Hand Symbol Classification for Human-Computer Interaction Using the Fifth Version of YOLO Object Detection

Sugiarto Wibowo[1*] and Indar Sugiarto[2]
[1,2]Department of Electrical Engineering, Faculty of Industrial Technology, Petra Christian University
Jawa Timur 60236, Indonesia
Email: [1]osugiartow@gmail.com, [2]indi@petra.ac.id

***Abstract*—Human-Computer Interaction (HCI) nowadays mostly uses physical contact, such as people using the mouse to choose something in an application. However, there are certain problems that people face in using conventional HCI. The research tries to overcome some problems when people use conventional HCI using the computer vision method. The research focuses on creating and evaluating the object detection model for classifying hand symbols. The research applies the fifth version of YOLO with the architecture of YOLOv5m to classify hand symbols in real time. The methods are divided into three steps. Those steps are dataset creation consisting of 100 images in each class, training phase, and performance evaluation of the model. The hand gesture classes made in the research are 'ok', 'cancel', 'previous', 'next', and 'confirm', the dataset is made by the researchers custom. After the training phase, the validation results show 93% for accuracy, 99% for precision, 100% for recall, and 99% for F1 score. Meanwhile, in real-time detection, the performance of the model for classifying hand symbols is 80% for accuracy, 95% for precision, 84% for recall, and 89% for F1 score. Although there are differences, it still acceptable for the research and can be improved in future research.**

***Index Terms*—Hand Symbol Classification, Human-Computer Interaction, YOLO Fifth Version, Object Detection**

I. Introduction

HAND gestures are a medium of communication used by people with disabilities but can also be used with computers for some purposes [1]. Communication with computers is generally known as Human-Computer Interaction (HCI). It typically requires physical contact, such as using a mouse, keyboard, and touch screen [2, 3]. However, along with the development of conventional HCI technology, there are several obstacles, such as wet or dirty hands touching the device. As a result, it can cause the device to be cloudy or even damaged. Contactless HCI can overcome these problems. Several methods can be used for contactless HCI, such as computer vision techniques, voice control using speech recognition, wearable devices, and external sensors like radar sensors.

Hand gesture recognition can also take advantage of wearable devices, such as smartwatches, radar sensors, speech recognition, and others. The smartwatch has several sensors, one of which is Inertial Measurements Units (IMU). Its data can be used to detect hand movements. Then, the specific algorithms can recognize the data to determine the class of hand symbol [4]. The radar sensor detects the movement of the hand and converts the data into a 3D image. Then, the data are classified to determine the hand symbols that detected class [2]. Moreover, speech recognition is quite popular to be applied in HCI. It uses artificial intelligence to process what people say to translate it into words or take some action on the computer or application [5].

The utilization of computer vision to recognize hand gestures by utilizing a camera includes object detection, image segmentation, and pose estimation [6]. The object detection algorithm works by detecting objects and classifying them by class. Meanwhile, image segmentation detects the color of the hand and classifies the type. Then, pose estimation detects points from the human skeleton, such as the hand and body skeleton.

Object detection is one of the most used computer vision methods to recognize hand gestures and sign language. For example, the third version of You Only Look Once (YOLO) is used to recognize Indonesian sign language using a pre-trained weight which is darknnet53 [7]. The fourth version of YOLO controls the music player application by recognizing hand gestures [8]. The latest YOLO version, YOLOv5s (small),

Received: May 13, 2022; received in revised form: July 14, 2022; accepted: July 15, 2022; available online: March 17, 2023.
*Corresponding Author

is used to recognize hand gestures. YOLOv5s is the second smallest version of the fifth version of YOLO. The performance is quite fast, but the accuracy is not too high. So, modifying the backbone structure can improve accuracy [9].

Hand segmentation has a workflow in that hand color segmentation is performed on the image using a glove. Then, the algorithm predicts the detected hand shape to find the probability of a class and the likelihood of being closest to a class. The algorithm translates for action on the computer [10, 11]. Meanwhile, the algorithm of hand pose estimation works by detecting the skeleton of the human hand and measuring the relationship between the points on the hand [12]. The object detection method for hand symbol recognition in previous studies is YOLO version 3 [1].

The hand symbol detection technique that has become state-of-the-art uses computer vision. This technique is widely used to detect hand symbols because users do not need to use wearable devices to move freely. A computer vision algorithm can recognize the movement of hand symbols through a camera, infrared sensor, and depth sensor [6]. In addition, computer vision algorithms are developing rapidly over time.

The researchers use computer vision with an object detection method to recognize hand symbols in the research. Computer vision is also quite popular these days for its many applications, and it is still developed further to support daily life. Object detection is a computer vision method to recognize objects in an image or video [13]. Types of object detection are divided into two, namely single-stage and two-stage detectors. The main difference lies in the computational load, which causes different speed and accuracy performances. The single-stage detector has faster performance than the two-stage detector algorithm, but the accuracy of the two-stage detector is better [14]. The single-stage detector also has a weakness. It does not detect small objects very well [15]. Object detection has two processes: object recognition and classification processes against classes [10].

The object detection algorithm used in the research is a single-stage detector, YOLO, instead of the two-stage detector method. The researchers have a vision that this model can be applied to the Single Board Computer (SBC) for the following research. YOLO has a direct working stage initially. YOLO will resize the image pixels, and the convolutional network process is carried out on the resized image results. The final output produced is the detected object confidence [16]. Another reasonably popular single-stage detector algorithm is the Mobnet SSD developed by Tensorflow [9, 14]. The research objectives are to create and evaluate the object detection model, the fifth version of YOLO, to recognize and classify hand symbols.

Fig. 1. Diagram block of the research.

The contributions of the research are as follows. First, it implements the fifth version of the YOLO algorithm for hand symbol detection. Second, the research can prove that YOLO can be further developed for HCI. Third, it is the creation of a dataset hand symbol to support HCI.

II. Research Method

A. Object Recognition Method

YOLO is a state-of-the-art single-stage detector. Each version of YOLO has its advantages, namely yolov1, as a first step to detecting objects in real-time [16]. Next, the development of YOLO has become YOLOv2 which increases its speed and accuracy while making inferences [17]. Then, the subsequent improvement in YOLOv3 is the model's accuracy [18]. Moreover, the YOLOv4 has also been enhanced in terms of the speed and accuracy of the model [19]. The latest development of YOLO is YOLOv5. YOLOv5 uses a different framework, PyTorch. Meanwhile, the previous version uses DarkNet. In addition, YOLOv5 has one architecture difference. The YOLOv5 uses a focus structure with CSPdarknet53 for the backbone, while YOLOv3 uses Darknet53, and YOLOv4 uses CSPdarknet53. The focus structure has an effect that can reduce the use of CUDA memory, reduce layers, and increase forward propagation and backpropagation [20].

YOLO has a working method of dividing an image into several small pieces: s $\times$ s grids. Each grid will be responsible for the center point of the objects in each grid. The algorithm gives a bounding box and the confidence value of the detected object class in the grid. When detecting an object, the confidence value is the same as the Intersection Over Union (IOU) obtained from the calculation between the bounding box predictions and ground truth [15, 16].

B. Research Stage

The process for the research is shown in Fig. 1. It is divided into three main processes: dataset creation, training dataset, and model evaluation. Dataset creation is the process of capturing images and labeling images

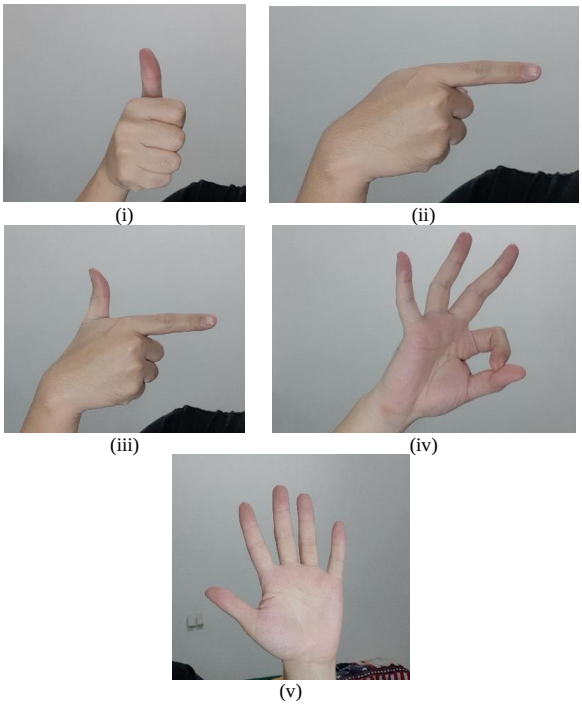

Fig. 2. (i) Cancel symbol, (ii) Previous symbol, (iii) Next symbol, (iv) Ok symbol, and (v) Confirm symbol.

into their classes. Then, all the labeled images are trained using a pre-trained model with some parameters. The researchers separate the dataset into training and validation data. Hence, the process of training the dataset begins. Training the dataset uses the pre-trained weight of the fifth version of YOLO. After the training end, the researchers evaluate the model for its performance. The evaluation process compares the training data validation and inference process. Then, the researchers give some conclusions regarding the model's performance from the evaluation step.

III. Results and Discussion

A. Dataset Creation

The hand symbol dataset is created using an external webcam with a resolution of 1080 px. The dataset of the hand symbols consists of five classes, namely ‘ok’, ‘cancel’, ‘previous’, ‘next’, and ‘confirm’. The creation of the hand symbol dataset is not based on the hand symbol used by persons with disabilities, but rather the researchers define the hand symbol themselves. Those five classes are chosen because they are general actions that the application has. The hand symbols visualization is shown in Fig. 2.

Figure 2 shows several symbols. First, all the fingers are up in the cancel symbol, and the palm faces the camera. Second, the previous symbol has the index finger pointed to the opposite of the hand. For example, when using the right hand, the person points to the left and vice versa. The remaining fingers are folded, and the back of the hand faces the camera. Third, in the next symbol, the thumb finger is up. The index finger is pointed to the opposite of the hand, and the remaining fingers are folded. The back of the hand also faces the camera. The difference between the previous and the next symbol is in the thumb finger. Fourth, the ok symbol shows the thumb and index finger shaping like O. The middle, ring, and little fingers are up, and the palm faces the camera. Last, in confirm symbol, the thumb finger is up, and the remaining fingers are folded. Then, the knuckle of the hand faces the camera.

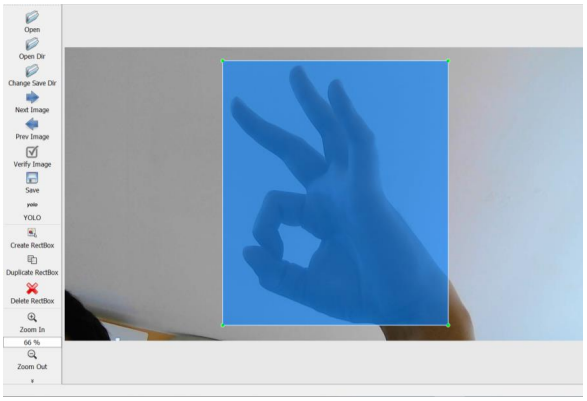

Fig. 3. Image labeling using labelImg.

The researchers make their dataset of approximately 100 photos for each class. Then, datasets are divided for training and validation. The researchers created their dataset because there are not many datasets for hand symbols for HCI purposes according to the needs. The dataset is also made using the left and right hands to detect hand symbols from both hands. The comparison of the dataset between training and validation is 80%:20%. Then, after dividing into training and validation datasets, the researchers perform the labeling process using labelImg. The process of labeling is in every image captured by the webcam. All the labeled images are used in the training phase.

The labeling process is shown in Fig. 3. It begins with blobbing the area of the hand symbol in the image to specify the class. Then, the process is repeated for every image. The blobbing area should be as minimum as possible. So, training the model's dataset is not distracted by the background with too many images. The labeling process is carried out for each image and saved in .txt format to perform training models using YOLO. The file contains class data of x, y, w, and h. Those values represent where the hand symbol is in the

image. An example of those values is shown in Fig. 4.

B. Dataset Training

The training phase is executed after all images have already been labeled according to their class. The training data process is carried out using Google Collab utilizing the Graphics Processing Unit (GPU). Google Collab provides a service for users to train models on the cloud. Besides GPU, there is also Tensor Processing Unit (TPU).

The pre-training model used for the training is YOLOv5m with the following specifications:

- Image size = 640 px
- Mean Average Precision (mAP) 0.50:0.95 = 45.20 (0.5:0.95 means IoU threshold between 0.5 and 0.95)
- Mean Average Precision (mAP) 0.5 = 63.9 (0.5 means the IoU threshold of 0.5)
- Size: 40.7 M (size of the model)

Then, the training process is carried out with the following parameters:

- Image size = 416 px
- Batch-size = 4
- Epoch = 300
- Weights = YOLOv5m.pt
- Cache = True
- Optimizer = Stochastic Gradient Descent (SGD)
- Data = Yet Another Markup Language (YAML) file containing directory path of train, validation, and hand symbol classes

YOLOv5m's architecture shown in Fig. 5 consists of three main parts: backbone, neck, and head. In the backbone, YOLOv5m uses multiple convolutional layers and C3 modules (which can be regarded as the specific implementation of Cross Stage Partial Network). Spatial Pyramid Pooling Fast (SPPF) layers are used between the neck and head part. Finally, the head part uses multiple convolutional layers, up-sampling resolution, concatenation operations, C3 modules, and detection layer. The hyperparameters are as follows.

- lr0= 0.01
- lrf= 0.1
- momentum= 0.937
- weight_decay= 0.0005
- warmup_epochs= 3.0
- warmup_momentum= 0.8
- warmup_bias_lr= 0.1
- box= 0.05
- cls= 0.5
- cls_pw= 1.0
- obj= 1.0
- obj_pw= 1.0
- iou_t= 0.2
- anchor_t= 4.0
- fl_gamma= 0.0
- hsv_h= 0.015
- hsv_s= 0.7
- hsv_v= 0.4
- degrees= 0.0
- translate= 0.1
- scale= 0.5
- shear= 0.0
- perspective= 0.0
- flipud= 0.0
- fliplr= 0.5
- mosaic= 1.0
- mixup= 0.0
- copy_paste = 0.0

The parameter of the training phase uses 416 px instead of 640 px because it can reduce the computation process, so the model's speed of classifying hand symbols is faster. The parameter cache is assigned true because it can make the training phase process faster. Then, the data parameter is used to determine the path of the training, validation data, and hand symbol classes. The remaining parameters are defaulted by the documentation. The duration of the training process with 300 epochs completed in 4.412 hours or 4 hours 24.72 minutes.

After the training process is carried out, it will produce a graph of precision, recall, F1 score, and mAP. These graphs can interpret the performance of the introduced model to recognize hand symbols. Precision is used to assess the model's ability to recognize hand symbols according to their class. For example, the webcam capture 'ok', and the model can classify the hand symbol correctly as 'ok'. Then, mAP assesses the model's accuracy when making inferences with certain IOU thresholds. The recall is used to assess the model's ability to recognize hand symbol objects when they appear in the image. For example, the model can classify 'cancel' correctly when the hand symbols are captured on the webcam many times. The F1 score assesses the balance of precision and recall. This metric can be adjusted depending on the needs. For example, if the model needs to be sensitive, the recall must be higher.

Figure 6 is a metric of the overall training results and a graph of the error reduction in object detection as the training progresses. Table I shows in detail the metric values shown in Fig. 6. It shows that the model has accuracy in hand symbol detection of 0.927 (92.7%) overall, with an IOU threshold between 0.5 and 0.95. Then, a precision of 0.99 (99.90%) shows the model's ability to classify hand symbols according to their

```
0 0.5177083333333333 0.500925925925926 0.403125 0.8851851851851852
```

Fig. 4. Result of the labeling process.

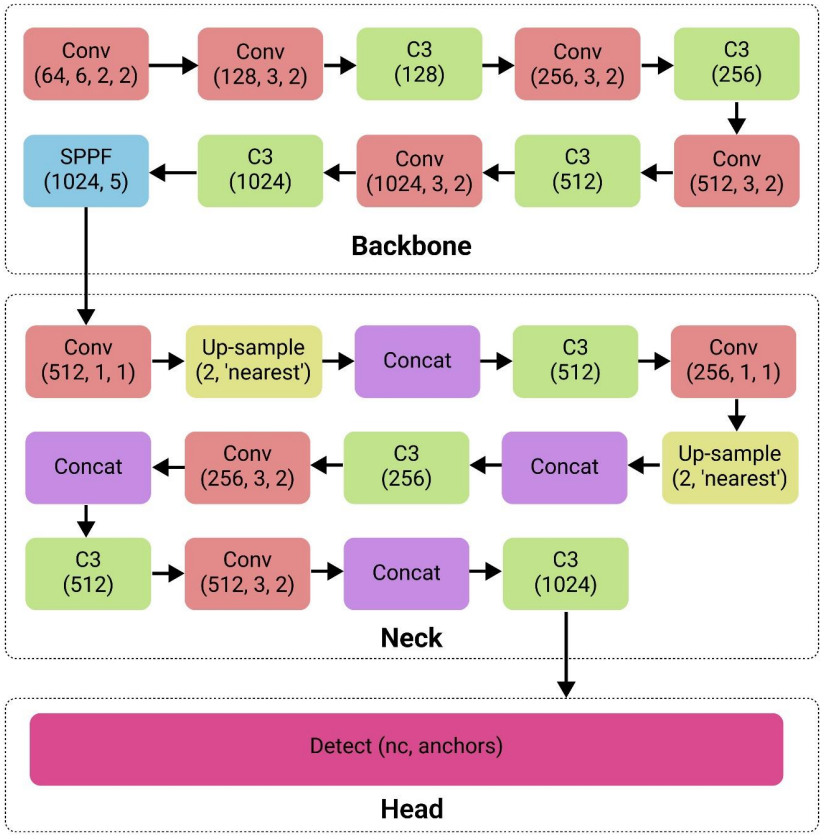

Fig. 5. The YOLOv5m architecture adapted from [21].

class. Meanwhile, a recall value of 1 (100%) indicates the model's ability to detect hand symbols in an image. It interprets the model's ability to predict each time an object appears. Then, an F1 score of 0.99 (99%) means the balance of the value of precision and recall so that it is close to 100%. However, the results of these metrics will decrease if the real-time inference is used for several factors, such as different environmental conditions (light, background, and others).

TABLE I
TRAINING DATASETS RESULT.

Class	Precision	Recall	mAP @0.5	mAP @0.5:0.95	F1 Score
Cancel	0.998	1	0.995	0.984	0.999
Confirm	0.999	1	0.995	0.913	0.999
Next	0.999	1	0.995	0.949	0.999
Ok	1.000	1	0.995	0.878	1.000
Previous	0.998	1	0.995	0.909	0.999
All	0.999	1	0.995	0.927	0.999

C. Inference Process

After completing the training, a model is produced and ready to be used for inference. The inference process is carried out on a laptop with the following specifications. Then, the inference process is executed using Visual Studio Code Terminal.

- CPU: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
- RAM: 12.0 GB
- OS: Windows 10 Pro
- Webcam Supported

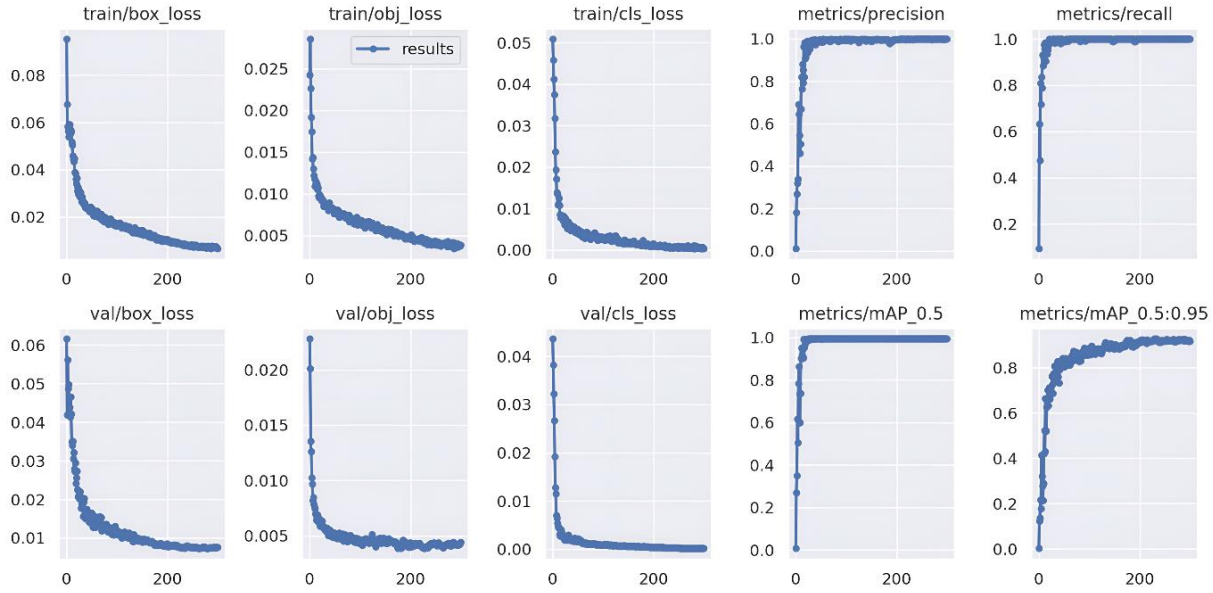

Fig. 6. Training result graph.

D. Discussion

The experiment is conducted by inference through a live image from a webcam by testing each class from a hand symbol 30 times. Tests are carried out at 1-second intervals for each detection, and the dominant class detected is determined. The experiment uses the researchers' hand, and it is also used for dataset creation in the same environment. The inference process is run through the terminal with the following command:

```
python detect.py --source 0 --weights "signLanguage
_v2.2.pt" --img-size 256 --conf-thres 0.70 --iou-thres 0.70
--save-crop
```

- Source = 0 indicates the image source from the webcam in real-time. It can be changed to an image or video path,
- Weights = signLanguage_v2.2.pt is the weight, which is the result of the training process. This model is based on the pre-trained model YOLOv5m that is already declared in the training phase.
- Img-size = 256 pixels is an adjustable image. The value is less than the training phase because the researchers need the model to classify hand symbol classes as fast as possible. As a note, the less value of img-size is, the faster the model classifies the class. However, the accuracy of the model can also decrease.
- Conf-thres = 0.70 is the threshold for confidence level when classifying classes.
- Iou-thres = 0.70 is the threshold for IOU when detecting an object that is stacked. This value is assigned based on the training results that the model has a high accuracy within the range of the IoU threshold between 0.50 and 0.95.

Next, an experiment is conducted to measure several data. It includes True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These data are used to calculate accuracy, precision, recall, and F1 scores. The calculation uses Eqs. (1)–(4).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}, \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3}$$

$$\text{F1 Score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{4}$$

Figure 7 shows the results metrics evaluation of experiments carried out. Those metrics show that the model sometimes classifies the wrong class. For example, the webcam captures ‘cancel’, but the model cannot recognize it. Furthermore, those metrics can be calculated as accuracy, precision, recall, and F1 score. The results of the calculations are shown in Table II.

The results show an accuracy level of 0.80 (80%). It shows the model's ability to predict the class correctly. Then, the precision result is 0.95 (95%), which means the model can classify hand symbols according to their class quite well. Moreover, a recall of 0.84 (84%) shows the model's ability to detect hand symbols when

		Predicted					
		Cancel	Confirm	Next	Ok	Previous	Background
Actual	Cancel	25	0	0	0	0	5
	Confirm	0	26	4	0	0	0
	Next	0	0	23	1	0	6
	Ok	2	0	0	19	0	9
	Previous	0	0	0	0	27	3
	Background	0	0	0	0	0	0

Fig. 7. Multiclass confusion matrix.

TABLE II
MODEL PERFORMANCE BASED ON EVALUATION METRICS INFERENCE PROCESS.

Class	Accuracy	Precision	Recall	F1 Score
Cancel	0.83	1.00	0.83	0.91
Confirm	0.87	0.87	1.00	0.93
Next	0.77	0.96	0.79	0.87
Ok	0.63	0.90	0.68	0.77
Previous	0.90	1.00	0.90	0.95
Summary	0.80	0.95	0.84	0.89

TABLE III
COMPARISON OF TRAINING AND INFERENCE METRICS.

Metrics	Validation After Training	Inference from Webcam
Accuracy	0.93	0.8
Precision	0.99	0.95
Recall	1.00	0.84
F1 Score	0.99	0.89

it appears many times. Then, an F1 score of 0.89 (89%) indicates that the precision value is higher than recall. Therefore, the classification of hand symbols based on class is more prioritized.

Table III compares the training results with the results of inference via a webcam directly. The accuracy value of the training results is obtained from the mAP value with a range of 0.50:0.95. When viewed from the total metric comparison, the model's performance has decreased in accuracy from 92.70% to 80%. Then, it also shows the precision from 99% to 95%, the recall from 100% to 84%, and the F1 score from 99.90% to 89%. The decrease in the model's performance is natural because sometimes the environment can affect the process, such as light level, background color, and others. Some solutions to increase the model's performance include increasing dataset creation for each class, varying the angle of the image dataset, using different hand sizes and colors, using different backgrounds, and others.

IV. CONCLUSION

Hand symbols recognition using the object detection method can be implemented for HCI purposes. The researchers choose the fifth version of YOLO algorithm to recognize hand symbols. The classes of the hand symbols are 'ok', 'cancel', 'previous', 'next', and 'confirm'. The researchers use the pre-trained model YOLOv5m and train it with the dataset of left-right hands. The performance of the trained model for classifying hand symbols is 80% for accuracy, 95% for precision, 84% for recall, and 89% for F1 score.

Nevertheless, the research has limitation. The dataset contains only the researchers' hand characteristics, whereas various hand characteristics should be added to make the model more robust and accurate. Then, in future research, the researchers can develop further by adding datasets with different hands' colors and sizes (characteristics), environments, hyperparameters, the latest yolo version architectures, and algorithms. In addition, further research can also focus on making the interface of the application that can be controlled using hand gestures. Future research can add more classes of hand symbols so it can support the advanced system.

ACKNOWLEDGEMENT

The researchers would like to thank Fakultas Teknologi Industri as well as Lembaga Penelitian dan Pengabdian Masyarakat of Petra Christian University for supporting the research through the special research grant No.09/HBK-PENELITIAN/LPPM-UKP/XI/2022. The researchers also acknowledge the financial support from the Ministry of Education, Culture, Research, and Technology of Indonesia under the research grant No. 02/AMD/SP2H/PT-L/LL7/2022.

REFERENCES

[1] A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaševičius, R. Maskeliūnas, and K. H. Abdulkareem, "Real-time hand gesture recognition based on deep learning YOLOv3 model," *Applied Sciences*, vol. 11, no. 9, pp. 1–15, 2021.

[2] S. Ahmed and S. H. Cho, "Hand gesture recognition using an IR-UWB radar with an inception module-based classifier," *Sensors*, vol. 20, no. 2, pp. 1–18, 2020.

[3] S. Parikh, S. Banka, I. Lautrey, and D. Yedurkar, "Human-computer interaction using dynamic hand gesture recognition to conveniently control the system," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 9, pp. 206–210, 2021.

[4] K. Nguyen-Trong, H. N. Vu, N. N. Trung, and C. Pham, "Gesture recognition using wearable sensors with bi-long short-term memory convolutional neural networks," *IEEE Sensors Journal*, vol. 21, no. 13, pp. 15 065–15 079, 2021.

[5] M. A. M. A. Shariah, R. N. Ainon, R. Zainuddin, and O. O. Khalifa, "Human computer interaction using isolated-words speech recognition technology," in *2007 International Conference on Intelligent and Advanced Systems*. Kuala Lumpur, Malaysia: IEEE, Nov. 25–28, 2007, pp. 1173–1178.

[6] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. Lille, France: IEEE, May 14–18, 2019, pp. 1–8.

[7] S. Daniels, N. Suciati, and C. Fathichah, "Indonesian sign language recognition using YOLO method," in *IOP Conference Series: Materials Science and Engineering*, vol. 1077. IOP Publishing, 2021, pp. 1–9.

[8] Z. Zheng, "Gesture recognition real-time control system based on YOLOV4," *Journal of Physics: Conference Series*, vol. 2196, pp. 1–9, 2022.

[9] D. Hu, J. Zhu, J. Liu, J. Wang, and X. Zhang, "Gesture recognition based on modified Yolov5s," *IET Image Processing*, vol. 16, no. 8, pp. 2124–2132, 2022.

[10] E. Padmalatha, S. Sailekya, R. R. Reddy, C. A. Krishna, and K. Divyarsha, "Sign language recognition," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 3, pp. 2128–2137, 2019.

[11] M. Rivera-Acosta, J. M. Ruiz-Varela, S. Ortega-Cisneros, J. Rivera, R. Parra-Michel, and P. Mejia-Alvarez, "Spelling correction real-time american sign language alphabet translation system based on YOLO network and LSTM," *Electronics*, vol. 10, no. 9, pp. 1–18, 2021.

[12] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 52–73, 2007.

[13] M. A. M. M. Asri, Z. Ahmad, I. A. Mohtar, and S. Ibrahim, "A real time Malaysian sign language detection algorithm based on YOLOv3," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2, pp. 651–656, 2019.

[14] P. Soviany and R. T. Ionescu, "Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction," in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. Timisoara, Romania: IEEE, Sept. 20–23, 2018, pp. 209–214.

[15] L. Aziz, M. S. B. H. Salam, U. U. Sheikh, and S. Ayub, "Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review," *IEEE Access*, vol. 8, pp. 170 461–170 495, 2020.

[16] S. Chaudhari, N. Malkan, A. Momin, and M. Bonde, "YOLO real time object detection," *International Journal of Computer Trends and Technology*, vol. 68, no. 6, pp. 70–76, 2020.

[17] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[18] ——, "YOLOv3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[19] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[20] U. Nepal and H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs," *Sensors*, vol. 22, no. 2, pp. 1–15, 2022.

[21] H. Liu, F. Sun, J. Gu, and L. Deng, "SF-YOLOv5: A lightweight small object detection algorithm based on improved feature fusion mode," *Sensors*, vol. 22, no. 15, pp. 1–14, 2022.