2025 International Conference on Advanced Machine Learning and Data Science (AMLDS) | 979-8-3315-1099-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/AMLDS63918.2025.11159332

Exploring the Performance Improvement Method of a YOLO-based Mask Appearance Detection Model under Small Data Sets

Cheng-Yuan Ho
Dept. of Information Management
National Taiwan University
Taipei, Taiwan
tommyho@ntu.edu.tw

Hsing-Yu Chang
Dept. of Computer Science and Information Engineering
Asia University
Taichung, Taiwan
sharon010228@gmail.com

***Abstract*—COVID-19 emerged at the end of 2019 and rapidly affected countries around the globe. In an effort to curb the spread of the virus, many nations implemented various measures, one of which was the widespread use of medical masks. Masks, being more readily available, easier to wear, and offering effective protection compared to goggles and protective clothing, have seen a dramatic increase in usage.**

In order to facilitate rapid preliminary inspection of mask appearance—specifically the completeness of the mask front and ear loops—on high-volume production lines, this study employs the YOLO object detection framework for mask appearance detection. However, the full YOLOv7 model, with its large architecture and high GPU memory consumption (up to 5.79 GB in our experiments), proves inconvenient for training on less advanced hardware, especially when the task at hand is relatively simple. Thus, YOLOv7-tiny was adopted as the baseline to alleviate computational demands (reducing maximum GPU memory usage to 1.46 GB). Nevertheless, training with YOLOv7-tiny revealed issues with insufficient accuracy. For instance, when the training dataset ranged from 160 to 750 images, the test accuracy varied between 28% and 95%. Consequently, modifications were made to the internal architecture of YOLOv7-tiny—such as adding convolution layers, pooling layers, and implementing cross-layer interactions—to improve accuracy on small datasets while maintaining comparable training, testing, and validation speeds. Under the same dataset conditions (with 160 to 750 training images), the proposed method increased test accuracy by 5% to 32%, achieving overall accuracy levels between 37% and 99%.

Keywords—object detection, deep learning, mask, small datasets

I. INTRODUCTION

Since the outbreak of COVID-19 in late 2019, which impacted the entire world, research and media reports have highlighted that mask-wearing can prevent virus transmission. Consequently, the demand for masks surged globally—at one point even leading to shortages in Taiwan. The government intervened by imposing purchase limits to curb both panic buying and opportunistic price hikes. Despite these measures, the severity of the pandemic further increased mask usage.

In various sectors such as healthcare and service industries, masks are used extensively, with frequent replacement during events like seasonal influenza outbreaks. Traditionally, quality control in factories has relied on manual inspection. However, with evolving times and the advent of artificial intelligence, many factories have adopted automated quality control methods. These methods not only reduce labor costs but also accelerate production. Although initial investments in machinery and equipment might be required, the long-term cost and time savings are substantial.

Given that masks are mass-produced continuously under the dual pressures of the pandemic and seasonal flu, production lines inevitably generate defective masks. Manual inspection is time-consuming, costly, and prone to errors due to fatigue, which may result in misclassification of masks. To address these issues, this study focuses on rapid preliminary detection of mask appearance—specifically verifying the integrity of the ear loops—to promptly identify production errors and facilitate corrective measures.

In this study, we modified the architecture of the YOLOv7-tiny model and named the modified version the Reorganized YOLOv7-tiny model. We then compared its accuracy, speed, and training time against the original YOLOv7-tiny model through the following experimental procedure:

1. Dataset Construction: Capture and classify mask images using a fixed camera.
2. Data Preprocessing: Annotate images using the MAKESENSE tool to mark categories and positions, and export annotations in YOLO text file format.
3. Data Splitting: Divide the annotated images into three datasets: Training, Validation, and Test sets.
4. Model Training and Validation: Train both the YOLOv7-tiny and the Reorganized YOLOv7-tiny models using the training and validation datasets (along with the YOLO-format annotation files), save the trained models, and then validate their accuracy using the test dataset.

Figures 1 and 2 illustrate the flowcharts for the training and validation processes, respectively. In Figure 1, the mask images are first classified into training, validation, and test sets. The training and validation sets are then fed into both models for training, with the resulting trained models, accuracies, and speeds being recorded for comparison. Figure 2 shows that the test set is used to evaluate both trained models, with the accuracy and inference times being documented.

The objective of this study is to develop methods to improve the performance of a mask appearance detection model using small datasets. Section II reviews the relevant literature; Section III describes research methodology, experimental environment, and dataset; Section IV details the model design and architecture;

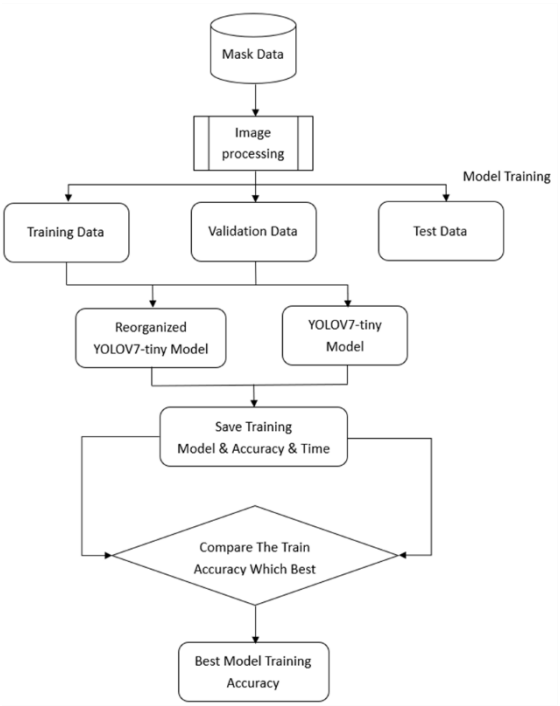

Fig. 1. Model Training Process.

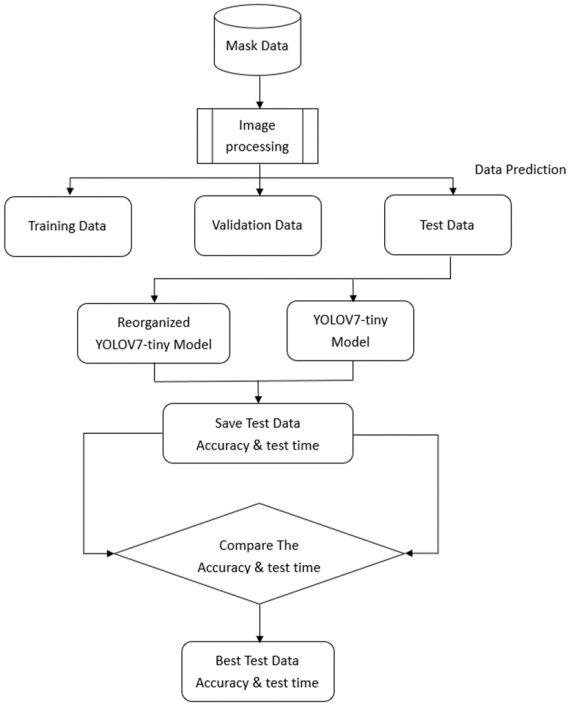

Fig. 2. Model Validation Process.

Section V presents the experimental results, validation, testing, and comparative analysis; and Section VI concludes the study and discusses encountered challenges.

II. Literature Review

Object detection—the technology of localizing objects in an image along with identifying their categories and associated confidence scores—can be broadly classified into two categories. The first is the one-stage approach, which detects and classifies objects in a single step (e.g., YOLO (You Only Look Once) [1] and EfficientDet [2]). The second is the two-stage approach, which first generates candidate regions and then classifies them (e.g., R-CNN [3], Fast R-CNN [4], and Faster R-CNN [5]).

YOLO represents a family of object detection models, evolving from YOLOv1 in 2015 to the latest YOLOv12 in 2025. This research was conducted in 2022, at a time when YOLOv7 was recognized as the state-of-the-art in object detection. Its exceptional balance between accuracy and speed made it the best available technology for applications such as mask appearance detection in industrial settings [1].

Key optimizations in YOLOv7 include:

1. Model Re-parameterization:

This technique is implemented at both the model level and the module level. Model-level re-parameterization involves averaging the weights from multiple training sessions (using different datasets or iteration counts), while module-level re-parameterization splits a module into several branches during training, which are later merged into an equivalent module for inference.

2. Model Scaling:

Common factors for scaling include resolution (input image size), stage (number of feature pyramids), depth (number of layers), and width (number of channels). It has been observed that in concatenation-based architectures, later layers can be adversely affected by the scaling of earlier layers. To address this, the authors proposed a compound model scaling strategy, where increases in depth are accompanied by proportional increases in width.

3. Coarse for Auxiliary and Fine for Lead Loss:

Incorporating an auxiliary head in the model can lead to better convergence and overall performance. While conventional methods assign labels separately to the lead head and the auxiliary head, the YOLOv7 authors introduced two label assignment strategies. In one strategy, the robust learning ability of the lead head is leveraged by having the auxiliary head learn from it, allowing the lead head to focus on other unlearned features; in the other strategy, a coarse label is used to compensate for the auxiliary head's weaker learning ability, ensuring important information is not missed. This dynamic label assignment strategy improves the training process.

Reference [6] employed a CNN-based architecture for image recognition to assess mask-wearing conditions. The study pointed out that while most existing literature focuses on plain masks, the increasing diversity in mask designs necessitates models that can recognize patterned masks and improve accuracy. The differences between our study and reference [6] are: i) Our study focuses on mask detection in the factory supply chain, whereas reference [6] targets mask detection on individuals; ii) Our dataset consists of plain masks, while reference [6] investigates masks with various patterns; and iii) We modified the YOLOv7-tiny architecture, whereas reference [6] modified the VGG16 architecture.

Reference [7] in 2022 used models such as ResNet152V2 [8], InceptionV3 [9], and Xception [10] while reference [11] used various CNN models. Both references [7, 11] tested the effects

of image preprocessing on defect detection in masks on production lines. In contrast, i) our study involves architectural modifications to the model, whereas references [7, 11] utilize existing models; and ii) we did not perform image grayscaling, background filtering, contour extraction and skeletonization, which references [7, 11] specifically applied.

III. Research Methodology

A. Experimental Environment

This study utilizes the following hardware: a desktop computer (detailed equipment and operating system information are provided in Table 1), a small industrial assembly line, an action camera (model: Insta360 One R), and a small lighting device to capture mask images during production line operation.

The desktop computer used for modeling and prediction is configured as described in Table 2. Due to specific version requirements for the YOLOv7 model and its associated packages, a clean environment is necessary—created, for example, via Anaconda—to install essential packages such as NumPy, OpenCV-Python, Pandas, PyTorch, and TensorBoard. For detailed version specifications, please refer to the YOLOv7 GitHub repository provided by the authors.

B. Dataset Description

The dataset in this study is divided into three main categories:

Category 1: Normal – Masks with intact fronts and fully attached ear loops (Figure 3).

Category 2: Images where either both ear loops are missing (Figure 4) or one ear loop is missing while the other is intact (Figure 5).

Category 3: Images with broken ear loops, including cases where both ear loops are broken (Figure 6) or only one ear loop is broken (Figure 7).

The key reason for keeping Category 2 separate from Category 3 is that in Category 3, the ear loops were initially attached but broke due to poor fusion (as highlighted by the pink circles in Figures 6 and 7), whereas in Category 2, one or both ear loops were never attached, thus not a result of a fusion issue.

C. Dataset Processing

The mask images captured by the action camera were annotated using the MAKESENSE annotation tool to mark both the category and the position of each image (Figure 8). The annotated bounding boxes (bbox) were then exported as YOLO-format text files. In these text files, the numbers (from left to right) represent the class ID, X, Y (the relative center coordinates of the bbox with respect to the image dimensions), and W, H (the relative width and height of bbox, respectively). Figure 9 illustrates the YOLO coordinate annotation.

To evaluate the impact of small datasets on both the YOLO model and the proposed model, the primary dataset was divided into 12 subsets with varying numbers of images. Table 3 provides detailed statistics on the training and validation datasets for each subset.

TABLE I. Hardware and Operating System for Capturing Images

Hardware	CPU	Intel i5-9400F
	RAM	32GB
	GPU	Nvidia RTX3060*2
System	OS	Ubuntu 20.04

TABLE II. Hardware Configuration and Environment for Modeling

Hardware	CPU	12th Gen Intel(R) Core(TM)i5-12400 2.5GHz
	RAM	16 GB
	GPU	Nvidia RTX3070
System / Software / Packages	OS	Window 10
	CUDA	11.4
	Python	3.9.13
	Numpy	1.23.3
	Opencv-python	4.6.0.66
	Pandas	1.5.0

Fig. 3. Category 1 (Normal).

Fig. 4. Category 2 (Both Ear Loops Completely Missing).

Fig. 5. Category 2 (One Side Missing an Ear Loop, the Other Normal).

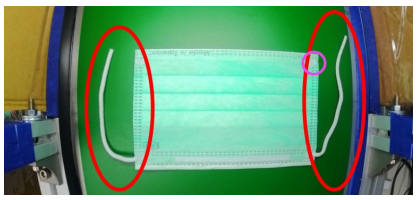

Fig. 6. Category 3 (Both Ear Loops Broken).

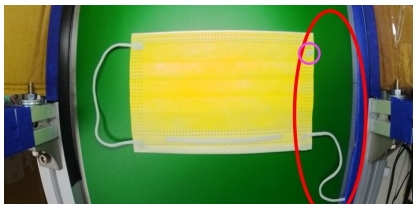

Fig. 7. Category 3 (One Side Ear Loop Broken).

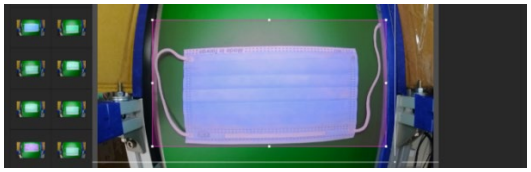

Fig. 8. Annotated Image.

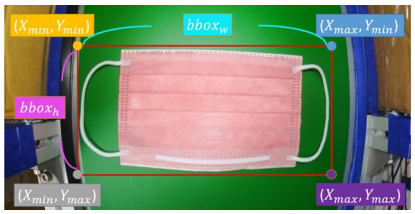

Fig. 9. YOLO Coordinate Annotation.

TABLE III. DETAILED DATA AND QUANTITIES OF TRAINING AND VALIDATION DATASETS

ID	Category 1 (number)	Category 2 (number)	Category 3 (number)	Total (number)
TRAINING DATA / VALIDATION DATA				
1	250 / 63	250 / 63	250 / 63	750 / 189
2	200 / 50	200 / 50	160 / 40	560 / 140
3	200 / 50	160 / 40	200 / 50	560 / 140
4	160 / 40	200 / 50	200 / 50	560 / 140
5	200 / 50	160 / 40	160 / 40	520 / 130
6	160 / 40	200 / 50	160 / 40	520 / 130
7	160 / 40	160 / 40	200 / 50	520 / 130
8	160 / 40	160 / 40	160 / 40	480 / 120
9	160 / 40	80 / 20	80 / 20	320 / 80
10	80 / 20	160 / 40	80 / 20	320 / 80
11	80 / 20	80 / 20	160 / 40	320 / 80
12	80 / 20	40 / 10	40 / 10	160 / 40

To evaluate the impact of small datasets on both the YOLO model and the proposed model, the primary dataset was divided into 12 subsets with varying numbers of images. Table 3 provides detailed statistics on the training and validation datasets for each subset.

IV. MODEL DESIGN

A. YOLOv7-tiny Architecture Overview

Figure 10 depicts the architecture of YOLOv7-tiny, which is divided into two main parts: the Backbone and the Head (corresponding to the left and right halves of the figure, respectively). The notation [from, number, module, args] of the program is used. Here, "from" indicates the source layer (most commonly -1, which represents the previous layer, though a specific layer number, such as 10, can also be used); "number" indicates how many identical modules are stacked; "module" denotes the module type; and "args" represents various arguments. Specifically, the Backbone consists of three modules:

1) CBL (Convolution + Batch Normalization + LeakyReLU): Used for feature extraction (see Figure 11).

2) C5: Integrates multiple feature maps to improve accuracy (see Figure 12).

3) MP (Max Pooling): Reduces the model's sensitivity to edges or insignificant features by limiting overfitting (see Figure 13).

The Head is composed of two modules:

1) CBC: Obtains feature maps at various scales to enhance overall performance and accuracy (see Figure 14).

2) UpSample: Uses the Nearest Neighbor Interpolation method to enlarge the image.

B. Design Philosophy of the Reorganized YOLOv7-tiny Architecture

In scenarios with small or insufficient datasets, our preliminary experiments revealed that both the YOLOv7 and YOLOv7-tiny models leave significant room for accuracy improvement. Thus, our aim was to develop a model architecture that strikes a balance—maintaining a level of complexity similar to YOLOv7-tiny while achieving higher accuracy, even rivaling more complex models that require vast amounts of training data.

To reduce sensitivity to edge-related features and to capture more comprehensive image features, an additional MP + C5 module was incorporated into the Backbone. Furthermore, to extract feature maps at multiple scales and enhance both overall performance and accuracy, the CBC module in the Head was replaced with SPPCSPC (Spatial Pyramid Pooling + Cross-Stage-Partial + Convolution), as shown in Figure 15. The internal structure of SPPCSPC uses CBS (Convolution + Batch Normalization + SiLU) for feature extraction (see Figure 16).

The key difference between CBS and CBL is that the SiLU activation function in CBS yields higher accuracy than the LeakyReLU used in CBL, while their computational speeds remain similar. Moreover, SPPCSPC extracts features three times—with two of these operations preceding max pooling—allowing the model to better learn both small and large feature points. Figure 17 presents the complete architecture of the Reorganized YOLOv7-tiny Model.

V. EXPERIMENTAL RESULTS AND ANALYSIS

As shown in Table 3, the training and validation datasets were divided into 12 subsets. The experiments used corresponding subsets—i.e., the model trained with the N-th subset of training data was validated using the N-th subset of validation data, where $1 \leq N \leq 12$.

A. YOLOv7-tiny Model Validation

Table 4 presents the average training time and validation accuracy of the YOLOv7-tiny model and proposed model, Reorganized YOLOv7-tiny. It can be observed that when the dataset comprises 480 or more images, YOLOv7-tiny achieves an accuracy of at least 90%, reaching up to 97.6%. The training time typically ranges between 74 and 120 minutes. However, when the number of training images is 360 or fewer, the accuracy declines dramatically—from 73% to as low as 21.6%.

B. Reorganized YOLOv7-tiny Model Validation

Using the same experimental procedures, training, and validation datasets, the Reorganized YOLOv7-tiny model in Table 4 achieves an accuracy of at least 93.6% (up to 98.6%) when the dataset contains 480 or more images, with an overall average accuracy of 96.10%. Moreover, its training time remains under two hours (ranging from 75 to 119 minutes), which is comparable to that of YOLOv7-tiny. Notably, when the training dataset comprises 360 images or fewer, the Reorganized YOLOv7-tiny model consistently outperforms YOLOv7-tiny. For example, with 320 training images, its accuracy exceeds 70% and can reach up to 89.60%—improving by a factor of 1.06 to 1.27 compared to YOLOv7-tiny.

C. Comparison of Training Time and Validation Accuracy

Table 5 summarizes the average training time and validation accuracy of both models across the 12 datasets. While the Reorganized YOLOv7-tiny model occasionally requires an extra couple of minutes (approximately 2 minutes more), its accuracy is consistently higher than that of the original YOLOv7-tiny model.

Additionally, experiments revealed that when using 320 images (Table 3, subsets 9-11) with a category distribution of 2:1:1, both models attain higher accuracy. In other distributions, there remains room for improvement. However, with extremely

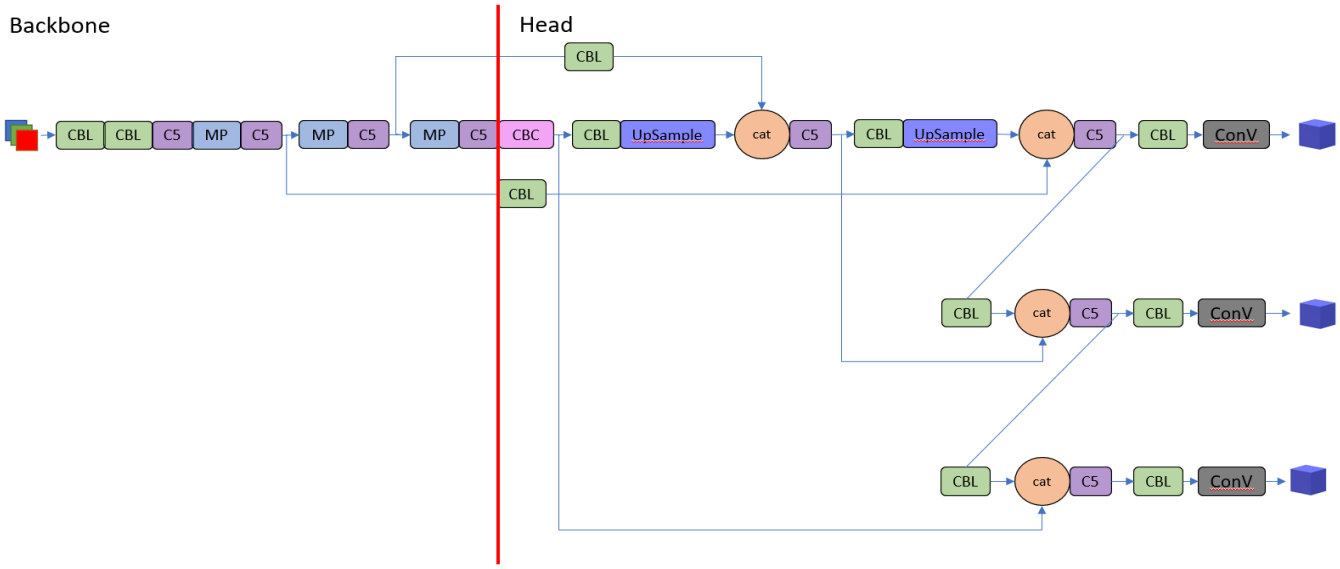

Fig. 10. YOLOv7-tiny Architecture.

Fig. 11. CBL Module.

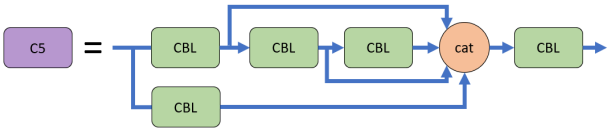

Fig. 12. C5 Module.

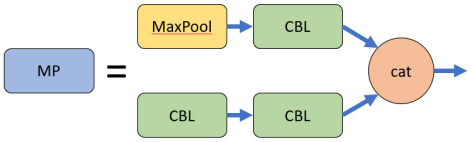

Fig. 13. MP Module.

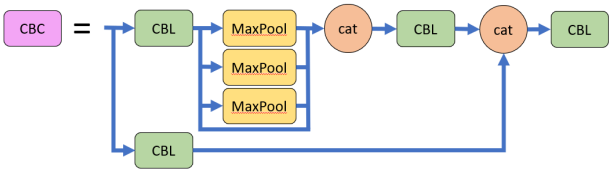

Fig. 14. CBC Module.

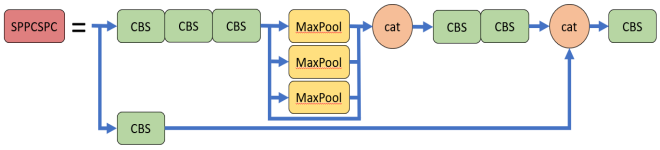

Fig. 15. SPPCSPC Module.

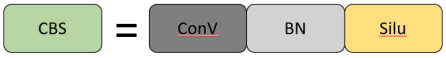

Fig. 16. CBS Module.

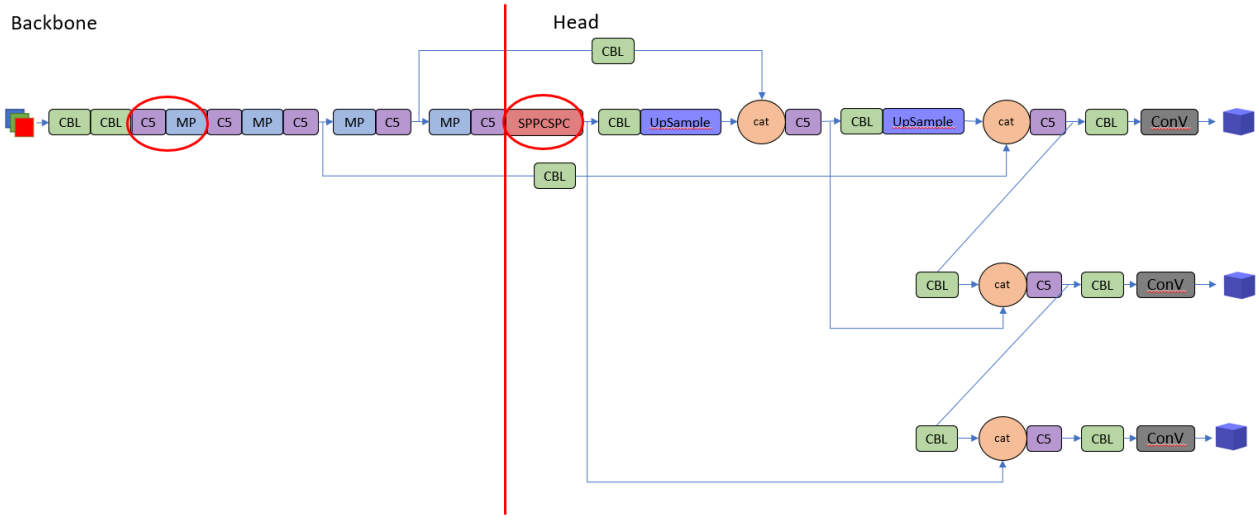

Fig. 17. Reorganized YOLOv7-tiny Architecture.

TABLE IV. TRAINING TIME AND ACCURACY OF TWO MODELS

ID	Total No. of Training Images	YOLOV7-tiny		Reorganized YOLOV7-tiny	
		Training Hours	Validation Accuracy	Training Hours	Validation Accuracy
1	750	2.016	97.60%	1.986	98%
2	560	1.384	96.30%	1.458	98.60%
3	560	1.471	93.60%	1.44	93.60%
4	560	1.405	91%	1.387	95%
5	520	1.341	97%	1.378	96%
6	520	1.336	95.60%	1.363	96%
7	520	1.356	94%	1.348	97%
8	480	1.229	90.60%	1.253	94.60%
9	320	0.819	73%	0.845	89.60%
10	320	0.82	68.60%	0.823	73.30%
11	320	0.826	55.30%	0.845	70.60%
12	160	0.413	21.60%	0.401	23.60%

TABLE V. AVERAGE COMPARISON OF VALIDATION ACCURACY AND TRAINING TIME BETWEEN THE TWO MODELS

Total Number of Training Dataset Images	YOLOV7-tiny Accuracy / Training Time (Hours)	Reorganized YOLOV7-tiny Accuracy / Training Time (Hours)
750	97.6% / 2.016	98.0% / 1.986
560	93.6% / 1.42	95.7% / 1.428
520	95.5% / 1.344	96.3% / 1.363
480	90.6% / 1.229	94.6% / 1.253
320	65.6% / 0.821	77.8% / 0.837
160	21.6% / 0.413	23.6% / 0.401

TABLE VI. TEST RESULTS OF TWO MODELS

ID	YOLOV7-tiny			Reorganized YOLOV7-tiny		
	Correctness	Error	Accuracy	Correctness	Error	Accuracy
1	100	5	95.20%	104	1	99.00%
2	100	5	95.20%	103	2	98.00%
3	97	8	92.30%	99	6	94.20%
4	97	8	92.30%	97	8	92.30%
5	95	10	90.40%	98	7	93.30%
6	100	5	95.20%	101	4	96.10%
7	93	12	88.50%	97	8	92.30%
8	100	5	95.20%	98	7	93.30%
9	75	30	71.40%	81	24	77.10%
10	76	29	72.30%	75	30	71.40%
11	56	49	53.30%	70	35	66.60%
12	30	75	28.50%	39	66	37.10%

limited data (e.g., 160 images), even with a 2:1:1 ratio, the accuracy of both models drops below 25%. Thus, for acceptable accuracy, the practical lower limit for training data for both models appears to be around 320 images.

D. Practical Test Accuracy Comparison

A set of 105 test images, which were not part of the training or validation datasets (comprising 65 images from Category 1 and 20 images each from Categories 2 and 3), was randomly selected to evaluate the practical performance of both models. In practice, 12 models (each corresponding to one of the 12 subsets from Table 3) from both YOLOv7-tiny and Reorganized YOLOv7-tiny architectures were tested on these 105 images, and their final accuracies were recorded (see Table 6).

Results from Table 6 show that for models trained on more than 320 images (subsets 1-8), both architectures deliver accuracies suitable for real-world production line applications. More importantly, the Reorganized YOLOv7-tiny model achieves practical accuracies between 92.30% and 99.00%, compared to 88.50% to 95.20% for YOLOv7-tiny. In cases using smaller datasets, the accuracy of the Reorganized YOLOv7-tiny model is generally superior to that of YOLOv7-tiny, with improvements of up to 1.3 times.

VI. CONCLUSION AND RECOMMENDATIONS

This study enhanced the model by adding extra layers to enable more comprehensive feature learning, ensuring that even with limited training data, a satisfactory level of accuracy can be maintained for preliminary classification. The results demonstrate that for relatively simple images such as masks, a small-scale model like YOLOv7-tiny—when appropriately modified—can achieve performance comparable to larger, data-intensive models.

Specifically, the Reorganized YOLOv7-tiny Model achieved around 70% accuracy when trained on a very limited dataset (approximately 320 images) and nearly 99% accuracy when sufficient data (about 750 images) were available. This shows that for simple images like masks, achieving high performance does not necessarily require a large dataset or heavy computational resources, thereby reducing data collection, annotation, and training times.

However, it is important to note that the camera used in this study was not high-end. When the speed of the production line increases, image blurring can significantly degrade model performance. Also, a slight wide-angle effect was observed, which might introduce some bias into the model. Improved imaging equipment is expected to yield even better results.

REFERENCES

[1] Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696.

[2] Mingxing Tan, Ruoming Pang, Quoc V. Le (2019). EfficientDet: Scalable and Efficient Object Detection. arXiv preprint arXiv:1911.09070.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524.

[4] Ross Girshick (2015). Fast R-CNN. arXiv preprint arXiv:1504.08083.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv preprint arXiv:1506.01497.

[6] Ting-Guang Jiang (2022). The patterned face mask detection. https://hdl.handle.net/11296/bas7n2.

[7] Ting-Han Wang (2022). Mask inspection management systems. https://hdl.handle.net/11296/v36z65.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). Deep Residual Learning for Image Recognition. arXiv preprint arXiv:1512.03385.

[9] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna (2015). Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567.

[10] Franois Chollet (2016). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint arXiv:1610.02357.

[11] LIANG, CHIA-WEI (2022). Study on Defect Detection of Medical Mask: Combination of Deep Learning and Machine Vision. https://hdl.handle.net/11296/gx3nw5.