Acadlore Transactions on AI and Machine Learning
https://www.acadlore.com/journals/ATAIML

Mask Wearing Detection Based on YOLOv5 Target Detection Algorithm under COVID-19

Jiuchao Xie[1], Rui Xi[2*], Daofang Chang[3]

[1] Institute of Logistics Science and Engineering, Shanghai Maritime University, 201308 Shanghai, China
[2] College of Mechanical and Electronic Engineering, Shandong Agricultural University, 271018 Taian, China
[3] College of Logistics Engineering, Shanghai Maritime University, 201308 Shanghai, China

[*] Correspondence: Rui Xi (xirui@sdau.edu.cn)

Received: 07-12-2022 **Revised:** 08-14-2022 **Accepted:** 09-02-2022

Citation: J. C. Xie, R. Xi, and D. F. Chang, "Mask wearing detection based on YOLOv5 target detection algorithm under COVID-19," *Acadlore Trans. Mach. Learn.*, vol. 1, no. 1, pp. 40-51, 2022. https://doi.org/10.56578/ataiml010106.

Abstract: Deep learning methods have been widely used in object detection in recent years as a result of advancements in artificial intelligence algorithms and hardware computing capacity. In light of the drawbacks of current manual testing mask wearing methods, this study offers a real-time detection method of mask wearing status based on the deep learning YOLOv5 algorithm to prevent COVID-19 and quicken the recovery of industrial production. The algorithm normalizes the original dataset, before connecting the data to the YOLOv5 network for iterative training, and saving the ideal weight data as a test set. The training and test results of the suggested approach are presented visually on a tensor board. With the help of cameras, this technique can collect faces, identify masked faces, and present prompts for mask use. According to experiment results, the suggested algorithm can match the requirements of real-world applications and has a high detection accuracy and good real-time performance.

Keywords: Algorithm; COVID-19; Real-time; Mask; YOLOv5

1. Introduction

In recent years, face recognition has become crucial to computer vision and digital image processing. The use of artificial intelligence technology is widespread. The impact of computer vision and natural language processing algorithms is notable when compared to the real effects of existing products, and is superior to existing manual or machine learning approaches. Target identification algorithms are employed in a variety of fields, including aerospace detection, traffic safety, and industrial equipment product detection, to name but a few [1, 2]. These are the primary application of computer vision algorithms. Visual algorithms perform better in situations involving several targets, a vast area, and multiple overlaps than conventional target identification and recognition techniques [3].

The rapid outbreak of a new pneumonia epidemic has gravely disrupted people's daily lives. In contrast to typical influenza, this pandemic can spread through human saliva and has a potent infection rate. Currently, China requires people to wear masks when using public transportation (trains, subways, planes, etc.) and in places where people congregate (shopping malls, hospitals, farmers' markets, etc.) due to the global spread of COVID-19, which puts the lives and property of 7 billion people in danger [4-7]. By preventing the virus from spreading over the initial barrier and isolating human saliva and airflow, wearing a mask can successfully protect both you and others. Manual inspection is the primary means of determining whether the personnel is wearing masks [8]. The staff will manually measure the temperature and oversee the use of masks by the passengers. During the busiest times of the day, detection is frequently overloaded. Much labor is wasted, and is unappreciative and ineffective. Often, human resources are unable to handle the high load. Some inspectors are stationed at the fixed entrances and exits, but not 24 hours a day [9-11]. In regions with a large people flow, the inspectors are very likely to overlook some passengers, which undermines the prevention and management of the epidemic.

This research decides to take the well-known YOLOv5 target detection algorithm as the fundamental algorithm, trying to determine whether the face is wearing a mask accurately and in real-time. Face data were randomly obtained from the Internet, re-labeled, and then proofread through labeling. The trained model may be quickly and readily deployed on mobile devices in addition to having great recall and accuracy. The camera can be used to gather data, detect data, and swiftly judge the mask wearing situation. The proposed method is crucial to the current effort to manage the epidemic, and worthy of investment.

A few domestic academics are now researching mask wear detection. To detect the wearing of masks, Huangxiao Deng proposed using transfer learning and the retina face network, and achieved an average precision (AP) of 86.5% on the validation set. Junjie Xiao employed the YCrCb and YOLOv3 techniques. The accuracy rate of wearing recognition was 82.5%, while the AP for mask detection was 89%. A mask identification approach was proposed by Zuodong Niu et al. to enhance the performance of the retina face network in natural scenes. The test results demonstrate the method's effectiveness in detection. The YOLOv4 approach, which has the best detection accuracy and speed and meets the detection needs of wearing masks in most situations, was employed by Junlin Guan et al. The lightweight CNN mask detection method was proposed by Wang et al. [12]. They also examined the pyramid box Lite model, the keras model based on the SSD algorithm, and the mask detection model based on the center face. After testing and analyzing these three models, they provided the benefits, drawbacks, and applicability of various approaches. The pyramid box algorithm of the Baidu company, the DFS algorithm of the Didi company, and the mask detection of Huawei's model arts platform were all studied by Zhengyu Xie et al., with the aim to offer a mask wearing detecting technology suitable for rail transit stations.

With the speedy advancement of technology, our cameras and computers are now able to combine, analyze, and process data to rapidly and effectively determine whether people are wearing masks. This study proposes a mask wearing detection algorithm based on YOLOv5, in light of the research above, with the goal of addressing the challenge of recognizing occluded targets and small targets. The obtained image is first normalized, after which the YOLOv5 model is trained on the training image to produce the best network weights. Then, the test image is identified, and thoroughly analyzed. Experimental results show that the algorithm is more successful and practical than the approach suggested by the aforementioned researchers in terms of detection speed and accuracy.

Users can utilize the proposed system for detection at any time, which offers 24-hour service under typical Internet conditions. The detection is as quick and precise as possible. The system is flawless in terms of safety, usability, and maintenance. The system can rapidly, correctly, and automatically determine whether people are wearing masks in images or videos, which has significant commercial potential and application possibilities.

2. YOLOv5

YOLO is simple to implement from the perspective of its principle. Many other algorithms can currently find targets, but they consume up too much resources [13-15]. It is difficult for some embedded devices to keep up with the demand. YOLO is frequently employed in large, quickly completed projects. In addition to images, it can also recognize videos in real time [16-18]. The original one-stage target detection algorithm of YOLO overcomes the disadvantages of the traditional two-stage target detection algorithm. It can complete classification and positioning of the targets in one step.

A continuous iterative method is adopted by the YOLO series target detection algorithms. Modern technologies have contributed to the YOLO series' advancement into YOLOv5 [19, 20]. The two most well-known of YOLO algorithms are YOLOv3 and YOLOv5. In May 2020, Ultralytics LLC proposed YOLOv5, which can process 140 frames per second and reason about images in as little as 0.007 seconds. This is fast enough for real-time video image identification. The construction is more compact than previous YOLO algorithms. YOLOv5's weight data file, which is 27 MB in size, is 1/9 the size of YOLOv4's. YOLOv5's extremely little weight file, which can be transported on mobile devices with lesser setup, is one of its benefits.

YOLOv5 can quickly train its own data sets and is incredibly user-friendly using the PyTorch framework. This framework is simpler to implement than the Darknet framework used by YOLOv4. A lot of computer vision technology is incorporated into the code, making it simple to read and excellent for learning and referencing. The environment can be configured with ease, the model can be trained fast, and batch reasoning can deliver findings in real time. It is capable of accurately predicting the input of a single image, a batch of images, video, and even a webcam port. PyTorch weight files can be readily converted to the onxx format used by Android, where they can subsequently be converted to OpenCV format or, via Core ML, to iOS format where they may then be utilized directly in mobile applications. YOLOv5 has a remarkable target identification speed that can reach 140 frames per second.

2.1 Network Structure

Overall, YOLO series algorithms are composed of input, backbone, neck and prediction.

The four network structures of the YOLOv5 series are the YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The YOLOv5s network is the smallest of the four. While it has slightly lesser accuracy than the other three, it offers the fastest detection speed. Figure 1 compares the performance of YOLOv5 networks. Compared with YOLOv5s, the other three networks continue to deepen and extend, which improves accuracy while marginally slowing down detection speed. This study chooses the YOLOv5s target detection model as the fundamental algorithm, which should be deployed to mobile or embedded terminals and should have a high detection speed.

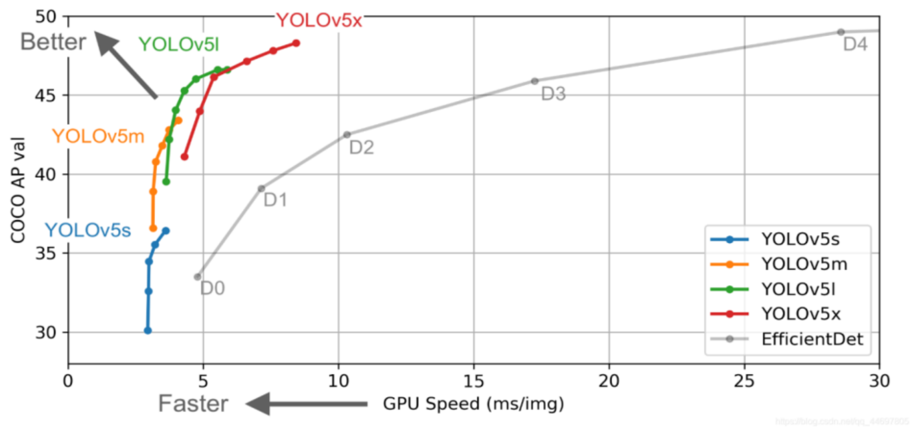


Figure 1. Performance comparison of YOLOv5 networks

2.2 Input Terminal

The input terminal adopts the mosaic data enhancement technique, adaptive anchor box and adaptive image scaling. The effectiveness of small target detection is considerably enhanced by mosaic data enhancement, through the operations of random scaling, random clipping, and random arranging.

The main principle is to randomly cut a selected image and three random images, and splice them into a training image. This can enhance the image background and combine the four images to increase batch size. Four images will also be calculated during the batch normalization, enabling YOLOv5 to batch itself.

YOLOv5 features anchor boxes with initial length and width for various datasets. There are a number of preset borders. The construction of the training samples—that is, the labels we apply—according to the offset of the actual border location from the preset boundary occurs during the training process. This is analogous to the preset boundary "framing" the target in its potential position before adjusting in accordance with those preset borders.

The anchor box is defined as follows: Each box is described by the height and width of the border. One may initially think that this anchor box is not fixed. Numerous points can be created using the box on the image. The initial anchor box does not need to indicate the central location because we already have a central point, which is the point of the feature map produced by the subsequent network. The model's final effect is directly adjusted by choosing the right anchor.

This method is integrated in the code, hence V5 is not fixed for V3 and V4. The best anchor box value from several training sets will be determined adaptively for each training.

The image size is typically varied, but when we want to start the network training, we need to make sure that it is constant. But if we solely utilize resizing, our results will be impacted by image distortion.

Consequently, a more effective technique—letterbox adaptive image scaling—is used. The size of the image is consistently maintained by the train. It creates a single, huge image the same size by combining the parts of the four images.

The missing image is filled with gray edges to create a fixed size via letterbox adaptive image, which tries to maintain the aspect ratio.

2.3 Backbone Network

The backbone network employs the focus structure and CSP structure.

The focus module's input channel has been increased by four times in YOLOv5 to increase computational power without compromising data. Block slicing is initially applied to the feature map, after which the results are concatenated before being sent to the subsequent modules. YOLOv5's most recent version switches out the focus module for a 6*6 convolution layer. Although the two requiring the same amount of computation, some GPU devices will benefit more from the 6*6 convolution effect. YOLOv5s converts 608*608*3 images into 304*304*12 feature maps after focus processing, which somewhat enhances the functionality of the feature map. The feature map will then go through 32 convolution kernels once again to create a feature map with 304*304*32 nodes. The focus module in YOLOv5s eventually makes use of a convolution kernel with a size of 32.

In V3 and V4, there is no focus structure. The V5 mode's innovation lies in this structure. Slicing is the key to the focus module (Figure 2).

In YOLOv5s, the CSP structure splits the original input into two branches, performs convolution operations on each branch, reduces the number of channels, applies bottleneck * N on one branch, and then concatenates the two branches to equalize the input and output sizes of the Bottlenneck CSP, so that the model can learn more features.

In YOLOv4, the CSPnet design concept is used as a reference, and the CSP structure is designed in the backbone network. Only the backbone network in V4 uses CSP structure, which is how V5 and V4 differ from one another. Two CSP structures are designed in V5. Take YOLOv5s network as an example, the backbone network adopts CSP1_ X structure, neck adopts CSP2_ X structure.

Compared with the residual structure of YOLOv3. The structure of CSP net is not complex, so it can be considered that there are large residual edges in CSP. CSP net can also be easily applied to ResNet and ResNeXt, and its architecture is shown in Figure 3. Since only half of the characteristic channels pass through Res (x) Blocks, it is no longer necessary to introduce bottleneck. This makes it possible to theoretically lower the memory access cost (MAC) when performing fixed floating-point operations (FLOP).

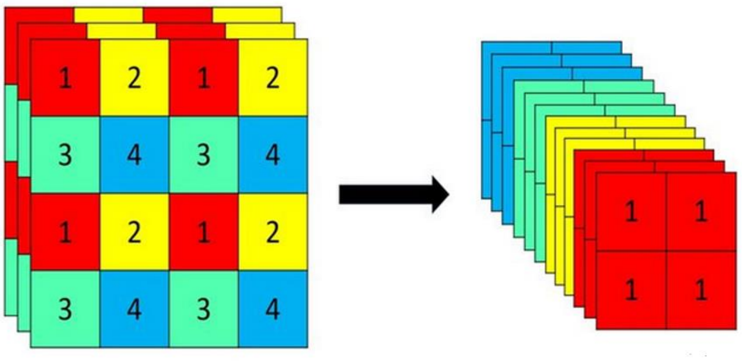


Figure 2. Slicing operation

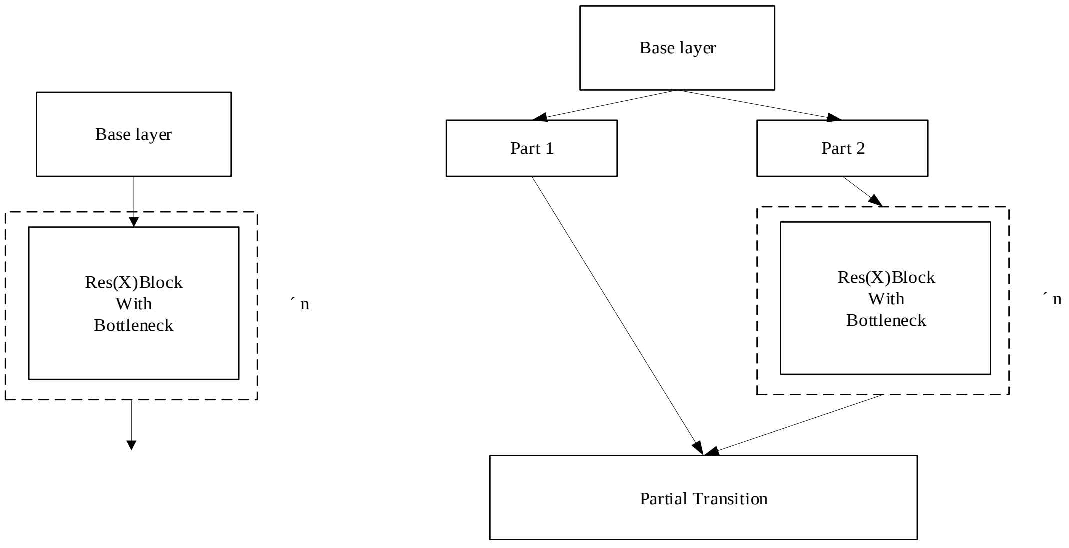


Figure 3. CSPnet structure

2.4 Neck

As much information as possible is gathered by the "neck" structure, which is located between the head and the backbone, before the information obtained by the backbone is fed back to the head. By limiting the loss of small target information, this structure is crucial in the transmission of small target information. This is accomplished by increasing the feature map's resolution once more so that features from various backbone layers can be pooled to enhance overall detection performance.

Comparatively speaking, the neck components—basically consisting of CBS, Upsample, Concat, and CSP (C3) without shortcut—are quite straightforward.

The FPN+PAN structure is also utilized in the design of Neck structure. FPN builds feature pyramids with the traditional structure in subgraph (a) of Figure 4, and uses top-down side connections to build high-level semantic feature maps on all sizes; The pan's construction is not unusual. The underlying target information is highly fuzzy after traversing the multilayer network in the midst of FPN. As a result, PAN strengthens and adds a bottom-up route to compensate for and balance out the positional information in Figure 4.

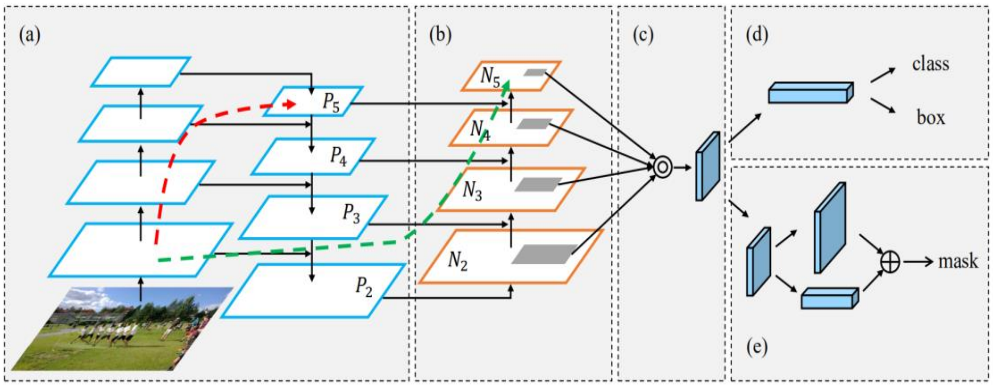


Figure 4. PANnet structure

2.5 Activation Function, Optimization Function and Loss Function

The activation function non-linearizes the neural network, and determines whether the perceptron is excited. This nonlinearity of the activation function enables the deep network to learn complex functions. YOLOv5 uses two activation functions, namely Leaky ReLU and Sigmoid.

The YOLOv5 model is optimized by Adam, whose performance facing sparse gradients is optimized through deviation correction. During model training and optimization, Adam achieves better optimization quality and speed by making each parameter to adapt to the learning rate.

In YOLOv5, the loss function of the bounding box is GIoU loss:

$$\mathrm{GIoU} = \mathrm{IoU} - |\mathrm{Ac} - \mathrm{U}| \;/\; |\mathrm{Ac}| \tag{1}$$

As an upgraded version of IoU, GIoU not only inherits the merits, i.e., effective comparison of the similarity between two arbitrary shapes and scale invariance, and makes up for the disadvantage that IoU cannot measure the distance between non-overlapping frames.

3. Methodology

YOLOv5’s recognition model can detect some objects more effectively on its own and has a quick recognition speed and good effect. To achieve the exclusive detection of small targets like masks and enhance the precision of mask wearing recognition, the mask wearing detection system developed in this research merely requires further network structure optimization and YOLOv5 model parameter adjustments.

3.1 Training Algorithm

The inputs of the training algorithm are the face mask dataset image and tag file. The following parameters are initialized: the number of training iterations, learning rate, batch size, size of input image, network configuration yaml file, IoU threshold of tag and anchor, loss coefficient, data enhancement coefficient. Each input image is preprocessed through brightness adjustment, as well as contrast, saturation and mosaic processing. Finally, the algorithm outputs the detection model that performs best in this training.

The model is trained in the following steps: Firstly, prepare the original data, and divide them into a training set, a test set, and a verification set. Next, import data configurations and initial parameters, and preprocess the input data. After that, load the network model, extract the features of the input image, and locate and classify the objects. With the increase of iteration times, SGD is used to update and optimize each group of parameters in the network. If the current iteration is not the last round, produce the map of the current model on the validation set; If the calculated model has better performance, update the stored best model. After a set number of iterations, output the model with the optimal performance, and the model being trained the latest.

3.2 Overall Implementation

The data gathered by the camera must first be preprocessed using the interface function of the free OpenCV package; the data is then prepared, the face mask images are screened and labeled, and the data set is randomly divided into the training, test, and verification sets; the model is trained by the proposed algorithm, and the most effective model for detecting facial masks is obtained; the obtained model is tested on the test data, and the final recognition results are drawn on the test images, namely, the position of the face and the mask's wearing status.

3.3 Mask Wearing Identification

The procedure of mask wearing recognition is depicted in Figure 5. The target to be detected first enters the face detection module in the form of an image to carry out the matching facial area recognition and marking operations at the beginning of the mask wearing recognition process; then, the mask wearing recognition module is activated to frame the face, and create an image; After classifying the images into groups based on whether or not the mask is worn, the module outputs the image recognition result about whether or not the mask is worn.

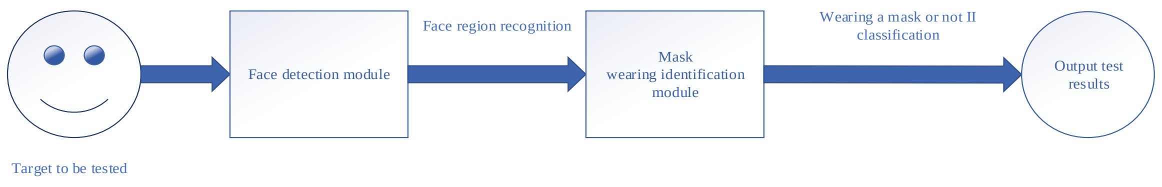


Figure 5. Flow chart of mask wearing identification

4. System Design

4.1 Dataset Production

The mask wearing statuses are predefined into two classes: wearing masks and not wearing masks. The two classes are labeled as face and mask, respectively. It is important to prevent mistaking mouth-covering actions (e.g., covering your mouth with hands or clothes) as mask wearing, the data about the mouth being covered by hands or clothes are added specifically to the dataset. The visualization results are obtained by analyzing the dataset: the objects in the dataset are relatively uniform in distribution, and most of them are small and medium-sized. There is occlusion between objects, which is normal in daily scenes. However, the class distribution is a bit unbalanced, and should be alleviated in data preprocessing.

The dataset folder has two subfolders: The image subfolder, and the label subfolder. The two subfolders store image and label txt files, respectively. The directory of images corresponds to that of labels. This is very important, for YOLO first reads the image path, and then directly replaces the image with label to find the label file.

Data labeling can frame, track, and transcribe images, text, voice, video, and other data, making the data suitable for AI and machine learning. LabelImg, written in Python and designed with QT, is the most common visual image calibration tool for image annotation, which is crucial for deep learning-based object detection. Fast-R-CNN, YOLO, SSD and other datasets required by the target detection network can be handled by this tool to calibrate the targets in the image. The generated XML file follows the Pascal VOC format.

Here, LabelImg is employed to train YOLOv5 model on our datasets. In the virtual environment, LabelImg is installed through PIP instructions, and the software is directly executed in the command line to start the data annotation process. Because YOLOv5 is applied in this design, the annotation format is directly changed to YOLO for annotation. The specific annotation process is as follows: open the image directory, set the directory for saving annotation files, and select automatic saving. Start labeling, framing, target labeling, save, and then switch to the next to continue labeling.

The labeling produces a series of txt files, which are a comment file for target detection. The names of the txt files and image files correspond one by one. Next, we need to open the specific annotation file, which contains the following contents: Each line in the txt file represents a target, which is distinguished by spaces, representing the its class ID, the X coordinate and Y coordinate of the normalized center point, and the W and h of the target box. The subsequent steps include: modify the dataset configuration file, assign the completed data into training set, test set and verification set by 7:2:1, and place the images and label txt files in the right subfolders.

4.2 Model Training

The YOLO model is originally able to recognize simple objects. After training, the model will get the ability to frame the face and judge whether the subject wears a mask. The model is trained in the following steps:

First, create a mask under the data directory. Next, configure the yaml's file, i.e., the dataset configuration file, with four contents, namely, the path of the validation image of the training set, the path of the validation image, the number of classes of the dataset, and the class alias of the dataset.

Hence, create a mask under YOLOv5s models. Next, configure the yaml's model configuration file, which configures the datasets required for training, with two classes: face and mask.

Afterwards, select the pretraining weight file. The purpose is to shorten the training time, and achieve better accuracy. The 5.0 version of YOLOv5 provides several pretraining weights, which suit different needs. The greater the weight, the better the training accuracy, and the slower the detection. In this paper, the most suitable pretraining weight is YOLOv5s.pt.

Finally, run the program for 100 epochs on the dataset, and set the image data as a batch of 4. The program will scan the data before training the model.

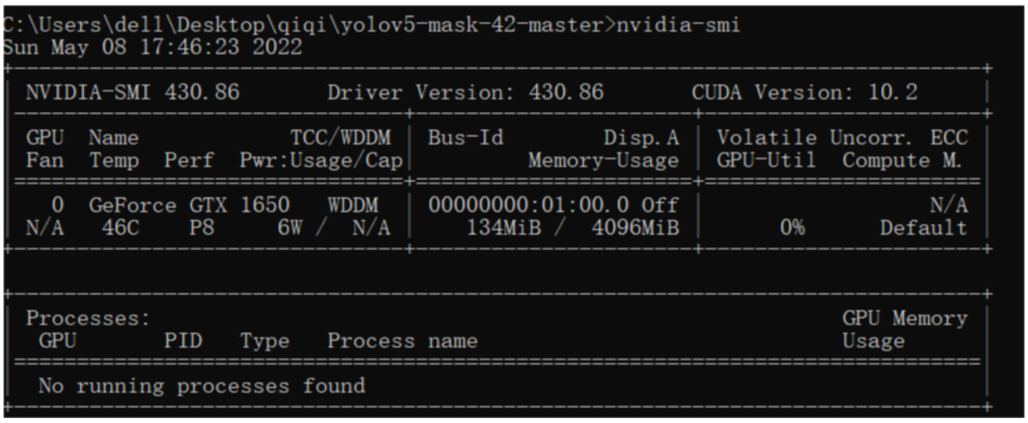
4.3 UI Interface

The UI image interface is developed by configuring PyQt in the PyCharm compilation environment and encapsulating it with PyQt5, aiming to realize image mask detection, video mask detection and camera real-time mask detection. When the interface starts, load the model, and set the directory of TMP to output the intermediate processing results. In the output, face and mask identify the target without and with mask, respectively.

5. Experiments and Results Analysis

5.1 Main Parameters

Run the "NVIDIA SMI" command to see the model, memory, driver version of the current graphics card, the process using the card, and CUDA version (Figure 6).



```
C:\Users\dell\Desktop\qiqi\yolov5-mask-42-master>nvidia-smi
Sun May 08 17:46:23 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 430.86       Driver Version: 430.86       CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1650   WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   46C    P8     6W /  N/A |    134MiB /  4096MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figure 6. Equipment parameters

5.2 Performance Metrics

Our model was evaluated by precision, recall, mean Average Precision (mAP) and frames per second (FPS). Accuracy is a measure of accuracy, while recall is a measure of coverage. The higher the accuracy and recall, the better the recognition effect of mask wearing status. The mAP measures the recognition accuracy. The larger the mAP, the better the recognition effect. FPS measures the number of frames of images transmitted per second. The higher the value, the faster the recognition.

5.3 Results

The loss value of YOLOv5 includes position loss, confidence loss and class loss. Figure 7 shows the convergence curve of loss value after 100 iterations of training. The results show that our model achieved a good fitting effect.

The confusion matrix indicates the classification accuracy. It can be seen that the detection accuracy of face and mask classes were 0.90 and 0.98, respectively.

Figure 8 shows the performance metrics of YOLOv5 model for mask wearing recognition. After 100 iterations, the model achieved convergence. During model training, accuracy and recall rate improved stably, and the mAP remained at a high level. The maximum precision, recall, and mAP of the trained model reached 0.987, 0.996 and 0.981, respectively.

On a test set of 349 random images, the P-R curve was drawn for our model by calculating the highest accuracy under different recall rates (Figure 9). The test results show that our model recognized objects quickly in practical application, and basically met the speed requirements of real-time detection.

The training results demonstrate the high detection accuracy of our model. No object was missed by the model, despite the high number of objects in the images. The model also effectively filtered interferences like mouth covering by hands or clothes.

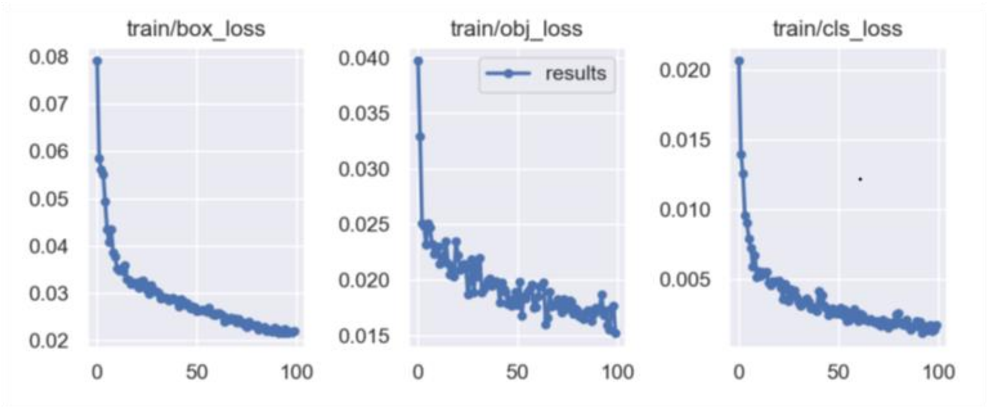


Figure 7. Training loss curves

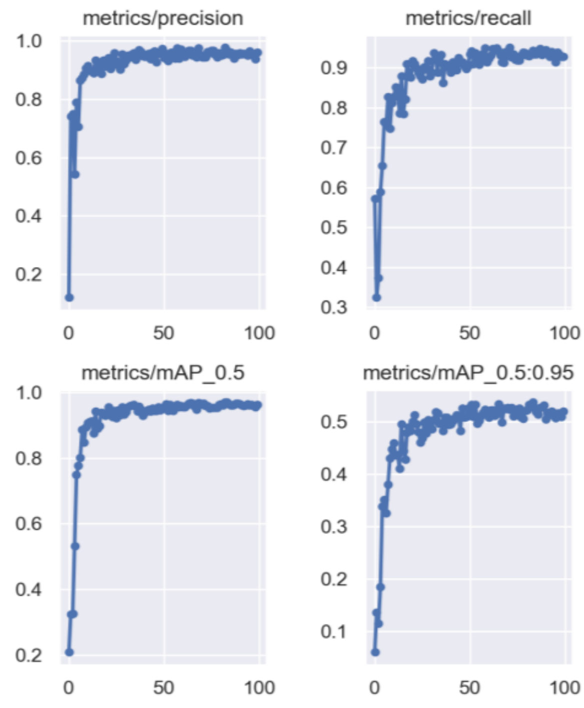


Figure 8. Metrics of YOLOv5 model

The proposed YOLOv5 algorithm, a real-time detection method, reached a speed of 130 f/s and an accuracy rate of about 95%. Meanwhile, it lowered the performance requirements of the hardware, and controlled the amount of calculation, thus meeting the needs of real-time detection.

Figure 10 and Figure 11 show the results of the UI interface, which supports the functions of image mask wearing detection, video mask wearing detection and camera real-time mask wearing detection. The results of image mask wearing detection show that our model can accurately identify the mask wearers and non-mask wearers, in the face of dense crowds. Hence, YOLOv5 algorithm is superior in mask wearing detection.

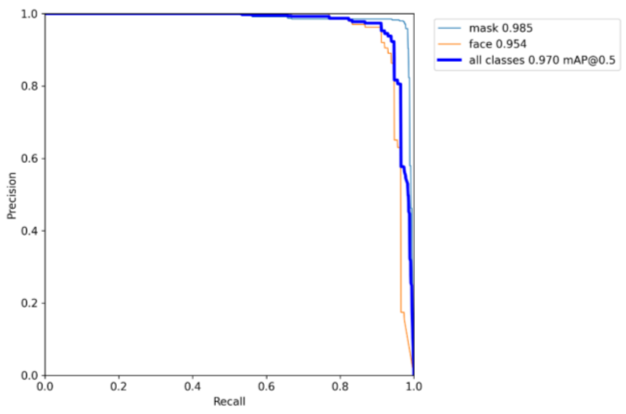


Figure 9. P-R curve

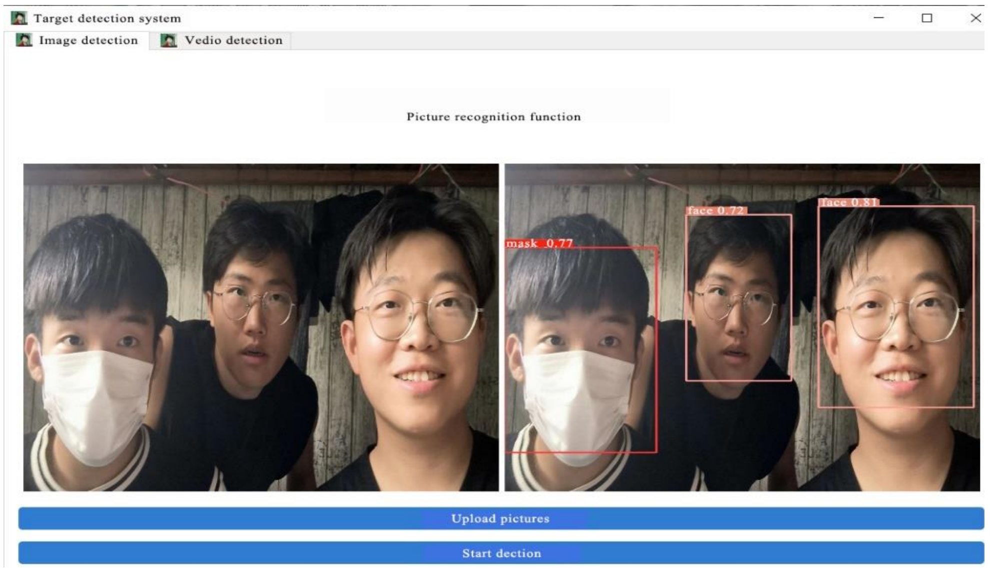
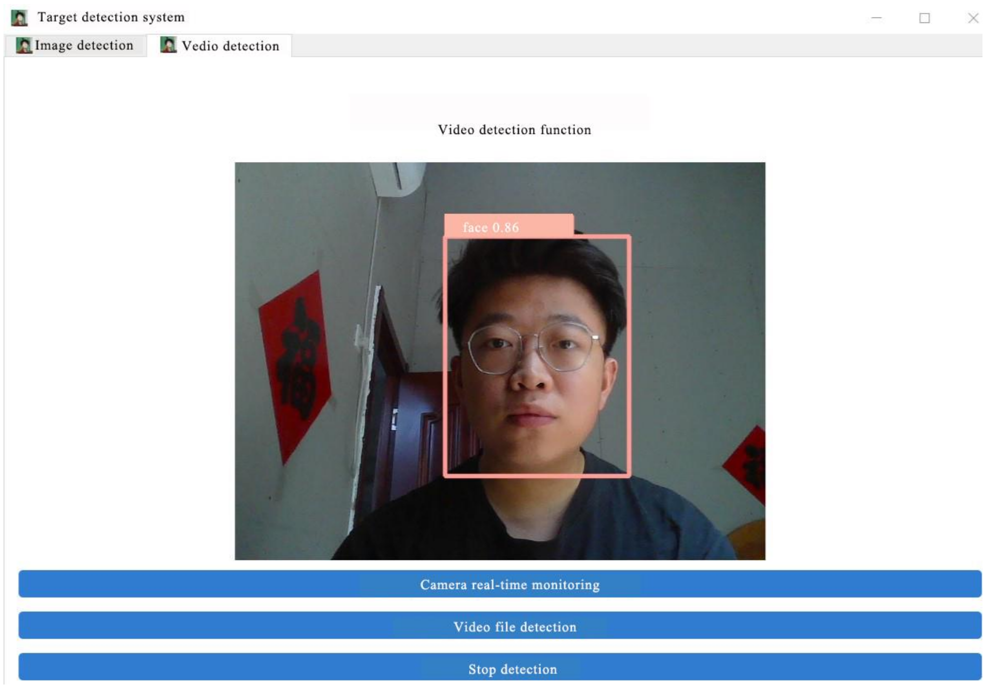


Figure 10. Image mask wearing test



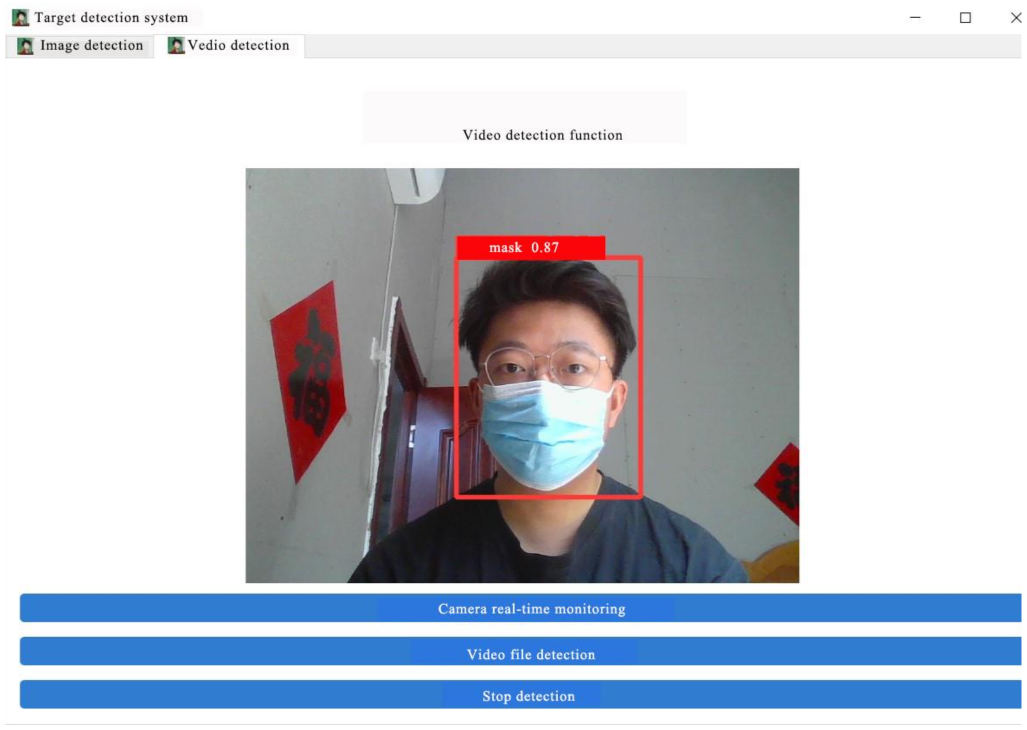


Figure 11. Camera real-time mask wearing detection

5.4 Algorithm Comparison

To further validates its superiority, our algorithm was compared with SSD, fast-R-CNN and YOLOv3, three common target detectors, using the control variate technique. The results are displayed in Table 1. The comparison shows that our algorithm significantly outshines the other algorithms in mAP and recognition speed. The excellent performance is realized with a small size, low cost, and high efficiency.

Table 1. Comparison between algorithms

Detection algorithm	mAP/%	FPS	Mask/%	Face/%
SSD	74.12	16.8	73.20	70.56
Fast-R-CNN	77.55	8.7	74.09	76.77
YOLOv3	80.66	50.54	76.75	79.36
Our algorithm	98.10	70.33	96.64	97.23

6. Conclusions

This study proposes a YOLOv5-based mask wearing identification approach, and develops and applies a lightweight YOLOv5 enhanced network to enhance target detection speed and accuracy while minimizing model parameters. K-means++ clustering is applied to customize the datasets, and the anchor box size of the target samples is optimized. Experiments demonstrate that even in complicated situations, the model maintains good accuracy, detection speed, and robustness. The algorithm performs well in terms of multiple targets, quick detection times, and small model parameters.

The model may considerably reduce dependence on the hardware environment, compress and accelerate reasoning speed, maximize model accuracy, and best suit the needs of actual applications. It can be used in conjunction with the mask wearing identification system to provide efficient public health protection monitoring, advance the intellectualization, scientific level, and humanization of the mask wearing detection process, and is crucial for fostering a secure environment in public.

There are still a few missed detections and false detections in the real detection because of uneven illumination, target occlusion, dense crowd, and other issues. We will continue to investigate this issue and contribute appropriately to the prevention and management of epidemics in the future. In addition to figuring out how to effectively deploy the mobile terminal model, the follow-up work will also verify and enhance the suggested model in real-world applications. At the same time, the integration of mask wearing and other related functions will result in a more functional safety and health supervision system, better suited to the current societal demands.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] H. Li, A. Z. Wu, Q. Q. Fang, M. Zhang, Q. Liu, Q. W. Liu, and W. Chen, "Lightweight mask R-CNN for long-range wireless power transfer systems," In *2019 11th International Conference on Wireless Communications and Signal Processing*, (WCSP 2019), Xi'an, China, October 23-25, 2019, IEEE, pp. 1-6. https://doi.org/10.1109/WCSP.2019.8927856.

[2] R. Gavrilescu, C. Zet, C. Foșalău, M. Skoczylas, and D. Cotovanu, "Faster R-CNN: An approach to real-time object detection," In *2018 International Conference and Exposition on Electrical and Power Engineering*, (EPE 2018), Iasi, Romania, October 18-19, 2018, IEEE, pp. 165-168. https://doi.org/10.1109/ICEPE.2018.8559776.

[3] M. D. Pramita, B. Kurniawan, and N. P. Utama, "Mask wearing classification using CNN," In *2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications*, (ICAICTA 2020), Tokoname, Japan, September 8-9, 2020, IEEE, pp. 1-4. https://doi.org/10.1109/ICAICTA49861.2020.9429029.

[4] J. Zhang, F. Han, Y. Chun, and W. Chen, "A novel detection framework about conditions of wearing face mask for helping control the spread of COVID-19," *IEEE Access,* vol. 9, pp. 42975-42984, 2021. https://doi.org/10.1109/ACCESS.2021.3066538.

[5] B. Wang, Y. Zhao, and C. L. P. Chen, "Hybrid transfer learning and broad learning system for wearing mask detection in the COVID-19 era," *IEEE T. Instrum. Meas.*, vol. 70, pp. 1-12, 2021. https://doi.org/10.1109/TIM.2021.3069844.

[6] M. Yu, S. Zou, A. Jia, and X. Cheng, "Recognition of the standardization of wearing masks during the epidemic of COVID-19," In *2021 IEEE Asia-Pacific conference on image processing, electronics and computers*, (IPEC 2021), Dalian, China, April 14-16, 2021, IEEE, pp. 728-732. https://doi.org/10.1109/IPEC51340.2021.9421236.

[7] J. Vadlapati, S. Senthil Velan, and E. Varghese, "Facial recognition using the OpenCV Libraries of Python for the pictures of human faces wearing face masks during the COVID-19 pandemic," In *2021 12th International Conference on Computing Communication and Networking Technologies*, (ICCCNT 2021), Kharagpur, India, July 06-08, 2021, IEEE, pp. 1-5. https://doi.org/10.1109/ICCCNT51525.2021.9579712.

[8] C. Ratanaubol, P. Wannapiroon, and P. Nilsook, "Video-based facial recognition develop for accurately identify people wearing surgical masks," In *2021 3rd International Conference on Computer Communication and the Internet*, (ICCCI 2021), Nagoya, Japan, June 25-27, 2021, IEEE, pp. 19-22. https://doi.org/10.1109/ICCCI51764.2021.9486818.

[9] C. Ratanaubol, P. Wannapiroon, and P. Nilsook, "Video-based facial recognition develop for accurately identify people wearing surgical masks," In *2021 3rd International Conference on Computer Communication and the Internet*, (ICCCI 2021), Nagoya, Japan, June 25-27, 2021, IEEE, pp. 19-22. https://doi.org/10.1109/ICCCI51764.2021.9486818.

[10] K. Zhang, X. Jia, Y. Wang, H. Zhang, and J. Cui, "Detection system of wearing face masks normatively based on deep learning," In *2021 International Conference on Control Science and Electric Power Systems*, (CSEPS 2021), Shanghai, China, May 28-30 2021, IEEE, pp. 35-39. https://doi.org/10.1109/CSEPS53726.2021.00014.

[11] R. K. Shukla, A. K. Tiwari, and V. Verma, "Identification of with face mask and without face mask using face recognition model," In *2021 10th International Conference on System Modeling & Advancement in Research Trends*, (SMART 2021), MORADABAD, India, December 10-11, 2021, IEEE, pp. 462-467. https://doi.org/10.1109/SMART52563.2021.9676204.

[12] L. Wang, Y. Lin, W. Sun, and Y. Wu, "Improved faster-RCNN algorithm for mask wearing detection," In *IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference*, (IMCEC 2021), Chongqing, China, June 18-20, 2021, IEEE, pp. 1119-1124. https://doi.org/10.1109/IMCEC51613.2021.9482098.

[13] T. H. Wu, T. W. Wang, and Y. Q. Liu, "Real-time vehicle and distance detection based on improved YOLOv5 network," In *2021 3rd World Symposium on Artificial Intelligence*, (WSAI 2021), Guangzhou, China, June 18-20, 2021, IEEE, pp. 24-28. https://doi.org/10.1109/WSAI51899.2021.9486316.

[14] S. Yu, H. Li, F. Gui, Y. Yang, and C. Lv, "Research on mask wearing detection algorithm based on YOLOv5," In *2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence*, (ICIBA 2021), Chongqing, China, December 17-19, 2021, IEEE, pp. 625-630. https://doi.org/10.1109/ICIBA52610.2021.9688011.

[15] W. Zhang, H. Yan, Y. Liu, X. Wang, and J. Huang, "Mask wearing detection based on improved YOLOv3," In *2021 International Conference on Computer Information Science and Artificial Intelligence*, (CISAI 2021), Kunming, China, September 17-19, 2021, IEEE, pp. 194-197. https://doi.org/10.1109/CISAI54367.2021.00044.

[16] S. Chen and W. Lin, "Embedded system real-time vehicle detection based on improved YOLO network," In *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference*, (IMCEC 2019), Chongqing, China, October 11-13, 2019, IEEE, pp. 1400-1403. https://doi.org/10.1109/IMCEC46724.2019.8984055.

[17] C. Jiang, H. Zhang, Y. Yue, and X. Hu, "AM-YOLO: Improved YOLOV4 based on attention mechanism and multi-feature fusion," In *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference*, (ITOEC 2022), Chongqing, China, March 4-6, 2022, IEEE, pp. 1403-1407. https://doi.org/10.1109/ITOEC53115.2022.9734536.

[18] R. Liu and Z. Ren, "Application of YOLO on mask detection task," In *2021 IEEE 13th International Conference on Computer Research and Development*, (ICCRD 2021), Beijing, China, January 5-7, 2021, IEEE, pp. 130-136. https://doi.org/10.1109/ICCRD51685.2021.9386366.

[19] M. H. Nugraha and D. Chahyati, "Tourism object detection around monumen nasional (monas) using YOLO and RetinaNet," In *2020 International Conference on Advanced Computer Science and Information Systems*, (ICACSIS 2020), Depok, Indonesia, October 17-18, 2020, IEEE, pp. 317-322. https://doi.org/10.1109/ICACSIS51025.2020.9263240.

[20] B. Strbac, M. Gostovic, Z. Lukac, and D. Samardzija, "YOLO multi-camera object detection and distance estimation," In *2020 Zooming Innovation in Consumer Technologies Conference*, (ZINC 2020), Novi Sad, Serbia, May 26-27, 2020, IEEE, pp. 26-30. https://doi.org/10.1109/ZINC50678.2020.9161805.