https://doi.org/10.2352/EI.2023.35.11.HPCI-229

WearMask: Fast In-browser Face Mask Detection with Serverless Edge Computing for COVID-19

Zekun Wang, Data Science Institute, Vanderbilt University, Nashville, TN
Pengwei Wang, Department of Information Science and Engineering, Shandong University, Qingdao, China (Corresponding author)
Peter C. Louis, Department of Biomedical Informatics, Vanderbilt University Medical Center, Nashville, TN
Lee E. Wheless, Department of Biomedical Informatics, Vanderbilt University Medical Center, Nashville, TN
Yuankai Huo, Department of Computer Science, Vanderbilt University, Nashville, TN

Abstract

The COVID-19 epidemic has been a significant healthcare challenge across the world. COVID-19 is transmitted predominately by respiratory droplets generated when people breathe, talk, cough, or sneeze. Wearing a mask is the primary, effective, and convenient method of blocking 80% of respiratory infections. Therefore, many face mask detection systems have been developed to supervise hospitals, airports, publication transportation, sports venues, and retail locations. However, the current commercial solutions are typically bundled with software or hardware, impeding public accessibility. In this paper, we propose an in-browser serverless edge-computing-based face mask detection solution, called Web-based efficient AI recognition of masks (WearMask), which can be deployed on common devices (e.g., cell phones, tablets, computers) with internet connections using web browsers. The serverless edge-computing design minimizes hardware costs (e.g., specific devices or cloud computing servers). It provides a holistic edge-computing framework for integrating (1) deep learning models (YOLO), (2) high-performance neural network inference computing framework (NCNN), and (3) a stack-based virtual machine (WebAssembly). For end-users, our solution has the advantages of (1) serverless edge-computing design with minimal device limitation and privacy risk, (2) installation-free deployment, (3) low computing requirements, and (4) high detection speed. Our application has been launched with public access at facemask-detection.com.

Introduction

Since November 2019, the COVID-19 epidemic had been a major social and healthcare issue in the United States. Only during 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 though many states 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 to 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 lightweight 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.

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. 1. 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. Most internet users are familiar with web browsers, which introduce almost no extra learning burdens for deploying our WearMask. We aggregate a holistic solution by combining serverless edge-computing and deep learning-based object detection, without 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(facemask-detection.com) (Fig. 1).

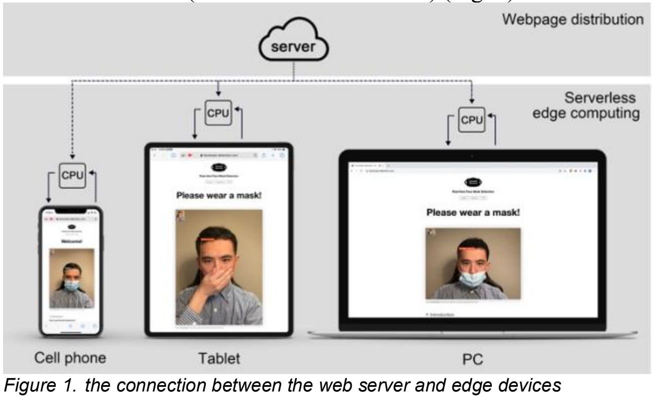


Figure 1. the connection between the web server and edge devices

Due to the lightweight device-agnostic design, the proposed method would be a cost-efficient solution for public facilities and small businesses. For example, setting up a commercial mask detection solution, on average 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 many public facilities, it is essential to find an affordable alternative.

Table 1. Solution comparison

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

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.

Background

Face Mask Detection

Historically, most of the papers focused on performing face recognition while wearing a mask or with other obstructions, while few previous studies have been conducted to determine if a subject is wearing a mask. Since the COVID-19 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 and others developed an alarm system for the wearing of 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 facemask-wearing, incorrect facemask-wearing, and correct facemask-wearing, 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. 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.

In-browser Serverless Edge Computing

The typical way of designing an 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 complete the inference by calling computing operations written in JavaScript. However, JavaScript is not a typical programming language in the deep learning field, with less community support.

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 completing 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. 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. 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.

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. For example, in our attempt, this process introduces the operator called

TensorScatterUpdate, which is not supported by this readable web format model.

Method

Data Collection

The data used to train the WearMask model consists of two types: (1) faces with masks, and (2) faces without masks. The normal face data (without masks) were collected from the WIDER FACE dataset [20]. The existing mask datasets were divided into two categories, including real faces with real masks (MAFA, RMFD [21], MMD) and real faces with generated masks (SMFD). Considering that the generated mask images influence model generalization ability, we used real images only. Briefly, the MAFA is composed of face pictures with various faces with heterogeneous masks, backgrounds, and annotation information. The RMFD only contains faces without background. Considering that the pictures taken by the camera would have a background in the real-world scenario, and the background images without faces would help the model improve precision, the MAFA dataset was finally selected as training data. However, the face in MAFA only contains the regions below the eyebrows, while the WIDER FACE dataset marks the whole face's position. To reconcile the differences between the annotation protocols of the two datasets, the partial annotations in WIDER FACE were extended to whole faces, following the protocol that the marked bounding box range was from below the hairline to above the lower edge of the chin, and the left and right borders do not include the ears. Moreover, we also collected some samples of incorrectly wearing masks and some edge conditions.

In the data annotation, we used the CDC mask guidance as a criterion to distinguish whether the mask was worn properly or not. The guidance requirements include: (1) must cover the nose, (2) must cover the mouth, (3) must fit under the chin. (4) must be snug on the face.

As a result, the final training dataset contained 4065 images from MAFA, 3894 images from WIDER FACE, and 1138 additional images from the internet. In general, a total of 9,097 images with 17,532 labeled boxes were divided into 80% training and validation, and 20% testing.

Modeling

To deploy deep learning models on edge devices without advanced GPUs, the efficiency and size of such models are essential. In general, the smaller the model is, the fewer computational resource is needed on edge devices. The YOLO-Fastest model is employed as the detection method in the WearMask solution, as a lightweight version of the You only look once (YOLO) [4] object detection model. YOLO is one of the most widely used fast object detection algorithms. As opposed to the traditional anchor-based detection algorithms (e.g., Faster-RCNN [22]), YOLO divides the image into predefined grids to match the target objects. The grid definition alleviates repetitive anchor computation, thus improving the computational efficiency dramatically. To further improve the computational speed, YOLO-Fastest was proposed as an open-source object detection model. In the YOLO-Fastest implementation, the EfficientNet-lite [23] is employed as the encoder. The total model size is only 1.3 MB (MegaByte), compatible with low computing power scenarios such as edge devices.

To further speed up the convergence of the training process, the COCO dataset (Microsoft Common Objects in Context, 80 categories) [23] was employed to pre-train the detection backbone. The model was then further trained by the face mask datasets as a transfer learning practice [24], by adjusting the network to suit new object definitions.

Serverless Edge-Computing

Moreover, as opposed to current cloud-computing based implementations, we implemented the WearMask as a serverless edge-computing framework. For face mask detection, the cloud computing solution is limited by the availability of enough internet bandwidth for real-time camera video streams, with high costs for cloud services. By contrast, our serverless edge-computing design minimizes hardware costs by utilizing users' existing devices. Beyond the lower hardware costs, the edge-computing design has another critical advantage of low privacy risk. In the WearMask framework, the video data are processed locally without uploading processes (e.g., upload to cloud servers), which is essential as a healthcare-related method.

In-browser Deployment

We further implement the edge-computing strategy as an in-browser deployment by aggregating NCNN and WebAssembly (WASM) architectures (Fig. 2). NCNN is an open-source optimized inference framework for mobile platforms developed by Tencent. It is implemented in pure C++ without third-party library dependencies. Herein, the size of the compiled model will be minimized, with decent computing efficiency on edge devices, especially on devices with ARM architecture chips. It supports custom layers, which means it works when there are custom operators in the model. Moreover, it has provided tools to convert from various frameworks to NCNN. WASM is a low-level language that runs in the browser. It is in binary form, which is faster than JavaScript programs. Moreover, prevalent web browsers, such as Chrome, Edge, Firefox, and WebKit (Safari), have already supported WASM with minimal generalizability barriers.

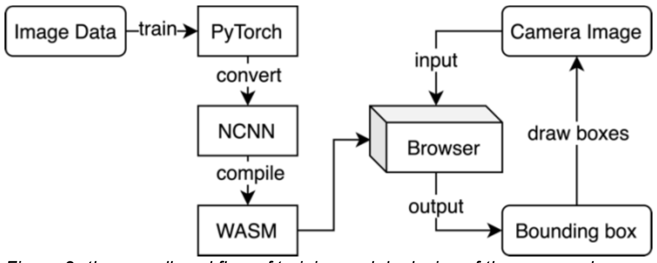


Figure 2. the overall workflow of training and deploying of the proposed WearMask framework

Using the NCNN library, we first converted the PyTorch model into an NCNN model with a model size of 581 KB. Then we created a new C++ program to streamline the detection process from image preprocessing to finally output the category, confidence, and box position, using the NCNN library for inference. After compiling this C++ program into WASM format, the entire framework was executed as a function in JavaScript. To visualize the detection results, we used HTML and CSS to render the detected bounding boxes with original face images.

Results

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 YOLO-V3 [25] from Ultralytics [26].

Results

The qualitative results of our proposed WearMask framework are presented in (Fig. 3). It can 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.

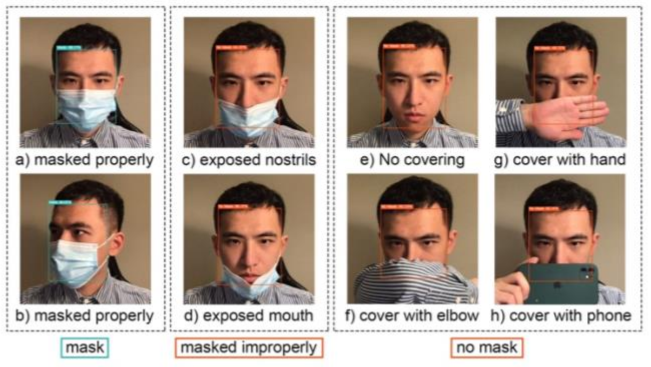


Figure 3. the different detection results. a) and b) show the cases with wear masks properly, c) and d) are the examples that wear mask improperly, and e), f), g) and h) are the cases of not wearing the mask.

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 is presented in Fig. 4.

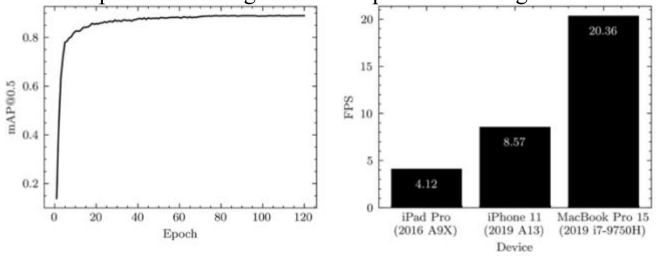


Figure 4. The left figure shows the testing mAP @ 0.5 of different training epochs. The right figure shows the different Frames Per Second (FPS) using different edge devices.

After converting the PyTorch model to an NCNN model with WASM, we established a website and published it at facemask-detection.com, to demo the in-browser deployment.

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, and (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). It shows that the performance of this model is at a comparable level with prior arts. Moreover, this study aims to build a lightweight edge computing framework for face mask detection rather than to achieve the best detection accuracy. More testing examples are presented in (Fig. 5).

Discussion

Current mask detection solutions typically require specialized equipment or dedicated environments for deployments, which are often unscalable, high-cost, or not flexible. To address such issues, we proposed a new edge-computing based face mask detection method, called WearMask, with the following features:

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. Easy deployments. The framework is device-agnostic (e.g., across computers, laptops, cell phones, or tablets) and compatible with major operating systems (e.g., Windows, macOS, Linux, Android, and iOS).
3. Installation free. No installation process and environmental settings are required to use the application. Users only need to visit our web page and enable camera permissions to trigger the software.
4. Low privacy risk. Since the program is run locally without exchanging data, there was no need to save any content or upload any data to the server. For privacy purposes, the model and data can be disconnected from the Internet after they are loaded.

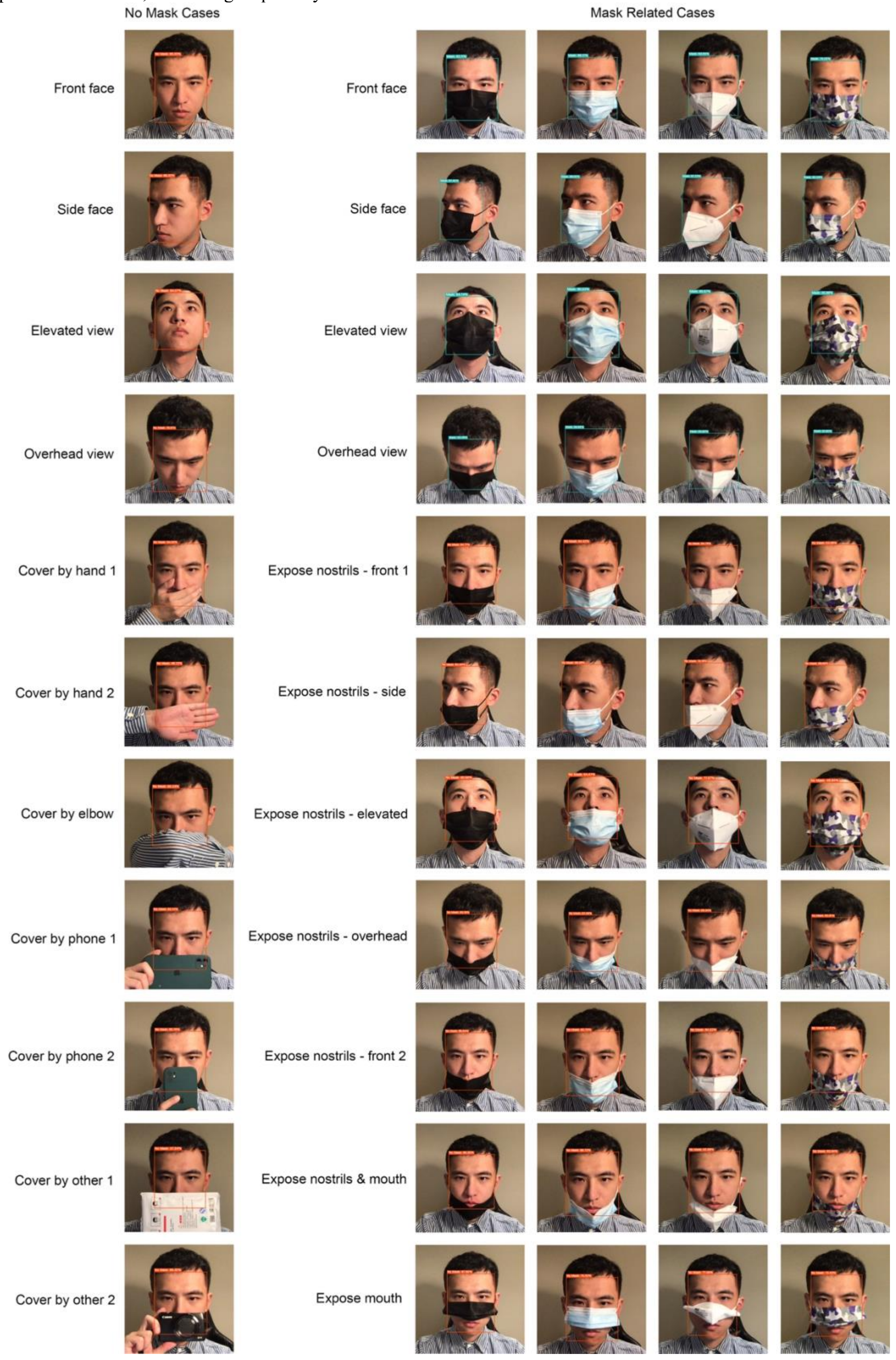
The proposed mask detection solution also had several limitations. Since most common devices do not support infrared detection, the program cannot detect human body temperature as professional equipment. As a reminder-only device, it cannot force a person to wear a mask if they insist on not wearing 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 where 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.

Conclusion

In this paper, we proposed a system agnostic no-installation face mask detection solution to remind people who are not wearing a mask or wearing a mask improperly. As a serverless edge-computing design, it can be run locally on various edge devices, with a low risk of privacy, low required network bandwidth, and low response time. The deployment scheme tackled insufficient support of deep learning from the JavaScript community, by aggregating NCNN and WASM. Our WearMask solution is a general framework where the detection algorithm can be replaced with other lightweight deep learning models. During the COVID-19 epidemic, our WearMask solution can monitor and remind people to Wear masks, alleviating respiratory infections.

Figure 5. More testing examples using the proposed WearMask tool masks. The results follow the CDC mask guide criteria.



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. Schu¨nemann, A. Elharakeh, A. Bognanni, T. Lotfi, M. Loeb, et al., The Lancet (2020)

[3] C. for Disease Control, Prevention, et al., Document modifi´e le **10** (2020)

[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, 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, M. Holman, Communications of the ACM 61(12), 107 (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, Z. Luo, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017), pp. 2682-2690

[10] A. Nieto-Rodr´ıguez, M. Mucientes, V.M. Brea, in *Iberian Conference on Pattern Recognition and Image Analysis* (Springer, 2015), pp. 138–145

[11] B. QIN, D. LI, Sensors (Basel, Switzerland) (2020). DOI 10.3390/s20185236

[12] M. Loey, G. Manogaran, M.H.N. Taha, N.E.M. Khalifa, Measurement 167, 108288 (2020)

[13] M. Loey, G. Manogaran, M.H.N. Taha, N.E.M. Khalifa, Sustainable Cities and Society p. 102600 (2020)

[14] G.J. Chowdary, N.S. Punn, S.K. Sonbhadra, S. Agarwal, 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., 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., 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., in 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16) (2016), pp. 265-283

[20] S. Yang, P. Luo, C.C. Loy, X. Tang, 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., arXiv preprint arXiv:2003.09093 (2020)

[22] S. Ren, K. He, R. Girshick, J. Sun, IEEE transactions on pattern analysis and machine intelligence 39(6), 1137 (2016)

[23] M. Tan, Q.V. Le, arXiv preprint arXiv:1905.11946 (2019)

[24] S.J. Pan, Q. Yang, IEEE Transactions on knowledge and data engineering 22(10), 1345 (2009)

[25] J. Redmon, A. Farhadi, arXiv preprint arXiv:1804.02767 (2018)

[26] G. Jocher, Ttayu, Josh Veitch-Michaelis, Gabriel Bianconi, Fatih Baltac 谋, 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, D. Rueckert, arXiv preprint arXiv:1804.05338 (2018)

Author Biography

Zekun Wang received her BS in statistics from the Central South University (2018) and his master's degree in Data Science from Vanderbilt University (2021). Since then, he has worked as a Research Engineer at Indeed, Inc. His work focused on applied machine learning solutions for small businesses.