

Modification Of Yolov11 Nano And Small Architecture For Improved Accuracy In Motorcycle Riders Face Recognition Based On Eye

Randy Ardiansyah^{*1}, I Gde Putu Wirarama WW², Ario Yudo Husodo³

^{1,2,3}University of Mataram, Indonesia

Email: ²i2s02410019@student.unram.ac.id

Received : Apr 9, 2025; Revised : Jul 21, 2025; Accepted : Jul 31, 2025; Published : Oct 21, 2025

Abstract

Face recognition still faces challenges in identifying faces covered by masks and helmets with open visors, such as those commonly used by motorcyclists, especially when entering parking areas. To improve the accuracy of face recognition in these conditions, this study proposes nano and small versions of the YOLOv11 modification, which is an internal version. Modifications are made to the neck section and the DySample module is added in place of the UpSample module to improve the model's capabilities. Experiments were conducted using a self-generated dataset consisting of 50 classes. The results show that the modified nano version achieves 99.3% accuracy at the same mAP50 as YOLOv11n and YOLOv12n. At mAP50-95, it shows a 1.6% accuracy improvement compared to YOLOv11n and YOLOv12n with 75% accuracy. Meanwhile, the modified small version achieved an accuracy improvement of 1.3% and 1.2% compared to YOLOv11n and YOLOv12n, respectively, reaching 76.1% on mAP50-95, although the accuracy on mAP50 remained the same as YOLOv11n and 0.1% superior to YOLOv12n. However, recall and precision did not show significant improvement in both as well as the increase in model parameters. However, the model is still in the nano and small versions. Therefore, the model can be implemented on edge devices. This research is important for the field of computer vision, especially in the context of face recognition. The contribution of this research is the improvement of the accuracy of the mAP50-95 metric in eye-based face recognition, which is relevant for intelligent security systems with limited resources.

Keywords : Face Recognition, mAP50, mAP50-95, Nano, Small, YOLOv11

This work is an open access article and licensed under a Creative Commons Attribution-Non Commercial 4.0 International License



1. INTRODUCTION

Facial recognition technology has achieved a high level of accuracy in full face conditions, as shown in various studies using various approaches such as AB-FR[1], CNN plus cosine annealing (CA) [2], PCA plus Triplet Similarity and similarity metric[3], SIFT and CNN[4], Faster R-CNN with VGG-16 feature extraction[5], E3FRM[6], LDA and OpenCV[7], GhostFaceNetV2-1 (MS1MV3) [8], R-CNN[9], and CNN by [10],[11],[12]. This success is supported by the model's ability to extract important features such as eyes, nose, mouth, and overall facial contours. However, the performance of the facial recognition system can decrease significantly when most of the face is blocked by objects such as masks, glasses, or helmets. Therefore, it is difficult to identify the obstructed face[13]. A number of studies have actually addressed the challenges of facial recognition, such as mask objects using the YOLOv4 and CNN approaches[14], R34-AMaskNet[15], Convolutional Visual Self-Attention Network [16]. These approaches have shown good recognition performance, but these facial recognition models are only designed to address masked conditions. This is not enough in the context of motorcyclists, because the face is often obstructed by a helmet and mask simultaneously in open visor conditions which often leave the eyebrows and eyes exposed. This causes the extraction process to be very limited in these facial conditions. An example of such a face can be shown in Figure 1 and can be seen at the link <https://drive.google.com/drive/folders/1-0zJg-tKd17-hERiSiUE4eKKmgAkdJGz?usp=sharing>.



Figure 1. Face covered by helmet and mask

As seen in Figure 1, to be able to recognize a covered face, eyebrows and eyes can be used as the recognition domain. However, eyebrows are often partially covered, making them less than optimal for motorcycle face recognition, especially when entering a parking area. Such identification is needed in the security field, especially to prevent vehicle theft. Utilizing eyes is a solution in face recognition when the face is in a condition like Figure 1. Therefore, a model is needed that can focus on recognition with eyes. Approaches that can be done alone include classical machine learning or deep learning methods. However, classical machine learning methods must use feature extraction such as Local Binary Pattern, HOG (Histogram of Oriented Gradients), and others. This process is less efficient because it involves accurate eye detection and involves several steps. For example, using Haar-cascade, facial landmarks, or local eyebrow active shape models in detecting eyes and eyebrows, followed by cropping, feature extraction, and classification using classical machine learning, as done by [17] using Naïve Bayes classification. Although this method achieves good accuracy, it is susceptible to eye detection errors, which disrupt the feature extraction process and reduce recognition accuracy. On the other hand, [18] also used scale space (ss) and DWT with SVM classification for eye-based recognition but only achieved an accuracy of 76.04% and still did eye cropping.

In contrast to deep learning methods that can perform automatic feature extraction [19], especially in object detection. Therefore, it can detect and classify eyes directly without having many stages like in classical machine learning methods, this is the basis for selecting the deep learning method in this study. Motorcyclist facial recognition itself uses the eye domain due to facial occlusion [20], making the performance of the object detection model for recognition can be reduced, therefore it is necessary to modify the object detection model to improve accuracy in recognizing occluded faces. Some deep learning methods used in object detection and classification include You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD), and Mask-RCNN. In this study, the YOLO method was chosen because it uses one stage detection, which means the neural network process is carried out only once, compared to Mask-RCNN which requires two neural network processes or two stages. Although SSD also uses a one-stage process, YOLO's performance is better in accuracy on the COCO dataset. The accuracy (mAP) of YOLOv4 itself on COCO is 43%. Meanwhile, SSD gets an mAP of 35.1%, as does Mask-RCNN which only gets 40.3%[21]. On the other hand, research [22] shows that the mAP of YOLOv7 is more than 50% of SSD and Mask R-CNN.

YOLO itself has various versions, starting from version 1 to version 13. This research itself uses YOLO version 11, the reason for choosing this version compared to versions 12 and 13, because version 13 does not have a concrete comparison metric. While for version 12 due to the recommendation from ultralytics to continue using YOLOv11. Version 11 itself still uses the FPN or Feature Pyramid Network structure [23] by using three components in its architecture, namely the backbone that performs feature extraction, the neck that functions for multifusion on large-scale features, and the detection head that is used in detecting and classifying an object [24]. Improvements in feature extraction through improvements to the backbone and head architecture produce the highest accuracy with few parameters, so it is computationally efficient without sacrificing accuracy.

YOLOv11 itself has several versions such as nano, small, medium, large, and extra-large. In this study, the nano and small versions were chosen because they are suitable for low-resource hardware. The reason for choosing low-power devices is because the device cost is affordable, the energy required is not high, and the size is ideal compared to high-power hardware. Low-power hardware itself such as ESP32, ESP8266, Raspberry Pi, and others. In further research, it is recommended to use Raspberry Pi, because it has the availability of GPU and can be combined with the Raspberry Pi Camera Module, making it suitable for the nano and small version of the YOLO model to be implemented in recognizing motorcyclists' faces showing the eye domain, with implementation at the parking lot entrance and outside the parking lot to increase security.

Based on these various reasons, there is still no research that specifically develops or modifies the YOLOv11 nano and small models in detecting eyes for facial recognition of motorcyclists when wearing helmets with open visors and masks. This is the gap in this research. Therefore, the aim of this research is to focus on modifying YOLOv11 on nano and small architectures to improve the accuracy of facial recognition of motorcyclists based on eyes that are obstructed by helmets and eyebrows so as to overcome the limitations of classical machine learning methods. This research is expected to improve safety in parking lots by recognizing motorcyclists' faces.

2. METHOD

A flowchart is required for the stages of this research. These stages are used to support the research until it achieves its objectives. The flowchart is shown in Figure 2, which includes six stages in support of this research.

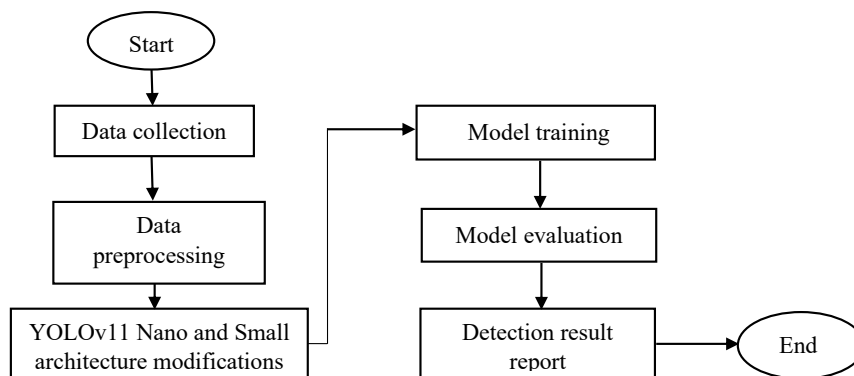


Figure 2. Research flowchart

Figure 2 shows that the first stage of this research is data collection. Data collection is shown to capture sample videos to be recognized wearing masks and helmets with open windows. The next stage is data processing to crop the sample faces. This aims to be able to label the bounding box for each sample. Next, the third stage is to modify the architecture, of course by referring to references on how to modify. The next step is to train the model in Google Colab, and then evaluate its performance. This evaluation also compares the performance of the YOLOv11n and YOLOv12n models for comparison. The final stage is visualizing the detection and recognition results. These detection results are taken from the model with the best performance among the trained models.

2.1. Data Collection

The data used in this study is self-collected data or in other words using primary data. The data is taken using a Realme 6 Pro cell phone with the data in the form of image data from a motorcyclist who will be taken from videos with camera specifications, namely 64 MP camera resolution, Wide, f/2.1,

4K@30 FPS, distance 0.5 m to 1.5 m. The number of videos collected amounted to 50 videos with each video containing 1 subject so that the number of subjects for recognition amounted to 50. The number of videos collected amounted to 50 videos with each video containing 1 subject so that the number of subjects for recognition amounted to 50. An example of how to collect data can be found in Figure 2.



Figure 2. Example of how to collect data

Figure 2 shows an example of data collection using the camera specifications described in the previous paragraph. The approximately 5- to 7-second videos were captured in bright sunlight, brightly lit nights, and cloudy weather. These videos were then processed to create facial images based on the conditions.

2.2. Data Processing

The data processing used in this research has several stages. The stages used in processing data in order to get a dataset that is ready to be used to train and test the model. These stages can be shown in Figure 3.

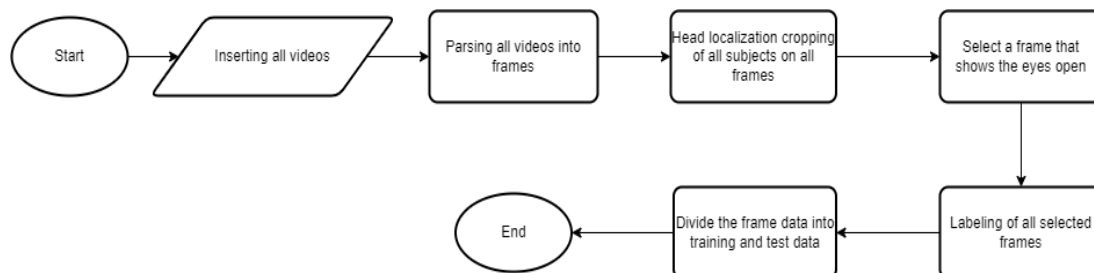


Figure 3. Data processing stages

Data processing in Figure 3 is done first by cutting into several frames that will produce several images. The image will be cropped to take the location of the face that uses a helmet and mask. The cropped data will be used in training and testing the model. The number of frames taken from each video is 140, with each frame checking whether the eyes of each class are closed or not, if they are closed then they will be deleted. The total frames used are 6354 data. 6354 Data will be labeled with YOLO mark tools with a total of 50 classes, to mark the eyes. An example of labeling can be seen in Figure 4.

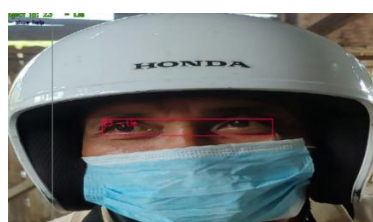


Figure 4. Example of labeling data

Figure 4 shows a face image covered by a helmet with an open visor and a mask that is labeled by directing the bounding box to the eye area of the sample. The next process, as shown in Figure 3, is to divide the training data and test data with a ratio of 80:20. The reason for using the 80:20 ratio is because the amount of training data is small so that for training the data must be enlarged, so that the training data uses 80% of the original data. The total training and test data used in this study are 5081 and 1272. This dataset has an age range for each class between 15 and 60 years, with 31 women for each class and 19 men for each class. An example of the dataset image can be shown in Figure 5.



Figure 5. Model training and test image data

Figure 5 shows an example of the data used to train the model. The labeling coordinates for the bounding box in Figure 4 are actually located directly in a file with the same image name. Therefore, during training, the bounding box in the example data in Figure 5 is not visible.

2.3. Architectural Modifications

Our architecture modification has several stages in order to improve the accuracy of the model. The stages of the modification can be shown in Figure 6.

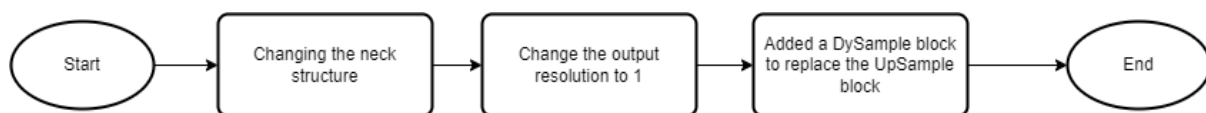


Figure 6. Modification Stages

Figure 6 are first, the modified architecture replaces the existing neck structure with a structure that suits the dataset conditions and changes the output resolution to 1 which previously used an output layer of 3, this modification is based on modifications made by [25], who did the same thing to YOLOv3 by cutting the neck so that its accuracy increased, and changing the resolution to 1 because the detection frame used is the same. Second, the UpSample block is replaced with the DySample block. DySample itself can improve efficiency in resources by performing dynamic convolution and by utilizing a method that samples points in upsampling. UpSample demands a lot of power on substantial computation and has many parameters that can hinder the ability of a model to remain lightweight. DySample incorporates a sampling-to-learn approach to sampling. The approach reduces the consumption of computational resources and improves the performance of image resolution without any additional load [26]. The design of DySample can be shown in Figure 7.

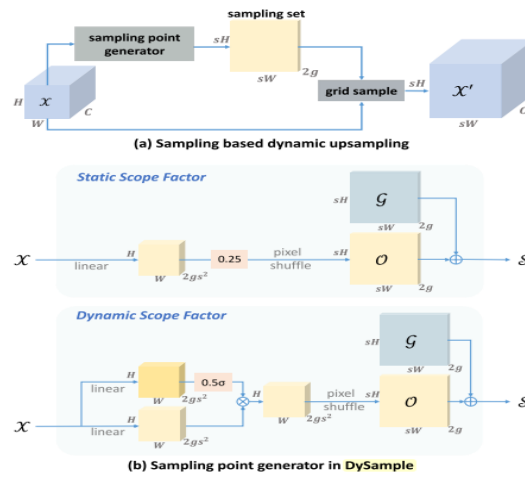


Figure 7. DySample Block Design [27]

The DySample method, as shown in Figure 8, has an input size of $C \times H_1 \times W_1$ and a sampling set δ of size $2 \times H_2 \times W_2$. The first two dimensions specifically indicate the x and y coordinates. Grid_sample as a function is used to take new or re-samples from the input feature map X and uses coordinates with the sample set δ . This is achieved by using bilinear interpolation which produces a feature map (X) of size $C \times H_2 \times W_2$. The process can be defined as follows:

$$X' = \text{grid_sample}(X, \delta) \quad (1)$$

The upsampling scale factor is set to s and the dimension of the feature map (X) is $C \times H \times W$. A linear layer is used with input and output channels still being $2S^2$ and C. The output offset itself is adjusted to the channel O which is $2S^2 \times H \times W$ in size. Then the offset O is adjusted by the pixel reordering algorithm, so that it becomes $2 \times sH \times sW$. The sampling set in the form of δ is generated by the superposition of the offset and also the sampling grid from G. The process is defined as follows:

$$O = \text{linear}(X) \quad (2)$$

$$\delta = G + O \quad (3)$$

The final process is the feature map defined as X' which has been upsampled with dimensions $C \times sH \times sW$. This process is generated using the sampling set and of course the grid_sample function, as in equation 1 [26]. Figure 8 itself shows the results of DySample. The use of DySample also outperforms 5 upsampling techniques such as CARAFE, SAPA, FADE, bilinear interpolation, and nearest neighbor interpolation on tasks such as object detection, semantic segmentation, instance segmentation, panoptic segmentation, monocular depth estimation [27]. After all the processes have been carried out, the keyword YOLOv11 architecture modification is specified in the yaml file. The architecture between YOLOv11 and the modified YOLOv11Mod can be shown in Figure 8.

Figure 8 shows the architectural differences between YOLOv11 and YOLOv11Mod. The main differences between the two images are the different neck and head used, as well as the presence of the DySample technique. Two DySample blocks are used in the neck itself, replacing UpSample, to increase the resolution capability of the target image without overburdening the existing computational load. Comparison of the detection and classification methods of YOLOv11 and YOLOv12 methods, nano and small versions, as state-of-the-art methods for object detection can be shown in Table 1.

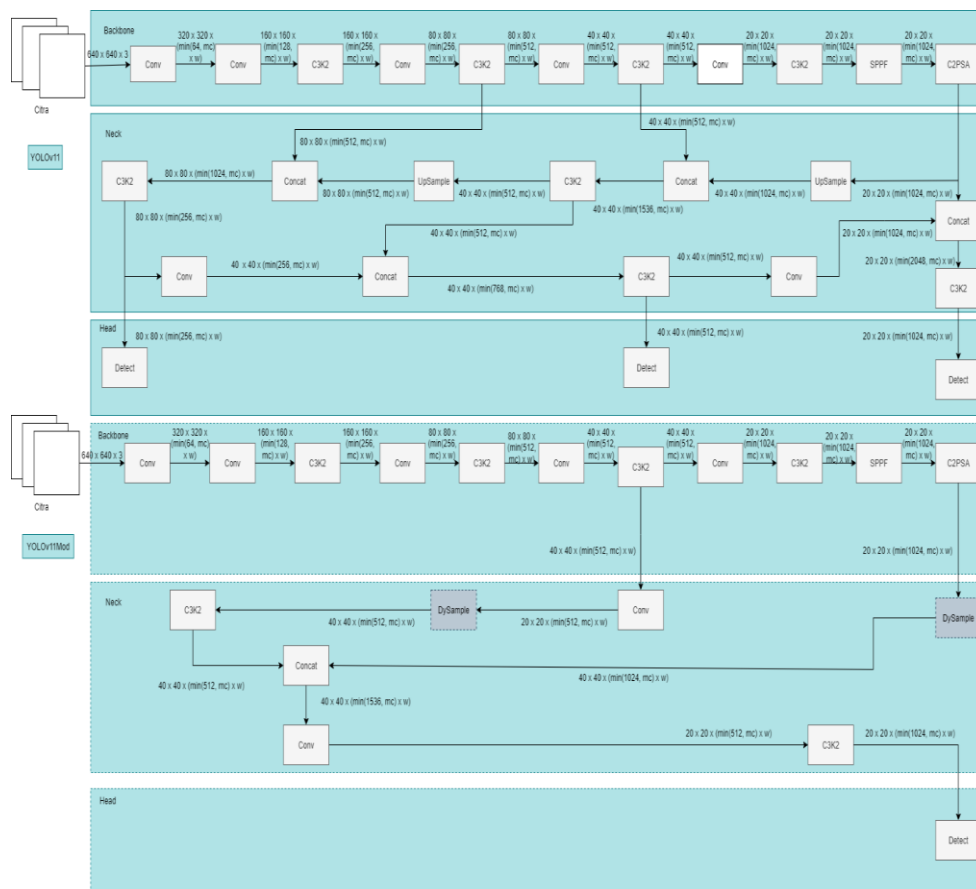


Figure 8. Differences between YOLOv11 and YOLOv11Mod architecture

Table 1. Comparison of YOLOv11, YOLOv12, and YOLOv11Mod methods on nano and small versions

Methods	Number of layers	Number of parameters	Number of Detect blocks	Model input resolution
YOLOv11n	314	2.591.902	3	640×640
YOLOv11nMod	235	2.692.394	1	640×640
YOLOv11s	314	9.432.150	3	640×640
YOLOv11sMod	235	10.684.306	1	640×640
YOLOv12n	159	2.566.478	3	640×640
YOLOv12s	159	9.250.230	3	640×640

Table 1 shows a comparison between YOLOv11 and YOLOv12, both nano and small, using the same input resolution of 640×640 . The differences lie in the number of detects and the parameters used by the model. Our proposed model has additional parameters compared to the original or internal version.

2.4. Model Training

Model training is done by setting the hyperparameters used for YOLO, in this study the hyperparameters used use imgsiz 640, epoch 100. The use of epoch 100, imgsiz 640 is used because previous research [28] which improved YOLOv8 also used the same configuration. The model training process is carried out on Google Colab with keyword examples as follows:

```

1. model = YOLO("YOLOv11nori.yaml")
2. d="YOLOv11/data.yaml"
3. results = model.train(data=d, epochs=100, imgsiz=640, resume=True)

```

2.5. Model Evaluation

Model evaluation in this study uses four evaluation metrics, namely precision, recall, mAP50, and mAP50-95 to measure the model's ability. The mAP50 metric is also used to determine accuracy as done in[25]. The addition of the mAP50-95 metric is used to evaluate the accuracy of a model very strictly which provides an assessment in detection ability[29]. The four metric equations can be explained by equations 4, 5, 6, and 7 as follows:

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

$$Recall = \frac{TP}{TP+FN} \quad (5)$$

$$mAP50 = \frac{1}{N} \sum_j^K AP_{50,i} \quad (6)$$

$$mAP50 - 95 = \frac{1}{N} \sum_{i=1}^N \cdot \frac{1}{K} \sum_{j=1}^K AP_{i,j} \quad (7)$$

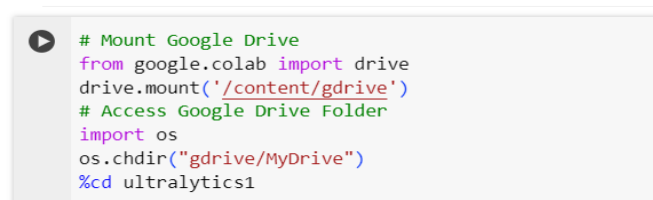
Where:

- TP (True Positive): Positive samples that are correctly recognized (true detection).
- FP (False Positive): Negative samples that are misrecognized as positive samples or false detections.
- FN (False Negative): Positive samples but skipped by the model (skipped detection).
- N: Number of target classes.
- $AP_{50,i}$: The average precision in category i for the IoU threshold is 0.5.
- K: Number of IoU thresholds (10 values ranging from 0.5 to 0.95).
- $AP_{i,j}$: Average precision.

In equation 4, precision measures the proportion of positive samples (True Positives) among all samples predicted as positive. Meanwhile, recall in equation 5 measures the proportion of positive samples that were correctly and successfully detected. Meanwhile, mAP is the average precision value across all classes[30]. We use equations 6 and 7 for mAP, as explained in the previous paragraph.

2.6. Implementation Plan

The programming language that will be used in this research is python with the pytorch library and also ultralytics in building the model to be used. The first stage is to prepare training on google colab, where we train models from datasets that have been collected from 50 Indonesian people who use masks and helmets. The face recognition performed focuses on eye detection. The training process uses google colab using the Tesla T4 GPU. The process carried out is to mount the drive that has stored the prepared dataset, then enter the folder to download the ultralytics library. The process can be shown in Figure 9.



```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
# Access Google Drive Folder
import os
os.chdir("gdrive/MyDrive")
%cd ultralytics1
```

Figure 9. Signing in to Google Drive

Figure 9 shows the process of logging into a Google account in Google Colab. This process allows you to download the Ultralytics repository through Google Colab and access the Ultralytics folder to add DySample. The download process itself is shown in Figure 10.


```
# Clone the ultralytics repository
!git clone https://github.com/ultralytics/ultralytics

# Navigate to the cloned directory
%cd ultralytics
```

Figure 10. Ultralytics library download process in Google Colab

Figure 10 shows the download of Ultralytics and its entry into the Ultralytics folder. After entering the Ultralytics folder, DySample is added to the nn/Extramodule/DySample.py folder. The DySample block code in DySample.py can be found on GitHub by the Dysample developer in its publication at [27]. The next step is to add the block to task.py so that it can be used in the modified YOLOv11. The addition is shown in Figure 11.

```
import torch
import torch.nn as nn
from .Extramodule import *
from ultralytics.nn.modules.SHA import SHA_BLOCK, C2PSA_SHA

elif m is ResNetLayer:
    c2 = args[1] if args[3] else args[1] * 4
elif m is nn.BatchNorm2d:
    args = [ch[f]]
elif m in [DySample]:
    args=[ch[f], *args[0:]]
```

(a)

(b)

Figure 11. Addition of DySample block in tasks.py

In Figure 11, the addition of blocks is done by importing the module that has been added to the Extramodule as in part a, the next process is so that in the training process the DySample block can be read in the yaml file to match the YOLOv11Mod architecture according to the modifications that have been made, then we add the DySample block to be the same as module m and the arguments or args will be filled by the number of channels of the first feature and the addition of the previous arguments as in part b. Finally, the library installation process is carried out using the code in Figure 12.

```
# Install the package in editable mode for development
!pip install -e .
```

Figure 12. Installation of ultralytics library

Figure 12 shows the installation of the ultralytics library after installing DySample. The final implementation process for further research is implementation on a Raspberry Pi. The implementation plan for the YOLOv11Mod model can be seen in Figure 13, where the model is used to recognize driver faces to improve security in parking areas.

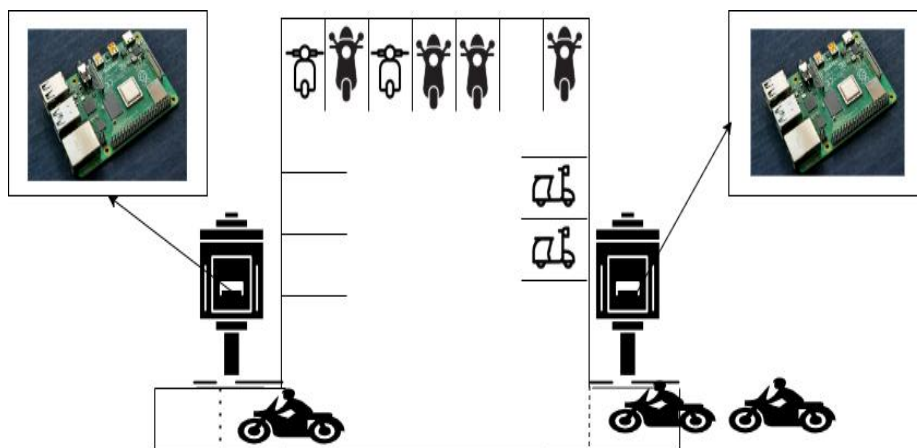


Figure 13. Model Implementation into Raspberry Pi

Figure 13 itself shows an overview of the implementation plan for future research. This process creates a model that can be implemented in real-world conditions and controls the entry and exit of vehicles into the parking lot.

2.7. Research novelty

The novelty offered in this research is the first modification of the architecture of YOLOv11, where the modification is to change the neck structure according to the dataset conditions where the data focuses on eye detection in vision, then the neck structure is adjusted to the conditions of eye detection on a large scale, and change the output resolution to 1 which previously used an output layer of 3, this modification is based on modifications made by [25], who did the same thing in YOLOv3 by cutting the neck or modifying it so that its accuracy increased.

Second, we replace the UpSample block with the DySample block because the UpSample block in YOLOv11 still uses nearest neighbor interpolation while DySample is a new upsample technique, the performance comparison of the upsample technique itself can be seen in [27], where DySample excels in object detection, semantic segmentation, instance segmentation, panoptic segmentation, and monocular depth estimation.

Third, we also offer data novelty, where the data used is primary data collected from 50 Indonesian subjects, ranging in age from 15 to 60 years old. The data taken in this study was taken at various times ranging from morning, afternoon, evening, and night. The data offered also from 50 classes has a total of 19 male classes and 31 female classes.

3. RESULT

3.1. Model performance evaluation

We evaluated the model using mAP50, mAP50-95, precision, and inference time. We compared these results with the baseline model and YOLOv12 in both the nano and small versions. The evaluation results are explained in the next subsection, along with our comparisons with the same dataset.

3.1.1. Model accuracy comparison

After the training process, the modified model and the original model can be compared in accuracy with the mAP50 and mAP50-95 metrics. We also trained YOLOv12 to serve as a comparison to the modified YOLOv11. The model versions compared here are the nano version and the small version. The comparison on mAP50 can be shown in Figure 14.

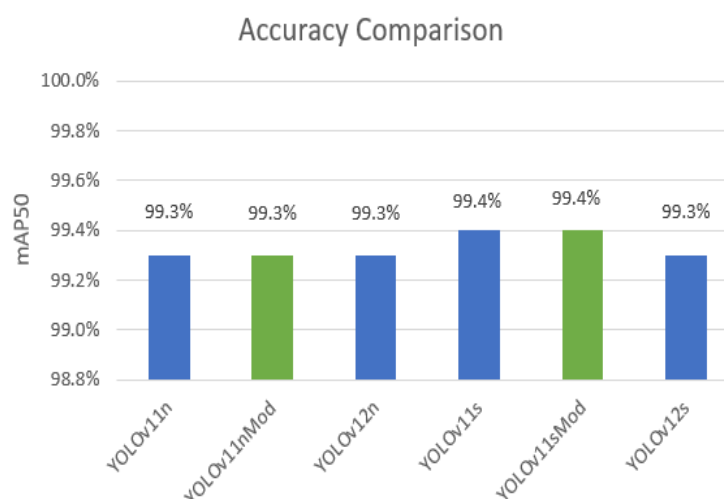


Figure 14. Accuracy comparison with mAP50 metric

In Figure 14, it can be seen that our proposed model in the nano version successfully obtained an accuracy of 99.3%, the same as the original YOLOv11 and YOLOv12 nano versions. The small version only obtained the same accuracy of 99.4% for the modified model and the original model, but for YOLOv12s it only obtained 99.3%. A comparison of the accuracy with the mAP50-95 metric was also used to determine the model's capability strictly, which can be shown in Figure 15.

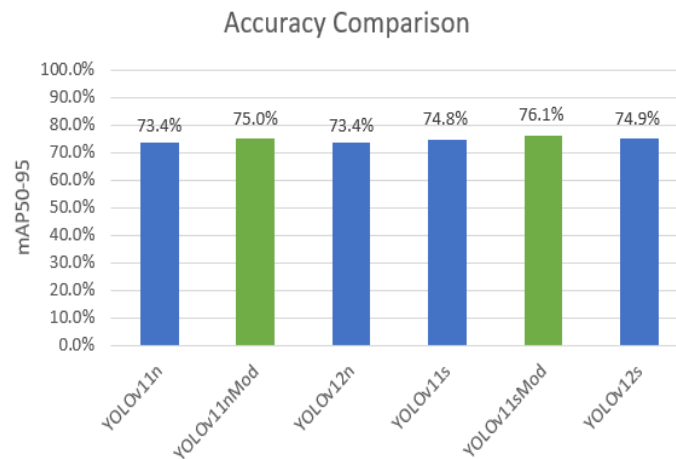


Figure 15. Accuracy comparison with mAP50-95 metric

Figure 15 shows the comparison results with a more stringent metric. Our proposed modified model achieved 75% and 76.1% accuracy. This results in an accuracy improvement over the baseline models, YOLOv11n and YOLOv11s. The improvements were 1.6% and 1.3% for the nano and small versions, respectively. Meanwhile, with YOLOv12 on the nano and small versions, our proposed model achieved 1.6% and 1.2% superiority, respectively.

3.1.2. Comparison of model recall

The test results after training for total recall for all classes in all models can be shown in Figure 16. The results in Figure 16 are directly compared with the baseline model (YOLOv11) and also the YOLOv12 model in the nano and small versions. For more details, see Figure 16.

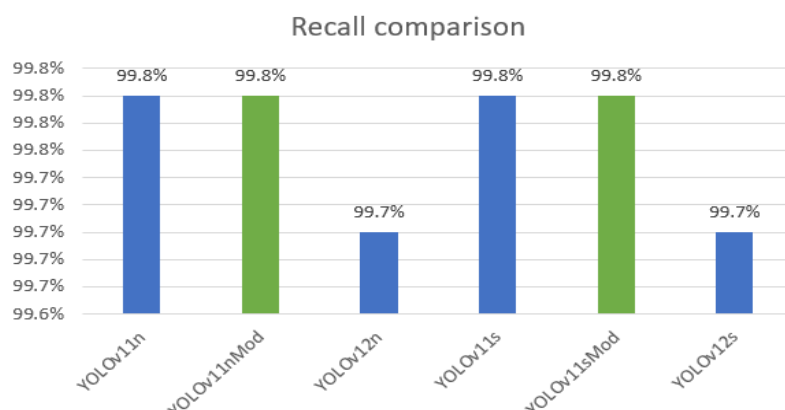


Figure 16. Recall comparison

Figure 16 shows that the nano version of our proposed model does not experience any improvement over the baseline model. The recall obtained based on Figure 16 for our proposed model is 99.8% the same for both the nano and small versions. However, the baseline and our proposed model outperform YOLOv12 in both the nano and small versions by 0.3%.

3.1.3. Comparison of model precision

The test results for the model's precision are shown in Figure 17. Similar to the recall comparison in the previous subsection, the model precision comparison of the total across all classes for all models is also compared. The results of the model's precision performance are shown in Figure 17.

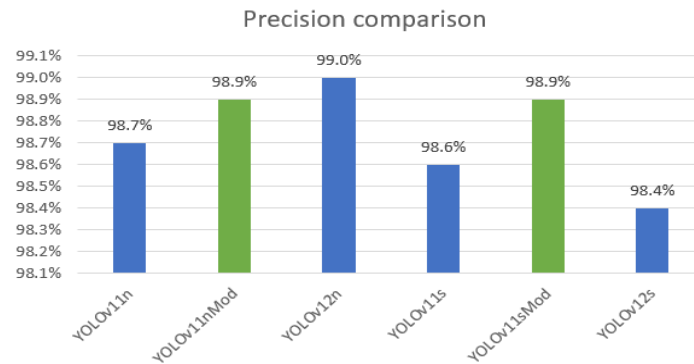


Figure 17. Precision comparison

Figure 17 shows that with the modifications we made to the nano version, there was a 0.2% increase in precision compared to the baseline (YOLOv11n). However, it still lost 0.1% to YOLOv12n, which had 99% precision. On the other hand, in the small version, our precision remained the same, but the baseline and YOLOv12 models experienced a decrease in precision. Although our nano model lost to YOLOv12n, its accuracy on the mAP50-95 metric was superior to YOLOv12n.

3.1.4. Comparison of inference model time

We measured the inference of all models when using test data of 1272 images on Google Colab using the T4 GPU. The inference time of the models themselves can be seen in Figure 18.

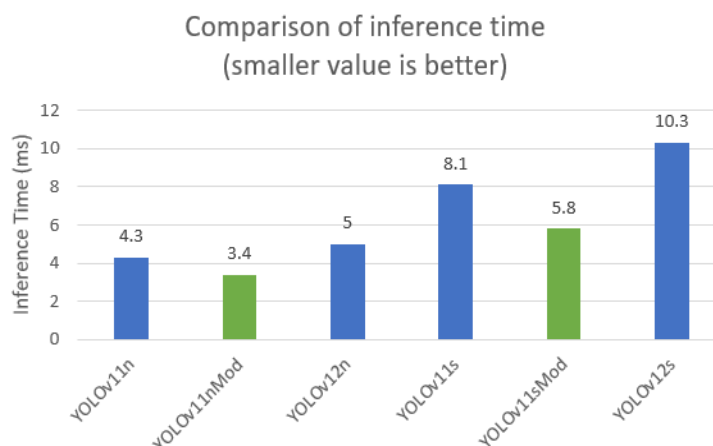


Figure 18. Comparison of inference time

In Figure 18, it can be seen that our proposed model in the nano version has the smallest inference time of 3.5 ms compared to YOLOv11n and YOLOv12n. The small version of our model also has the smallest inference time compared to the small version of the state-of-the-art models YOLOv11 and YOLOv12 with an inference time of 5.8 ms. These results show that our proposed model is superior in inference time in both nano and small versions compared to YOLOv11 and YOLOv12.

3.2. Detection Result Testing

We tested the detection results by using the trained model to detect and recognize the faces of several samples, an example of which can be shown in Figure 19.

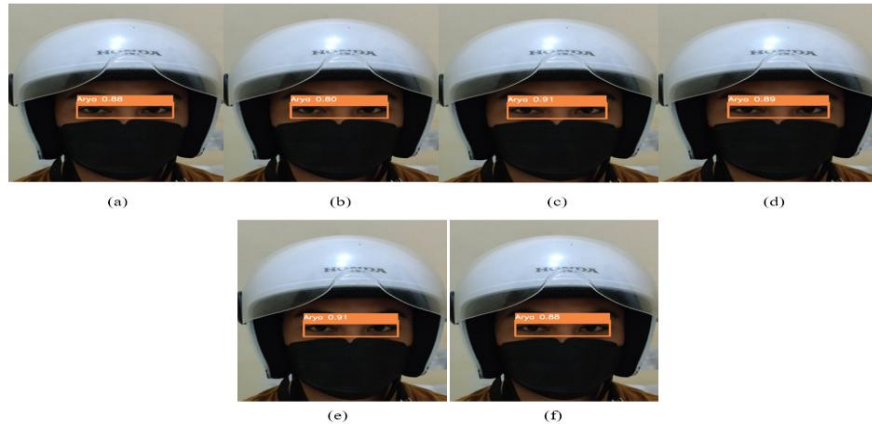


Figure 19. Example of detection and recognition results

Figure 19 shows the detection and recognition results performed by 6 different models, labels a, b, c, d, e, and f are YOLOv11n, YOLOv11nMod, YOLOv11s, YOLOv11sMod, YOLOv12n, and YOLOv12s models. It is clear that labels a, b, c to label f do not have differences in the location of the bounding box, but the only difference is the level of confidence. In the nano version, YOLOv12 has the highest confidence value, while in the small version, YOLOv11 has the highest confidence value, for clearer detection and recognition we use YOLOv11sMod because it gets the highest accuracy with the mAP50-95 metric compared to the other 5 models in providing detection and recognition results for several samples from several classes shown in Figure 20.



Figure 20. Detection and recognition results of some samples

Figure 20 is the result of some eye detection-based recognition results, with the recognition label directly in the bounding box above the eye. The recognition performed by our proposed model from the

example of Figure 20 always has a confidence score above 0.5 in performing recognition so that the recognition carried out is good.

4. DISCUSSIONS

This study successfully demonstrates that the proposed method is capable of producing high face recognition accuracy, equivalent to the accuracy obtained from full face recognition and when the face is covered by a mask. This demonstrates the superiority of the developed approach in handling conditions where the face is not fully visible. Various previous studies have demonstrated high levels of accuracy in full face recognition. Studies such as [3], [5], [10], [11], [1], [7], [9], [8], [4], and [12] show consistent success in this regard. For example, [10], [11], [4], and [12] use Convolutional Neural Network (CNN)-based methods, with [4] combining SIFT feature extraction, successfully achieving accuracies of 98%, 97.9%, 100%, and 95.46%, respectively. In addition, other approaches such as PCA with Triplet Similarity[3], Faster R-CNN[5], AB-FR[1], LDA[7], R-CNN[9], GhosFaceNetV2-1 (MSIMV3)[8] also recorded high results, with accuracies of 98.75%, 99.3%, 99.53%, 97.29%, 99.6%, and 99.86%. Despite the impressive results, all of these approaches rely on full-face images, which makes them less effective in conditions where the face is covered by a mask or other objects.

This phenomenon has led to the emergence of various follow-up studies, such as those conducted by [16], [6], [2], [15], [31], and [14], which focused on face recognition even though part of the face is covered. Studies [2], [31], and [14] using the CNN approach showed high accuracy, 93.58%, 94.1%, and 99.53%, respectively. Similarly, the CVSAN[16], E3FRM[6], and R34-AmaskNet[15] methods recorded accuracies of 99.2%, 96.3%, and 94.3%. However, although the results are satisfactory, the accuracy will decrease as more parts of the face are covered. To overcome this limitation, several studies have focused on parts of the face that remain visible even though much of the face is covered, such as the eyes and eyebrows. The study by [17] used KAZE, HOG, and SING feature extraction classified with the Naïve Bayes algorithm and achieved an accuracy of 97.87%. Meanwhile [18] utilizes multiscale analysis through scale-space, Discrete wavelet transform (DWT), and curvature-based methods, and uses SVM classification. The result, the recognition accuracy of the eye area is 76.04% and for eyebrows is 98.6%. However, the weakness is in the accuracy of eye recognition. For comparison, the method proposed in this study adopts the YOLOv11 approach in two versions, namely nano and small with modifications in eye detection-based recognition. The results themselves, successfully provide model accuracy performance reaching 99.3% for the nano version and 99.4% for the small version. In the mAP50-95 metric, there is an increase in accuracy from the baseline of 1.6% for the nano version and 1.3% for the small version of the original version. Of course, with a lower inference time.

This is the main contribution of this research. These results were achieved by modifying the architecture, by cutting a special layer for large objects, eyes on the face and adding the DySample technique. This is based on research [25] which successfully increased the accuracy of small detection from special layer cutting for small objects and DySample which can increase the resolution capability of the target image without burdening the existing computational load. That is what made modifications to the object's condition so that there was an increase in the model's ability in the mAP50-95 metric with a lower inference time.

5. CONCLUSION

Our proposed model outperforms YOLOv11 and YOLOv12 based on the mAP50-95 metric for both the nano and small versions. In the nano version, while our model's mAP50 accuracy is comparable to YOLOv11n and YOLOv12n, our model outperforms YOLOv11n by 1.6% in the mAP50-95 metric, achieving 75% accuracy. In the small version, our model also shows a 1.3% improvement in mAP50-95 accuracy compared to YOLOv11s and a 1.2%

improvement in YOLOv12s. However, for the mAP50 metric, our model has the same accuracy as YOLOv11s and is 0.1% better than YOLOv12s. However, precision and recall do not show significant improvements for both the nano and small versions. The improvement in mAP50-95 accuracy is achieved by architectural modifications, which, although leading to parameter improvements, are still within the nano and small categories. Therefore, the main contribution of this research is the improvement of the accuracy of the mAP50-95 metric with shorter inference time, thus supporting practical implementation on devices such as the Raspberry Pi in field applications. Suggestions for further research include adding data with different lighting conditions for each class and more varied data collection distances, as well as its implementation on a Raspberry Pi device for field implementation.

REFERENCES

- [1] X. Qi, C. Wu, Y. Shi, H. Qi, K. Duan, and X. Wang, "A Convolutional Neural Network Face Recognition Method Based on BiLSTM and Attention Mechanism," *Comput. Intell. Neurosci.*, vol. 2023, no. 1, pp. 1–14, 2023, doi: 10.1155/2023/2501022.
- [2] W. C. Cheng, H. C. Hsiao, and L. H. Li, "Deep Learning Mask Face Recognition with Annealing Mechanism," *Appl. Sci.*, vol. 13, no. 2, p. 732, 2023, doi: 10.3390/app13020732.
- [3] B. Bazatbekov, C. Turan, S. Kadyrov, and A. Aitimov, "2D face recognition using PCA and triplet similarity embedding," *Bull. Electr. Eng. Informatics*, vol. 12, no. 1, pp. 580–586, 2023, doi: 10.11591/eei.v12i1.4162.
- [4] H. Benradi, A. Chater, and A. Lasfar, "A hybrid approach for face recognition using a convolutional neural network combined with feature extraction techniques," *IAES Int. J. Artif. Intell.*, vol. 12, no. 2, pp. 627–640, 2023, doi: 10.11591/ijai.v12.i2.pp627-640.
- [5] G. Rajeshkumar *et al.*, "Smart office automation via faster R-CNN based face recognition and internet of things," *Meas. Sensors*, vol. 27, no. November 2022, p. 100719, 2023, doi: 10.1016/j.measen.2023.100719.
- [6] A. Shah, B. Ali, M. Habib, J. Frnda, I. Ullah, and M. Shahid Anwar, "An ensemble face recognition mechanism based on three-way decisions," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 4, pp. 196–208, 2023, doi: 10.1016/j.jksuci.2023.03.016.
- [7] D. Sunaryono, J. Siswantoro, and R. Anggoro, "An android based course attendance system using face recognition," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 33, no. 3, pp. 304–312, 2021, doi: 10.1016/j.jksuci.2019.01.006.
- [8] M. Alansari, O. A. Hay, S. Javed, A. Shoufan, Y. Zweiri, and N. Werghi, "GhostFaceNets: Lightweight Face Recognition Model From Cheap Operations," *IEEE Access*, vol. 11, no. March, pp. 35429–35446, 2023, doi: 10.1109/ACCESS.2023.3266068.
- [9] A. T. Putra, K. Usman, and S. Saidah, "Webinar Student Presence System Based on Regional Convolutional Neural Network Using Face Recognition," *J. Tek. Inform.*, vol. 2, no. 2, pp. 109–118, 2021, doi: 10.20884/1.jutif.2021.2.2.82.
- [10] E. Tanuwijaya, A. S. A. Setiawan, A. R. Arianindita, and T. Kristanto, "Human Face Recognition on Image Video Conference Application Using Siamese Network With Skip Connection Smaller Vgg Model," *J. Tek. Inform.*, vol. 4, no. 5, pp. 1119–1125, 2023, doi: 10.52436/1.jutif.2023.4.5.981.
- [11] E. Tanuwijaya, R. L. Lordianto, and R. A. Jasin, "Recognition of Human Faces in Video Conference Applications Using the Cnn Pipeline Pengenalan Wajah Manusia Pada Aplikasi Video Conference Menggunakan Metode Pipeline Cnn," *J. Tek. Inform.*, vol. 3, no. 2, pp. 421–427, 2022, doi: 10.20884/1.jutif.2022.3.2.219.
- [12] H. O. Ikromovich and B. B. Mamatkulovich, "FACIAL RECOGNITION USING TRANSFER LEARNING IN THE DEEP CNN," *Web Sci. Int. Sci. Res. J.*, vol. 5, no. 3, pp. 1–14, 2023, doi: 10.17605/OSF.IO/NRMK2.
- [13] O. A. Naser, S. M. S. Ahmad, K. Samsudin, M. Hanafi, S. M. B. Shafie, and N. Z. Zamri, "Facial recognition for partially occluded faces," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 30, no. 3,

- pp. 1846–1855, 2023, doi: 10.11591/ijeecs.v30.i3.pp1846-1855.
- [14] F. Firdaus and R. Munir, “Masked Face Recognition using Deep Learning based on Unmasked Area,” in *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, 2022, pp. 1–6. doi: 10.1109/ICPC2T53885.2022.9776651.
 - [15] M. Zhang, R. Liu, D. Deguchi, and H. Murase, “Masked Face Recognition With Mask Transfer and Self-Attention Under the COVID-19 Pandemic,” *IEEE Access*, vol. 10, no. January, pp. 20527–20538, 2022, doi: 10.1109/ACCESS.2022.3150345.
 - [16] Y. Ge, H. Liu, J. Du, Z. Li, and Y. Wei, “Masked face recognition with convolutional visual self-attention network,” *Neurocomputing*, vol. 518, no. C, pp. 496–506, 2023, doi: 10.1016/j.neucom.2022.10.025.
 - [17] S. Ramachandra and S. Ramachandran, “Region specific and subimage based neighbour gradient feature extraction for robust periocular recognition,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 7961–7973, 2022, doi: 10.1016/j.jksuci.2022.07.013.
 - [18] R. Lionnie, C. Apriono, and D. Gunawan, “Eyes versus Eyebrows: A Comprehensive Evaluation Using the Multiscale Analysis and Curvature-Based Combination Methods in Partial Face Recognition,” *Algorithms*, vol. 15, no. 6, p. 208, 2022, doi: 10.3390/a15060208.
 - [19] S. Noris and A. Waluyo, “Penerapan Deep Learning untuk Klasifikasi Buah Menggunakan Algoritma Convolutional Neural Network (CNN),” *J. Teknol. Sist. Inf. dan Apl.*, vol. 6, no. 1, pp. 39–46, 2023, doi: 10.32493/jtsi.v6i1.29648.
 - [20] L. Zhang, B. Verma, D. Tjondronegoro, and V. Chandran, “Facial expression analysis under partial occlusion: A survey,” *ACM Comput. Surv.*, vol. 51, no. 2, pp. 1–49, 2018, doi: 10.1145/3158369.
 - [21] P. Hidayatullah, *Buku Sakti Deep Learning: Computer Vision Menggunakan YOLO Untuk Pemula*. Bandung: Informatika, 2021.
 - [22] X. Mao *et al.*, “COCO-O: A Benchmark for Object Detectors under Natural Distribution Shifts,” *Proc. IEEE Int. Conf. Comput. Vis.*, no. 2020, pp. 6316–6327, 2023, doi: 10.1109/ICCV51070.2023.00583.
 - [23] Z. Li, Q. He, and W. Yang, “E-FPN: an enhanced feature pyramid network for UAV scenarios detection,” *Vis. Comput.*, vol. 41, no. 1, pp. 675–693, 2025, doi: 10.1007/s00371-024-03355-w.
 - [24] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, “TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2021-Octob, pp. 2778–2788, 2021, doi: 10.1109/ICCVW54120.2021.00312.
 - [25] P. Hidayatullah *et al.*, “Computer Methods and Programs in Biomedicine DeepSperm : A robust and real-time bull sperm-cell detection in densely populated semen videos,” *Comput. Methods Programs Biomed.*, vol. 209, no. C, p. 106302, 2021, doi: 10.1016/j.cmpb.2021.106302.
 - [26] Z. Lin, B. Yun, and Y. Zheng, “LD-YOLO: A Lightweight Dynamic Forest Fire and Smoke Detection Model with Dysample and Spatial Context Awareness Module,” *Forests*, vol. 15, no. 9, p. 1630, 2024, doi: 10.3390/f15091630.
 - [27] W. Liu, H. Lu, H. Fu, and Z. Cao, “Learning to Upsample by Learning to Sample,” in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 6004–6014. doi: 10.1109/ICCV51070.2023.00554.
 - [28] Z. Chen, J. Feng, K. Zhu, Z. Yang, Y. Wang, and M. Ren, “YOLOv8-ACCW: Lightweight Grape Leaf Disease Detection Method Based on Improved YOLOv8,” *IEEE Access*, vol. 12, no. June, pp. 123595–123608, 2024, doi: 10.1109/ACCESS.2024.3453379.
 - [29] L. He, Y. Zhou, and L. Liu, “Research and Application of YOLOv11-Based Object Segmentation in Intelligent Recognition at Construction Sites,” *Buildings*, vol. 14, no. 12, p. 3777, 2024, doi: 10.3390/buildings14123777.
 - [30] Perez and Caldentey, “A Novel YOLOv10-DECA Model for Real-Time Detection of Concrete Cracks Chaokai,” *Buildings*, vol. 14, no. 10, p. 3230, 2024, doi: 10.3390/buildings14103230.
 - [31] T.-H. Tsai, J.-X. Lu, X.-Y. Chou, and C.-Y. Wang, “Joint Masked Face Recognition and Temperature Measurement System Using Convolutional Neural Networks,” *Sensors*, vol. 23, no. 6, p. 2901, Mar. 2023, doi: 10.3390/s23062901.