YOLOv8nGM: An Improved Speed of YOLOv8n Object Detection

Chandra H. Heruatmadja [1,*], Harjanto Prabowo [2], Harco Leslie Hendric Spits Warnars [1], and Yaya Heryadi [1]

[1] Doctor of Computer Science Department, Binus Graduate Program, Bina Nusantara University, Jakarta, Indonesia
[2] Management Department, BINUS Business School Undergraduate Program, Bina Nusantara University, Jakarta, Indonesia
Email: chandra.heruatmadja@binus.ac.id (C.H.H.); harprabowo@binus.ac.id (H.P.); spits.hendric@binus.ac.id (H.L.H.S.W.); yayaheryadi@binus.edu (Y.H.)
*Corresponding author

***Abstract*—The rapid advancement of technology has significantly enhanced object detection tasks that require high speed and accuracy. Although the YOLOv8n model is known for its efficient and accurate detection capabilities, its detection process still involves multiple complex stages, limiting its deployment in challenging environments. This research investigates how the YOLOv8n architecture can be modified to enhance inference speed without compromising detection accuracy. The objective is to propose a modification to the YOLOv8n backbone by integrating a gating mechanism on top of the C2f module. The model, YOLOv8nGM, utilizes a sigmoid activation function to implement gating, converting network outputs into probabilities that enable selective computation depending on input complexity. This adaptive mechanism enables the network to bypass unnecessary calculations, thereby reducing computational overhead. Experiments on the benchmark dataset reveal that YOLOv8nGM achieves an average inference time of 8.72 ms per image, representing an approximately 5.4% improvement in speed compared to the original YOLOv8n. Additionally, the GPU memory consumption was reduced from 0.27 GB to 0.21 GB, signifying more efficient resource use. The mAP50 metrics on the validation and test datasets showed that the proposed model achieved minor improvements of 0.02% and 0.008%, respectively, compared to the original model, indicating that accuracy was maintained despite faster computation. These findings demonstrate that the gating mechanism effectively balances inference speed and precision, making the YOLOv8nGM model suitable for real-time object detection applications such as warehouse inventory counting. This study advances object detection technology by providing a practical and computationally efficient solution for environments that require rapid and accurate recognition.**

***Keywords*—deep learning, YOLOv8n, gating mechanism, object detection**

I. Introduction

Deep learning technology is growing rapidly in various sectors and applications of human life, as deep learning systems have been widely used to identify objects in images over the last few years [1]. However, most detection engines must undergo several phases or processes to achieve accurate detection [2]. They are also challenging to apply in complex environments, such as those with variations in lighting angles and intensity at different distances and positions. Variations in brightness, shadows, contrast, position, and posture between the background and the target object can result in differences in the accuracy of object detection results. These challenges have significantly hindered the progress of object detection technology, particularly in fields that demand high accuracy and precision, such as military navigation, spatial intelligence surveillance, robotic vision, autonomous driving, and human interaction [3].

These limitations have led to the development of a new object detection method called YOLO, which stands for "You Only Look Once". Launched in 2023, YOLO is described as a real-time detection system capable of identifying and classifying objects within a single image pass [2, 4]. By the time this research was conducted, the latest version, YOLOv8, features a novel backbone architecture named CSPDarknet-AA, which is an enhanced iteration of the CSPDarknet series known for its efficiency and firm performance in object detection tasks [5].

YOLOv8's architecture consists of three main modules: the head, backbone, and neck. The head module generates the final outputs, including bounding box coordinates, object confidence scores, and class labels. The backbone is a complex Convolutional Neural Network (CNN) designed to extract features at multiple scales from the input images, capturing both low-level textures and high-level semantic details critical for precise object detection. The neck module refines and merges these multi-scale features using an enhanced version of the Path Aggregation Network (PANet), which optimizes information flow across different feature levels to detect objects of various sizes effectively. Additionally, YOLOv8 employs an anchor-free method for bounding box prediction, simplifying the process by reducing the number of hyperparameters and enhancing adaptability to objects with diverse aspect ratios and scales [6, 7].

Manuscript received June 30, 2025; revised July 20, 2025; accepted August 25, 2025; published December 5, 2025.

Although the detection results of YOLOv8 are promising, further research is being conducted to enhance its capabilities for specific needs by modifying the existing model structure. The YOLOv8 model detection technology, inserted and embedded in the drone, can detect small objects with noticeable, accurate, and precise image results [8]. In automotive industry, YOLOv8 can also be modified by replacing the SIoU bounding box with Wise-IoUv3 which is used as a detector sensor on cars that can detect conditions around the vehicle clearly and precisely such as other cars, cyclists in very dim environments, and even able to detect pedestrians with petite portrait sizes and difficult to recognize [9].

In line with the results of previous studies, there is Transformer-YOLOv8, a model that uses a YOLOv8 backbone network with gating mechanism and a transformer-based detection head which installed as a sensor can detect several objects around the car in the form of vehicles such as cars, taxis, and buses, pedestrians, and surrounding city infrastructure such as tall buildings, road signs, and trees [10]. In the manufacturing industry, the modified YOLOv8, which replaces the SIoU bounding box with EfficiCIoU, produces the EV-YOLOv8 model. This model can detect small targets, such as minor damage on the surface of wind turbine blades [11]. YOLOv8 enhanced with SPPCSPC module to replace SPPF module and add SPD layer can be used to detect various defects on metal surface, including punching, crescent cracks, waist folds, rolled pits, water spots, silk spots, inclusions, oil spots, creases, and weld lines, which is very useful for the steel industry [5].

YOLOv8 detection sensor technology is also used in areas not related to manufacturing technology, in the livestock sector, YOLOv8n installed in pigeon cages can detect the quantity and quality of eggs such as egg size, egg shell thickness, and so on so that breeders can use it to increase the productivity and breeding of egg-producing pigeons [12], and in the agricultural sector, YOLOv8s installed in orchards can detect the quantity and quality of peaches such as fruit size, number of peach seedlings, and so on so that farmers can use them to increase productivity and cultivation of peaches [13].

Based on the various studies explained, this study aims to modify the structure of the YOLOv8n model to achieve a better detection speed without compromising its accuracy level in a retail warehouse environment. The speed of YOLOv8 is crucial for facilitating inventory calculation by quickly and accurately identifying stock items within a limited time window for inventory counting. This study will examine the effect of modifying the YOLOv8n backbone with a gating mechanism to measure inference speed and GPU consumption.

II. Literature Review

A. Related Work

The rapid development of object detection technology enables many tasks to be completed easily and effectively, one of which is the YOLO deep learning model, which can detect objects quickly and accurately. However, several other studies have been conducted to improve the performance of YOLO itself, one of which involves a gating mechanism. The gating mechanism is used in many artificial neural network models to facilitate the backpropagation step [14]. For example, the gating mechanism can control flow information in network neurons, and it is often used in models such as LSTM and GRU to manage flow information and maintain dependence term long [15]. Gating can involve several mechanisms, such as a bistable neuron that switches between states to organize flow information [16]. Likewise, the gating mechanism can also be applied to the YOLO deep learning model. The YOLO model is designed to detect objects in real-time, striking a balance between high speed and accuracy. Notably, YOLOv8n, a variant of YOLOv8, offers improvements in accuracy detection and computational efficiency [6, 17]. This YOLOv8n model is applied as shown in the sign, then cross-detects vehicles, and introduces face detection [18–20]. The review results in Table I present findings from several articles that discuss gating mechanisms and YOLO.

TABLE I. Review Results from Several Articles Related to the Gating Mechanism and YOLOv8n

No.	Ref.	Year	Country	Result
1.	[13]	2024	China	YOLO-Peach is a technology developed using the YOLOv8n model to enhance efficiency and operational scientific management in gardens, specifically through help detection and automatic quantification of seeds and fruits in peaches
2.	[16]	2020	China	A new gating mechanism utilizing an RNN in a recurrent system can overcome a failed gating switch and its associated weak performance. This mechanism can expand gating activation, which can help RNN reach a minimum possibility.
3.	[21]	2021	China	EIoU was developed from Intersection over Union (IoU) as an algorithm evaluation metric, utilizing a framework for detecting vehicles, to enhance detection performance.
4.	[5]	2024	China	YOLO-ADS can effectively detect defects in metal.
5.	[22]	2019	China	A gating mechanism applied to a Deep Neural Network (DNN) can capture more useful information, thereby enhancing the performance of the gating mechanism.
6.	[23]	2024	US	Deep learning is known to help in the process of making associated decisions with recurrent convolutional neural networks.
7.	[8]	2023	China	Drone-YOLO can help detect objects in UAV images that are very small in size, such as those in drone images.
8.	[24]	2021	South Korea	An assisted gating system with DNN technology is known to increase effectiveness and provide more economical power.
9.	[14]	2020	UK	Collaborative gating systems with chrono initialization and ordered neurons can enhance performance in gating systems that memorize synthesis tasks, classify pictures sequentially, model language, and reinforce learning, particularly when the usage term is prolonged.

10.	[25]	2017	Australia	Recurrent neural networks in gating systems with a stacked convolution approach can enhance efficiency by enabling the parallel processing of consecutive tokens.
11.	[26]	2020	US	Researchers developed a convolutional neural network to support language modeling. With a novel gating mechanism, the network enhances security while consuming minimal power.
12.	[27]	2023	China	The detection method bubble has essential applications in industry. YOLOv8n was developed with a Deformable Convolution Network (DCN) to replace the original C2f module, which is known to capture features important from the target, especially bubbles, in a more effective and precise manner compared to the native YOLOv8n module.
13.	[28]	2020	Canada	Canada Network GoogLeNet and Inception-v2, Inception-v3, and Inception-v4 can significantly increase the accuracy of CNN-based models and reduce computational costs.
14.	[6]	2024	Pakistan	Pakistan YOLOv8n can be easily modified or developed to meet specific needs, such as integration with the Python package and CLI. It is a sophisticated solution that can detect continuous object development.

B. *Gating Mechanism*

The gating mechanism was first introduced by Szegedy *et al.* [29], who proposed a new CNN architecture designed to enhance network performance in image recognition tasks without significantly increasing computational complexity. The core of this innovation is the inception module, which utilizes multiple convolutional filter sizes in each layer and combines the results.

In this research study, the idea behind this gating mechanism is to allow the deep learning model to adaptively decide whether to perform an expensive complete computation (full block pass) or a lighter and cheaper computation (light pass) based on the complexity of the input features at a particular stage [28]. When the gating condition is met, in this case, if the neural network model considers the feature to be complex enough, the model will perform convolution through the standard module of YOLOv8n, the C2f module. However, if the feature is considered simple, the model only performs simple convolution with a kernel size of 1×1, thus saving time without sacrificing accuracy [30].

III. Materials and Methods

A. *Data Preparation*

The dataset used in this research is a proprietary dataset captured directly from a retail goods inventory warehouse, comprising a total of 4332 images with a resolution of 1080×1080 pixels as illustrated in Fig. 1. The images undergo an augmentation process to add variations, thereby enriching the dataset and enhancing the model's detection capabilities during training. The augmentation technique for the dataset is carried out using random brightness and contrast adjustments, rotation, blurring, and resizing to 640×640 pixels to ensure compatibility with YOLO, via the Python Albumentation library [31].

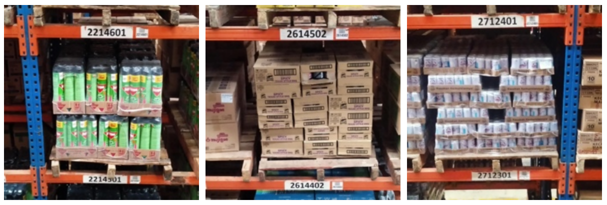
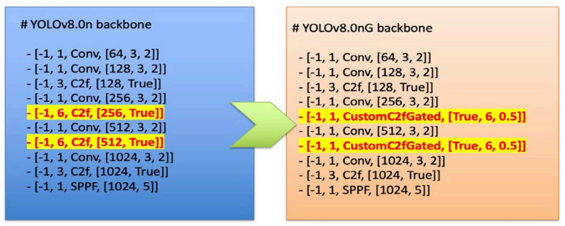


Fig. 1. Image dataset for training, validation, and testing.

Inside the images dataset, there are three image categories, consist of 3359 images of boxes, 500 images of plastic wrap, and 79 images of sacks, where the images are not publicly available, which are then divided proportionally using the stratified sampling technique [32]. The dataset is divided into 80%–20%, where 80% is used for training, and 20% for validation and testing, resulting in a division of 3464 images for training, 434 images for validation, and 434 images for testing. In addition, to assess the performance of the proposed model on a larger dataset. The images dataset are not publicly available due to restricted acess into the warehouse, however this study also conducts model testing using the COCO open dataset with a greater number of images and classes. The COCO dataset is openly available from Microsoft at https://cocodataset.org/#download [33].

B. *YOLOv8nGM Model*

The proposed model, named YOLOv8nGM, is a modification of the YOLOv8n model, as illustrated in Fig. 2. YOLOv8n was chosen because, during this research, YOLOv8 was the most widely used version of YOLO for research and remains relevant to deep learning research. Among the various versions of YOLOv8, YOLOv8n was reported to be the fastest, more compact, and with fewer Floating-Point Operations Per Second (FLOPS) [34].



```
# YOLOv8.0n backbone

 - [-1, 1, Conv, [64, 3, 2]]
 - [-1, 1, Conv, [128, 3, 2]]
 - [-1, 3, C2f, [128, True]]
 - [-1, 1, Conv, [256, 3, 2]]
 - [-1, 6, C2f, [256, True]]
 - [-1, 1, Conv, [512, 3, 2]]
 - [-1, 6, C2f, [512, True]]
 - [-1, 1, Conv, [1024, 3, 2]]
 - [-1, 3, C2f, [1024, True]]
 - [-1, 1, SPPF, [1024, 5]]
```

```
# YOLOv8.0nG backbone

 - [-1, 1, Conv, [64, 3, 2]]
 - [-1, 1, Conv, [128, 3, 2]]
 - [-1, 3, C2f, [128, True]]
 - [-1, 1, Conv, [256, 3, 2]]
 - [-1, 1, CustomC2fGated, [True, 6, 0.5]]
 - [-1, 1, Conv, [512, 3, 2]]
 - [-1, 1, CustomC2fGated, [True, 6, 0.5]]
 - [-1, 1, Conv, [1024, 3, 2]]
 - [-1, 3, C2f, [1024, True]]
 - [-1, 1, SPPF, [1024, 5]]
```

Fig. 2. Proposed YOLOv8nGM backbone.

YOLOv8 has been extensively tested for its accuracy and speed, which are superior to those of other deep learning models, such as RetinaNet and Mask R-CNN, in both of which can detect images quickly and accurately [35].

The selection of the YOLOv8n model variant, rather than other variants of YOLOv8, is motivated by the study’s objective to demonstrate the superior processing speed of the fastest model among its peers which is crucial while maintaining its accuracy level in a retail warehouse environment as shown in Table II. In addition, compared to other versions of YOLO, YOLOv8 offers state-of-the-art performance in terms of accuracy and speed, building upon the advancements made in previous YOLO versions—YOLOv5, YOLOv6, and YOLOv7—as demonstrated in Figs. 3 and 4 [36].

TABLE II. VARIOUS VERSIONS OF YOLOV8

Model	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	FLOPS (B)
YOLOv8n	80.4	0.99	8.7
YOLOv8s	128.4	1.20	28.6
YOLOv8m	234.7	1.83	78.9
YOLOv8l	375.2	2.39	165.2
YOLOv8x	479.1	3.53	257.8

Other study has also shown that each YOLOv8 variants achieves higher throughput with roughly the same number of parameters compare to its predecessors, suggesting architectural optimizations that enhance hardware efficiency [37].

Moreover, the gating mechanism approach in other YOLO models requires further research because there are fundamental architectural differences between one YOLO model and others [38, 39].

The changes made to the original YOLOv8n model involve adding a condition to the convolution processes, based on the results of the layers above it, which depends on the complexity of the features expressed in the calculation of the sigmoid activation function, ranging from 0 to 1. The sigmoid activation function is used to convert the network output into interpretable probabilities by mapping the input to an output in the range of 0 to 1 [40]. A feature is stated as simple if the sigmoid value is 0, and a feature is noted as complex if the sigmoid value is 1.

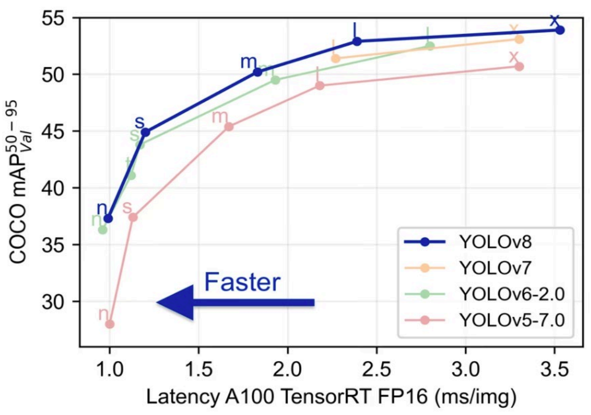


Fig. 3. YOLO versions & variants speed comparison.

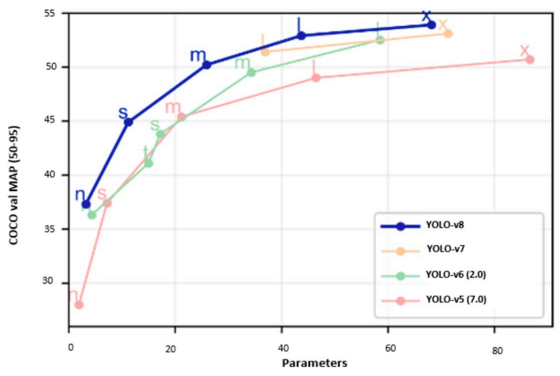


Fig. 4. YOLO throughput comparison.

Using the addition of this logic function, there is an insertion into the C2f backbone section to implement the concept of the gating mechanism by adding a condition statement and sigmoid function from the original YOLOv8n backbone [27], which is illustrated in Fig. 5.

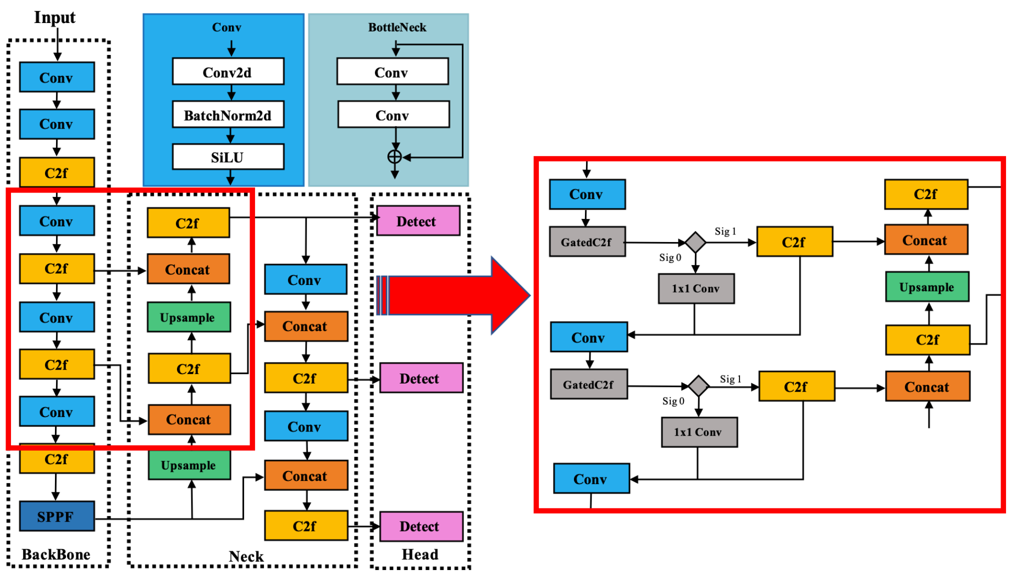
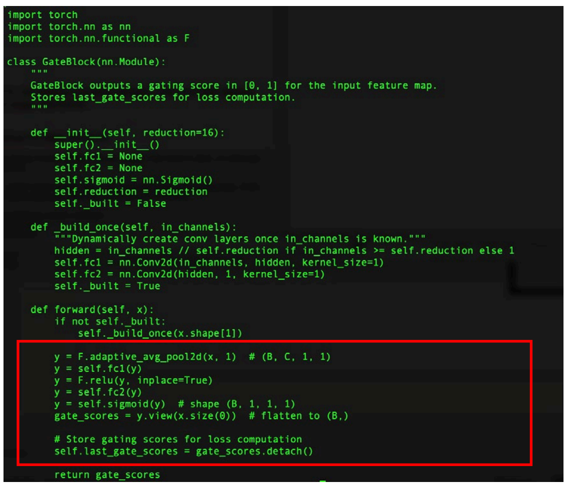


Fig. 5. Modified C2f section.

The C2f module in YOLOv8 replaces earlier components in previous YOLO, such as C2 and C3. It enhances gradient flow and detection accuracy while maintaining the model's efficiency and lightness. The C2f module is located in the backbone and neck of YOLOv8, enhancing the processing, sharing, and refinement of feature maps across layers. This module is optimized to handle complex object detection tasks without having heavy computational resources [41]. However, the C2f module still utilizes at least 64 extraction filters, which creates a research gap for non-complex features that can be extracted using simple filters on a 1×1 convolutional layer [30], rather than being passed through the complex C2f module.

The model revision depicted in Fig. 5 is implemented by augmenting the YOLOv8n backbone with extra code components, as described subsequently:

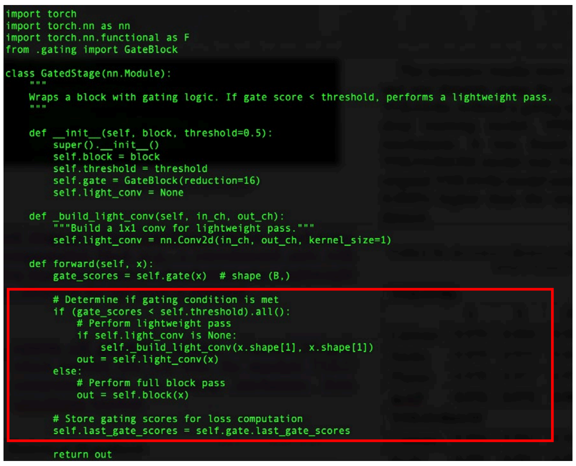
gating.py defines GateBlock object class, outputting a scalar gating score ∈ [0, 1] per input, indicative of feature "complexity" or "significance" as illustrated in Fig. 6.



```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class GateBlock(nn.Module):
    """
    GateBlock outputs a gating score in [0, 1] for the input feature map.
    Stores last_gate_scores for loss computation.
    """

    def __init__(self, reduction=16):
        super().__init__()
        self.fc1 = None
        self.fc2 = None
        self.sigmoid = nn.Sigmoid()
        self.reduction = reduction
        self._built = False

    def _build_once(self, in_channels):
        """Dynamically create conv layers once in_channels is known."""
        hidden = in_channels // self.reduction if in_channels >= self.reduction else 1
        self.fc1 = nn.Conv2d(in_channels, hidden, kernel_size=1)
        self.fc2 = nn.Conv2d(hidden, 1, kernel_size=1)
        self._built = True

    def forward(self, x):
        if not self._built:
            self._build_once(x.shape[1])

        y = F.adaptive_avg_pool2d(x, 1)  # (B, C, 1, 1)
        y = self.fc1(y)
        y = F.relu(y, inplace=True)
        y = self.fc2(y)
        y = self.sigmoid(y)  # shape (B, 1, 1, 1)
        gate_scores = y.view(x.size(0))  # flatten to (B,)

        # Store gating scores for loss computation
        self.last_gate_scores = gate_scores.detach()

        return gate_scores
```

Fig. 6. source code of gating.py.

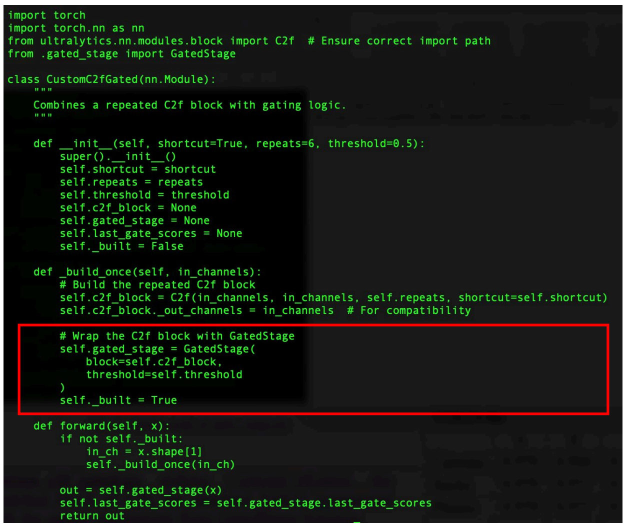
gated_stage.py defines GatedStage object class, which wraps a computational block (e.g., a convolutional unit) with gating logic as illustrated in Fig. 7; scores below a threshold invoke a lightweight pass, otherwise the full block is executed.



```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from .gating import GateBlock

class GatedStage(nn.Module):
    """
    Wraps a block with gating logic. If gate score < threshold, performs a lightweight pass.
    """

    def __init__(self, block, threshold=0.5):
        super().__init__()
        self.block = block
        self.threshold = threshold
        self.gate = GateBlock(reduction=16)
        self.light_conv = None

    def _build_light_conv(self, in_ch, out_ch):
        """Build a 1x1 conv for lightweight pass."""
        self.light_conv = nn.Conv2d(in_ch, out_ch, kernel_size=1)

    def forward(self, x):
        gate_scores = self.gate(x)  # shape (B,)

        # Determine if gating condition is met
        if (gate_scores < self.threshold).all():
            # Perform lightweight pass
            if self.light_conv is None:
                self._build_light_conv(x.shape[1], x.shape[1])
            out = self.light_conv(x)
        else:
            # Perform full block pass
            out = self.block(x)

        # Store gating scores for loss computation
        self.last_gate_scores = self.gate.last_gate_scores

        return out
```

Fig. 7. source code of gate_stage.py.

custom_c2f_gated.py defines CustomC2fGated object class, the modified block of C2f, the standard YOLO component with the gating mechanism from GatedStage function as illustrated in Fig. 8.



```python
import torch
import torch.nn as nn
from ultralytics.nn.modules.block import C2f  # Ensure correct import path
from .gated_stage import GatedStage

class CustomC2fGated(nn.Module):
    """
    Combines a repeated C2f block with gating logic.
    """

    def __init__(self, shortcut=True, repeats=6, threshold=0.5):
        super().__init__()
        self.shortcut = shortcut
        self.repeats = repeats
        self.threshold = threshold
        self.c2f_block = None
        self.gated_stage = None
        self.last_gate_scores = None
        self._built = False

    def _build_once(self, in_channels):
        # Build the repeated C2f block
        self.c2f_block = C2f(in_channels, in_channels, self.repeats, shortcut=self.shortcut)
        self.c2f_block._out_channels = in_channels  # For compatibility

        # Wrap the C2f block with GatedStage
        self.gated_stage = GatedStage(
            block=self.c2f_block,
            threshold=self.threshold
        )
        self._built = True

    def forward(self, x):
        if not self._built:
            in_ch = x.shape[1]
            self._build_once(in_ch)

        out = self.gated_stage(x)
        self.last_gate_scores = self.gated_stage.last_gate_scores
        return out
```

Fig. 8. Source code of modified C2f block.

C. *Statistical Hypothesis*

To validate whether the difference in object identification processing speed using the proposed model compared with the original model is statistically significant, statistical tests are carried out on the results of each validation and testing process using the modified model and the original model. The statistical test carried out used a significance test of 2 independent variables, in which statistical tests generally use the t-test for normally distributed data, or the Mann-Whitney test for non-normally distributed data [42], using the research hypothesis:

H_0 : μYOLOv8n = μYOLOv8nGM, there is no significant difference between the speed of the original model and the proposed model.

H_a : μYOLOv8n ≠ μYOLOv8nGM, there is a significant difference between the speed of the original model and the proposed model.

μYOLOv8n is the mean of the process speed of the YOLOv8n model, and μYOLOv8nGM is the mean of the processing speed of the YOLOv8nGM model.

IV. Results and Discussion

A. *Accuracy and Speed*

First, both the original YOLOv8n, without a gating mechanism, and the YOLOv8nGM models with a gating mechanism were trained on a 3464-image dataset that contains cartons, plastic wrap, and sacks, for 330 epochs. Each epoch consists of 216 iterations, and each iteration consists of 16 images. After the training of both models was completed, the accuracy level was measured using the mAP50 metric. mAP50-95 refers to the mean average precision calculated over a range of IoU thresholds from 0.50 to 0.95, with increments of 0.05. Generally, the mAP50-95 index is the most stringent, as it considers a broader range of IoU thresholds, while mAP50 has the lowest threshold and less stringent requirements.

The accuracy results were obtained from the validation and test datasets using the original deep learning model as presented in Tables III and IV, YOLOv8n, without a gating mechanism, and the modified deep learning model, YOLOv8nGM, with a gating mechanism. It was found that the mAP50 of the YOLOv8nGM model was 0.02% higher than that of the original YOLOv8n model using the validation dataset, and 0.008% higher than the original model using the test dataset.

TABLE III. ACCURACY RESULTS FROM YOLOV8N VS YOLOV8NGM WITH VALIDATION DATASET

DL Model	Class	1	2	3	4	5	mAP@50
YOLOv8n	Cartons	0.970	0.966	0.960	0.956	0.986	**0.968**
	Sacks	0.979	0.972	0.955	0.942	0.991	**0.968**
	Plastic	0.978	0.977	0.971	0.963	0.989	**0.976**
	mAP@0.5	**0.976**	**0.972**	**0.962**	**0.954**	**0.989**	**0.970**
YOLOv8nGM	Cartons	0.966	0.963	0.956	0.969	0.974	**0.966**
	Sacks	0.992	0.982	0.972	0.992	0.939	**0.975**
	Plastic	0.972	0.968	0.964	0.976	0.994	**0.975**
	mAP@0.5	**0.977**	**0.971**	**0.964**	**0.979**	**0.969**	**0.972**

TABLE IV. ACCURACY RESULTS FROM YOLOV8N VS YOLOV8NGM WITH TEST DATASET

DL Model	Class	1	2	3	4	5	mAP@50
YOLOv8n	Cartons	0.971	0.966	0.959	0.955	0.986	**0.967**
	Sacks	0.930	0.928	0.921	0.907	0.966	**0.930**
	Plastic	0.964	0.961	0.957	0.947	0.982	**0.962**
	mAP@0.5	**0.955**	**0.952**	**0.946**	**0.936**	**0.978**	**0.953**
YOLOv8nGM	Cartons	0.969	0.964	0.958	0.971	0.976	**0.968**
	Sacks	0.954	0.946	0.935	0.957	0.956	**0.950**
	Plastic	0.970	0.966	0.963	0.973	0.963	**0.967**
	mAP@0.5	**0.964**	**0.959**	**0.952**	**0.967**	**0.965**	**0.961**

With an mAP value of 0.961 at an IoU of 0.5 on the test dataset, or an increase in accuracy of 0.008, it can be seen that the proposed model does not experience a decrease in accuracy. Based on the average performance measurement metrics of the object detection model using an Intersection over Union (IoU) threshold of 0.5, a mAP0.5 value greater than 0.7 is considered a very accurate model [43]. Hence, a value of 0.961 is more than enough.

The accuracy of object detection using the proposed model can also be demonstrated by the results, as shown in Fig. 9, where the proposed model successfully detects all objects, including various categories of boxes, plastic wrap, and sacks.

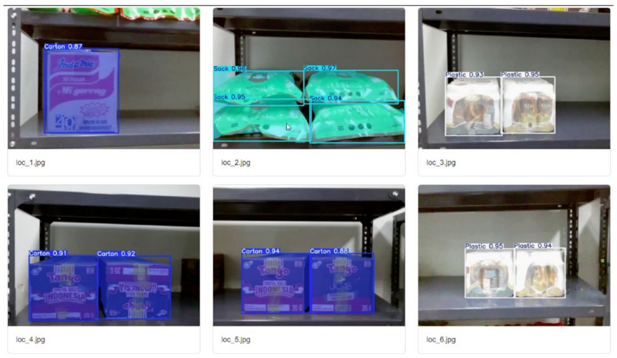


Fig. 9. Object detection using proposed model (YOLOv8nGM) in real environment.

Speed inference is the speed at which an artificial intelligence model generates output based on the data that has been given. In applications that utilize real-time object detection [44], the achieved speed is based on comparing the results of object identification with the validation and test datasets as illustrated in Figs. 10 and 11.

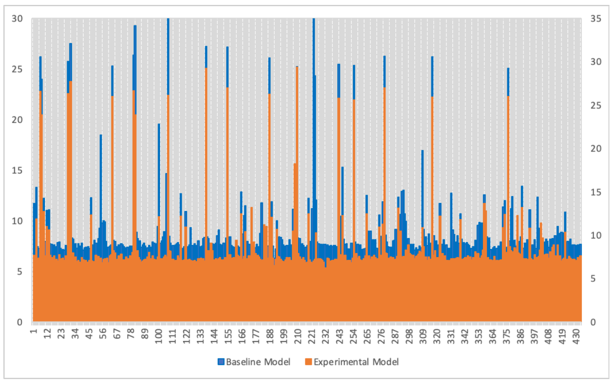


Fig. 10. Speed comparison from YOLOv8n vs YOLOv8nGM with validation dataset.

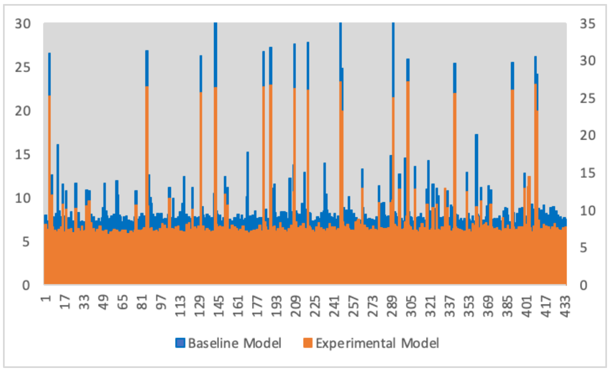


Fig. 11. Speed comparison from YOLOv8n vs YOLOv8nGM with test dataset.

The original YOLOv8n model, without a gating mechanism, achieves an average processing speed of 9.22 ms per image with validation dataset. In comparison, with the validation dataset, the YOLOv8nGM model with a gating mechanism has a processing speed of 8.72 ms per image, which is 0.5 ms (5.4%) faster than the original YOLOv8n model as presented in Table V, and for test dataset, the YOLOv8nGM model with a gating mechanism has a processing speed of 8.59 ms per image, which is 0.35 ms (3.9%) faster than the original YOLOv8n model as presented in Table VI.

TABLE V. SPEED RESULT FROM YOLOV8N VS YOLOV8NGM WITH VALIDATION DATASET

DL Model	1	2	3	4	5	Mean
YOLOv8n	9.86 ms	9.15 ms	8.96 ms	9.03 ms	9.10 ms	**9.22 ms**
YOLOv8nGM	9.03 ms	8.64 ms	8.59 ms	8.68 ms	8.64 ms	**8.72 ms**
Standar Dev.	4.38	4.34	4.24	4.19	4.16	

TABLE VI. SPEED RESULT FROM YOLOV8N VS YOLOV8NGM WITH TEST DATASET

DL Model	1	2	3	4	5	Mean
YOLOv8n	9.02 ms	8.89 ms	8.93 ms	8.83 ms	9.02 ms	**8.94 ms**
YOLOv8nGM	8.68 ms	8.82 ms	8.45 ms	8.53 ms	8.48 ms	**8.59 ms**
Standar Dev.	4.19	4.09	4.19	4.07	4.11	

Following the experiments on the COCO dataset, the proposed YOLOv8nGM model demonstrated superior object-identification speed, as illustrated in Fig. 12. The chart shows that, out of the 5000 COCO images evaluated, the majority yielded faster inference times with the proposed model, only a few images exhibited slower performance compared with the original model. On average, the proposed model required 7.87 ms per image, whereas the original model needed 8.29 ms, corresponding to an improvement of 0.42 ms (5.06%).

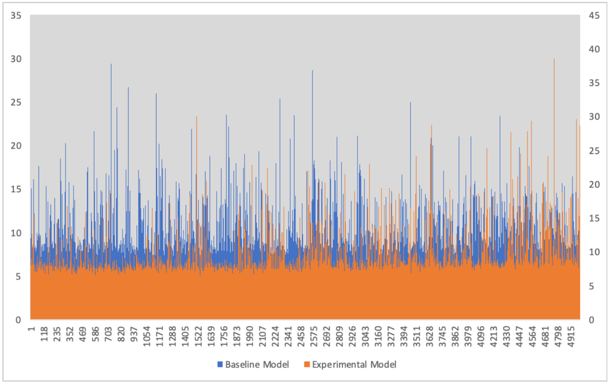


Fig. 12. Speed comparison from YOLOv8n vs YOLOv8nGM with COCO dataset.

This result is consistent with the result obtained from earlier test on a primary dataset. Nevertheless, this increase in processing speed slightly compromises the accuracy when compared with the original model. Based on the comparison of mAP@50 accuracy values across all 80 classes in the COCO dataset, 15 classes demonstrated that the proposed model achieved higher predictive accuracy results than the original model (18.75%), as shown in Table VII. However, it was observed that 52 out of the 80 COCO dataset classes (65%) showed that the proposed model had an accuracy difference of less than 2% compared to the original model. This represents a trade-off for improved speed (5.06% faster) [36].

TABLE VII. ACCURACY COMPARISON YOLOV8N VS YOLOV8NGM WITH COCO DATASET

Model		YOLOv8nGM		YOLOv8n	
Class	Images	mAP50	mAP 50–95	mAP50	mAP 50–95
person	2693	0.713	0.398	0.71	0.396
airplane	97	0.79	0.47	0.789	0.484
boat	121	0.369	0.156	0.335	0.143
fire hydrant	86	0.792	0.562	0.784	0.556
stop sign	69	0.675	0.579	0.671	0.577
sheep	65	0.632	0.371	0.627	0.364
cow	87	0.648	0.387	0.631	0.375
kite	91	0.524	0.242	0.523	0.258
wine glass	110	0.379	0.21	0.373	0.203
spoon	153	0.125	0.0488	0.122	0.0489
potted plant	172	0.312	0.131	0.308	0.132
tv	207	0.69	0.493	0.683	0.493
cell phone	214	0.396	0.248	0.365	0.226
book	230	0.127	0.0503	0.121	0.0468
vase	137	0.416	0.258	0.414	0.263

B. GPU Consumption

A GPU consumption test was performed to determine the quality of the proposed model. The more GPUs consumed, the higher the power usage, temperature, and memory usage, which can reduce the GPU's lifespan [45].

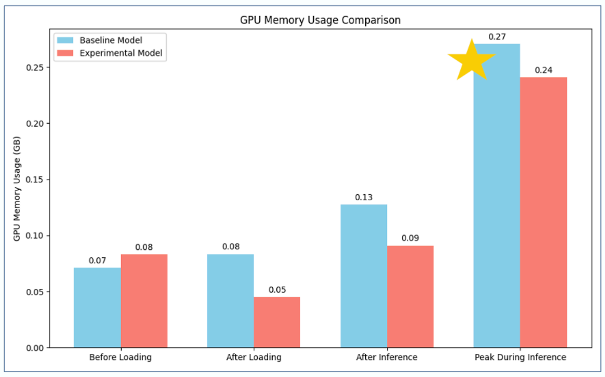


Fig. 13. GPU consumption of validation dataset.

Fig. 13 illustrates GPU consumption during the processing of the validation dataset. The original model, YOLOv8n without a gating mechanism, consumes 0.27 GB of memory, while the proposed model, YOLOv8nGM with a gating mechanism, consumes only 0.24 GB. The YOLOv8nGM model has more economical power usage and less memory consumption.

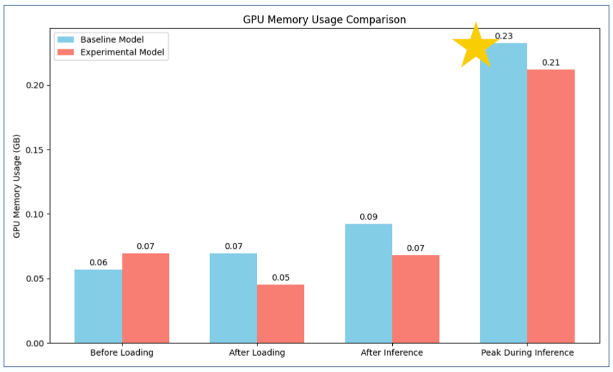


Fig. 14. GPU consumption of test dataset.

Based on Fig. 14, GPU consumption while processing the test dataset, the YOLOv8n model consumes 0.23 GB of memory, while the YOLOv8nGM model uses only 0.21 GB. This shows that the proposed model has more economical power usage and less memory consumption.

C. Mann-Withney Test

The results of the normality test on the object identification using the validation dataset showed that the sample data was not normally distributed (Table VIII), as indicated by the Sig value. Kolmogorov-Smirnov 0.00, or less than 0.005 [46], therefore, the statistical test proceeds with the Mann-Whitney significance test.

TABLE VIII. NORMALITY TEST OF VALIDATION DATASET

Statistic Test	Measurement	Base	Experiment
Kolmogorov-Smirnov[a]	Statistic	0.295	0.296
	df	434	434
	Sig.	0.000	0.000
Shapiro-Wilk	Statistic	0.602	0.610
	df	434	434
	Sig.	0.000	0.000

Based on the Mann-Whitney significance test on the validation dataset, it was found that the speed of the proposed model was significantly different from that of the original model, as indicated by the Asymp. Sig (2-Tailed) value of 0.000 (Tables IX and X), less than 0.005 [47], so that $H0$ is rejected, and Ha is accepted, or there is a significant difference between the speed of the original model process, and the proposed model using the validation dataset.

TABLE IX. MANN-WITHNEY HYPOTHESIS OF VALIDATION DATASET

DL Model	N	Mean Rank	Sum of Ranks
Base	434	538.91	233887.50
Experiment	434	330.09	143258.50
Total	868	-	-

TABLE X. STATISTIC RESULT OF VALIDATION DATASET

Measurement	Score
Mann-Withney U	48863.500
Wilcoxon W	143258.500
Z	−12.270
Asymp. Sig. (2-tailed)	0.000

The same statistical test was performed on the object identification using the test dataset for both deep learning models. From the results of the normality test, the sample data was not normally distributed, as indicated by the Sig. The Kolmogorov-Smirnov value indicator is 0.000 (Table XI), which is less than 0.005. From the results of the Mann-Whitney significance test, it was obtained that the test results of the proposed model were significant compared to the original model, indicated by the Asymp. Sig (2-Tailed) value of 0.002 (Tables XII and XIII), less than 0.005, so that $H0$ is rejected, and Ha is accepted, or there is a significant difference between the processing speed of the original model and the proposed model using the test dataset.

TABLE XI. NORMALITY TEST OF TEST DATASET

Statistic Test	Measurement	Base	Experiment
Kolmogorov-Smirnov[a]	Statistic	0.307	0.292
	df	434	434
	Sig.	0.000	0.000
Shapiro-Wilk	Statistic	0.608	0.631
	df	434	434
	Sig.	0.000	0.000

TABLE XII. MANN-WITHNEY HYPOTHESIS OF TEST DATASET

DL Model	N	Mean Rank	Sum of Ranks
Base	434	460.62	199910.00
Experiment	434	408.38	1777236.00
Total	868		

TABLE XIII. STATISTIC RESULT OF TEST DATASET

Measurement	Score
Mann-Withney U	82841.000
Wilcoxon W	177236.000
Z	−3.070
Asymp. Sig. (2-tailed)	0.002

Based on the experiment results and statistical tests, it can be concluded that modifying the YOLOv8n deep learning model with a gating mechanism (YOLOv8nGM) has a significant impact on the process speed, without compromising the model’s accuracy.

V. CONCLUSION

This study successfully demonstrates the proposed model YOLOv8nGM, which adds a gating mechanism to the original YOLOv8n backbone network enhance computational efficiency in deep learning models without compromising accuracy. The proposed model reveal statistically significant enhancements in processing speed, with validation dataset results showing p = 0.000 and test dataset confirming robust performance with p = 0.002 using the Mann-Whitney U test, validating the effectiveness of adaptive computation strategies.

Beyond speed optimization, the model exhibits superior accuracy metrics, achieving mAP@0.5 of 0.961 compared to the baseline YOLOv8n's 0.953, representing a meaningful +0.008 improvement. The research further confirms substantial resource efficiency gains, including reduced GPU memory consumption to 0.21 GB and more efficient power utilization, critical factors for edge deployment in real-world industrial applications.

This work represents a significant contribution to efficient computer vision, offering a practical solution for industrial automation, smart warehousing, and real-time surveillance systems where speed, accuracy, and resource efficiency are simultaneously critical. The YOLOv8nGM model successfully bridges the gap between high-performance detection and practical deployment, setting a new standard for adaptive deep learning architectures.

Future research directions should explore extending gating mechanisms to another YOLO components, by modifying the gating mechanism, either on the backbone, neck, or head of the YOLOv8n architecture, to produce a more sensitive detector with clearer, more accurate, and more precise results. The robust statistical methodology employed, including appropriate non-parametric testing given non-normal data distributions, strengthens confidence in these findings and provides a template for future optimization studies.

Conflict of Interest

The authors declare no conflict of interest

Author Contributions

CHH conceptualization, methodology, and writing original draft; HP, data analysis, and writing analysis data result; HLHSW collecting data, analyzing data, and writing discussion; YH data extraction, manuscript revision, and editing. All authors had approved the final version.

Acknowledgment

The authors wish to thank the reviewers for their valuable and constructive comments and suggestions which helped to improve the manuscript.

References

[1] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539

[2] A. Timilsina. (March 2024). YOLOv8 architecture explained! [Online]. Available: https://abintimilsina.medium.com/yolov8-architecture-explained-a5e90a560ce5

[3] C. Ruiner, C. Debbing, V. Hagemann, M. Schaper, M. Klumpp, and M. Hesenius, "Job demands and resources when using technologies at work—Development of a digital work typology," *Employee Relations: The International Journal*, vol. 45, no. 1, pp. 190–208, 2023. doi: 10.1108/ER-11-2021-0468

[4] F. F. Adedoyin and B. Christiansen, *Generative AI and Multifactor Productivity in Business*, United Kingdom: IGI Global, 2024.

[5] Z. Gui and J. Geng, "YOLO-ADS: An improved YOLOv8 algorithm for metal surface defect detection," *Electronics (Switzerland)*, vol. 13, no. 16, 3129, Aug. 2024. doi: 10.3390/electronics13163129

[6] M. Yaseen, "What is YOLOv8: An in-depth exploration of the internal features of the next-generation object detector," arXiv preprint, arXiv:2408.15857, Aug. 2024.

[7] T. Zhao *et al.*, "Resting-state brain networks alterations in adolescents with internet gaming disorder associate with cognitive control impairments," *Front. Psychiatry*, vol. 15, Sep. 2024. doi: 10.3389/fpsyt.2024.1404050

[8] Z. Zhang, "Drone-YOLO: An efficient neural network method for target detection in drone images," *Drones*, vol. 7, no. 8, 526, Aug. 2023. doi: 10.3390/drones7080526

[9] S. L. Pan and R. Nishant, "Artificial intelligence for digital sustainability: An insight into domain-specific research and future directions," *Int. J. Inf. Manage*, vol. 72, 102668, 2023. doi: 10.1016/j.ijinfomgt.2023.102668

[10] D. Nimma *et al.*, "Object detection in real-time video surveillance using attention based transformer-YOLOv8 model," *Alexandria Engineering Journal*, vol. 118, pp. 482–495, Apr. 2025. doi: 10.1016/j.aej.2025.01.032

[11] H. Wang, Q. Shen, Q. Dai *et al.*, "Evolutionary variational YOLOv8 network for fault detection in wind turbines," *Computers, Materials and Continua*, vol. 80, no. 1, pp. 625–642, 2024. doi: 10.32604/cmc.2024.051757

[12] T. Jiang *et al.*, "Improved YOLOv8 model for lightweight pigeon egg detection," *Animals*, vol. 14, no. 8, Apr. 2024. doi: 10.3390/ani14081226

[13] Y. Shi, S. Qing, L. Zhao *et al.*, "YOLO-Peach: A high-performance lightweight YOLOv8s-based model for accurate recognition and enumeration of peach seedling fruits," *Agronomy*, vol. 14, no. 8, Aug. 2024. doi: 10.3390/agronomy14081628

[14] A. Gu, C. Gulcehre, T. Paine, M. Hoffman, and R. Pascanu, "Improving the gating mechanism of recurrent neural networks," in *Proc. 37th International Conference on Machine Learning*, 2020, pp. 3800–3809.

[15] Z. Liang *et al.*, "Low-rank Adaptation with Gating Mechanisms in large language models, an improved method for fine-tuning: G-LoRA," in *Proc. Ninth International Symposium on Advances in Electrical, Electronics, and Computer Engineering (ISAEECE 2024)*, vol. 13291, 2024, pp. 1086–1091. doi: 10.1117/12.3033402

[16] Z. Cheng *et al.*, "Refined gate: A simple and effective gating mechanism for recurrent units," arXiv preprint, arXiv:2002.11338, May 2020.

[17] Y. Feng, L. Bo, L. Pan, and W. Di, "Research on aerial photography target detection methods," in *Proc. 2024 36th Chinese Control and Decision Conference (CCDC)*, 2024, pp. 4238–4243. doi: 10.1109/CCDC62350.2024.10587654

[18] U. Nishitha, V. Lokesh, T. Kaushik, and R. P. Singh, "Vehicle detection in unmanned aerial imagery through advance you only look once architectures," in *Proc. 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10724269

[19] N. Choudhary, R. Sharma, D. Upadhyay, A. Verma, and V. Jain, "Enhanced traffic sign recognition using advanced YOLOv8 model," in P*roc. 2024 4th International Conference on Intelligent Technologies (CONIT)*, 2024, pp. 1–4. doi: 10.1109/CONIT61985.2024.10626450

[20] V. Sharma, N. Singh, and R. Prasad, "Assessing YOLOv8 as a classifier for detection of synthetically generated facial imagery," in *Proc. 2024 11th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2024, pp. 99–104. doi: 10.1109/SPIN60856.2024.10511616

[21] Z. Yang, X. Wang, and J. Li, "EIoU: An improved vehicle detection algorithm based on vehiclenet neural network," in *Proc. Journal of Physics: Conference Series*, May 2021. doi: 10.1088/1742-6596/1924/1/012001

[22] L. You, W. Guo, L. Dai, and J. Du, "Deep neural network embeddings with gating mechanisms for text-independent speaker verification," in *Proc. of the Annual Conference of the International Speech Communication Association, INTERSPEECH, International Speech Communication Association*, 2019, pp. 1168–1172. doi: 10.21437/Interspeech.2019-1746

[23] H. Jin, K. Yun, and E. Kim, "Gating mechanism in deep neural networks for resource-efficient continual learning," *IEEE Access*, vol. 10, pp. 18776–18786, 2022. doi: 10.1109/ACCESS.2022.3147237

[24] X. Zhao, "Deep learning based visual perception and decision-making technology for autonomous vehicles," *Applied and Computational Engineering*, vol. 33, no. 1, pp. 191–200, Feb. 2024. doi: 10.54254/2755-2721/33/20230265

[25] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th International Conference on Machine Learning, Sydney*, 2017.

[26] A. Makkuva, S. Oh, S. Kannan *et al.*, "Learning in gated neural networks," in *Proc. 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

[27] T. Chen and Q. Zeng, "Research on bubble detection based on improved YOLOv8n," *IEEE Access*, vol. 12, pp. 9659–9668, 2024. doi: 10.1109/ACCESS.2024.3353196

[28] A. Kapoor, R. Shah, and R. Bhuva, "Understanding inception network architecture for image classification," Technical Report, University of Waterloo, 2020. doi: 10.13140/RG.2.2.16212.35204

[29] C. Szegedy, W. Liu, Y. Jia *et al.*, "Going deeper with convolutions," in *Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[30] T. Ahmad. 1×1 convolution: Explainer. *Medium*. [Online]. Available: https://medium.com/@tauseefahmad12/1x1-convolution-explainer-ef482ec49508

[31] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, Dec. 2019. doi: 10.1186/s40537-019-0197-0

[32] J. Sadaiyandi, P. Arumugam, A. K. Sangaiah, and C. Zhang, "Stratified sampling-based deep learning approach to increase prediction accuracy of unbalanced dataset," *Electronics (Switzerland)*, vol. 12, no. 21, Nov. 2023. doi: 10.3390/electronics12214423

[33] Microsoft. Common Object in Context (COCO) dataset. *Microsoft*. [Online]. Available: https://cocodataset.org/#download

[34] Ultralytics. (Jan. 2023). YOLOv8. *Ultralytics*. [Online]. Available: https://docs.ultralytics.com/models/yolov8/

[35] C. Heruatmadja, H. Prabowo, H. L. H. S. Warnars, and Y. Heryadi, "Suitable deep learning based for high accuracy object detection in inventory management: Systematic literature review," in *Proc. International Conference on Informatics and Computational Sciences*, 2024.

[36] Ultralytics. (Jan. 2023). Explore Ultralytics YOLOv8. *Ultralytics*. [Online]. Available: https://docs.ultralytics.com/models/yolov8/

[37] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, Jul. 2023. doi: 10.3390/machines11070677

[38] H. Song, J. Wang, and Y. Zhang, "Detection of abandoned objects based on Yolov9 and background differencing," *Signal Image Video Process*, vol. 19, no. 1, 54, 2024. doi: 10.1007/s11760-024-03609-z

[39] Z. Jia, S. Sun, and G. Liu, "Real-time traffic sign detection based on weighted attention and model refinement," *Neural Process Lett*, vol. 55, no. 6, pp. 7511–7527, 2023. doi: 10.1007/s11063-023-11271-8

[40] A. Parti. (Apr. 2024). Understanding activation functions in neural networks. [Online]. Available: https://pareto.ai/blog/activation-function-in-neural-networks

[41] A. Morse. (June 2025). What is c2f in YOLOv8. [Online]. Available: https://dev.to/angelomorse/what-is-c2f-in-yolov8-1ojc

[42] N. Nachar, "The Mann-Whitney U: A test for assessing whether two independent samples come from the same distribution," *Tutorials in Quantitative Methods for Psychology*, vol. 4, no. 1, pp. 13–20, 2008.

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 779–788.

[44] I. K. Suartama *et al.*, "Development of e-learning oriented inquiry learning based on character education in multimedia course," *European Journal of Educational Research*, vol. 9, no. 4, pp. 1591–1603, 2020. doi: 10.12973/EU-JER.9.4.1591

[45] G. Dosymbetova *et al.*, "Neural network-based active cooling system with iot monitoring and control for LCPV silicon solar cells," *IEEE Access*, vol. 11, pp. 52585–52602, 2023. doi: 10.1109/ACCESS.2023.3280265

[46] D. S. Dimitrova, V. K. Kaishev, and S. Tan, "Computing the Kolmogorov-Smirnov distribution when the underlying CDF is purely discrete, mixed or continuous," *Journal of Statistical Software*, vol. 95, pp. 1–42, 2020.

[47] R. W. Emerson, "Mann-Whitney U test and t-test," *Journal of Visual Impairment & Blindness*, vol. 117, no. 1, pp. 99–100, 2023. doi: 10.1177/0145482X221150592