CIoU loss, DropBlock regularization, CmBN, Mosaic data augmentation, Mish activation, SAT, CSP, and WRC. These changes make the YOLOv4 better and more accurate than the previous versions. (Khalili & Shakiba 2022, 3)

3.4.2 YOLO V5 And YOLO V5s

Currently, the framework of YOLO series has been gradually developed to YOLO V5, which has been further optimized and improved on previous frameworks such as V4 and V3. According to the depth of network and the width of feature map, V5 series can be divided into four models: YOLO V5s, YOLO V5m, YOLO V5l and YOLO V5x. (Zhang, Qian, Si, Zeng & Wang 2021, 1-7)

YOLO V5 has some similarities with its predessor. For instance, the architecture of it can still be divided into four parts, which are input, backbone, neck, and prediction. The main improvements of YOLO V5 lie in these four parts. In the input part, Mosiac data augmentation is used. Meanwhile, operations for data augmentation like random scaling, random cropping, and random arrangement show good benefits for the final results. Besides, YOLO V5 adds auto anchor box algorithm and adaptive black-edge padding in input part. The most important improvements of YOLO V5 lie in the backbone part. The focus structure and CSP are added to improve the feature extraction. In the neck part, the FPN and PANet are used. And the CSP2 structure are used in the neck part of YOLO V5, which is inspired by CSPnet. The training loss function of YOLO V5 is consist of CIOU loss for bounding box, classification loss and object detection loss. CIOU loss is written as: $CIOUloss=1-(IOU-d22d2c-v21-IOU+v)$, where v is written as: $v=4\pi 2$, where wgt and hgt represent the width and height of the ground truth bounding box. wp and hp mean the width and height of the predicted bounding box. The overview of YOLO V5 is shown in below figure. (Jiang 2022, 830-832)

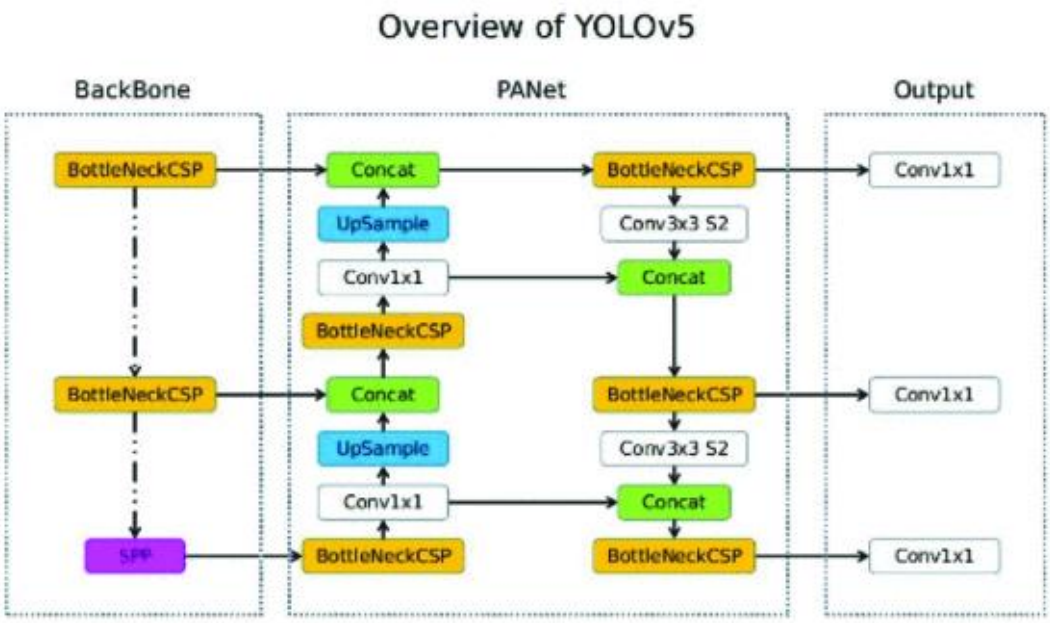


FIGURE 6. The overview of YOLO V5 (adapted from Jiang 2022)

YOLO V5 includes 4 versions which are YOLO V5s, YOLO V5m, YOLO V51, and YOLO V5x. Among them, YOLO V5s has the smallest size and a weight of only 27 MB. For YOLO V5, includes V5s, V5m, V5l or V5x, its backbone, neck and head are consistent. The only difference is the depth and width. Modifying these two parameters can adjust the network structure of the model. Depth multiple is used to control the depth of the model. For example, the depth of V5s is 0.33, while the depth of V5l is 1. In other words, the number of bottlenecks in V5l is three times that of v5s. width_ Multiple is used to control the number of convolution kernels. The width of V5s is 0.5, the width of V5l is 1, and the width of V5x is 1.25. This means that the number of convolution cores of V5s is half of the default setting, so the first layer of backbone in YOLO V5s yaml file is actually [[-1, 1, focus, [32, 3]]. If the goal is a very lightweight detection model, YOLO V5s is preferred. (Zhang, Yin, Liu, Zhou & Zhou 2021, 670)

YOLO V5s is the simplest version with the smallest weight file and has a high recognition speed. The framework of YOLO V5s is shown in Figure 7 below. There are some blocks, such as Focus, Convolution, BottleneckCSP and SPP. The Focus is to use concat function to connect the 4 slices by copying 4 copies of the input picture and then slice it. (Wu, Wang & Liu 2021, 24-26)

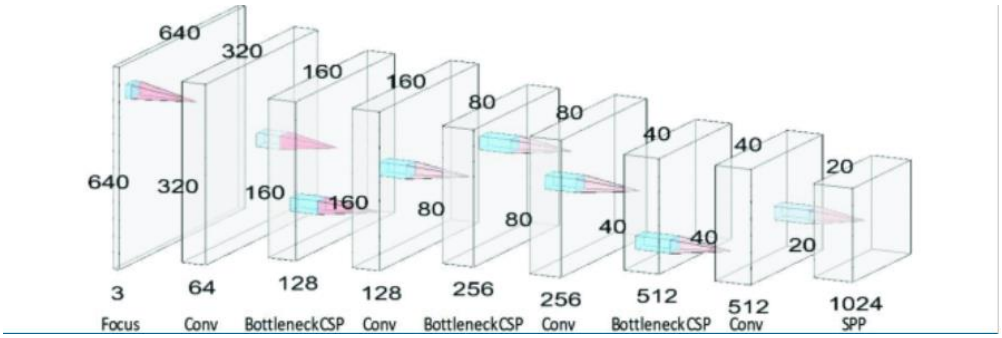


FIGURE 7. The structure of the YOLO V5s backbone (adapted from Wu, Wang & Liu 2022)

The convolution module includes a specific structure diagram composed of a convolution layer, a Batch_Norm layer, a LeakyReLU layer, as shown in below figure. The BottleneckCSP is divided into two parts, Bottleneck and CSP. The bottleneck is actually a classic residual structure, first a 1x1 Convolution layer, then a 3x3 Convolution layer, and finally, a residual structure to add to the initial input. The structure diagram of residual is also shown in Figure 8 below. (Wu et al. 2021, 25-26)

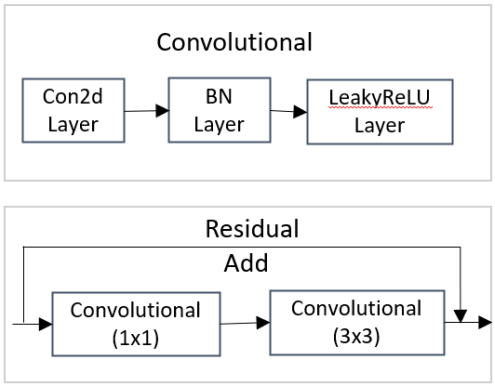


FIGURE 8. First Bottlenneckcsp structure of YOLO V5s ((adapted from Wu, Wang & Liu 2022)

The CSP structure of YOLO V5s is to divide the original input into two branches, respectively perform convolution operations to halve the number of channels, and then branch one for the Bottleneck x N operation, then use the concat function to combine branch one and branch two, so that the input and output of BottleneckCSP It is the same size, the purpose is to let the model learn more features. (Wu et al. 2021, 27-28)

The SPP structure of YOLO V5s. Take YOLO V5s as an example, the input of SPP is 512x20x20, after a 1x1 Convolution layer, 256x20x20 is output, and then it is down-sampled through three parallel Max-pool Layers, and the result is added with the initial feature to output 1024x20x20, and finally, 512 convolution kernels is used to restore it to 512x20x20. (Wu et al. 2021, 255)

In summary, YOLO V5 is more powerful than its predecessors. First, when YOLO V3 and YOLO V4 train different data sets, the calculation of the value of the initial anchor box is run by a separate program. But YOLO V5 embeds this functionality in the code. During each training, YOLO V5 adaptively calculates the optimal anchor frame value in different training sets. It is also flexible to turn off the automatic calculation anchor box in users' code if the users feel that the calculation anchor box is not working very well. Second, in common target detection algorithms, different images are different in length and width. Therefore, the common way is to uniformly scale the original image to a standard size, and then send it to the detection network. In YOLO V5 code, the letterbox function of dataset.py is modified to adaptively add the least number of black edges to the original image. The black edge at both ends of the image height becomes less, and the amount of calculation in reasoning will also be reduced, that is, the speed of target detection will be improved. This simple improvement resulted in a 37 percent increase in reasoning speed, which is pretty impressive. Third, positive samples of YOLO V5 were increased. This is the same as using multi-anchors for single ground truth in YOLO V4. Finally, Mosaic data is enhanced in YOLO V5. Cutmix only uses two pictures for splicing, while Mosaic data adopts the methods of random scaling, random cropping and random arrangement of four pictures for splicing. This can enrich data sets and reduce GPU usage. In comparison to YOLO-v3 SPP and Faster RCNN, YOLO V5 performed the best. The miss detection rate of YOLO V5 was 1.78% lower than that of YOLO V3, and it is a lighter model than YOLO V4(Udatewar, Desai, Godghase, Arunkumar & Pranali et al. 2021, 2).

3.5 PyQt

PyQT is basically a framework that provides tools for Graphical User Interface design. The PyQT framework is the python binding of the QT framework, which is very popular GUI design repository. It was developed by Phil Thompson and is a successful fusion of the Python programming language and the Qt library. The Qt library is one of the most powerful. PyQt implements a set of Python modules. It has over 300 classes and nearly 6,000 functions and methods. It is a multi-platform toolkit that runs on all major operating systems, including UNIX, Windows and Mac. (Rempt 2015, 7)

4 OVERALL SYSTEM DESIGN

The main purpose of this system is to establish a mask recognition system that runs smoothly. In recent years, many epidemic diseases have been rampant. Wearing masks can effectively stop the spread of these diseases. However, it is difficult to quickly recognize people weather wearing mask or not by staff's judgment. The system will help managers in these public places to recognize people weather wearing mask or not as accurately as possible. At the same time, the availability, cost and maintainability of the system should be considered before development.

4.1 System Requirements Analysis

Novel Coronavirus, scarlet fever, monkeypox and other infectious diseases ravage the world, and at present there is no specific cure. Wearing masks can effectively cut off the transmission of these diseases and is an effective measure to prevent infection. However, some people cannot wear masks in real time due to special circumstances, which may cause hidden dangers to the safety of people around them. In particular, shopping malls, subways, schools and other personnel intensive places, If the management of public places only relies on staff to visually check the wearing of masks, it will result in a huge waste of human resources. With time passing, computer vision technology enables human to integrate the image captured by the camera with computer analysis and processing. This allows people to use the machine to quickly and efficiently detect whether people are wearing masks. This system can automatically, accurately and quickly identify whether people in videos or pictures are wearing masks, which has important application scenarios. (Sakshi, Gupta, Yadav & Kumar 2021, 212-216)

4.1.1 Performance Requirement

First, the system provides round-the-clock service under normal Internet conditions, and users can use the system for detection at any time. Second, the detection speed should be fast and the detection result should be as accurate as possible. Third, the security, operability and maintainability of the system should be thorough.

4.1.2 Operation Requirements

The system adopts YOLO V5 model, Pytorch deep learning framework and COCO data set. The operating system is Windows 10. These technologies have been very mature at present. In the case of the emergence of a large number of high-performance network, they still reflect the advantages of fast speed, large amount of consumption, high reliability and low price. They are perfectly suited to the needs of the system.

4.2 System Function Design

When using this system, users can detect in two ways, real-time or upload local pictures and videos. After preprocessing the image, the system locates the target the region containing the face. Then, the region will be positioned, and recognized whether to wear a mask. Finally, the recognition results will be the output.

According to the above description, the system should have the following three functions. One is real-time detection. The system uses a camera to obtain real-time images of the scene. The second is the picture and video files detection. The image will be multi-step processed in this function, such as image feature extraction, target location and target recognition. Third, image or video detection result will be displayed. The fourth function is to output results. This function can output the result of the objection recognition that has been recognized.

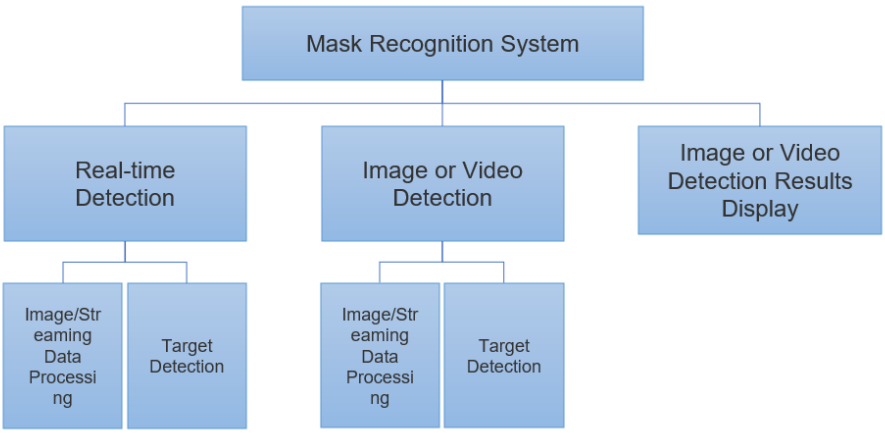


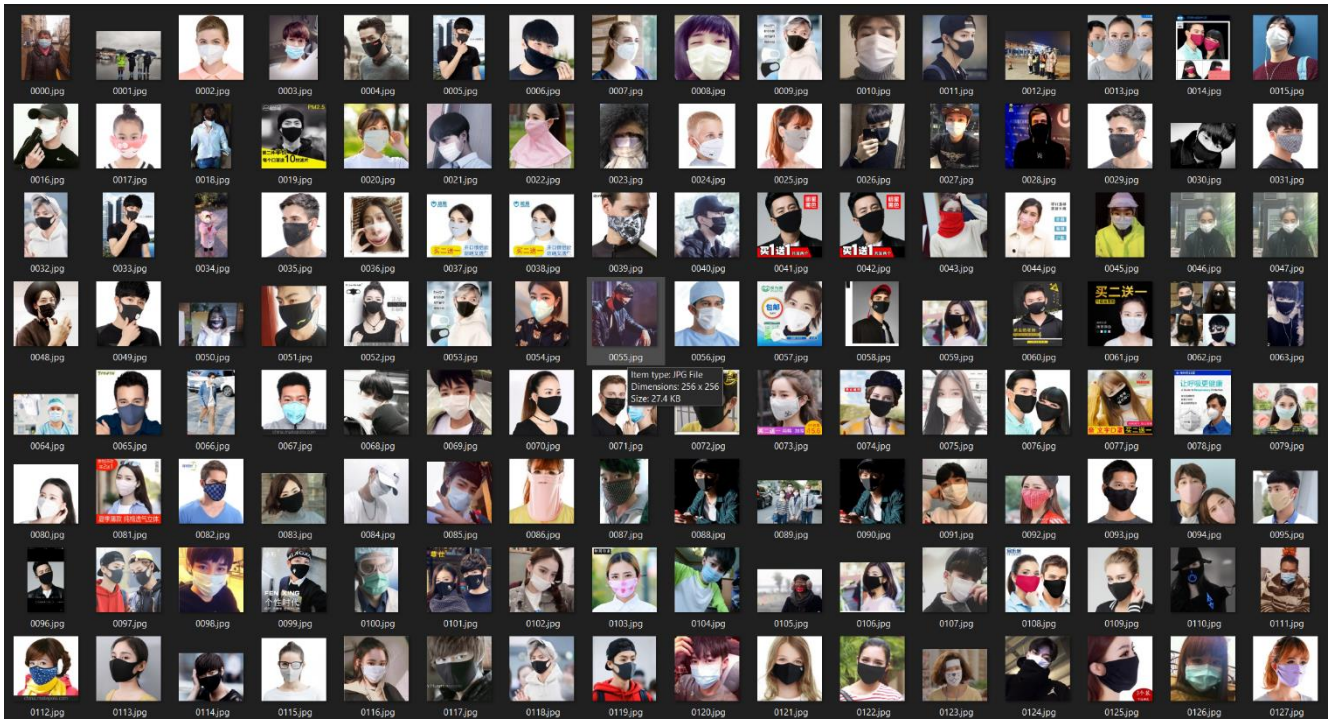
FIGURE 9. System function structure diagram

5 SYSTEM IMPLEMENTATION

The first step is to install YOLO V5, PyTorch, and environment dependencies. Second, 6000 photos with masks and faces were taken from a public data set. The number of people, the angle, the color of the mask in the picture should be varied. And, the next step is to train the model. Training data requires the configuration of operating parameters and the consolidation of data sets. After the model is trained, the effect of the model needs to be tested. Real-time detection captures stream data through the camera and converts the stream data into detectable frames.

5.1 Data Preparation

About 6000 images were collected from a public data set, and as mask wearing detection data set. At the meantime, the number of images with and without masks were basically the same. Sample images will be labelled in two categories, one masked and one no_mask. There are 4000 different images in the training set, and 1000 different images in the validation set. Different type of data should be prepared, such as different angles or different colors of masks. Such training is conducive to improving the accuracy of the detection of wearing mask.



PICTURE 3. Part of pictures from dataset

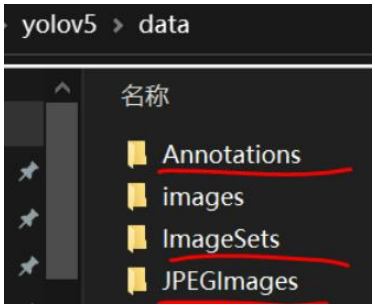
5.2 Model Training

In order to ensure that the model can still maintain the detectability of calibration after training, real faces are selected as training samples for the detection of this system. The collected pictures were classified and faces were marked masked or no_mask. The location annotated box uses the results from the public dataset.

5.2.1 Data Set Marking

Annotations of the data have been obtained from external datasets. These images, obtained from a third party, were tagged as masked or no_mask. The format of data annotation file is XML. With all these resourse, the next step is to create a dataset. Training data requires the configuration of operating parameters and the consolidation of data sets. The dataset needs to be configured before model training.

First of all, the data folder needs to be modified. The files under data are used to configure the training set, test set, and validation set path, which are the hyperparameters configuration files. It also includes the number of classes to configure object detection and the class name. Firstly, the folder JPGImages, Annotations, and ImageSets under data should be created. Place all images in the JPGImages folder. Place all xml files under the Annotations folder. Next, the make_txt.py file in the root directory should be created. The purpose is to generate two dataset classification txt files train and val in ImageSets folder. File tain.txt should contains the training picture path. The path to the test image should be set in val.txt. And then, the label.py file created in the root directory need to run to convert the xml file marked with image information to txt format. In addition, three files train, val and test in txt format appeared under the data file to save the path of the picture. At this point, the data set is all done.



PICTURE 4. The folder data

5.2.2 Configuration File Modification

Next, configuration files, including coco.yaml and model.yaml files, need to be modified. Since this system is not using coco datasets, a separate dataset yaml file needs to be configured. Copy coco-yaml from the data directory and name it mask.yaml. Then make three changes to mask.yaml. First, change the path of the train, test and val files to the path generated in the previous step. In the second place, the number in nc represents the category of the data set, which is changed to 2. Third, annotate the name of the class for the dataset in the names field, masked and no_mask. Then, the model.yaml file needs to be modified. There are four models, 5s, 5m, 5l, 5x, under directory models. This system uses YOLO V5s. Thus, nc in YOLOv5s.yaml needs to be modified into 2.

```yaml
# tarin and val data as 1) directory: path/images/, 2)
train: data/train.txt
val: data/val.txt
test: data/test.txt

# number of classes
np: 2

# class names
names: ['masked', 'no_mask']
```

PICTURE 5. Data set configuration file modification

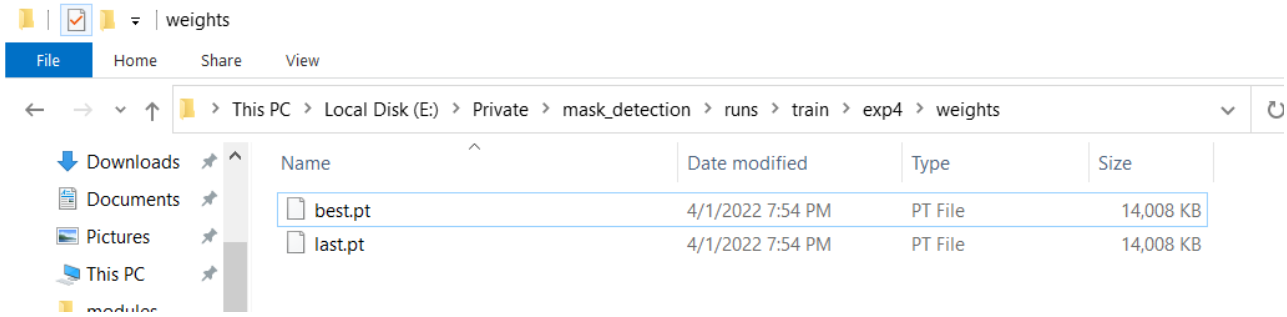
5.2.3 Model Training

The last step is to train the model. The path of weights, yaml, and data need to be modified into the paths of YOLO V5s.yaml and mask.yaml. Field epochs is the number of iterations and can be customized. In this training, 125 training sessions were conducted. Batch-size specifies the number of images that can be processed at a time. Here, batch-size is changed to 32 according to the conditions of computer hardware. Then, run train.py.

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'best.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/mask.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=125)
    parser.add_argument('--batch-size', type=int, default=32, help='total batch size for all GPUs')
```

PICTURE 6. File train.py

After the training is complete, the corresponding exp directory is generated under the YOLO V5 root directory. The weight files are best.pt and last.pt, respectively. The best.pt contains the weights with the best performance, which can be used for the detection of the corresponding dataset. The last.pt represents the weight of the last epoch. Save the files separately. Otherwise, the files may be replaced in subsequent operations. The best.pt file needs to be copied to the root directory of YOLO V5.



PICTURE 7. Weights file

At this point the training of the model is complete. In the end, the training took 0.557 hours totally. After the training, the data of all models will be saved automatically. And the weight files best.pt and last.pt have been saved in path mask_detection/runs/train/exp4/weights. YOLO V5 does a great job of calling up a number of libraries for visualization. Thus, the users can easily see the results of training.

5.2.4 Model Test

At the end of train.py, when the model has been trained, val.py is called to test the model. Set noval to false in the run method. After setting, run model tests only once after model training is complete. Set data to the path of mask.yaml and use the trained weights file best.pt for weights. Adjust other parameters appropriately. After the test is complete, view evaluation indicators such as mAP to view test results.

```
def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--data', type=str, default=ROOT / 'data/mask.yaml', help='dataset.yaml path')
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'best.pt', help='model.pt path(s)')
    parser.add_argument('--batch-size', type=int, default=32, help='batch size')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='inference size (pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.001, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.6, help='NMS IoU threshold')
    parser.add_argument('--task', default='val', help='train, val, test, speed or study')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
```

PICTURE 8. val.py file

5.2.5 Problems Encountered In Training

In the early stage of training, there are some warnings. Presumably it was due to the lack of tool library. However, it does not affect the normal operation of the project. Therefore, there is no error when running the program for the first time, but the training state cannot be entered after the hyperparameters are displayed. After troubleshooting, the problem was found to be caused by the large batch_size setting. At the beginning, batch_size is set to 64, but due to device hardware limitations, the program froze. This problem was solved when batch_size was modified. Afterwards, it was also found that a pathname in train.py was incorrectly written. The weights/YOLOv5s.pt was corrected as weight/YOLOv5s.pt. After the path is changed, the tool runs properly.

6 IMPLEMENTATION EFFECT

The image below is labels.jpg. The first figure shows the amount of data in the training set. The second diagram is the labels display. The third graph is (x, y) of center. The fourth figure shows the height and width of the label.

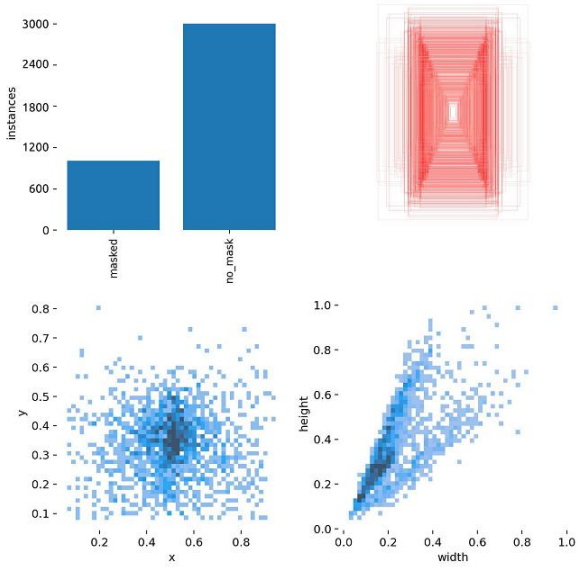


FIGURE 10. labels.jpg file

The image below is results.png. YOLO V5 uses GIOU loss as the loss of bounding box, and box_loss is the mean value of GIOU loss function. The smaller the value is, the more accurate the box is. Graph obj_loss is the mean value of target detection loss. The smaller the value, the more accurate the target detection. Graph cls_loss indicates the average value of loss classification. The smaller the value is, the more accurate the classification is. Graph precision is the precision of all the positive classes found divided by all the positive classes found. Graph recall is the accuracy of positive, that is, how many positive samples have been found. From the perspective of real results, recall describes how many real positive examples in the test set are selected by the binary classifier, that is, how many real positive examples are recalled by the binary classifier. Graph val box_loss is the verification set bounding box loss. Graph val obj_loss detects the mean value of loss for the validation set target. Graph val cls_Loss indicates the average value of loss classification in the verification set. Graph mAP refers to the area surrounded by precision and recall as two axes, and m represents the average, the number after _ represents the threshold for determining IOU as positive and negative samples, and _0.5:0.95 represents the

mean value after the threshold is set to 0.5:0.05:0.95. Graph mAP_0.5 represents the average mAP with a threshold greater than 0.5. Graph mAP_0.5:0.95 represents the average mAP at different IOU thresholds, i.e., from 0.5 to 0.95, step size 0.05.

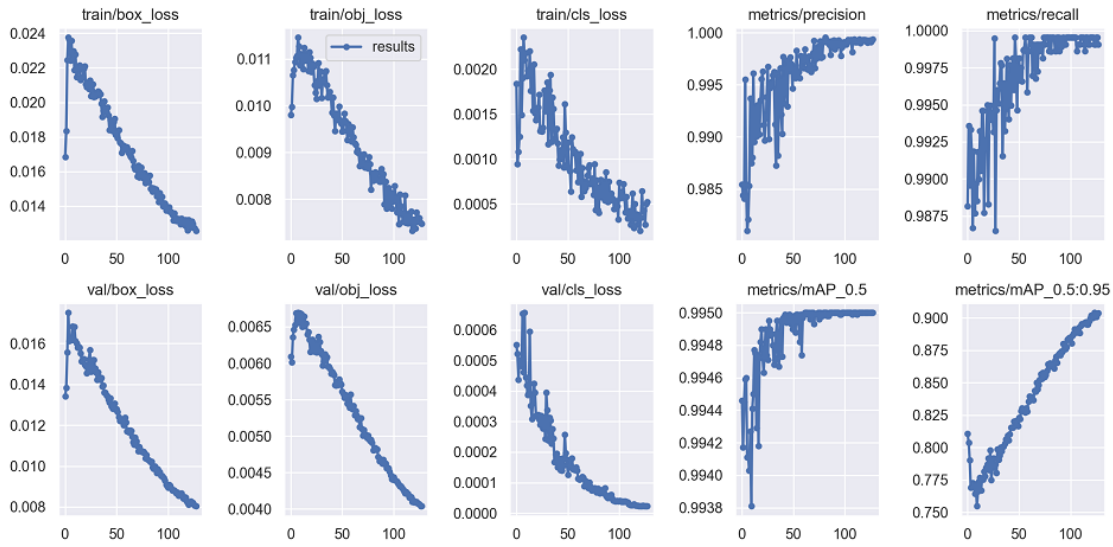
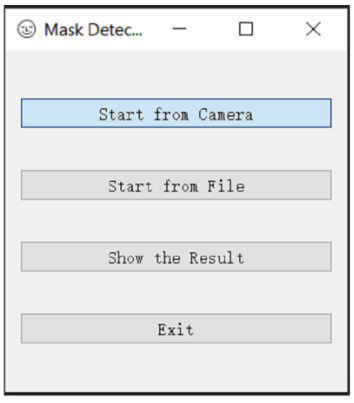


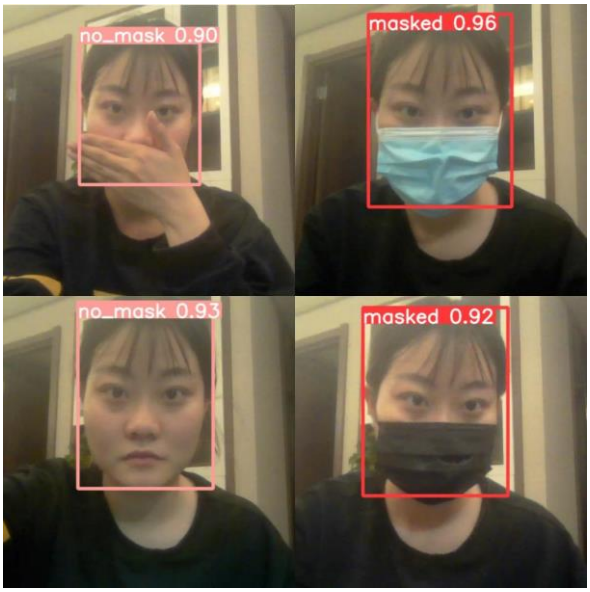
FIGURE 11. results.png

Run the program, and a simple program window made with Qt will pop up. Users can select functions including real-time detection, picture or video detection, and presentation of picture or video detection results. Click the button "Start from Camera" to open the camera and display the image and detection results in real time from the window. Click the button "Start from file" to detect all files in the directory data/images. The detected result will be in runs/detect/exp. Users can access these results through the "Show the Results" button. When the user wants to end the program, the user can exit by clicking the Exit button.



PICTURE 9. Program window

After the final experiment, the result is shown in the figure below. For those who wear masks, the system selects their faces with red boxes and writes masked and the probability. For those who did not wear a mask, a pink box was used to mark the face with the word no_mask and the probability. For the case of occlusion, different angles and different styles of masks, the test results of this model are good. However, when the detected object is too small, the effect is not very ideal. This is the bottleneck of YOLO V5s.



PICTURE 10. Result of real-time detection

7 CONCLUSION

This system uses YOLO V5 deep convolutional network to conduct data processing training on mask wearing detection data set. After repeated training on the data set, the detection of mask wearing has achieved ideal results. This shows that the experiment and the method is valuable. However, some problems were found during testing. For example, the system may fail to recognize the detection object if it is too small or too dense. Detection in this case is not ideal. In addition, there is a short time delay in real-time detection. The subsequent improvement will proceed from these points to further improve network design, improve detection accuracy, and improve detection efficiency and performance.

REFERENCES

Adarsh, P., Rathi, P., Kumar, M., 2020. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model, in: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, pp. 687–694.

Adusumalli, H., Kalyani, D., Sri, R. K., Pratapteja, M., & Rao, P. P., 2021. Face Mask Detection Using OpenCV. In: Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021). IEEE, pp.1304-1309.

Deng, D., 2021. Image Recognition Based on Sparse Enhancement and Collaborative Filtering Algorithm. In: 2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE). IEEE, pp.2426-2429.

Dinesh, S., & Vairavel, G., 2022. VLSI implementation of reconfigurable processing modulefor binary and grayscale image processing. In: 2014 International Conference on Electronics and Communication Systems (ICECS). IEEE, pp.1-4.

He, H., 2022. Yolo Target Detection Algorithm in Road Scene Based on Computer Vision, in: 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC). IEEE, pp. 1111–1114.

Jiang, P., Ergu, D., Liu, F., Cai, Y., Ma, B., 2022. A Review of Yolo Algorithm Developments. Procedia Computer Science 199, 1066–1073. https://doi.org/10.1016/j.procs.2022.01.135

Jiang, Y., 2022. COTS recognition and detection based on Improved YOLO v5 model, in: 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP). IEEE, pp. 830–833.

Jordan, M. I. & Mitchell, T. M., 2015. Machine learning: Trends, perspectives, and prospects. Science. 349(6245), 255–260. DOI: 10.1126/science.aaa8415.

Kaehler, A., Bradski, G., 2016. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. “O’Reilly Media, Inc.”

Khalili, S., Shakiba, A., 2022. A face detection method via ensemble of four versions of YOLOs, in: 2022 International Conference on Machine Vision and Image Processing (MVIP). IEEE, pp. 1–4.

Kim, S., Wimmer, H., Kim, J., 2022. Analysis of Deep Learning Libraries: Keras, PyTorch, and MXnet, in: 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA). IEEE, pp. 54–62.

Kotsiantis, S., Zaharakis, I. and Pintelas, P., 2006. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, 26(3), pp.159-190.

Krizhevsky, A., Sutskever, I. and Hinton, G., 2017. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), pp.84-90.

Kumar, G., Bhatia, P.K., 2014. A Detailed Review of Feature Extraction in Image Processing Systems, in: 2014 Fourth International Conference on Advanced Computing & Communication Technologies. IEEE, pp. 5–12.

Liu, Z., Deng, Y., Ma, F., Du, J., Xiong, C., Hu, M., Zhang, L., Ji, X., 2021. Target detection and tracking algorithm based on improved Mask RCNN and LMB, in: 2021 International Conference on Control, Automation and Information Sciences (ICCAIS). IEEE, pp. 1037–1041.

Lü, C., Wang, X., Shen, Y. 2013. A stereo vision measurement system Based on OpenCV, in: 2013 6th International Congress on Image and Signal Processing (CISP). IEEE, pp. 718–722.

Mahesh, B., 2020. Machine Learning Algorithms - A Review. International Journal of Science and Research (IJSR) 9, 381–386.

Mane, S., Mangale, S., 2018. Moving Object Detection and Tracking Using Convolutional Neural Networks, in: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, pp. 1809–1813.

Moru, D.K., Borro, D., 2019. A machine vision algorithm for quality control inspection of gears. The International Journal of Advanced Manufacturing Technology 106, 105–123. https://doi.org/10.1007/s00170-019-04426-2

Nejkovic, V., Radenkovic, M., Petrovic, N., 2021. Ultramarathon Result and Injury Prediction using PyTorch, in: 2021 15th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS). IEEE, pp. 249–252.

Nti, I., Quarcoo, J., Aning, J. and Fosu, G., 2022. A mini-review of machine learning in big data analytics: Applications, challenges, and prospects. Big Data Mining and Analytics, 5(2), pp.81-97.

Park, S., Jeong, S., Park, Y., Kim, S., Lee, D., Mo, Y., Jang, D. and Park, K., 2021. Phenological Analysis of Sub-Alpine Forest on Jeju Island, South Korea, Using Data Fusion of Landsat and MODIS Products. Forests, 12(3), p.286.

Prakash, K.B., Kanagachidambaresan, G.R., 2021. Programming with TensorFlow: Solution for Edge Computing Applications. Springer Nature.

Raghunandan, A., Mohana, Raghav, P., Aradhya, H.V.R., 2018. Object Detection Algorithms for Video Surveillance Applications, in: 2018 International Conference on Communication and Signal Processing (ICCSP). IEEE, pp. 563–568.

Rani, P., Kotwal, S., Manhas, J., Sharma, V. and Sharma, S., 2021. Machine Learning and Deep Learning Based Computational Approaches in Automatic Microorganisms Image Recognition: Methodologies, Challenges, and Developments. Archives of Computational Methods in Engineering, 29(3), pp.1801-1837.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788.

Redmon, J., Farhadi, A., 2017. YOLO9000: Better, Faster, Stronger, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 6517–6525.

Rempt, B., 2001. GUI programming with python. Oregon: Command Prom

Sakshi, S., Gupta, A.K., Singh Yadav, S., Kumar, U., 2021. Face Mask Detection System using CNN, in: 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). IEEE, pp. 212–216.

Shrivastava, R., 2013. A hidden Markov model based dynamic hand gesture recognition system using OpenCV, in: 2013 3rd IEEE International Advance Computing Conference (IACC). IEEE, pp. 947–950.

Udatewar, P., Desai, A., Godghase, G., Nair, A., & Kosamkar, P., 2021. Personal Protective Equipment Kit Detection Using Yolo V5 And Tensorflow. In: 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON). IEEE, pp.1-8.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., 2018. Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience 2018, 1–13. https://doi.org/10.1155/2018/7068349

Wu, T. H., Wang, T. W., & Liu, Y. Q., 2021. Real-time vehicle and distance detection based on improved yolo v5 network, in: 2021 3rd World Symposium on Artificial Intelligence (WSAI). IEEE, pp. 24–28.

Yun, G., Oh, S. and Shin, S., 2021. Image Preprocessing Method in Radiographic Inspection for Automatic Detection of Ship Welding Defects. Applied Sciences, 12(1), p.123.

Zhang, L., Yin, L., Liu, L., Zhuo, R., Zhuo, Y., 2021. Forestry Pests Identification and Classification Based on Improved YOLO v5s, in: 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS) . IEEE, pp. 670–673.

Zhang, X., Li, B., & Hu, H. 2017. Scale-aware hierarchical loss: A multipath RPN for multi-scale pedestrian detection, in: 2017 IEEE Visual Communications and Image Processing (VCIP). IEEE, pp. 1–4.

Zhang, X., Qian, B., Si, H., Zeng, L., Wang, H., 2021. Research on the computer intelligent recognition of electric appliance by Yolo V5 algorithm. Journal of Physics: Conference Series 2083, 032078. https://doi.org/10.1088/1742-6596/2083/3/032078

Zhang, Y., Fan, Q., Bao, F., Liu, Y. and Zhang, C., 2018. Single-Image Super-Resolution Based on Rational Fractal Interpolation. IEEE Transactions on Image Processing, 27(8), pp.3782-3797.

Zou, Z., Shi, Z., Guo, Y. and Ye, J., 2022. Object Detection in 20 Years: A Survey. [online] arXiv.org. Available at: <https://arxiv.org/abs/1905.05055v1> [Accessed 14 September 2022].

APPENDIX 1
Part of core code

```
import os
import sys
import torch
import cv2 as cv2

from PyQt5.QtWidgets import QPushButton, QVBoxLayout, QWidget, QApplication
from detect import run
from PyQt5.QtGui import QIcon

if torch.cuda.is_available():
    dev = '0'
else:
    dev = 'cpu'
run_dict = {
    'weights': 'best.pt',
    'source': 0,
    'imgsz': [640, 640],
    'conf_thres': 0.60,
    'iou_thres': 0.40,
    'max_det': 10,
    'device': dev,
    'view_img': False,
    'save_txt': False,
    'save_conf': False,
    'save_crop': False,
    'nosave': True,
    'classes': None,
    'agnostic_nms': False,
    'augment': False,
    'visualize': False,
    'update': False,
    'project': 'runs/detect',
    'name': 'exp',
    'exist_ok': False,
    'line_thickness': 3,
    'hide_labels': False,
    'hide_conf': False,
    'half': False,
}

run_dict_file = {
    'weights': 'best.pt',
```

```
    'source': 'data/images',
    'imgsz': [640, 640],
    'conf_thres': 0.60,
    'iou_thres': 0.40,
    'max_det': 10,
    'device': dev,
    'view_img': False,
    'save_txt': False,
    'save_conf': False,
    'save_crop': False,
    'nosave': False,
    'classes': None,
    'agnostic_nms': False,
    'augment': False,
    'visualize': False,
    'update': False,
    'project': 'runs/detect',
    'name': 'exp',
    'exist_ok': False,
    'line_thickness': 3,
    'hide_labels': False,
    'hide_conf': False,
    'half': False,
}

class WindowClass(QWidget):
    def __init__(self,parent=None):
        super(WindowClass, self).__init__(parent)
        self.setWindowTitle('Mask Detection')
        self.setWindowIcon(QIcon('icon.png'))
        self.btn_1 = QPushButton("Start from Camera")
        self.btn_2 = QPushButton("Start from File")
        self.btn_3 = QPushButton("Show the Result")
        self.btn_4 = QPushButton("Exit")


        self.btn_1.setCheckable(True)
        self.btn_1.toggle()

        self.btn_1.clicked.connect(lambda :self.wichBtn(self.btn_1))
        self.btn_2.clicked.connect(lambda :self.wichBtn(self.btn_2))
        self.btn_3.clicked.connect(lambda :self.wichBtn(self.btn_3))
        self.btn_4.clicked.connect(lambda :self.wichBtn(self.btn_4))

        self.resize(300,300)
        layout=QVBoxLayout()
        layout.addWidget(self.btn_1)
```

```
        layout.addWidget(self.btn_2)
        layout.addWidget(self.btn_3)
        layout.addWidget(self.btn_4)

        self.setLayout(layout)

    def wichBtn(self,btn):
        print("The button clicked is：" , btn.text())
        if btn.text() == 'Exit':
            sys.exit()
        if btn.text() == 'Start from Camera':
            run(**run_dict)
        if btn.text() == 'Show the Result':
            path = os.getcwd() + r'\runs\detect'
            self.showResult(path)
        if btn.text() == 'Start from File':
            run(**run_dict_file)

    def showResult(self,path):
        paths = os.listdir(path)
        last_folder_path = os.path.join(path,paths[len(paths)-1])
        files = os.listdir(last_folder_path)
        for name in files:
            if any(name.endswith(extension) for extension in ['.png', '.jpg', '.jpeg', '.PNG', '.JPG', '.JPEG']):
                image = cv2.imread(os.path.join(last_folder_path,name))
                self.cv_show('result',image)

    def cv_show(self, name, img):
        cv2.imshow(name, img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    win = WindowClass()
    win.show()
    sys.exit(app.exec_())
```