A Novel IoT-Enabled System for Real-Time Face Mask Recognition Based on Petri Nets

Victor R.L. Shen [1]

[1]National Taipei University

November 7, 2023

Abstract

Due to Coronavirus Disease 2019 (COVID-19), many countries have formulated pandemic prevention regulations, requiring the masses to wear a face mask before entering public places and taking public transportations. However, if the entrances of some places are manually checked to see whether people are wearing a face mask or not, it becomes not only labor-intensive and time-consuming, but also inefficiently checking each passer-by. Therefore, this paper aims to develop a face mask recognition system based on an edge computing platform. The traditional manual inspection control method is replaced by artificial intelligence (AI) technology to achieve automatic recognition and control. As an edge computing platform, Jetson Nano is an embedded system equipped with an AI platform, which can be used for object detection and image classification. Developed by Ultralytics LLC, a YOLOv5 model using the PyTorch framework runs on the edge computing platform, featuring high speed, high precision, and small size. According to the model training results, the average precision (AP) reaches 95.41%, while the mean average precision (mAP) records 94.42%. The average single-class running time is 0.016 seconds, and the file size of training model is 3.8MB. The recognition distance is up to 8m, and the maximum face rotation angle is 90°. In addition, a Petri net software tool, WoPeD, with graphical features based on mathematical theories, is used to verify the mask recognition system; and ensures the system has acceptable precision and recall values.

A Novel IoT-Enabled System for Real-Time Face Mask Recognition Based on Petri Nets

Cheng-Ying Yang, *Member, IEEE*, Yi-Nan Lin, Victor R.L. Shen, *Senior Member, IEEE*, Frank H.C. Shen, and Chien-Chi Wang

***Abstract*—Due to Coronavirus Disease 2019 (COVID-19), many countries have formulated pandemic prevention regulations, requiring the masses to wear a face mask before entering public places and taking public transportations. However, if the entrances of some places are manually checked to see whether people are wearing a face mask or not, it becomes not only labor-intensive and time-consuming, but also inefficiently checking each passer-by. Therefore, this paper aims to develop a face mask recognition system based on an edge computing platform. The traditional manual inspection control method is replaced by artificial intelligence (AI) technology to achieve automatic recognition and control. As an edge computing platform, Jetson Nano is an embedded system equipped with an AI platform, which can be used for object detection and image classification. Developed by Ultralytics LLC, a YOLOv5 model using the PyTorch framework runs on the edge computing platform, featuring high speed, high precision, and small size. According to the model training results, the average precision (*AP*) reaches 95.41%, while the mean average precision (*mAP*) records 94.42%. The average single-class running time is 0.016 seconds, and the file size of training model is 3.8MB. The recognition distance is up to 8m, and the maximum face rotation angle is 90°. In addition, a Petri net software tool, WoPeD, with graphical features based on mathematical theories, is used to verify the mask recognition system; and ensures the system has acceptable precision and recall values.**

***Index Terms*—Edge Computing, Face Mask Recognition, Object Detection, Petri Net, YOLOv5.**

I. INTRODUCTION

SINCE the outbreak of COVID-19 in 2019, massive public places have facilitated the spread of virus, which is infectious through air, droplet, and contact. To monitor the rampant pandemic that causes severe complications or death, governments have introduced various control regulations so that everyone must carry out a mandate to wear a face mask in public places and to keep social distancing [1]. All designated personnel have been assigned at the entrances of many buildings to check face mask wearing, which leads to additional management burdens and costs [2-3].

In the era of Internet-of-Things (IoT), a lot of sensors and devices collect and process datasets from the surrounding environment, transmit them to cloud centers, and receive feedback signals over the Internet for connection and perception. But transmitting massive amounts of heterogeneous datasets, perceiving complex environments from these datasets, and then making smart decisions in a timely manner are even more difficult [4].

Currently, artificial intelligence (AI), especially deep learning, gains a significant success in various areas, such as speech recognition, computer vision, and natural language processing (NLP). The integration of AI with IoT thus evolves into the era of AI of Things (AIoT). This study has selected the YOLO model and Jetson Nano as its training model and edge computing platform, respectively. AI-Enabled algorithms have advantages of high precision, rapid recognition, small size, and lower costs [5].

This paper aims to replace human supervision in public places or at the entrance to public transportations with an AI-Based platform, which enables automatic face mask detection and recognition. Therefore, the edge computing system should be empowered by strong computing capabilities, whereas a deep learning model featuring efficiency and precision is preferred to pave the way for the face mask recognition system on the edge computing platform [6].

Problem Statement: Nowadays, the real-time face mask recognition methods are not modeled or verified for the soundness and integrity of the system design workflow [4]. Hence, it remains to be proven whether their system models are feasible or not. Additionally, the edge computing systems are troubled by inaccurate recognition and short distance due to the lack of AI-Based systems and the burden of large file size for deep learning [5]. Consequently, this paper proposes a smart face mask recognition system to solve the problems mentioned above.

The remainder of this paper is organized as follows:

This work was supported in part by the Ministry of Science and Technology, Taiwan, under grants MOST 110-2637-E-131-005 and MOST 111-2221-E-845-003-.

Cheng-Ying Yang is with the Department of Computer Science, University of Taipei, Taipei City, Taiwan (e-mail: cyang@utaipei.edu.tw).

Yi-Nan Lin is with the Department of Electronic Engineering, Ming Chi University of Technology, New Taipei City, Taiwan (e-mail: jnlin@mail.mcut.edu.tw).

Victor R. L. Shen is with the Department of Information Management, Chaoyang University of Technology, Taichung City; and Department of Computer Science and Information Engineering, National Taipei University, New Taipei City, Taiwan (email: rlshen@mail.ntpu.edu.tw).

Frank H.C. Shen is with the Department of Electronic Engineering, Fu Jen Catholic University, New Taipei City, Taiwan (email: frank50391@hotmail.com).

Chien-Chi Wang is with the Department of Electronic Engineering, Ming Chi University of Technology, New Taipei City, Taiwan (email: lky7956@gmail.com).

Section II discusses the literature review of IoT-Enabled systems. The IoT-Enabled system with hardware components and software tools are described in Section III. The system verification and performance evaluation with experimental results are presented in Section IV. Finally, the conclusion is remarked in Section V.

II. Literature Review

This section focuses on the studies and applications related to this paper, including the YOLO model for deep learning, software tools, Petri net theory, and related works.

A. YOLO Model

Proposed by Joseph Redmon, et al. in 2016, You Only Look Once (YOLO) is a neural network algorithm designed to detect objects through computer vision and pattern recognition (CVPR) [7-8]. YOLO contains three basic testing steps: The first step is to adjust images input in the pixel format of 448 x 448. In the second step, these adjusted images are examined on the convolutional neural network (CNN) to predict the number of categories and calculate their probability. In the third step, the recognition results are exported according to the calculated confidence level.

When detecting objects, as an end-to-end computing and training model, YOLO gathers all images on a single CNN and predicts the probability of each category in the whole image. With each image cut into grids for measuring S x S pixels, YOLO then detects the objects in all grids to predict their bounding boxes that have five datasets each, including the central coordinates (X and Y) of the grid boundaries, the width and height (w and h) of the whole image, and the confidence level. Each grid also predicts the probability of each category. Lastly, the recognition results are obtained based on the bounding box of each grid and the probability of each category.

The YOLO architecture contains 24 convolutional layers and 2 fully connected layers. The front layers identify the category characteristics of images, while the rear layers export the predicted coordinates and probabilities. YOLO distinguishes itself from its reference GoogLeNet model by using 3 x 3 convolutional layers and 1x1 fully connected layers to reduce dimensions for a smaller computing amount.

B. Software—Anaconda and LableImg

As one of the most popular programming languages, Python can be applied to AI, deep learning, and the Internet of Things (IoT). With many development modules, the software is available for use after being downloaded through the instruction, *pip install*. However, as many downloads are often interrupted by issues of compatibility and the installation order of modules, their solutions may take time and need more efforts. Hence, Anaconda has been launched as open-sourced and free software that can save time from downloading by simplifying environment installation with several setup packages and capabilities of supporting multitask operating systems [9]. Its rich and complete tools allow users to finish implementing their programs or studies more conveniently.

Often used to make datasets for deep learning, LableImg is an image annotation tool that labels the object bounding boxes in images according to object characteristics [10]. It can label multiple types of characteristics and export them as Pascal VOC (i.e. a file type in XML), Create ML (i.e. a file type in json), or YOLO (i.e. a file type in txt).

C. Petri Net Theory

Proposed by a German mathematician, Dr. Carl Adam Petri, in 1962, Petri net is a system modeling and simulation tool based on mathematical theories and graphical characteristics, primarily used for the analysis of workflow networks [11-12]. As more experts and scholars have been engaged in relevant studies, the theory with applications of Petri net (PN) has widely evolved into more sub-classes, such as colored Petri net, timed Petri net, and fuzzy Petri net [13].

Petri net is composed of four graphical elements representing the operation of a system workflow, including *place*, *transition*, *arc*, and *token*, as listed in Table I.

TABLE I
Basic elements of Petri net

Element	Notation	Function
Place	○	Represents the state of objects or events in a system.
Transition	□	Represents changes in the state of objects or events in a system.
Arc	→	Represents the transition direction of objects or events in a system.
Token	●	Represents a moving object or event in a system.

The basic elements of Petri net are defined as follows [14]:

Petri Net $= \{P, T, F, W, M\}$

$P = \{p_1, p_2, p_3, \ldots, p_n\}$ denotes a finite set of places.

$T = \{t_1, t_2, t_3, \ldots, t_n\}$ denotes a finite set of transitions.

$F \subseteq (P \times T) \cup (T \times P)$ denotes a finite set of arcs, also known as flow relation.

$W = F \rightarrow \{0, 1, 2, 3, \ldots, n\}$ denotes a weight on an arc.

$M = P \rightarrow \{0, 1, 2, 3, \ldots, n\}$ denotes the number of tokens on a place.

Each arc has a preset weight (i.e. initially set as 1), and a token is an object on a place. Multiple tokens can be located on the same place. When one or several tokens are found in one input place and consistent with the weight on an arc, the tokens may be triggered and transitioned to the output place. Different combinational architectures have different implications.

D. Related Works

Both Chen [4] and Tsai [5] proposed a face mask recognition system and trained a model through deep learning on an edge computing platform to recognize face masks in real time. However, as the soundness and integrity of the system workflow have not been verified yet, the feasibility of their system development cannot be guaranteed. Since their edge computing platform is not equipped with the AI-Based system, the real-time function of image recognition is troubled by larger model volume and shorter recognition distance. Liu [6] adopted the face recognition of OpenCV and found that the system lacked enough memory size when operated on the Jetson Nano system, despite the overall favorable recognition.

III. Proposed System

This section presents the software and hardware requirements, system environment, and system verification tool. The software includes LableImg, YOLOv5, and Petri nets; and the hardware includes a personal computer as a model training platform.

A. System Architecture

The flowchart of model training falls into two parts, including *gathering the datasets of mask-wearing images* and *training the YOLOv5 model*, as shown in Fig. 1. First, this study gathered mask-wearing images by shooting via the Internet and set the parameters and the proportions of datasets, namely, the training datasets with 75%, the verification datasets with 20%, and the testing datasets with 5%, before YOLOv5 model training is performed. Characteristics of the image datasets are labeled with LabelImg to produce files in the YOLO format. With these files prepared, the YOLOv5 model for real-time face mask recognition can be trained. The model training led to the weighted files and training results [15-16]. The system design flowchart is divided into three parts, namely, *real-time image transmission*, *face mask recognition*, and *the analysis of recognition results*. The recognition results are output to the display through the face mask recognition system, as shown in Fig. 2.

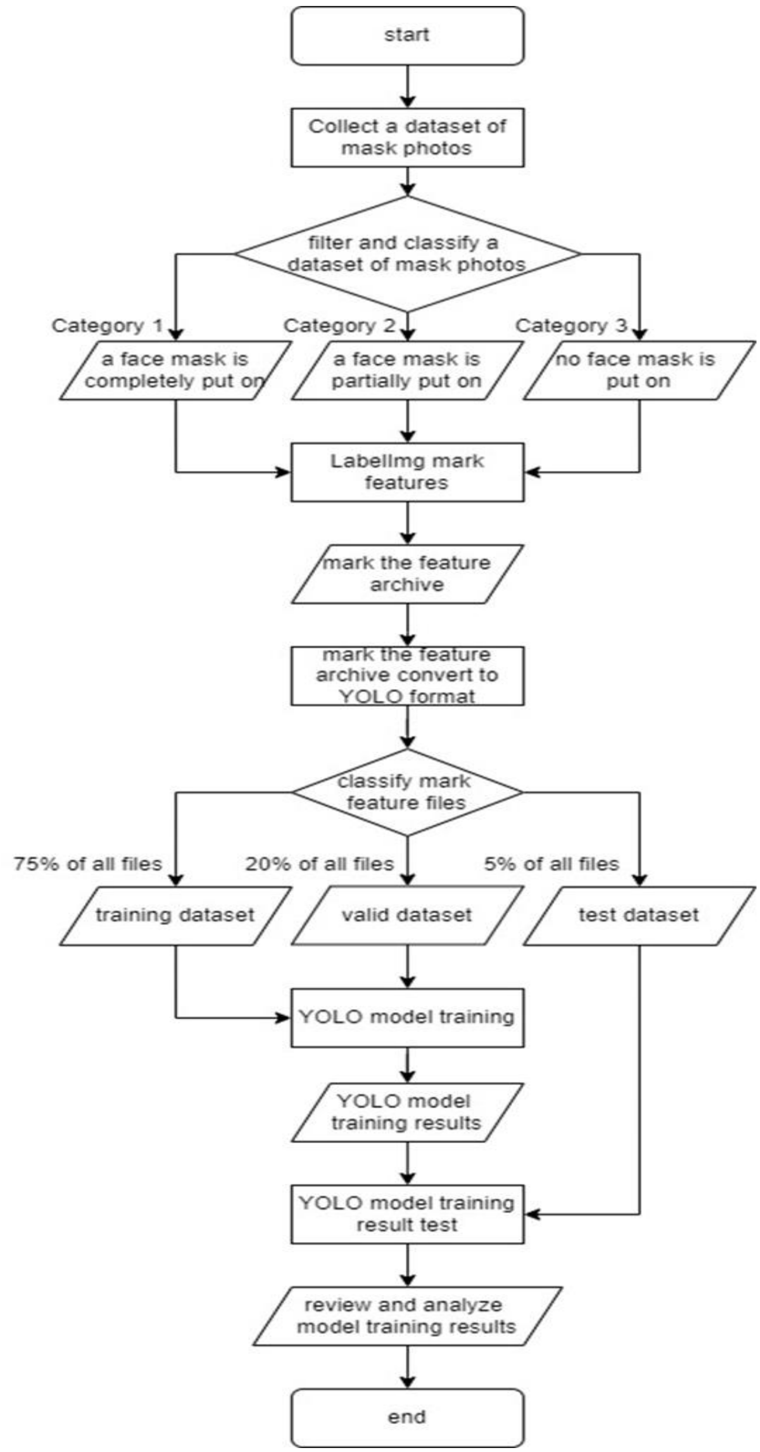

Fig. 1. Flowchart of model training.

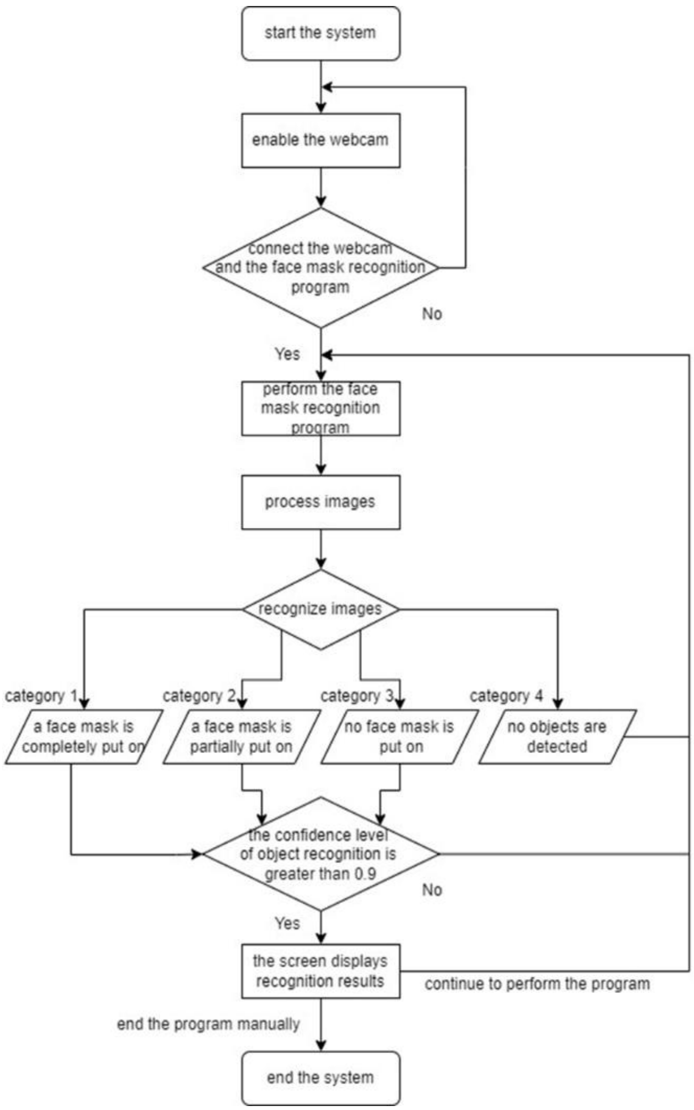

Fig. 2. Flowchart of system modeling.

B. Software Architecture

LableImg (software), YOLOv5 (the model), and Petri net (the simulation tool) were used. Specifically, LableImg serves to draw the bounding boxes of face masks, YOLOv5 acts to train the face mask recognition system, and Petri net acts to verify the feasibility and integrity of the system design workflow.

1) **Image Format of LabelImg**

Before drawing bounding boxes, category names should be entered in the file called "classes. txt". Three categories of characteristics are covered, namely, "with_mask (i.e. a face mask is completely put on)", "mask_incorrect (i.e.a face mask is partially put on)", and "without_mask (i.e. no face mask is put on)". Files were saved in the YOLO format, including the categories of bounding boxes, as well as the central coordinates, *X* and *Y*, and the corresponding width and height of normalized bounding boxes.

The coordinates of bounding boxes labeled with LableImg are shown in Fig. 3; and the calculation formulas and definitions of the bounding boxes are presented in Eqns. (1-6).

$$w = Xmax - Xmin \tag{1}$$
$$h = Ymax - Ymin \tag{2}$$
$$X_{center} = \frac{(X\,min + X\,max\,)}{2} * \frac{1}{w} \tag{3}$$
$$Y_{center} = \frac{(Y\,min + Y\,max\,)}{2} * \frac{1}{h} \tag{4}$$
$$\mathrm{YOLO}_{\ w} = \frac{(X\,max - Xmin\,)}{w} \tag{5}$$
$$\mathrm{YOLO}_{\ h} = \frac{(Y\,max - Ymin\,)}{h} \tag{6}$$

$Xmax$: Maximum *x*-coordinate of a bounding box
$Xmin$: Minimum *x*-coordinate of a bounding box
$Ymax$: Maximum *y*-coordinate of a bounding box
$Ymin$: Minimum *y*-coordinate of a bounding box
w: Width of a bounding box
h: Height of a bounding box
X_{center}: Central coordinate *X* of a normalized bounding box
Y_{center}: Central coordinate *Y* of a normalized bounding box
YOLO_{w}: Width of a normalized bounding box
YOLO_{h}: Height of a normalized bounding box

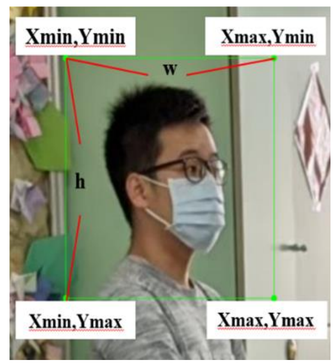

Fig. 3. Coordinates of a bounding box labeled with LableImg.

2) **Simulation Tool – WoPeD**

The Petri net simulation tool is Workflow Petri Net Designer (WoPeD) [17-18], an open-sourced software system developed by Baden-Wurttemberg Cooperative State University Karlsruhe in line with GNU Lesser General Public License (LGPL). It was mainly designed to model, simulate, and analyze the workflow networks. In addition to WoPeD, other software tools include Platform Independent Petri net Editor (PIPE) [19] and HPetriSim (HPSIM) [20], etc.

C. Hardware Architecture

A model training platform, an edge computing platform, and a webcam were deployed. To be specific, a personal computer was used as a training platform; Jetson Nano was used as an edge computing platform for running the face mask recognition program; and a webcam was used for transmitting images to the image recognition system.

1) **Model Training Platform**

The hardware requirements of a personal computer used as the model training platform are listed in Table II. With a highly efficient CPU and a graphic processor with high computing capabilities, the YOLOv5 deep learning model was trained for the face mask recognition.

TABLE II
REQUIREMENTS OF THE PLATFORM FOR MODEL TRAINING

Item	Specification
CPU	AMD Ryzen™ 5 3600 (3.6GHz)
Graphic Processor	NVIDIA® GeForce RTX™ 2060
Memory	32GB
Operating System	Windows

2) **Edge Computing Platform**
Nvidia Developer Jetson-Nano-Developer-Kit [21] acts as an edge computing platform for face mask recognition, whose hardware requirements are listed in Table III. Jetson Nano is an embedded module as AI platform, which can be applied to AI algorithms for object detection and image classification [22-23].

TABLE III
REQUIREMENTS OF JETSON NANO (

Item	Specification
CPU	Quad-core ARM A57 (1.43 GHz)
Graphic Processor	128-core NVIDIA Maxwell™
Memory	4GB 64-bit LPDDR4 25.6GB/s
Display	HDMI and DP
USB	4 x USB 3.0 and 1 x USB 2.0 Micro-B
Size	100 x 80 x 29mm
Operating System	Ubuntu

3) **Webcam**
HP w200 [24] was adopted as the webcam transmitting real-time images, featuring excellent compatibility and ease of use (with a plug-and-play USB), whose hardware requirements are listed in Table IV.

TABLE IV
REQUIREMENTS OF HP W200 (WEBCAM)

Item	Specification
Image Resolution	1,280 x 720p, max. 30fps
Wide Angle	D: 68.6°
Zooming Type	Fixed focus
Focal Length	60cm to infinity
Connection Type	USB 2.0
Size	80 x 28.5 x 24mm

IV. SYSTEM VERIFICATION AND MAIN RESULTS

Based on the system architecture specified above, this section discusses the experimental results and uses the Petri net simulation tool to model, verify, and analyze the system architecture.

A. Modeling and Verification of Petri Net

Based on the flowchart of system architecture, the system built a Petri net model and used 18 places, 23 transitions, and 46 arcs. Figs. 4 and 5 show the semantical analysis results and the PN modeling of system design, respectively. The verification results revealed that the system design flow is completely consistent with Petri net’s structural properties with soundness and integrity without any errors. The interpretation of places and transitions is listed in Table V.

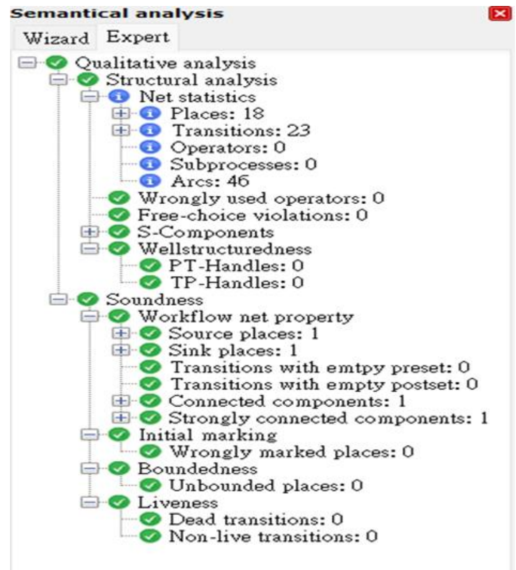

Fig. 4. Verification results of Petri net model.

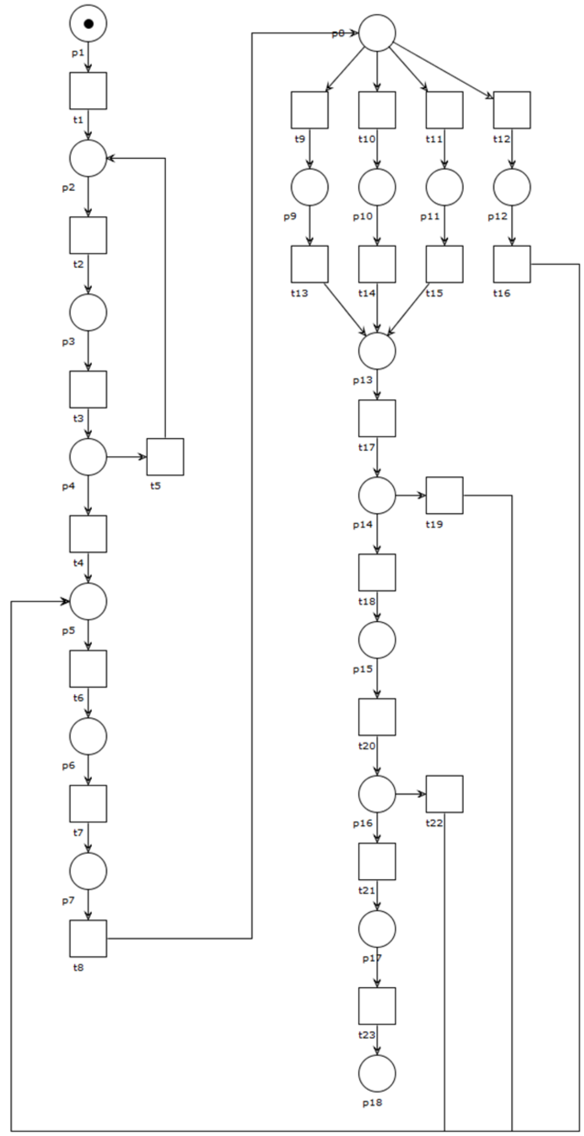

Fig. 5. Petri net modeling the flowchart of system design.

TABLE V
INTERPRETATION OF PLACES AND TRANSITIONS

Place	Interpretation	Transition	Interpretation
p_1	enable Jetson Nano	t_1	start the system
p_2	run Jetson Nano	t_2	enable the webcam
p_3	run the face mask recognition program (detection hardware)	t_3	connect the webcam and the face mask recognition program
p_4	the connection between the webcam and the program	t_4	Yes
p_5	enter real-time images from the webcam	t_5	No
p_6	run the face mask recognition program--recognize categories	t_6	perform the face mask recognition
p_7	generate pre-processed results	t_7	process images
p_8	generate the category results of face mask recognition	t_8	recognize images
p_9	generate the category of with_mask	t_9	category 1
p_{10}	generate the category of mask_incorrect	t_{10}	category 2
p_{11}	generate the category of without_mask	t_{11}	category 3
p_{12}	generate the category of detecting no objects	t_{12}	category 4
p_{13}	filter the confidence level of object recognition	t_{13}	a face mask is completely put on
p_{14}	generate the results of filtering the confidence level of object recognition	t_{14}	a face mask is partially put on
p_{15}	the category of object recognition is correct	t_{15}	no face mask is put on
p_{16}	choose the recognition category in the output	t_{16}	no objects are detected
p_{17}	end the face mask recognition program	t_{17}	the confidence level of object recognition is more than 0.9
p_{18}	turn off Jetson Nano	t_{18}	Yes
		t_{19}	No
		t_{20}	the screen displays recognition results
		t_{21}	end the program manually
		t_{22}	continue to run the program
		t_{23}	end the system

The Petri net modeling is explained as follows: The start system of the PN model is represented by the initial marking of the place p_1 containing one token. This enables the firing of the transition t_1 which moves the token from place p_1 to place p_2. First, start with p_1 (enable Jetson Nano) to fire t_1 (start the system), followed by p_2 (run Jetson Nano) to fire t_2 (enable the webcam). Place p_3 (run the face mask recognition program-detect hardware) is to fire t_3 (connect the webcam and the face mask recognition program) and to evaluate p_4 (the connection between the webcam and the program). Connection success guides to fire t_4 (Yes), through p_5 (enter real-time images from the webcam); otherwise, connection failure leads to firing t_5 (No) and returns to p_2 to reconnect the webcam. After firing t_6 (perform the face mask recognition program), place p_6 (run the face mask recognition program--recognize categories) is activated and followed by p_7 (generate pre-processed results) after firing t_7 (process images). With t_8 (recognize images) fired, place p_8 (generate the category results of face mask recognition) is found. For example, if t_9 (Category 1) is fired, place p_9 (generate the category of with_mask) will go to fire t_{13} (a face mask is completely put on). If t_{10} (Category 2) is fired, place p_{10} (generate the category of mask_incorrect) will be operated to result in firing t_{14} (a face mask is partially put on). If t_{11} (Category 3) is fired, place p_{11} (generate the category of without_mask) will be operated to result in firing t_{15} (no face mask is put on). If t_{12} (Category 4) is fired, p_{12} (generate the category of detecting no objects) will be operated to fire t_{16} (no objects are detected), and the system will return to p_5, where images are re-entered. After firing t_{13}, t_{14}, or t_{15}, p_{13} (filter the confidence level of object recognition) will be conducted. Firing t_{17} (the confidence level of object recognition is more than 0.9) is followed by p_{14} (generate the results of filtering the confidence level of object recognition). If the confidence level is more than 0.9, the system will fire t_{18} (Yes) and then go through p_{15} (the category of object recognition is correct). Otherwise, the system will fire t_{19} (No) and return to p_5 to re-enter images. Firing t_{20} (the screen displays recognition results) is followed by p_{16} (choose the recognition category in the output end). To end the recognition program, the system will fire t_{21} (end the program manually) and go through p_{17} (end the face mask recognition); otherwise, it will return to p_5 to input images again after firing t_{22} (continue to perform the program). Finally, the system will fire t_{23} (end the system) to reach p_{18} (turn off Jetson Nano).

B. Experimental Results

The training results of YOLOv5 model indicates that the system could recognize face masks in the proximity and in well-lit conditions, with the confidence level more than 90%, as shown in Figures 6-8. It is still working well to recognize face masks even when a hand covers the face or there are multiple persons in the image, as shown in Figure 9. To examine the scenarios under different conditions, this paper adjusted distance and lighting conditions, and found that the face angle, coupled with the former two parameters, might cause errors or failure in face mask recognition. For example, the dark places at night, the distance from the camera, and the face rotation angle, all of which are prone to causing the system failure in recognizing face masks.

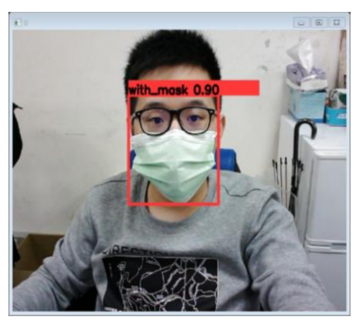

Fig. 6. With_mask.

Fig. 7. Mask_incorrect.

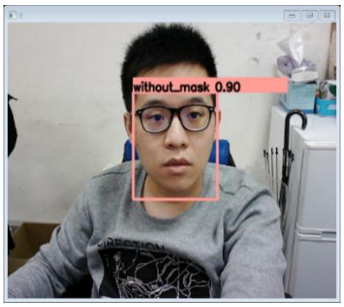

Fig. 8. Without mask.

Fig. 9. The face covered with a hand (left) and recognition of more than one face (right).

Take the face angle for example. The system cannot recognize face masks when people rotate their face by more than 90°, no matter how they wear a face mask, as shown in Figs. 10 and 11.

Fig. 10. Face rotation by degrees less than 90° (left) and more than 90° (right) with a face mask.

Fig. 11. Face rotation by degrees less than 90° (left) and more than 90° (right) without a face mask.

When the sufficient lighting power density reaches 115 Lux at different distances, the results of face mask recognition with or without a face mask are tested in Cases 1 through 8. When the face is located at more than 8m away from the camera, the system cannot recognize the face masks.

Case 1: 1m (left) and 2m (right) away from the camera with a face mask.
Case 2: 3m (left) and 4m (right) away from the camera with a face mask.
Case 3: 5m (left) and 6m (right) away from the camera with a face mask.
Case 4: 7m (left) and 8m (right) away from the camera with a face mask.
Case 5: 1m (left) and 2m (right) away from the camera without a face mask.
Case 6: 3m (left) and 4m (right) away from the camera without a face mask.
Case 7: 5m (left) and 6m (right) away from the camera without a face mask.
Case 8: 7m (left) and 8m (right) away from the camera without a face mask.

When the insufficient lighting power density reaches 3 Lux at different distances, the recognition results of with or without a face mask are tested in Cases 9 through 14.
Case 9: 1m (left) and 2m (right) away from the camera in badly-lit condition.
Case 10: 3m (left) and 4m (right) away from the camera in badly-lit condition.
Case 11: 5m away from the camera with a face mask in badly-lit condition.
Case 12: 1m (left) and 2m (right) away from the camera without a face mask in badly-lit condition.
Case 13: 3m (left) and 4m (right) away from the camera without a face mask in badly-lit condition.
Case 14: 5m away from the camera without a face mask in badly-lit condition.

The optimal training results of all categories are listed in Table VI for precision and listed in Table VII for recall, respectively; and their calculation formulas are defined in Eqns. (7-8) [3].

$$\text{Precision} = \frac{TP}{TP+FP} \tag{7}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{8}$$

where *T* (True) denotes the correct recognition.
F (False) denotes the incorrect recognition.
P (Positive) denotes the positive recognition.
N (Negative) denotes s the negative recognition.
TP (True Positive) denotes the recognition (positive) consistent with the actual situation.
FP (False Positive) denotes the recognition different from the actual situation.
FN (False Negative) denotes the recognition (negative) different from the actual situation.

TABLE VI
TRAINING RESULTS OF THE YOLOV5 MODEL FOR PRECISION

Category	***TP***	***FP***	**Precision**	**Average Precision**
With_mask	307	12	0.962	0.928
Mask_incorrect	37	8	0.822	
Without_mask	11	0	1	

TABLE VII
TRAINING RESULTS OF THE YOLOV5 MODEL FOR RECALL

Category	***TP***	***FP***	**Precision**	**Average Precision**
With_mask	307	43	0.877	0.763
Mask_incorrect	37	22	0.627	
Without_mask	11	3	0.785	

The YOLOv5 model is trained 300 times, 207 of which have the optimal training results, with precision, 0.928, and recall, 0.763. The precision and recall are shown in Figs. 12 and 13, respectively.

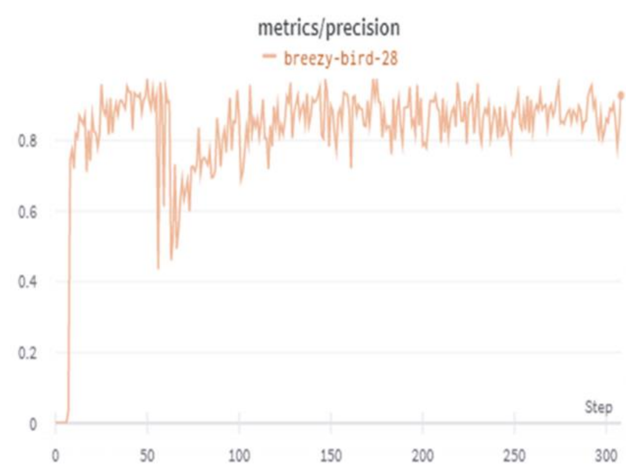

Fig. 12. Training results of the YOLOv5 model for precision.

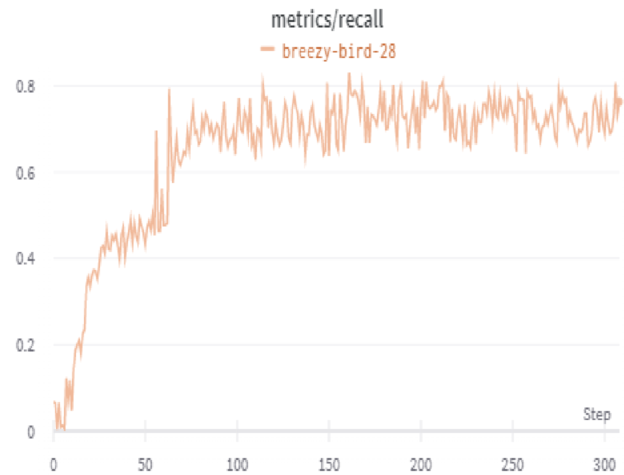

Fig. 13. Training results of the YOLOv5 model for recall.

Average Precision (*AP*) represents the average precision of a single category and includes the recognition results of multiple images instead of a single image. Mean Average Precision (*mAP)* denotes the average precision of all categories after their *AP* values are averaged. In terms of the model training results, the *AP* value with_mask is 0.944, while the *AP* value without_mask is 0.755. The *AP* and *mAP* values are 0.876 and 0.858, respectively, for the mask_incorrect cases. All the experimental results are shown in Fig. 14.

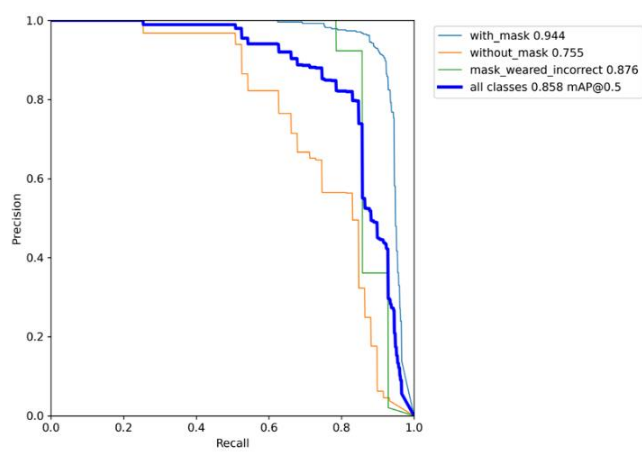

Fig. 14. Training results of the YOLOv5 model for *AP* values of all categories.

C. Functional comparison

To fully support the claim that the proposed face mask recognition system is more feasible and acceptable than other existing ones, different research methods were compared with one another, including the proposed one, C.-W. Chen [4], M.-T. Tsai [5], X. Kong, et al. [23], and Y. Chen, et al. [25]. The results of functional comparison are all listed in Table VIII. Compared with other existing systems, this study has obtained the promising *AP* and *mAP* values after training the model; and the running time takes 0.016 seconds on average in each category. The recognition distance is 8m; and the maximum face rotation angle reaches 90°.

In addition, the PN model was used as a verification mechanism, ensuring the feasibility of the entire system. This allows users to gain more stable execution when using the YOLOv5 models. Moreover, PN model was used to verify that the system design workflow was sound with integrity [25]. The main results show that the average precision is as high as 94.42% with better performance.

TABLE VIII
Comparison with previous studies

Method / Feature	**Proposed** (%)	C.-W. Chen [4]	M.-T. Tsai [5]	X. Kong et al. [23]	Y. Chen et al. [25]
AP value of with_mask %	95.41	89.12	88.92	90.37	91.27
AP value of without_mask %	90.52	85.12	84.93	89.76	90.18
AP value of mask_incorrect %	93.41	87.22	86.72	89.97	90.86
mAP %	94.42	84.67	85.56	89.65	90.54
Recall %	92.31	82.16	83.65	86.14	88.13
Development board	Nvidia Jetson-Nano-Developer-Kit 4GB	Raspberry Pi 4 Model B 8GB	Nvidia Jetson-Nano-Developer-Kit 4GB	ECMask inspired by FaceBoxes	Wechat Applet with Matlab
Training model	YOLOv5	YOLOv4	YOLOv3 tiny	OpenVINO	Matlab
Number of frames	63	22	56	61	62
Recognition distance	~ 8m	~ 6m	~ 6m	~ 8m	~ 7m
Face rotation angle	90°	75°	70°	80°	90°
Lighting	Recognition success in the condition of 3 Lux	Recognition success in dim lighting	Recognition success in dim lighting	Recognition success in dim lighting	Recognition success in dim lighting
Petri net verification	Yes	N/A	N/A	N/A	N/A

"N/A" denotes unavailable or No.

V. Conclusion

The smart face mask recognition system has been successfully implemented on an edge computing platform, empowered by AI-Based algorithms. It can recognize whether a face mask is worn in a highly precise manner, featuring lower costs, high speed, and small size. The contributions of this system are summarized as follows:

1. Petri net was used for system modeling and verification. The verification results are totally consistent with the basic properties of Petri nets, indicating that the system has soundness and integrity without any errors caused in its system design workflow. If the system has no errors, then it will accelerate the system production.
2. The low-cost (~ 3,000 NTD) NVIDIA Jetson Nano is adopted, which is an embedded module equipped with AI-Enabled platform. In addition, the YOLOv5 model has the following advantages:
   (1) High precision: *AP* value is 95.41% and *mAP* value is 94.42%.
   (2) High recognition speed: The running time reaches 0.016 second on average in each category.
   (3) Small file size: The training results show that the file size is 3.8MB.
   (4) Far recognition distance: The recognition distance is 8m, and the face rotation angle is 90°.
3. Compared with the previous studies, the promising *AP* and *mAP* values of the trained system are obtained. Meanwhile, its overall recognition speed is faster by 30%, and the recognition distance is farther by 1m, and the face rotation angle is wider by 15°, all of the performance metrics outperform other studies.

However, the following factors may cause the failure in the real-time face mask recognition system.

1. Lighting conditions: The lighting power density is less than 3 Lux.
2. Recognition distance: More than 8m is away from the camera.
3. Face rotation angle: The face rotation angle is more than 90°.

To overcome the above difficulties, the future studies are required to include more datasets regarding the environment with better lighting conditions and the webcams with higher resolution.

Acknowledgment

This work was supported by the Ministry of Science and Technology, Taiwan, under grants 110-2637-E-131-005- and MOST 111-2221-E-845-003-.

References

[1] I. B. Venkateswarlu, J. Kakarla, and S. Prakash, "Face mask detection using MobileNet and global pooling block," *Procs. of 2020 IEEE 4th Conference on Information & Communication Technology (CICT)*, pp. 1-6, Dec. 2020, Chennai, India.
[2] S. Reddy, S. Goel, and R. Nijhawan, "Real-time face mask detection using machine learning/ deep feature-based classifier for face mask recognition," *Procs. of 2021 IEEE Bombay Section Signature Conference (IBSSC)*, pp. 1-6, Nov. 2021, Gwalior, India.
[3] B. Wang, Y. Zhao, C. L. Philip Chen, "Hybrid transfer learning and broad learning system for wearing mask detection in the COVID-19 era," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-12, Mar. 2021.
[4] C. W. Chen, "Design of an intelligent epidemic prevention system combining face and mask recognition and forehead temperature measurement", *Master Thesis*, Graduate School of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan, 2021.

[5] M. T. Tsai, “Application of image recognition technology in mask recognition and warning system”, *Master Thesis*, Graduate School of Electrical Engineering, I-Shou University, Taiwan, 2021.
[6] J. J. Liu, “Research and application of face mask recognition based on convolutional neural network and body temperature measurement”, *Master Thesis*, Graduate School of Computer Science & Information Engineering, National United University, Taiwan, 2021.
[7] S. I. Ali, S. S. Ebrahimi, M. Khurram, and S. I. Qadri, “Real-time face mask detection in deep learning using convolutional network,” *Procs. of 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, Jun. 2021, Bhopal, India.
[8] S. Luo, J. Yu, Y. Xi, and X. Liao, “Aircraft target detection in remote sensing images based on improved YOLOv5,” *IEEE Access*, vol. 10, pp. 5184-5192, Jan. 2022.
[9] Anaconda. [Online] Available: https://www.anaconda.com. May 2022.
[10] LabelImg. [Online] Available: https://github.com/tzutalin/labelImg. May 2022.
[11] Y.-Y. Wang, A.-F. Lai, R.-K. Shen, C.-Y. Yang, V. R.L. Shen, Y.-H. Chu, “Modeling and verification of an intelligent tutoring system based on Petri net theory,” *Mathematical Biosciences and Engineering*, vol. 16, pp. 4947-4975, May 2019.
[12] N. Youssry and A. Khattab, “Accurate real-time face mask detection framework using YOLOv5,” *Procs. of 2022 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS)*, Jun. 2022, Cairo, Egypt.
[13] J. Zhang and D. Tao, “Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things,” *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7789 -7817, May 2021.
[14] S. K. Panda, M. Lin, and T. Zhou, “Energy-efficient computation offloading with DVFS using deep reinforcement learning for time-critical IoT applications in edge computing,” *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6611 -6621, Apr. 2023.
[15] YOLOv5 Teaching and Learning. [Online] Available: https://tw511.com/a/01/13004.html. May 2022.
[16] YOLOv5. [Online] Available: https://github.com/ultralytics/yolov5. May 2022.
[17] WoPeD. [Online] Available: https://woped.dhbw-karlsruhe.de/. May 2022.
[18] Q. Wang and G. Guo,“AAN-Face: Attention augmented networks for face recognition,” *IEEE Transactions on Image Processing*, vol. 30, pp. 7636 -7648, Sep. 2021.
[19] PIPE. [Online] Available: http://pipe2.sourceforge.net/. May 2022.
[20] HPetriSim. [Online] Available: https://github.com/Uzuul23/HPetriSim. May 2022.
[21] Jetson Nano. [Online] Available: https://www.nvidia.com/zh-tw/autonomous-machines/embedded-systems/jetson-nano/. May 2022.
[22] H. Shekhawat and P. Verma, “A real-time model for COVID 19 face-mask identification with “YOLOv4”,” *Procs. of 2022 IEEE World Conference on Applied Intelligence and Computing (AIC)*, Jun. 2022, Sonbhadra, India.
[23] X. Kong, K. Wang, S. Wang, X. Wang, X. Jiang, Y. Guo, G. Shen, X. Chen, and Q. Ni, “Real-time mask identification for COVID-19: An edge-computing-based deep learning framework,” *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15929 –15938, Nov. 2021.
[24] HP w200. [Online] Available: https://www.hp-imagesolution.com/intl/tw/product/category-139/1026.May 2022.
[25] Y. Chen, M. Hu, C. Hua, G. Zhai, J. Zhang, Q. Li, and S. X. Yang, “Face mask assistant: Detection of face mask service stage using mobile phone,” *IEEE Sensors Journal*, vol. 21, no. 9, pp. 11084 -11093, May 2021.

Cheng-Ying Yang received his M.S. degree in Electronic Engineering from Monmouth University, New Jersey, in 1991; and Ph.D. degree from The University of Toledo, Ohio, USA., in 1999. He is a member of IEEE Satellite & Space Communication Society. Currently, he is employed as Professor at The University of Taipei, Taiwan. His research interests include performance analysis of digital communication systems, signal processing, Petri net applications, and computer security.

Yi-Nan Lin received his B.S. degree in Electrical Engineering from National Taiwan Institute of Technology in 1989, M.S. degree in Computer Science & Engineering from Yuan Ze University, Tao-Yuan City, Taiwan, in 2000, and Ph.D. degree in Electrical Engineering from Chang Gung University, Tao-Yuan City, Taiwan, in 2009. Since 1990, he has joined the Department of Electrical Engineering at Ming Chi University of Technology (MCUT), Taishan District, New Taipei City, Taiwan. He is now a Full Professor of Electronic Engineering at MCUT. His current research interests include IoT system, error-control coding, digital transmission systems, Petri net applications, and embedded system design.

Victor R.L. Shen received his B.S. and M.E. degrees in Electronic Engineering of Industrial Education from National Taiwan Normal University, Taiwan, in 1975 and 1980, respectively. Also, he earned his M.S. degree in Electrical Engineering and Computer Science from the University of Illinois at Chicago, U.S.A., in 1987. And he received his Ph.D. degree in Electrical Engineering from National Taiwan University in 1997. From July 2002 to November 2002, he was a visiting Professor at Information Science Institute in the University of Southern California, U.S.A. Since February 2017, he has been hired as an Adjunct Professor at the Department of Computer Science and Information Engineering in National Taipei University; an Honorary Distinguished Professor at the Department of Electronic Engineering in Ming Chi University of Technology; and an Honorary Distinguished Professor at the Department of Information Management in Chaoyang University of Technology.

Dr. Shen is a Fellow of IET, life senior members of IEEE, ACM, and IEICE. His current research interests include e-Learning system development, Petri net theory and applications, artificial intelligence (AI), knowledge-based systems, specification and formal verification of digital systems, high-level synthesis of VLSI, information security, smart home, and. pattern recognition.

Frank H.C. Shen received his B.S. and M.S. degrees in Electronic Engineering from Fu Jen Catholic University, Taiwan, in 2004 and 2006, respectively. He has been working with wireless communication and electronic industry for 16 years and had many experiences in project management. Now, he is working at Quanta Corporation. As a technical manager, he is responsible for circuit design of 5G products. His research interests include artificial intelligence, wireless communication system, information management, and Petri net applications.

Chien-Chih Wang received his B.S. and M.S. degrees in Electronic Engineering from Ming Chi University of Technology (MCUT), Taiwan, in 2019 and 2022, respectively. Since 2019, he has joined the Department of Electronic Engineering at MCUT, Taishan District, New Taipei City, Taiwan. His current research interests include Internet of Things (IoT) system, image processing, and chat robot.