Alinteri J. of Agr. Sci. (2024) 39(2): 148-159
e-ISSN: 2587-2249
info@alinteridergisi.com

http://dergipark.gov.tr/alinterizbd
http://www.alinteridergisi.com/

WearMask: Rapid In-browser Face Mask Recognition for COVID-19 Using Serverless Edge Computing

Mrs.B.Mamatha1, Manda Nidisha2, Ramisetty Ajay3,Pathuri chaitanya srinivas4

1.Asst.Professor,Computer Science and Engineering, CMR Engineering College, medchal, T.S, India

2.B.Tech, Computer Science and Engineering, CMR Engineering College, medchal, T.S, India

Abstract- In the US, the COVID-19 pandemic has posed a serious threat to healthcare. 80% of respiratory infections can be prevented primarily, effectively, and conveniently by wearing a mask.Centers for Disease Control and Prevention (CDC), COVID-19infection is transmitted predominately by respiratory dropletsgenerated when people breathe, talk, cough, or sneeze. Wearinga mask is the primary, effective, and convenient method ofblocking 80% of all respiratory infections. Nevertheless, the public's accessibility is hindered by the fact that most commercial face mask detection devices available today come packaged with particular hardware or software. In this research, we offer a Webbased efficient AI recognition of masks (WearMask) in-browser serverless edge computing based face mask detection system that can be implemented on any common devices (e.g., mobile phones, tablets, machines) having web browsers and internet connections; no software needs to be installed. The additional hardware expenses (such as dedicated devices or cloud computing servers) are kept to a minimum by the serverless edge computing architecture.The additional hardware expenses (such as dedicated devices or cloud computing servers) are kept to a minimum by the serverless edge computing approach. The suggested approach offers a comprehensive edge computing framework that combines three components: (1) high-performance neural network inference computing framework (NCNN), (2) deep learning models (YOLO), and (3) a virtual machine that is stack-based (WebAssembly). Our web-based solution offers the following benefits to end users: (1) reduced device limitation and privacy risk due to its serverless edge computing design; (2) installation-free deployment; (3) low computing requirements; and (4) high detection speed. Our WearMask app is now available to the general public at facemask-detection.com.

KEYWORDS —COVID-19, mask-wearing, edge computing, deployment, deep learning, and edge device.

I. INTRODUCTION

SINCE November 2019, the COVID-19 epidemic had been a major social and healthcare issue in the United States. Only during the Thanksgiving week in 2020, there were 1,147,489 new confirmed cases and 10,279 new deaths from COVID-19 [1]. It is necessary to wear masks in public places [2]. Even with the successful development of many vaccines, wearing a mask is still one of the most effective and affordable ways to block 80% of all respiratory infections and cut off the route of transmission [3]. Even many states Z. Wang, Y. Huo were with the Data Science Institute, Vanderbilt University, Nashville, TN, 37215, USA P. Wang is the corresponding author, was with the Department of Information Science and Engineering, Shandong University, Qingdao, China, e-mail: wangpw@sdu.edu.cn P. Louis was with the Department of Biomedical Informatics, Vanderbilt University Medical Center, Nashville,TN, 37215, USA L. Wheless was with the Department of Dermatology, Vanderbilt University Medical Center, Nashville,TN, 37215, USA Y. Huo was with the Department of Computer Science, Vanderbilt University, Nashville, TN, 37215, USA have enforced people to wear masks in public places, there are still a considerable number of people who forget or refuse to wear masks, or wear masks improperly. Such facts would increase the infection rate and eventually bring a heavier load of the public health care system. Therefore, many face mask monitoring systems have been developed to provide effective supervision for hospitals, airports, publication transportation systems, sports venues, and retail locations. However, the current commercial face mask detection systems are typically bundled with specific software or hardware, impeding public accessibility. Herein, it would be appealing to design a light-weight device agnostic solution to enable fast and convenient face mask detection deployment. In this paper, we propose a serverless edge-computing based in-browser face mask detection solution, called Web-based efficient AI recognition of masks (WearMask), which can be deployed on any common devices (e.g., cell phones, tablets, computers) that have an internet connection and a web browser, without installing any software. Serverless edge-computing is a recent infrastructural evolution of edge-computing, in which computing resources are directly used by end-users. The features of existing computing strategies are listed in Tab. I. To maximize the flexibility and convenience of the deployment for small business, the serverless edge-computing design is introduced in our WearMask framework. As opposed to canonical edge computing, serverless edge-computing does not require extra hardware between a web-server and end-users. Web browsers (e.g., Chrome and Firefox) are used as the interfaces since they are the most widely accessible interface for users to access the internet, which is device and operating system (OS) agnostic. On the other hand, most internet users are familiar with web browsers, which introduce almost no extra learning burdens for deploying our WearMask software. To do so, we aggregate a holistic solution by combining serverless edge-computing and deep learning based object detection, without the requirement of having advanced GPU. The technical contribution of the proposed method is to provide a holistic serverless edge-computing framework with (1) deep learning models (YOLO [4]), (2) high-performance neural network inference computing

framework (NCNN [5]), and (3) a format running on the stack-based virtual machine (WebAssembly [6]). For end-users, the advantages of the proposed web-based solution are (1) minimal device limitation, (2) installation free design, (3) low computing requirements, and (4) high detection speed. Our WearMask application has been launched with public access as a website 1 (Fig. 1).

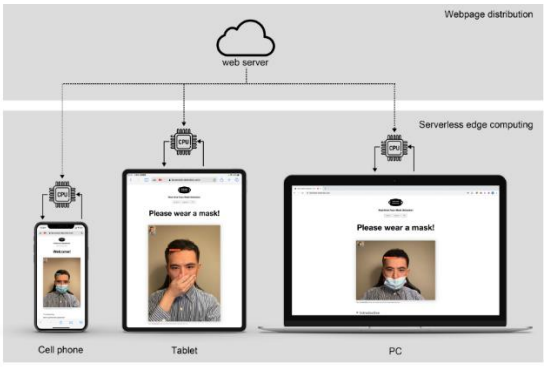

Fig. 1. This figure shows the connection between the web server and edge devices.

TABLE I
SOLUTION COMPARISON

Solution	Amazon Recognition PPE detection	Mask detector (Android APP)	MaskDetection (Kogniz.ai)	Mask detection machine	WearMask (Our solution)
No specific hardware	✓	✓	×	×	✓
No installation	×	×	✓	✓	✓
Cross platform	✓	×	✓	×	✓
Local computing	×	✓	×	✓	✓

Due to the light-weight device-agnostic design, the proposed method would be a cost-efficient solution for public facilities and small businesses. For example, to set up a commercial mask detection solution, averagely costs $1,000 - $4,000, leading to an extra burden for small businesses, especially considering the financial challenges during the pandemic. As the United States has more than 30.2 million small businesses [7] and a large number of public facilities, it is essential to find an affordable alternative. The remainder of the paper is organized as follows. First, we review recent works in face mask detection. Then, we discuss the data curation, model selection, and the training process (Yolo-Fastest [8]). Next, we describe the specific deployment process of the model (NCNN, WebAssembly). Finally, we discuss this scheme's existing strengths and weaknesses and other deployment schemes' advantages and disadvantages

II. RELATED WORKS

A. Face Mask Detection Historically, most of the papers focused on performing face recognition while wearing a mask or with other obstructions, while few, if any, previous studies have been conducted to determine if a subject is wearing a mask. Since the COVID19 pandemic, however, more efforts have been devoted to face mask detection due to the emergence of the need for reducing COVID-19 transmission. To detect occluded faces, Shiming Ge et al. established the MAsked FAces (MAFA) dataset, which contains 30,811 images and established the LLE-CNNs model, obtaining an Average Precision (AP) of 76.4% [9]. A. Nieto-Rodr´ıguez andothers developed an alarm system for the wearing of

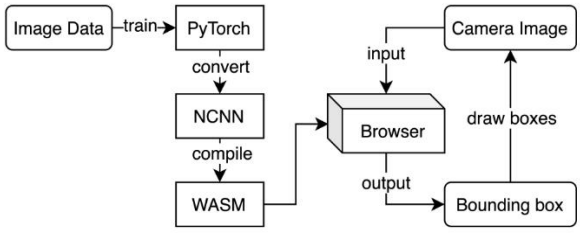

Fig. 2. This figure shows the overall workflow of training and deploying of the proposed WearMask framework.

f masks in the operating room, combining the face detector with the mask detector, and optimized for low False Positive rate and high Recall, and obtained 95% True Positive rate [10]. But its direction is limited to the surgical mask in the operating room. Bosheng Qin et al. established SRCNet and classified the mask-wearing situation into three categories: no facemaskwearing, incorrect facemask-wearing, and correct facemaskwearing, to classify images and obtain 98.70% accuracy [11]. However, it needs to crop the face area before face mask detection and concentrate on the face. Mohamed Loey et al. used ResNet-50 and SVM to obtain 99.49% accuracy on Real-World Masked Face Dataset (RMFD) [12]. Later, they used YOLO-V2 based on ResNet-50 for object detection in another publication and achieved 81% Average Precision (AP) on the Medical Masks Dataset (MMD) and Face Mask Dataset (FMD) [13]. G. Jignesh Chowdary et al. achieved 99.9% accuracy on Simulated Masked Face Dataset (SMFD) with an InceptionV3 based model [14]. Nevertheless, since SMFD is a small dataset generated using mask images. Results are more difficult to be generalized in complicated scenarios. As opposed to these studies, whose goals were to develop a more precise face mask detection algorithm, our work emphasizes leveraging lightweight serverless edge-computing based in-browser deployment of face mask detection with deep learning. To the best of our knowledge, no previous publications have investigated the serverless edge-computing based in-browser solution of face mask detection by aggregating deep learning models, serverless edge-computing frameworks, and stack-based virtual machines.

B. In-browser Serverless Edge Computing

The typical way of designing in-browser deep learning model is to use TensorFlow.js [15] or ONNX.js [16]. These methods use a specialized JavaScript library to read the model file and completes the inference through calling computing operations written in JavaScript. However, JavaScript is not a typical programming language in the deep learning field, with less community support. 1) ONNX.js: To use ONNX.js, we first need to convert the existing PyTorch [17] model into an ONNX (Open Neural Network Exchange) [18] model. ONNX defines various standard operators in machine learning and deep learning as an open format to facilitate developers to use ONNX as a relay to convert models from one framework to another. Now ONNX has supported PyTorch, TensorFlow [19], Caffe2, NCNN, and other common deep learning frameworks. ONNX.js is a JavaScript library that can directly read the ONNX model in the JavaScript environment for inference. However, there are limitations to complete in-browser deployments using ONNX.js: (1) It does not support INT64 format variables in the model. Since ONNX.js

runs in the JavaScript environment and JavaScript does not support INT64 format variables. The ONNX model, directly exported by PyTorch, contains many variables in the INT64 format. Therefore, we need to convert the INT64 component to INT32 format in the ONNX model. Nevertheless, even after replacing all variables with INT32 format, the ONNX model still does not work correctly. Some native ONNX operators only support INT64 as input to the node, such as ConstantOfShape. The official ONNX model interpreter no longer supports the modified operator. (2) Many operators are not supported. ONNX.js needs to read the model and call its built-in JavaScript operations based on the model content, which means that the ONNX.js library must have defined all operators before performing them. Because ONNX.js is a new niche, its supporting operators are relatively few. For example, the Resize operation is not supported. Considering the rapid emergence of new models and ideas nowadays, this is a significant limitation for model deployment. 2) TensorFlow.js: Compared to ONNX.js, TensorFlow.js is more widely used and supports a richer set of operators. To use TensorFlow.js, we first convert the model to a TensorFlow SavedModel format. Since there is no official way to convert a PyTorch model to TensorFlow, we adapt it to an ONNX model and then convert it into a TensorFlow SavedModel. Next, We convert the SavedModel to a particular readable web format model that can be read by TensorFlow.js in a JavaScript environment. Complex conversion chains mean lower reliability and more limitations. In its transformation, it is likely to introduce some uncommon operators in the model. For example, in our attempt, this process introduces the operator called TensorScatterUpdate, which is not supported for the particular readable web format model.

III. METHODS

A. Data Collection

The WearMask model was trained on two different types of data: faces with masks and faces without masks. The WIDER facial dataset provided the normal facial data (mask-free) [20]. genuine faces with genuine masks (MAFA, RMFD [21], MMD) and real faces with generated masks (SMFD) comprised the two categories of the current mask datasets. Since the generated mask images affect the model's capacity to generalize, we substituted real photographs. In a nutshell, the MAFA is made up of face images featuring different faces with diverse masks, backgrounds, and annotation data. All of the faces in the RMFD are background-free. Ultimately, the MAFA dataset was chosen as training data because it was thought that the absence of faces in the background photos would help the model become more accurate. Nevertheless, the WIDER FACE dataset marks the position of the entire face, whereas the MAFA face just includes the areas beneath the brows. The partial annotations in WIDER FACE were expanded to entire faces in order to balance the variations in the annotation processes of two datasets, as follows: according to protocol, the left and right boundaries do not encompass the ears, and the indicated bounding box ranged from below the hairline to above the bottom edge of the chin. Additionally, we gathered additional samples of people wearing masks wrongly as well as certain edge

circumstances. Thus, the final training dataset comprised 1138 extra images from the WIDER FACE, 4065 images from MAFA, and 3894 images from the online. Generally speaking, 80% of the 9,097 photos with 17,532 labeled boxes were used for training and validation, and 20% were used for testing.

B. Modeling

Deep learning models must be both efficient and large in order to be used on edge devices without powerful GPUs. Generally speaking, less computational power is required on edge devices for smaller models.As a more compact version of You only look once, the WearMask solution uses the YOLO-Fastest model as its detection approach. (YOLO) [4] model for object detection. One of the most popular quick object detection methods isYOLO. The grid definition significantly increases computational performance by reducing the need for repetitive anchor computation. The encoder used in the YOLO-Fastest implementation is the EfficientNet-lite [23]. With a total model size of just 1.3 MB (MegaByte), it can be used with edge devices and other low-power computing scenarios.

In order to expedite the training process, the detection system was pre-trained using the COCO dataset (Microsoft Common Objects in Context, 80 categories) [23] backbone. The model was then further trained by modifying the network to fit new object definitions using the face mask datasets as a transfer learning exercise [24].

C. Serverless Edge-Computing

Moreover, we developed the WearMask as a serverless edge computing framework in contrast to the existing cloud computing based ones. The cloud computing solution's capacity to detect face masks is restricted by the availability of sufficient internet capacity to stream live video from cameras, but cloud services are expensive. In contrast, our serverless edge computing concept leverages consumers' existing devices to reduce hardware expenses. Low privacy risk is another important benefit of the edge computing concept, in addition to cheaper hardware costs. The WearMask framework processes video data locally, eliminating the need for uploading procedures (such uploading to cloud servers), which is crucial for methods pertaining to healthcare.

D. In-browser Deployment

By combining the NCNN and WebAssembly (WASM) architectures, we further implement the edge-computing technique as an in-browser deployment (Fig. 2). Tencent created NCNN, an open-source optimized inference technology for mobile platforms. It is developed entirely in C++, with no outside assistance,library requirements. This will reduce the built model's size while maintaining a respectable level of computational efficiency on edge devices, particularly those with ARM architecture chips. When there are custom operators in the model, it functions since it supports custom layers. Additionally, it has offered tools for converting several

frameworks to NCNN. A low-level language called WASM is used by browsers. Because it is in binary form, JavaScript programs cannot match its speed. In addition, widely used web browsers including Chrome, Edge, Firefox, and WebKit (Safari) offer minimal support for WASM already. We first transformed the PyTorch model into an NCNN model with a 581 KB model size using the NCNN library. We then developed a fresh C++ application to expedite the Using the NCNN library for inference, the detection process proceeds from picture preprocessing to the final output of the category, confidence, and box position. This C++ application was compiled into WASM format, and then the complete framework was run in JavaScript as a function. Using HTML and CSS, we rendered the discovered bounding boxes with the original face photos to demonstrate the detection findings.

IV. EXPERIMENTS AND RESULTS

A. Experimental Setting

The training was performed on Google Colab (Tesla V100- SXM2-16GB). The PyTorch Version was 1.7.0 + cu 101. The training code was modified from the public code for YOLOV3 [25] from Ultralytics [26].

Fig. 3. This figure shows the different detection results. a) and b) show the cases with wear masks properly, c) and d) are the examples that wear mask improperly, and e), f), g) and h) are the cases of not wearing the mask.

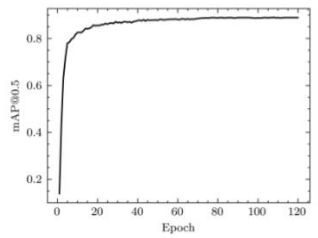

Fig. 4. This figure shows the testing mAP @ 0.5 of different training epochs.

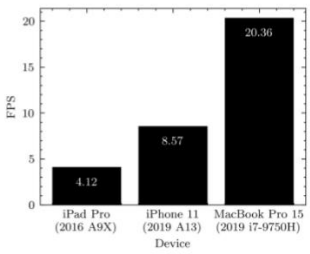

Fig. 5. This figure shows the different Frames Per Second (FPS) using different edge devices.

B. Results

The qualitative results of our proposed WearMask framework is presented in (Fig. 3). It is able to correctly identify the cases when a subject is not wearing the mask properly, such as when the nostrils or mouth are exposed. It also has a higher probability of recognizing that it is not a mask even if the object uses the hands, elbows, or other things such as cell phones to cover the face. The model is able to work on multiple faces within the same frame. From quantitative results, the trained YOLO-Fastest model achieved mean of Average Precision when IoU is 0.5 (mAP@0.5) of 0.89 after 120 epochs with batch-size 16(Fig. 4). The detection FPS on representative edge devices are presented is Fig. 5. After converting the PyTorch model to an NCNN model.

Fig. 6. This figure shows more testing examples using the proposed WearMask tool.

with WASM, we established a website and published it at facemask-detection.com, to demo the in-browser deployment.

C. Comparison with Previous Studies

Previous related works used various datasets with different tasks and evaluation strategies. For instance, some of the methods evaluated the performance of detection and classification using detection metrics, while other works only evaluated detection or classification, respectively. Therefore, most of the previous evaluation results cannot be directly compared with the results obtained in this paper. In this study, we compared our performance with the most similar settings among previous works, which (1) used real mask pictures instead of simulated mask pictures, (2) performed simultaneous object detection and classification. Among them, the LLE-CNNs model used by Shiming Ge et al. obtained an AP of 76.4% [9]. Mohamed Loey et al. used YOLOV2 with ResNet-50 and obtained an AP of 81% [13]. Our model achieved an AP of 89% (Fig. 4). Although the datasets are not completely same, it shows that the performance of this model is at a comparable level with prior arts. Moreover, this study aims to build a light-weight edge computing framework for face mask detection rather than to achieve the best detection accuracy. More testing examples are presented in (Fig. 6)

V. DISCUSSION

Present mask detection systems usually necessitate deployments in specialized areas or with specialized equipment, which are either expensive, inflexible, or unscalable. In order to solve these problems, we suggested a novel edge-computing based face mask detection technique known as WearMask, which has the attributes listed below:

(1) Serverless edge-computing design. The proposed framework can be deployed conveniently with minimal costs and high flexibility. The deep learning model size is minimized for edge devices.

(2) Simple implementations. The framework works with all major operating systems, including Windows, macOS, Linux, Android, and iOS, and is device-agnostic, meaning it may be used on any computer, laptop, smartphone, or tablet.

(3) Installation free. The program can be used without requiring an installation procedure or specific environmental conditions. To start the software, users merely need to go to our website and allow camera access.

(4) Minimal risk to privacy. There was no need to upload any data to the server or save any contents because the software runs locally without exchanging data. In order to protect privacy, Once loaded, the model and data can be unplugged from the Internet.

There were also a few limitations with the suggested mask detection method. The software cannot measure human body temperature as well as professional equipment because the majority

of everyday gadgets do not support infrared detection. As a tool for merely reminders, it If someone refuses to wear a mask, you cannot make them wear it.

The potential future improvements of this mask detection scheme will be as follows:

(1) The existing dataset divides the no mask and the wearing masks incorrectly into one category, which causes some misunderstanding in detection. Fine-grain classification would be helpful with more training data. (2) For a subject that does not wear the mask properly, the current scheme can recognize this case. However, it cannot remind the subject of the specific incorrect location, such as revealing the nostril or mouth. In the future, we can add an attention map [27] to specify the location that the mask is not worn properly. (3) Due to the system limitation in iOS, only Safari supports WebAssembly-related functions, and it does not support parallel computing features such as SIMD (Single instruction, multiple data). Therefore, when running on iOS, the Frames Per Second (FPS) is significantly lower than the performance of the same CPU level on other system devices. From our experiments, when SIMD-related features were enabled, the FPS was twice faster

VI. CONCLUSION

In order to alert individuals who are not wearing masks or are wearing them incorrectly, we presented a system-agnostic, no-installation face mask detection method in this work. As an With its serverless edge computing architecture, it has a reduced risk of privacy, a low necessary network bandwidth, and a quick reaction time when operated locally on a range of edge devices. By combining NCNN and WASM, the deployment plan addressed the JavaScript community's inadequate support for deep learning. Our WearMask solution is a generic framework that allows various lightweight deep learning models to take the place of the detection method. Our WearMask technology can monitor and remind people to wear masks during the COVID-19 pandemic, reducing the risk of respiratory diseases.

REFERENCES

[1] 1Point3Acres.com, “Global covid-19 tracker & interactive charts: Real time updates & digestable information for everyone,” https://coronavirus. 1point3acres.com/, 2020.

[2] D. K. Chu, E. A. Akl, S. Duda, K. Solo, S. Yaacoub, H. J. Schunemann, ¨ A. El-harakeh, A. Bognanni, T. Lotfi, M. Loeb et al., “Physical distancing, face masks, and eye protection to prevent person-to-person transmission of sars-cov-2 and covid-19: a systematic review and metaanalysis,” The Lancet, 2020.

[3] C. for Disease Control, Prevention et al., “Scientific brief: Community use of cloth masks to control the spread of sars-cov-2,” Document modifie le ´ , vol. 10, 2020.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779– 788.

[5] nihui et al., "Ncnn: A high-performance neural network inference framework optimized for the mobile platform," https://github.com/Tencent/ ncnn, 2020.

[6] A. Rossberg, B. L. Titzer, A. Haas, D. L. Schuff, D. Gohman, L. Wagner, A. Zakai, J. Bastien, and M. Holman, "Bringing the web up to speed with webassembly," Communications of the ACM, vol. 61, no. 12, pp. 107–115, 2018.

[7] SBA, "2018 small business profiles," https://www.sba.gov/sites/default/ files/advocacy/2018-Small-Business-Profiles-US.pdf, 2018.

[8] dog qiuqiu et al., "Yolo-fastest: Yolo universal target detection model combined with efficientnet-lite," https://github.com/dog-qiuqiu/ Yolo-Fastest, 2020.

[9] S. Ge, J. Li, Q. Ye, and Z. Luo, "Detecting masked faces in the wild with lle-cnns," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2682–2690

[10] A. Nieto-Rodr ´ıguez, M. Mucientes, and V. M. Brea, "System for medical mask detection in the operating room through facial attributes," in Iberian Conference on Pattern Recognition and Image Analysis . Springer, 2015, pp. 138–145.

[11] B. QIN and D. LI, "Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19," 2020.

[12] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic," Measurement , vol. 167, p. 108288, 2020.

[13] ——, "Fighting against covid-19: A novel deep learning model based on yolo-v2 with resnet-50 for medical face mask detection," Sustainable Cities and Society, p. 102600, 2020.

[14] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face mask detection using transfer learning of inceptionv3," arXiv preprint arXiv:2009.08369, 2020.

[15] D. Smilkov, N. Thorat, Y. Assogba, A. Yuan, N. Kreeger, P. Yu, K. Zhang, S. Cai, E. Nielsen, D. Soergel et al., "Tensorflow. js: Machine learning for the web and beyond," arXiv preprint arXiv:1901.05350 , 2019.

[16] Microsoft, "Onnx.js: run onnx models using javascript," https://github. com/microsoft/onnxjs, 2020.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., "Pytorch: An imperative style, high-performance deep learning library," in Advances in neural information processing systems, 2019, pp. 8026–8037.

[18] J. Bai, F. Lu, K. Zhang et al., "Onnx: Open neural network exchange," https://github.com/onnx/onnx, 2019.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., "Tensorflow: A system for largescale machine learning," in 12th {USENIX} symposium on operating systems design and implementation ({OSDI } 16), 2016, pp. 265–283.

[20] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "Wider face: A face detection benchmark," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 5525–5533.

[21] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei et al., "Masked face recognition dataset and application," arXiv preprint arXiv:2003.09093, 2020.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 6, pp. 1137–1149, 2016.

[23] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," arXiv preprint arXiv:1905.11946, 2019.

[24] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345– 1359, 2009.

[25] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[26] G. Jocher, "guigarfr, perry0418," Ttayu, Josh Veitch-Michaelis, Gabriel Bianconi, Fatih Baltacı, Daniel Suess, and WannaSeaU. ultralytics/yolov3: Video Inference, Transfer Learning Improvements, 2019.

[27] J. Schlemper, O. Oktay, L. Chen, J. Matthew, C. Knight, B. Kainz, B. Glocker, and D. Rueckert, "Attention-gated networks for improving ultrasound scan plane detection," arXiv preprint arXiv:1804.05338 , 20[28] M. Prabhakar A Novel Design and Development of The Conveyorcabinet with UV Lightchamber and Dry Fogging System for Decontamination of Objects for COVID -19 Safety, ICIRTCS-23, UGC Care, **ISSN / ISBN**978-93-91420-60-4pp.160