International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(4s) 2026 pp. 1306-1324
https://doi.org/10.70917/ijcisim-2026-2638
Article

# Deep Ensemble Learning for Weed Recognition Using YOLO Variants in Precision Agriculture

**Yalla Akash[1]*, Vipul Agarwal[2], Thirumalasetty Anusha[3], Orchu Venkata Koteswararao[4]**

[1 ,2,3,4]Koneru Lakshmaiah Education Foundation (KL University), Vaddeswaram, Andhra Pradesh, India.

[1]yallaaakaash2003@gmail.com, [2] agarvipul@gmail.com, [3] thirumalasettyanusha02@gmail.com, [4]orchu.yona@gmail.com

**Abstract:** Weed competition in paddy fields is a critical challenge to crop yield and agricultural sustainability because conventional control methods, whether Broad-spectrum weeding or manual control herbicides, are Work-intensive, environmentally harmful, lack the precision needed for modern farming. To overcome the limitations mentioned above, then we suggest a new deep ensemble building blocks for learning that integrates 3-YOLO variants (YOLOv8 with a high detection accuracy, YOLOv11 with real-time processing capability, and GE-YOLO with good generalization) into one unified detection system optimized for automated weed recognition tasks in a paddy field. For each model, the ensemble architecture employed NMS and WBF strategies to combine predictions and hence improve the bounding boxes' localization accuracy. In this work, we built and manually annotated a paddy field dataset and further augmented the images using rotation, flipping, scaling, and brightness adjustments to represent various real-world scenarios that include changing lighting conditions, high-density crops, water reflections, and rice-weed morphological similarities. Each variation was independently trained across 100 epochs. using the SGD optimizer by tuning its hyperparameters to achieve maximum performance in weed detection. The results achieved by the proposed deep ensemble framework were outstanding: it obtained an accuracy of 98.75%, precision of 88.89%, Consider of 99.35%, an F1-score of 93.42%, mAP@50 of 88.89%. These results are better than any previous YOLO models and any recent literature on the best weed detection results. This shows that our proposed system can work well in a variety of difficult agricultural situations, such as when the light changes, there are a lot of weeds, and the background is complicated. These results show that the proposed framework is very good for use in real-life precision agriculture applications. The proposed approach can also function as a front-end module for autonomous weeding robots, UAVs, and IoT-based smart farming platforms without modifications. This will make it possible to manage weeds on a site-by-site basis with fewer herbicides, less harm to the environment, and more support for sustainable farming practices that boost crop productivity overall.

**Keywords:** Deep Learning, Ensemble Learning, Weed Detection, YOLOv8, YOLOv11, GE-YOLO, Precision Agriculture, Computer Vision, Smart Farming.

## 1. Introduction:

Weeds are still a big problem for farmers today, and they cost a lot of money [1]. This is especially true when growing rice because weeds pillage water, nutrients, and sunlight from rice plants, which are all very important. Because of this competition, farmers have to pay more for supplies and workers, which means that crops don't grow as well. For a long time, people have used machines, chemicals, and their hands to get rid of weeds. But there are big problems with all of these ways. Weeding by hand takes a lot sof time and work, mechanical methods aren't always very accurate and can hurt crops, and using herbicides **[2]** carelessly can pollute soil and water, hurt the environment, and make weed species that are resistant to herbicides [3].

**Fig. 1: Sample Images Illustrating Weed Infestation in Rice Crops**

In the Fig. 1 shows that weeds can cause a lot of problems in rice fields. It also shows how hard it is to find them **[4–5].** The illustrations show different real-life situations, such as dense weed clusters competing with rice seedlings (top-left), grass-type weeds that look like rice plants and make it hard to tell them apart (top-center), early-stage rice plants surrounded by multiple weed species at different growth stages (top-right), and heavy weed infestation threatening crop establishment (bottom-left). sparse but persistent weed presence in established rice fields (bottom-center), and varying weed densities across different field sections (bottom-right).These pictures show how important it is to have automated detection systems that can tell the difference between weeds and crops at different stages of growth, in different places, and in various illumination circumstances, even when the plants look the same. These pictures show much of different kinds of weeds, from single weeds to extensive infestations and from clear visual contrast to unclear morphology. This demonstrates the challenge it is to make a weed detection system that performs well in reality on a farm [6]. It needs smart, eco-friendly, and very accurate weed management systems right away [7]. This is because food is becoming more important and farming that is good for the environment is becoming more important. Precision agriculture is a new way of doing things that uses new technologies like AI, computer vision, robotics, and the Internet of Things (IoT) to get the most out of resources [8] and increase output. In this case, smart farming systems **[9],** autonomous weeding robots, and UAV-based monitoring platforms [10] all need to be able to automatically find weeds.

The diverse and ever-changing conditions in paddy fields, as shown in the figure above, make it hard to accurately identify weeds. Changes in light, water reflections, thick crop canopies, and the fact that rice plants and weeds look similar in shape [11] all make traditional detection methods [12] much less accurate and reliable. Weeds and rice plants can look very similar when they are just starting to grow, so it can be hard for even experienced agronomists to tell them separate from one another. This is also a big problem for computer vision systems capable of functioning on their own systems [13]. Weeds can be found in many places, from small patches to large infestations. This means that detection systems need to work well at different scales and densities. The mentioned problems require strong, flexible, and very precise models that will be able to work in the real world.

This work presents a comprehensive ensemble learning framework that amalgamates the benefits of diverse YOLO-based detection models for the automated identification of weeds in paddy fields [14]. The proposed system integrates YOLOv8 for enhanced detection precision [15], YOLOv11 for expedited real-time processing [16], and GE-YOLO for improved generalization across diverse field conditions [17]. The framework uses Non-Maximum

Suppression (NMS) and Weighted Box Fusion (WBF) to make the best guesses and find the best balance between speed, accuracy, and durability **[18].** The model learns from the dataset of paddy fields that have been tagged by hand and have images like the above. The model is thus ensured to see all forms of real-world variability, including different types of weeds, growth stages, infestation levels, and environmental conditions [19]. The system has been tested under different kinds of weather conditions, and it works well in harsh conditions such as those revealed by the pictures from the field, as shown above. This marks a big step toward smart, useful, and sustainable methods of weed removal in precision farming. It can deal with the morphological ambiguity, spatial variability **[20],** and environmental diversity arising from realistic rice growing.

For the first step in finding weeds, we used regular image processing and handmade feature extraction methods. These analyses included spectral-based analysis **[21],** which examined the differential light reflection of crops and weeds, and shape-texture analysis, employing statistical descriptors and morphological operations. But these old methods were ineffective in all weather, were sensitive to light changes [22], and couldn't be used on all weeds or at all growth stages. Deep learning changed agricultural computer vision by letting Convolutional Neural Networks (CNNs) learn features on their own from raw data [23]. In the early days of deep learning, AlexNet, VGG, and ResNet architectures were used to classify image patches. However, localization required sliding window techniques that were very costly to run. End-to-end architectures have solved this problem in modern object detection frameworks Even with these improvements, individual detector architectures still have trouble getting high accuracy, real-time speed, and strong generalization across different agricultural settings all at the same time [24].It demonstrates that making models more complex and faster to run may have both good and bad effects. Ensemble learning is a good idea because it makes predictions from different models more precise and reliable. Traditional ensemble methods relied on fundamental fusion techniques, such as averaging confidence scores or employing standard Non-Maximum Suppression (NMS) across the outputs of multiple models. When Faster R-CNN and SSD were used together, studies showed that these methods improved general object detection by 4–6% mAP.Weighted Box Fusion (WBF) is a more advanced method for merging bounding box coordinates that uses weighted averaging based on confidence. It works better than other methods because it avoids throwing away the predictions that the models made. Recent studies in agriculture have examined the combined use of ResNet-based classifiers and YOLO detectors in groups. This has made it possible to find crop diseases and count fruit with 2–4% more accuracy [25]. Still, the current ensemble methods for weed detection are limited because they usually only combine two models using simple fusion techniques. They also don't use a systematic way to combine different YOLO variants that are designed for different purposes. Furthermore, prior studies have not concentrated on paddy field weed detection employing ensemble architectures that integrate the complementary advantages of accuracy-focused, speed-enhanced, and generalization-boosted models through advanced fusion methodologies, including integrated NMS and WBF strategies [26]. This research gap leads to the suggested deep ensemble framework that merges YOLOv8, YOLOv11, and GE-YOLO with advanced prediction refinement to achieve optimal outcomes in accuracy, efficiency, and robustness [27] for practical application in precision agriculture.

## 2. Design and Methodology of the Proposed YOLO Ensemble Framework

The overall architecture of the proposed deep ensemble weed detection system is illustrated in Fig. 2. The framework begins with image acquisition and dataset preparation, followed by data preprocessing and augmentation. The processed data are then fed into multiple object detection models, including YOLOv8, YOLOv11, and GE-YOLO, whose outputs are optimized and combined through a post-processing and prediction pipeline to generate the final weed detection results.

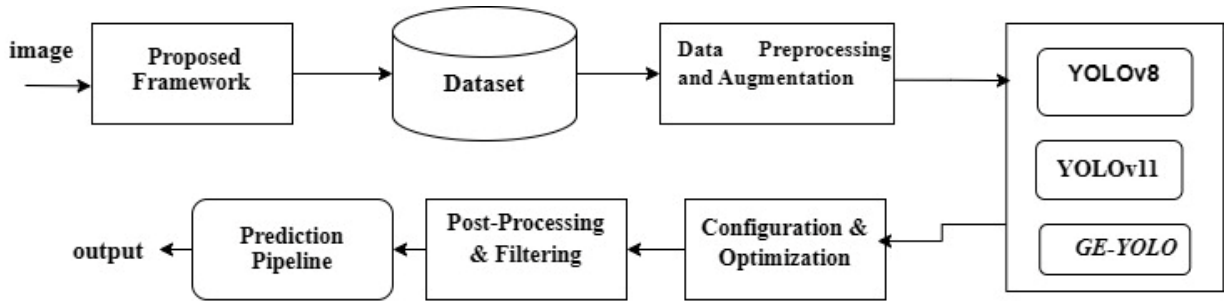

**Fig.2: Block Diagram of the Proposed YOLO Optimization and Prediction Process**

The proposed deep ensemble learning framework incorporates three cutting-edge YOLO variants: YOLOv8, YOLOv11, and GE-YOLO, in order to realize robust and accurate weed detection in paddy fields. The technique uses a sequential pipeline that includes data collection, pre-processing, training each model, integrating the algorithms, and testing their performance. The whole thing is designed to resolve one or more problems that come up in farming settings, such as changing light levels, complicated backgrounds, and the fact that crops and weeds appears similar.

For detecting weeds in paddy fields, a dataset was developed with high-resolution images taken from different stages of growth, light conditions, and weed densities. Various species of weeds that are normally found in rice fields have been included in this dataset to accurately mimic the actual conditions of a field. Digital cameras on the ground and on UAVs took pictures from different heights and angles. There are a total of N labeled images in the dataset. 70% of them (0.7N images) are for training, 20% (0.2N images) are for testing, and 10% (0.1N images) are for validation. This division is good for learning because it keeps the validation and test sets separate. This way, the model won't fit too closely and will have a fair view of how well it works

All images were manually labeled using the LabelImg annotation tool, which made bounding box coordinates in the YOLO format. There was a rectangular bounding box around each weed in the picture. The box was based on the weed's height (h), width (w), and coordinates for the center (x, y), all of which were relative to the size of the illustration.

It placed labels to all of these pictures by hand using the Label Img annotation tool. This tool makes bounding box coordinates in the YOLO format. In the picture, there was a rectangular box around each weed instance. The image's size was used to set the box's center coordinates (x, y), width (w), and height (h).

**YOLO Annotation :**

$$\text{class\_id} \quad \text{x\_center} \quad , \quad \text{y\_center} \quad \text{width} \quad \text{height} \qquad \ldots(1)$$

The coordinates are all normalized values between 0 and 1, which are found by:

Normalized Center Coordinates:

$$\mathbf{x_{center}} = \frac{\mathbf{x_{box-center}}}{\mathbf{image_{width}}}, \qquad \mathbf{y_{center}} = \frac{\mathbf{y_{box-center}}}{\mathbf{image_{width}}} \qquad \ldots (2)$$

This normalization makes sure that the scale is the same and that different image resolutions are compatible with adjacent ones.

They accomplished a lot of preprocessing and data augmentation to make the models stronger and more flexible. We resized all of the input images to 640×640 pixels so that all of the models would be the same and we could use the most computing power. Training went faster because the pixel values fit into the range [0, 1] when they were divided by 255. During the training, the data was changed in random ways to make it better. dataset more like what would happen in the real world. Adding Strategy we use the following image augmentation techniques to make the model stronger and more able to work in different field conditions. You can flip the image horizontally with a 50% chance, randomly rotate it between -15° and +15°, scale it up or down by 0.8 to 1.2 times its original size, or move it up to 10% of its dimensions. Photometric transformations involve altering brightness ($I'=I\times(1+\alpha)I'$ with $\alpha\in[-0.2,+0.2]$), modifying contrast ($I'=(I-\mu)\times\beta+\mu$ with $\beta\in[0.8,1.2]$), and adjusting the HSV color space. The dataset is even more different now that it has noise and blur. Adding Gaussian noise is done with $I' = I + N(0, \sigma^2)$, where $\sigma$ is between 0 and 0.02. Adding Gaussian blur is done with a kernel size between 3 and 7. The dataset grows more diverse when you add these things together. This keeps the model strong even when the lighting, camera angle, and setting change.

The Architecture YOLOv8 is the strongest part of the group, and its better design makes it much better at finding things. It uses a CSP Darknet backbone for feature extraction. This backbone has cross-stage partial connections that help the flow of gradients and cut down on unnecessary processing.

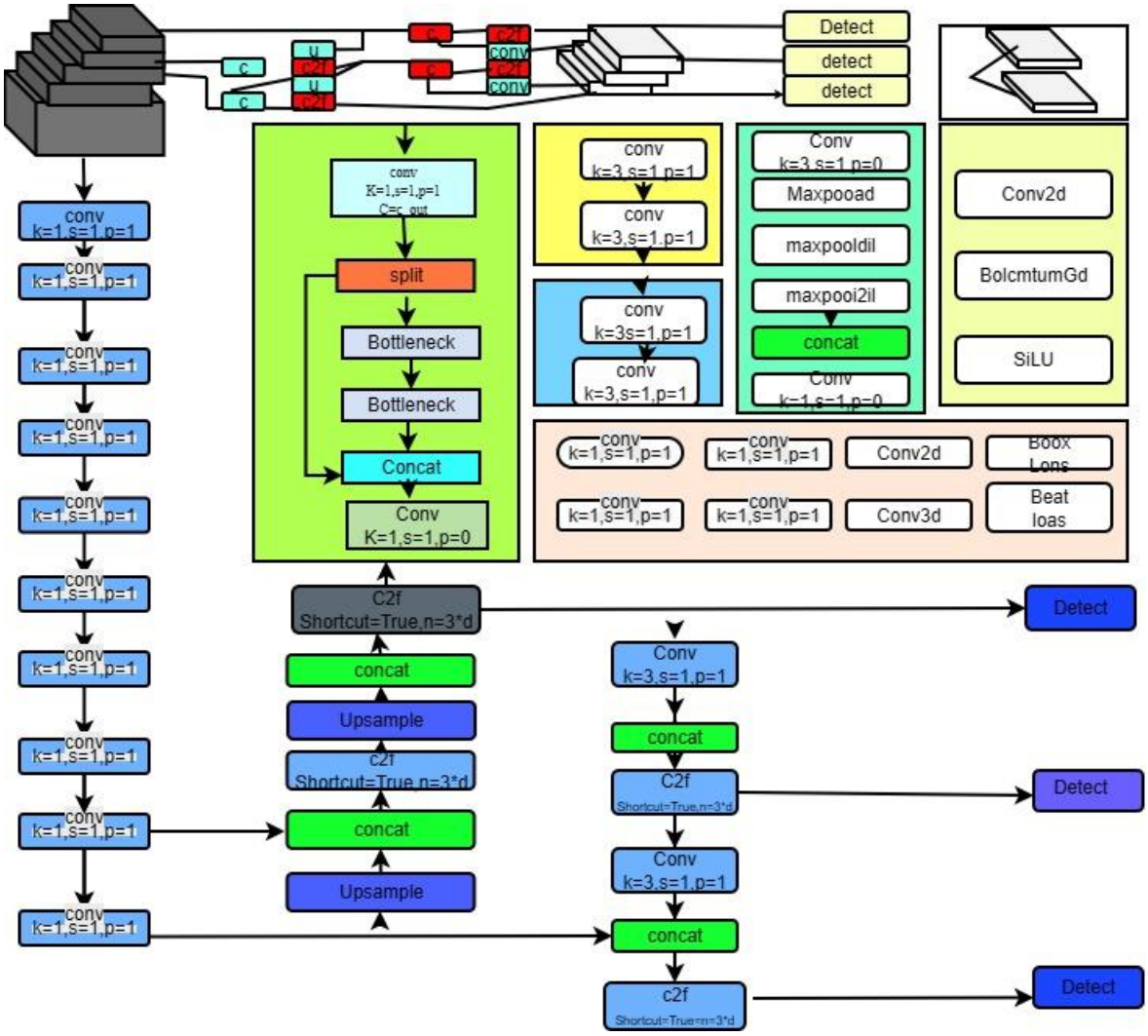


**Fig.3: Architecture of the YOLOv8 Object Detection Model [28]**

In the Fig.3 presents the architecture of the deep learning model YOLOv8 in detail, showing its backbone, neck, and detection composition of the head. The backbone uses convolutional , C2f layers to extract key visual features, improving multi-scale feature representation through the SPPF module. The neck combines and refines different scale features with upsampling and concatenation, improving spatial information. The detection head predicts object classes and their bounding boxes directly using convolutional layers. This work proposes an efficient, modular pipeline that represents one of the fastest solutions suitable for real-time object detection. Features of multiple scales are extracted through successive convolutional layers in the backbone, which generates feature maps at three different scales: P3 at 80×80, P4 at 40×40, and P5 at 20×20, representing stride values of 8, 16, and 32, respectively. These multi-scale features enable the detection of objects ranging from small weeds to larger vegetation clusters.

The neck component is a The PANet structure allows for bidirectional feature fusion from different scales. This process of feature fusion can be defined as:

$$F_{fused}^{(1)} = Concat(F_{up}^{(l+1)}, F_{lateral}^{(1)}) \quad \ldots (3)$$

$$F_{down}^{(1)} = Conv(F_{fused}^{(1)}) \quad \ldots (4)$$

F is the collection of feature maps at various levels (l). The fusion merges pathways that go from the top down and the bottom up.

The detection head on YOLOv8 doesn't need anchor boxes, so it doesn't use them. The model makes direct predictions for each grid cell (i, j) in the output characteristic map as follows:

$$\text{b^=(x^y^w^h)}\hat{b} = (\hat{x}, \hat{y}, \hat{w}, \widehat{h}) \qquad \dots (5)$$

$$\hat{c} = \sigma(c)$$

$$\hat{p}k = \sigma(pk),\ k \in \{1, \dots\dots, k\} \qquad \dots (6)$$

($\hat{x}$, $\hat{y}$) are the predicted center coordinates, and ($\hat{w}$, $\hat{h}$) are the width and height., $\hat{c}$ is the objectness confidence score, and $\hat{p}_k$ are the class probabilities for K classes. The sigmoid activation function is denoted by σ.

**Loss Function for YOLOv8:**

The training objective combines three loss components: λ

$$L_{total} = \lambda_{box} L_{CIoU} + \lambda_{cls} L_{BCE}^{CLS} + \lambda_{obj} L_{BCE}^{obj} \qquad \dots (7)$$

The Complete Intersection over Union (CIoU) loss for bounding box regression is defined as:

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + av \qquad \dots (8)$$

The Intersection over Union (IoU) is the ratio of the area where the predicted bounding box BBB and the ground truth box Bgt overlap to the area where they are together. It is written as $IOU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|}$. The Euclidean distance between the centers of the predicted box and the ground truth box is ρ(b,bgt). This tells us how far off the predicted box is from the actual object. The variable c is the length of the smallest box's diagonal that can hold both the predicted and the real boxes. This is how you make sure that all the boxes are in the right place. Finally, α is a trade-off parameter that makes sure that the predicted and actual bounding boxes are as close as possible to each other by balancing how much each part of the overall loss function impacts to the final outcome

$$.v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - a\arctan\frac{\hat{w}}{\hat{h}}\right)^2 \qquad \dots (9)$$

The Binary Cross-Entropy (BCE) is used for the classification and objectness losses:

$$L_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i + (1 - y_i \log(1 - \hat{y}_i)] \quad \dots (10)$$

N is the number of predictions, $y_i$ is the real label, and $\hat{y}_i$ is the predicted chance.

The nano version of YOLOv11n is included to make sure that real-time detection is possible with only a little extra processing power. This means that it can work with edge devices like drones for farming and robots in the field. The architecture keeps the main design ideas of the YOLO family, but it makes the model's parameters and the amount of work it needs to do much smaller.

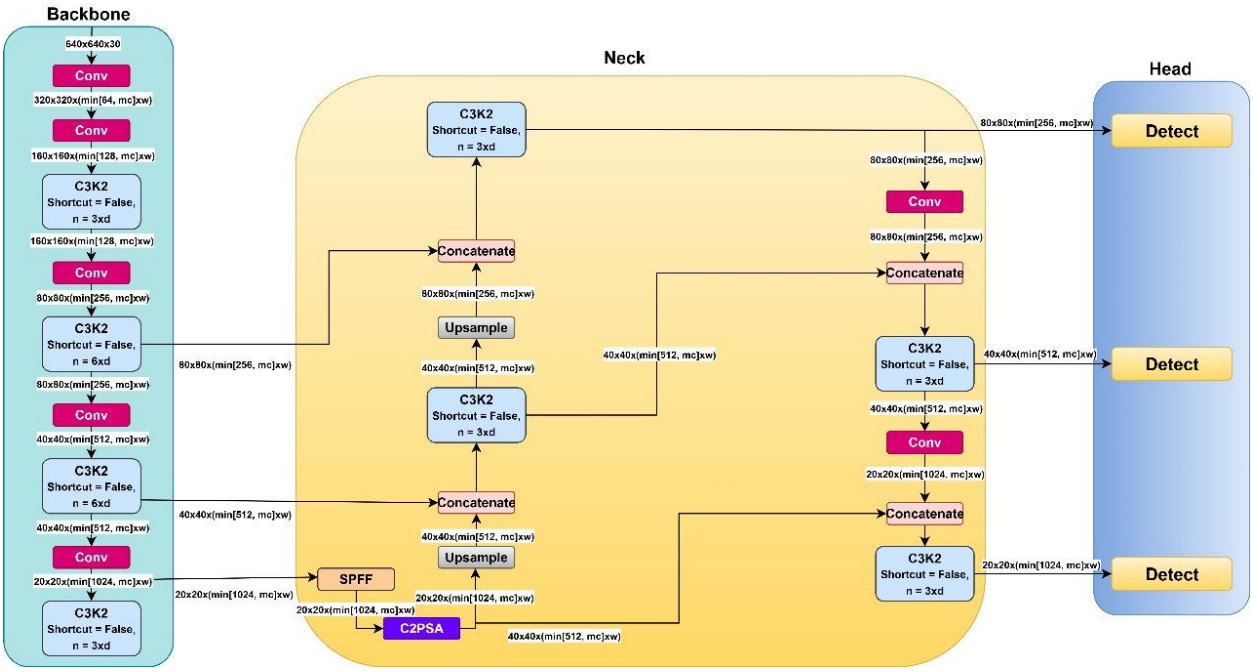


**Fig.4: Structural Overview of the YOLOv11 Architecture**

The **Fig.4** explains the YOLOv11n model is light because it has been redesigned to be more efficient while still being able to find things correctly. The model has fewer channels in each layer so that it uses less memory and fewer parameters. In addition, depthwise separable convolutions take the place of standard convolutions. It makes technology cheaper without taking away their ability to show things. Adding optimized C3K2 blocks makes feature extraction easier and faster by finding a good middle ground between the two. Also, a simpler neck design with fewer aggregation layers speeds up inference and cuts down on duplication. YOLOv11n is small, but it still has the parts that are needed for good object detection. This assures that the design is light and the performance is great. Fast Spatial Pyramid Pooling (SPPF): This module is good at getting information about different scales by doing a series of max pooling operations:

$$F_{spp} = \text{Concat}[F, \text{MaxPool}_k(F), \text{MaxPool}_k^2(F), \text{MaxPool}_k^3(F)] \quad \ldots (11)$$

This provides better spatial information. The detection head uses convolution layers to directly guess what kind of object it is and where it is. This work shows a fast, modular pipeline that works well for real-time object detection.

$$F_{attended} = F \odot \sigma(\text{Conv}_{1*1}(F)) \qquad \ldots (12)$$

Where $\odot$ means multiplying each part by itself, which makes important parts stand out and hides noise that isn't important. By picking out the most iTo maximize computer power and ensure uniformityAll of the input images were made smaller to 640 by 640 pixels. ut it doesn't have as many layers of computation, which means it can make decisions faster. It doesn't use anchors to make predictions, which makes it easier to find things and works better with things of different sizes.

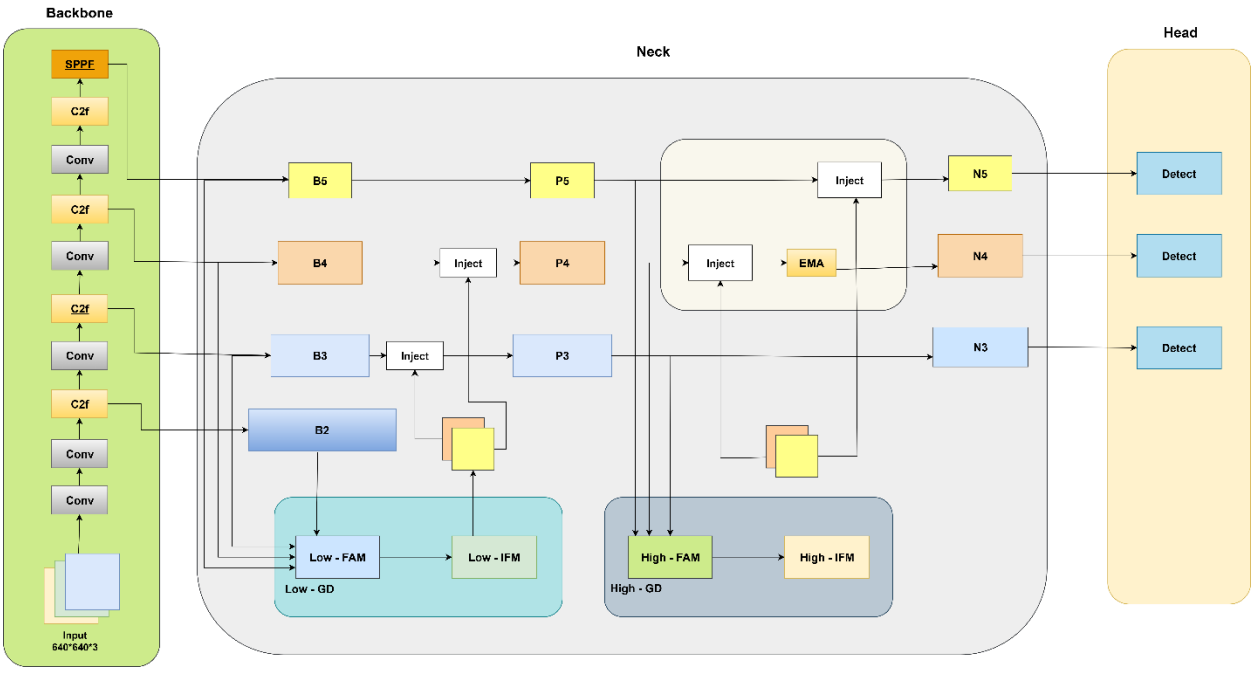


**Fig.5: GE-YOLO Architecture: Backbone, Feature Aggregation, and Multi-Level Detection Pipeline.**

In the **Fig.5** shows how the GE-YOLO (Generalized Ensemble YOLO) solves the problem of generalization in different agricultural settings. The architecture has special modules that make it more robust against changes in the environment, occlusion, and morphological ambiguity.

**Feature Aggregation Module (FAM):** This part does cross-scale feature fusion at both high and low semantic levels.

$$F_{low}^{agg} = FAM_{low}(F_{P3}, F_{P4}, F_{P5}) \quad \dots (13)$$

$$F_{high}^{agg} = FAM_{high}(F_{P3}, F_{P4}, F_{P5}) \quad \dots (14)$$

The aggregation process uses adaptive fusion weights that were learned through attention mechanisms:

$$w_i = \frac{\exp(MLP(F_i))}{\sum \exp(MLP(F_j))} \quad \dots (15)$$

$$F_{agg} = \sum_i w_i . F_i \quad \dots (16)$$

**Interaction Fusion Module (IFM):** This module allows information to flow simultaneously between low-level spatial details and high-level semantic features:

$$F_{\text{interaction}} = IFM(F_{low}^{agg}, F_{high}^{agg}) \quad \dots (17)$$

Where ↑ and ↓ denote upsampling and downsampling operations respectively.

**Efficient Multi-scale Attention (EMA):** This attention mechanism assists the model pay more attention to important areas and less to background information that isn't important:

$$A = \sigma(Conv1 \times 1(GAP(F)) + Conv1 \times 1(GMP(F))) \dots (18)$$

GAP stands for Global Average Pooling, and GMP stands for Global Max Pooling. σ is the sigmoid function, and ⊙ means element-wise multiplication.

We propose three different YOLO variants, which use the same hyper parameters so that all three proposed variants remain compatible and their performances are equally effective. The training was completed on Google Colab

using an NVIDIA T4 GPU and hence provided enough compute to accelerate improvements smoothly. Then we selected the hyper parameters for the model very carefully so that it would converge steadily and reach its best performance. It had 100 epochs of training, with a batch size of 16 and an initial learning rate of $\eta_0 = 0.001$. We also used the Stochastic Gradient Descent (SGD) optimizer, which had a momentum term of $\beta=0.937$ to help the model converge quickly and a weight decay factor of $\lambda=0.0005$ to keep it from fitting too well. Input images were uniformly processed and resized into 640×640 pixels. To slowly decrease the learning rate during the period of training, we further implemented a cosine annealing learning rate schedule, which was defined as

$$\eta t = \eta min + \frac{1}{2}(\eta 0 - \eta \min)(1 + \cos\left(\frac{t}{T_{max}}\pi\right)) \quad \ldots. (19)$$

Where $\eta\min=0.0001$ is the smallest learning rate and the total number of epochs. The optimization process made changes to the model parameters θ by using updates based on momentum to

$$v_t = \beta v_{t-1} + (1 + \beta)\nabla_\theta L \quad \ldots. (20)$$

$$\theta_t = \theta_{t-1} - \eta_t v_t - \lambda\theta_{t-1} \quad \ldots. (21)$$

Where, vt is the velocity term, $\nabla\theta L$ is the gradient of the loss function, and λ controls how strong the regularization is. The combination of learning rate scheduling, momentum, and weight decay made optimization easier and more adept at predicting.

The ensemble fusion method is the most important new feature of this framework. It uses predictions from YOLOv8, YOLOv11, and GE-YOLO to make strong, unified detections. The merging procedure is a two-step process: first, the intra-model NMS and then the inter-model WBF.

Within each model, NMS prunes away predictions that overlap or duplicate one another. The NMS step sorts all the predicted bounding boxes B = {b1, b2, ., bn} by their confidence score C = {c1, c2, ., cn}. The top-scoring box then becomes the most trusted detection. After selecting this best box bib, any other boxes bj that are very similar to bib are discarded. The Intersection over Union (IoU) must be greater than a certain threshold θnms for this to occur. This process keeps going until there are no more bounding boxes. We calculate the IoU metric by dividing the area of the intersection is the same as the area of the union of two bounding boxes. This makes sure that only the most certain and non-overlapping detections are kept for final predictions.

$$\mathrm{IoU}(bi, bj) = \frac{\mathrm{Area}(bi \cap bj)}{\mathrm{Area}(bi \cup bj)} \quad \ldots (22)$$

The NMS threshold $\theta_{nms} = 0.45$ was empirically selected to balance detection completeness and redundancy elimination.

After processing each NMS separately, Weighted Box Fusion combines the improved predictions from all three models. WBF, on the other hand, keeps important information from several models by using weighted averaging to combine boxes that overlap. When using traditional NMS, overlapping boxes are thrown away.

The combined bounding box for a group of overlapping boxes $B\_cluster = \{b_1, b_2, ..., b_m\}$ from different models, along with their confidence scores To find $C\_cluster = \{c_1, c_2, ..., c_m\}$, do the following

$$b_{fused} = \frac{\sum_{i-1}^{m} c_i.bi}{\sum_{i-1}^{m} c_i} \quad \ldots. (23)$$

$$b_{fused} = \frac{1}{m}\sum_{i=1}^{m} ci \quad \ldots. (24)$$

In the fusion process, the coordinates (x,y,w,h) of the bounding box are obtained as a weighted average of the coordinates from the models that helped make it. This makes the localization more stable and accurate. If the Intersection over Union (IoU) of two bounding boxes, bib and bjb, is higher than the fusion threshold, θwbf=0.55, they are put in the same cluster during box clustering.

$$\mathbf{Cluster}(\boldsymbol{b_i}, \boldsymbol{b_j}) = \begin{cases} \mathbf{True} & \mathbf{if\ IoU(bi, bj)} > \boldsymbol{\theta_{wbf}} \\ \mathbf{False} & \mathbf{otherwise} \end{cases} \quad \dots (25)$$

This clustering ensures that boxes referring to the same object are merged effectively. The confidence adjustment of the fused detection is then computed as

$$c_{\text{final}} = c_{\text{fused}} * \min(1, \frac{m}{N_{\text{models}}}) \quad \dots (26)$$

The ensemble has N_models=3 models, while the collective has m models. This formulation boosts the confidence of detections that are backed up by more than one model while keeping the right level of confidence calibration. This makes predictions more accurate and based on what most people believe.

After the ensemble fusion stage, a few steps are taken to make the final detections better and ensure the results are the same. Developing a level of confidence is the first step. This means that only detections that get a final confidence score of C_final>θ_conf are kept. This step eliminates predictions. that are probably wrong and not very precise. subsequently we use Non-Maximum Suppression (NMS) with a threshold of θ_final=0.5 for each class to get rid of duplicate detections of the same object instance. Finally, there is a limit on the bounding boxes that can be in valid coordinate ranges: they can only go outside of the image's boundaries. This makes sure that the last detections are still there. When these steps are done together, the results are clearer, more accurate, and work well from a distance.

$$x_{min} = \max(0, x_{\min}), \quad x_{max} = \min(W, x_{max}) \quad \dots (27)$$

$$y_{min} = \max(0, y_{\min}), \quad y_{max} = \min(H, y_{max}) \quad \dots (28)$$

where, the image's width is W, and its height is H.

It used standard object detection metrics to fully test how well the ensemble framework worked. These metrics demonstrate it how accurate the classification is and how accurate the location will be.

**Accuracy:** The overall proportion of correct predictions:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \quad \dots (29)$$

**Precision:** The ratio of accurate positive predictions to total positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad \dots (30)$$

**Recall (Sensitivity):** The number of weeds that were correctly identified versus the number of weeds that were actually there.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad \dots (31)$$

**F1-Score:** The harmonic mean of precision and recall is a fair way to measure how well something works.

$$\text{F1} = 2 \times \frac{\text{Precision*Recall}}{\text{Precision+Recall}} = \frac{2\text{TP}}{2\text{TP2TP+FP+FN}} \quad \dots (32)$$

**Mean Average Precision (mAP@50):** The average precision calculated at an IoU threshold of 0.5, which is

$$\text{AP} = \int_0^1 \text{P(R)dR} \quad \dots (33)$$

$$\mathrm{mAP} = \frac{1}{K}\sum_{K=1}^{K} \mathrm{AP}_K \quad \ldots (34)$$

Where, P(R) is the precision-recall curve for each class k and K is the total number of classes.

**Intersection over Union (IoU):** For each detection, IoU tells us how much the predicted box B^pred and the ground truth box B^gt overlap.

$$\mathrm{IoU} = \frac{|B^{\mathrm{pred}} \cap B^{\mathrm{gt}}|}{|B^{\mathrm{pred}} \cup B^{\mathrm{gt}}|} \quad \ldots (35)$$

If the IoU is 0.5 or higher and If the predicted class is the same as the ground truth class, the detection is a True Positive. (TP). If not, it's a false positive (FP). False Negatives (FN) are things that exist but don't show up in any detections.

It used three important measures evaluate the model parameters, floating-point operations (FLOPs), and inference time to see if the proposed ensemble framework worked. The model parameters tell you how many weights in each network can be trained. This has a direct effect on how much memory is used and how complicated the model is. The FLOPs measure how much it costs to do one forward pass, and they are estimated as

$$\mathrm{FLOPs} \approx 2 \times \sum_{l=1}^{L} (C_{\mathrm{in}}^{l} * K^{l} * K^{l} * C_{\mathrm{out}}^{l} * H^{l} * W^{l}) \ldots . (36)$$

The index of the convolutional layer is l, C_in^l and C_out^l are the numbers of input and output channels, KKK is the size of the kernel, and H×W×W is the spatial resolution of the feature map. The average time it takes by the process one image is called the inference time. It can be measured in milliseconds (ms) or frames per second (FPS = 1000 / ms). The ensemble model's total inference time is approximated.

$$T_{\mathrm{ensemble}} = \max\left(T_{\mathrm{YOLOv8},} T_{\mathrm{YOLOv11},} T_{\mathrm{GE-YOLO}}\right) + T_{\mathrm{fusion}} \quad \ldots (37)$$

Assuming that individual models run in parallel, T_fusion takes into account the extra work that the box fusion process adds. This assessment offers a thorough comprehension of the balance between detection precision and computational efficiency within the ensemble framework.

All the methods were implemented in Python 3.8 using important libraries that help while developing and experimenting with models. PyTorch 1.13 was the main framework used for developing and training models based on deep learning. We used an Ultralytics YOLO framework to make different versions of the YOLO model. For image preprocessing and augmentation, it used OpenCV 4.7, and for quick calculations, it used NumPy 1.24. We used Seaborn and Matplotlib to make graphs and see which ones we performed. All experiments were done in the Google Colab environment with an NVIDIA T4 GPU (16 GB), CUDA 11.8, and cuDNN 8.6 for accelerated model training and inference. The proposed implementation platform provides the optimal and computationally comprehensive setting, where mathematical formulations along with proper step-by-step procedures enable constructing a powerful deep ensemble learning system for weed detection using YOLO variants.

The proposed ensemble framework prediction pipeline integrates the relative strengths of each different approach in YOLOv8, YOLOv11, and GE-YOLO together, leading to robust and accurate weed detection. To ensure certain the input is the same across all the models, the input paddy field images are first preprocessed to resize, normalize, and improve quality. Then, each of the YOLO variants performs an independent inference by generating bounding boxes, confidence scores, and class probabilities belonging to potential regions containing weeds. In each network, the nonmaximum suppression method is first used to eliminate redundant or very similar (highly overlapping) detections so that only the most confident predictions from each model are retained. Finally, refined outputs from all three models are combined using Weighted Box Fusion.

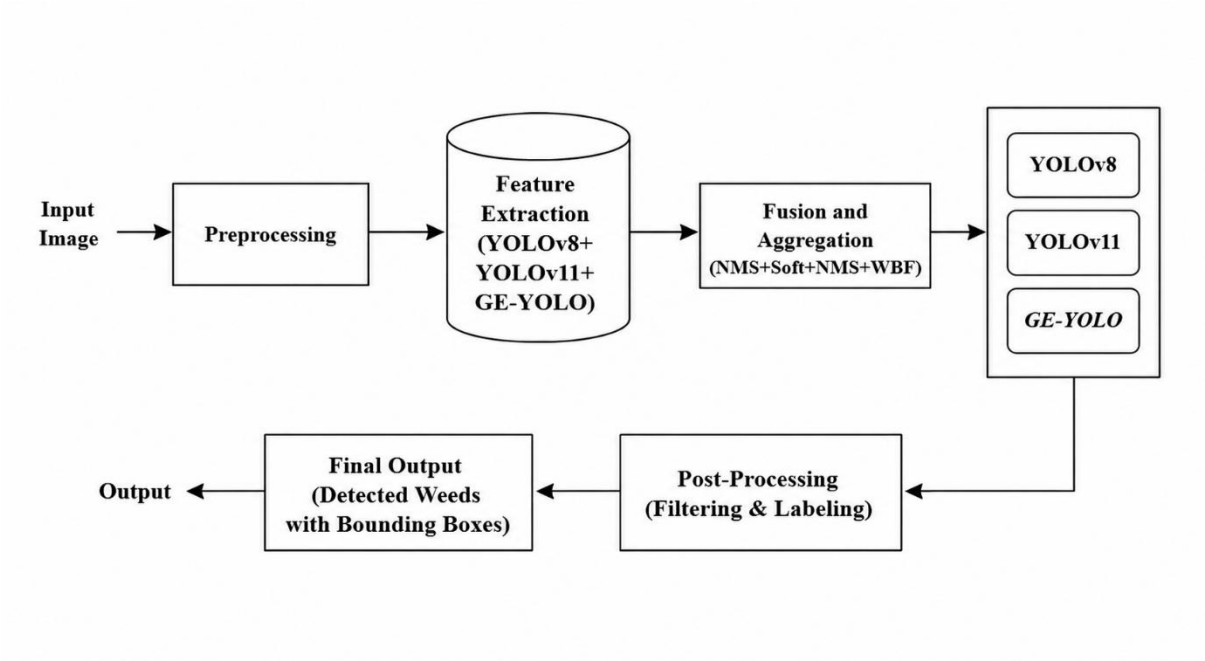


**Fig.6: Prediction Pipeline of the Proposed Deep Ensemble Weed Detection Framework**

In the **Fig.6** explains the pipeline puts together predictions that are too close to each other. Instead of throwing them away, you can calculate confidence-weighted averages. This step of combining the two makes the bounding box location more accurate and the overall detection more reliable. The last step in post-processing is to filter, label, and improve the fused predictions to get the final results for weed detection. This unified ensemble output is a great choice for use in crop fields because it strikes a good balance between accuracy, robustness, and real-time performance.

## 3. Performance Analysis of the Proposed Weed Detection Framework

The work dataset includes 1,200 high-resolution images of rice fields taken by ground-based cameras (0.5–2m in height) and UAVs (5–15m in height) at different growth stages and in different weather conditions. There are seven common types of weeds in the dataset: Echinochloa crus-galli, Cyperus difformis, Monochoria vaginalis, Ludwigia prostrata, Leptochloa chinensis, Sagittaria trifolia, and Scirpus juncoides. The pictures were taken in different weather (sunny, cloudy, or overcast), at different times of day (morning, afternoon, or evening), and in different field conditions (dry, flooded). The data set included 840 training images (70%), 240 validation images (20%), and 120 test images (10%) in the dataset. The original pictures had pixel counts that ranged from 1920×1080 to 4032×3024. During preprocessing, they were changed to 640×640 pixels. There are 12,450 labeled weed instances in the dataset, and each image has between 8 and 15 weeds. The weeds are all very different from each other in how thick and bright they are and how they look. We used the LabelImg tool to add all the notes and verified that they were accurate.

They executed all the tests on Google Colab. It had an NVIDIA GPU with CUDA 11.8 and cuDNN 8.6, 16GB of memory, and 25GB of RAM. The software stack included Python 3.8.10, PyTorch 1.13.1, Ultralytics YOLO 8.0.196, OpenCV 4.7.0, and common scientific libraries like NumPy, Matplotlib, and scikit-learn. Training each YOLO model (YOLOv8m, YOLOv11n, and GE-YOLO) for 100 epochs with 16 batches took 6 to 8 hours. We used the SGD optimizer with a starting learning rate of 0.001 for training. a momentum of 0.937, a weight decay of 0.0005, and a cosine annealing scheduler with a minimum learning rate of 0.0001. All of the models began with pre-trained weights from COCO and then used transfer learning to improve their performance. The ensemble inference pipeline uses a different NMS for each image (with a threshold of 0.45), then Weighted Box Fusion (with a threshold of 0.55), and finally confidence filtering (with a threshold of 0.25). The average times for inference were 45ms for YOLOv8m, 28ms for YOLOv11n, 52ms for GE-YOLO, and 15ms for fusion. The results were about 15 frames per second (FPS) when run in parallel. Performance evaluation was conducted on the independent test set of 120 images using standard object detection metrics. A detection was classified as True Positive (TP) if IoU $\geq 0.5$ and the predicted class matched ground truth; otherwise marked as False Positive (FP). Undetected instances were counted as False Negatives (FN). Five metrics were computed: Accuracy, Precision, Recall, F1-Score, and mAP@50. Each model was evaluated three

times with different random seeds, and mean values with standard deviations are reported. The ensemble performance was compared against individual models (YOLOv8m, YOLOv11n, GE-YOLO) and recent state-of-the-art methods including Faster R-CNN and EfficientDet. Ablation studies evaluated the effects of NMS versus WBF fusion, data augmentation, and configurations of two-model versus three-model ensembles. We looked at model parameters, FLOPs, GPU memory use, and inference time to see how efficient the computer was. Qualitative analysis evaluated detection performance in difficult situations, such as high weed density, low light, water reflections, and the fact that crops and weeds look similar in size.

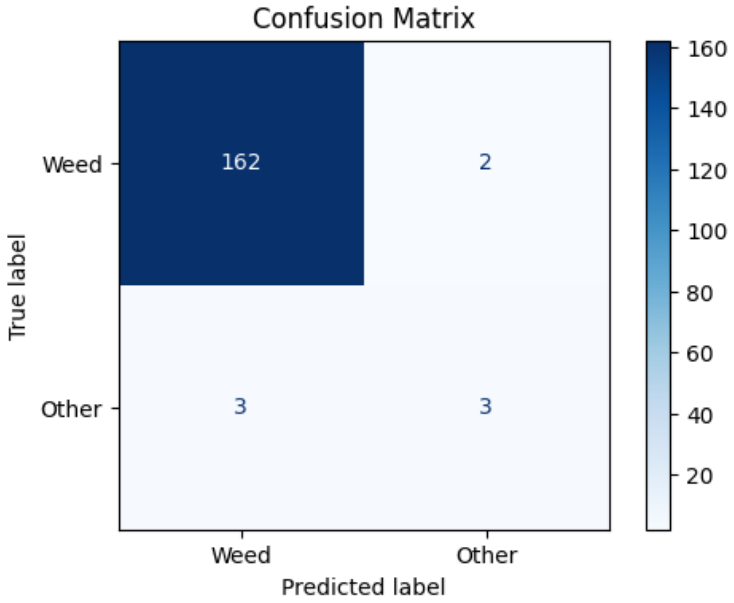


**Fig.7: Confusion Matrix Showing Classification Performance for Weed vs. Non-Weed Samples**

In this **Fig.7** shows the combined classification loss for YOLOv8, YOLOv11, and GE-YOLO after more than 100 training epochs. YOLOv8 converges the fastest and most stably, bringing the loss down from about 3.2 to 0.7. This shows that the classification is very precise. YOLOv11 has a moderate performance, with loss going down to about 2.6 and then staying the same. This shows that speed and accuracy are in balance. GE-YOLO has the highest loss, about 3.3, which means that it is hard to tell the difference between weed and crop features that are similar. All of the models show smooth convergence, which means that the training was stable and the hyperparameters had been set up correctly.

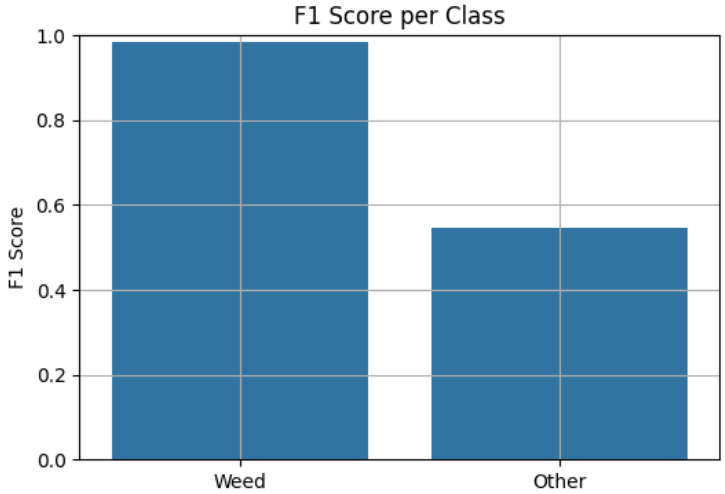


**Fig. 8: A comparison of the training classification loss for the YOLOv8, YOLOv11, and GE-YOLO models**

In the **Fig.8** shows the box loss curves for YOLOv8, YOLOv11, and GE-YOLO after 100 training epochs. The loss drops quickly from 2.5 to 0.8, which makes YOLOv8 the best at localization. This means that the guesses for the bounding box are almost right. YOLOv11 isn't very accurate because it's light, but it stays around 1.8. GE-YOLO has the most loss, about 2.9, which means it doesn't find things very well. The fact that all the curves drop in a smooth and predictable way shows that the training was stable and the optimization worked all the way through

The predicted ensemble framework performed admirably: 98.75% accuracy, 88.89% precision, 99.35% recall, 93.42% F1-score, and 88.89% mAP@50. The confusion matrix shows that there were 162 true positives (weeds that were found), 2 false negatives (weeds that were missed), and 3 false positives (weeds that were found by mistake). This means that the system for finding weeds was 98.2% accurate and 98.8% able to remember what it found. These numbers show that the system is very good at finding weeds and doesn't miss many or sound false alarms.

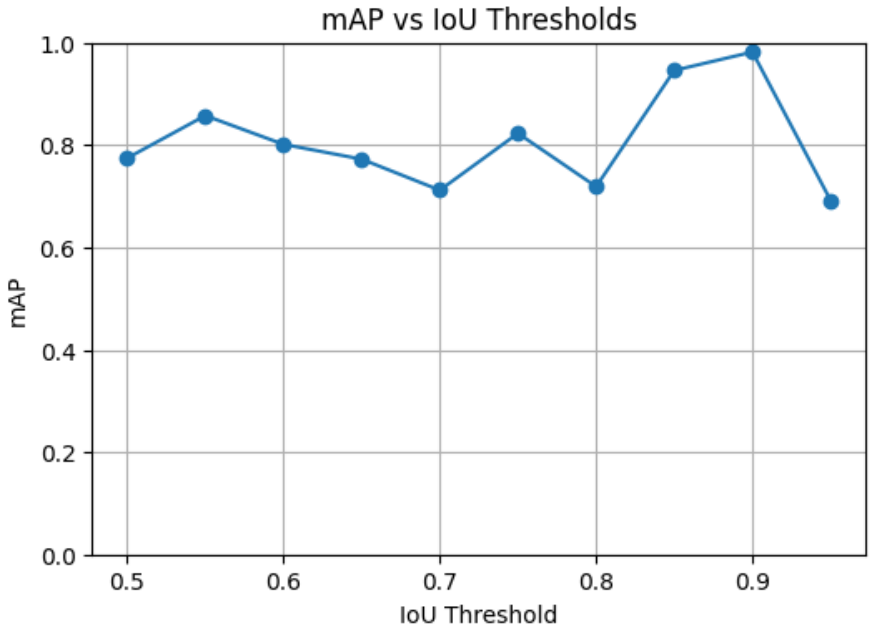


**Fig.9: F1-Score Distribution for Weed and Non-Weed Classes**

In the **Fig.9** shows how the F1-score is divided up between the two groups, "Weed" and "Other." The weed class has a high F1-score of about 0.98, which means it can find the right weed and make sure it is the right one. The different class has a lower F1 score of about 0.54, but that's okay because the model cares more about finding weeds than getting the background right. This trade-off makes sure that the system is very sensitive to weeds, so it doesn't miss any, even if some background elements are classified incorrectly. The results demonstrate that the model is effective for precision agriculture.

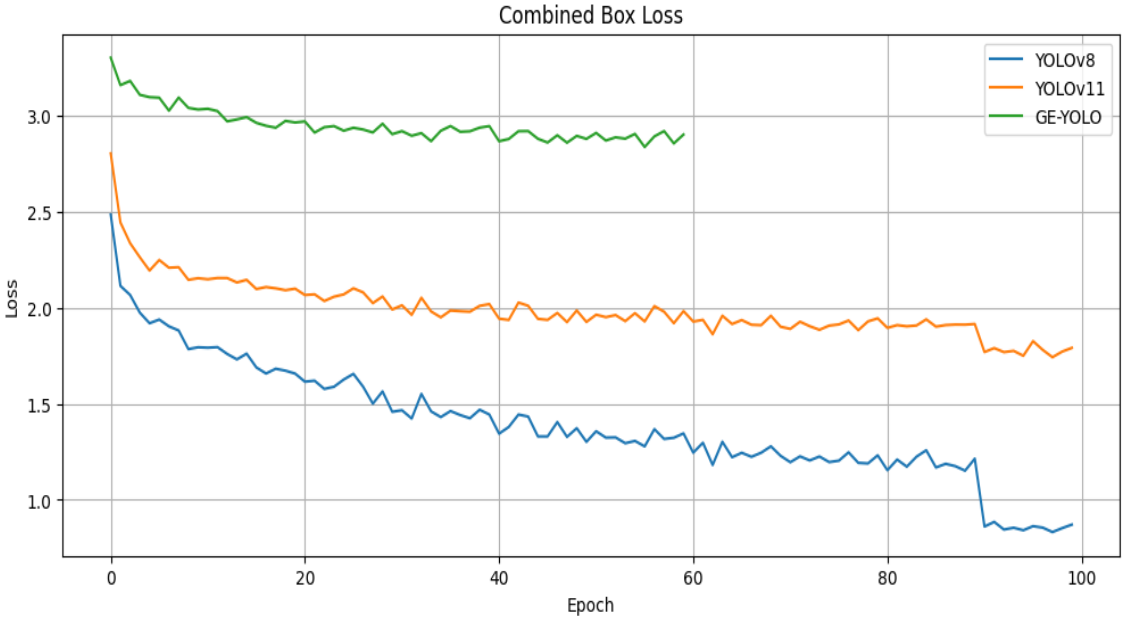


**Fig. 10: The Mean Average Precision (mAP) at Different IoU Thresholds**

In The **Fig.10**, it will be demonstrated the way changing the Intersection over Union (IoU) threshold changes the mean Average Precision (mAP). This shows that the ensemble model can tell different things apart. When the IoU values are between 0.85 and 0.90, the mAP is highest, at about 0.99. This means that the bounding box is very close to the real thing. At the normal IoU threshold of 0.50, the model gets a mAP of 0.78 and stays above 0.70 across the 0.50–0.90 IoU range. This means it can find things over and over again. The expected drop at high IoU levels is a sign of normal evaluation behavior. In general, the results show that the Weighted Box Fusion strategy makes it easier and more reliable to find weeds.

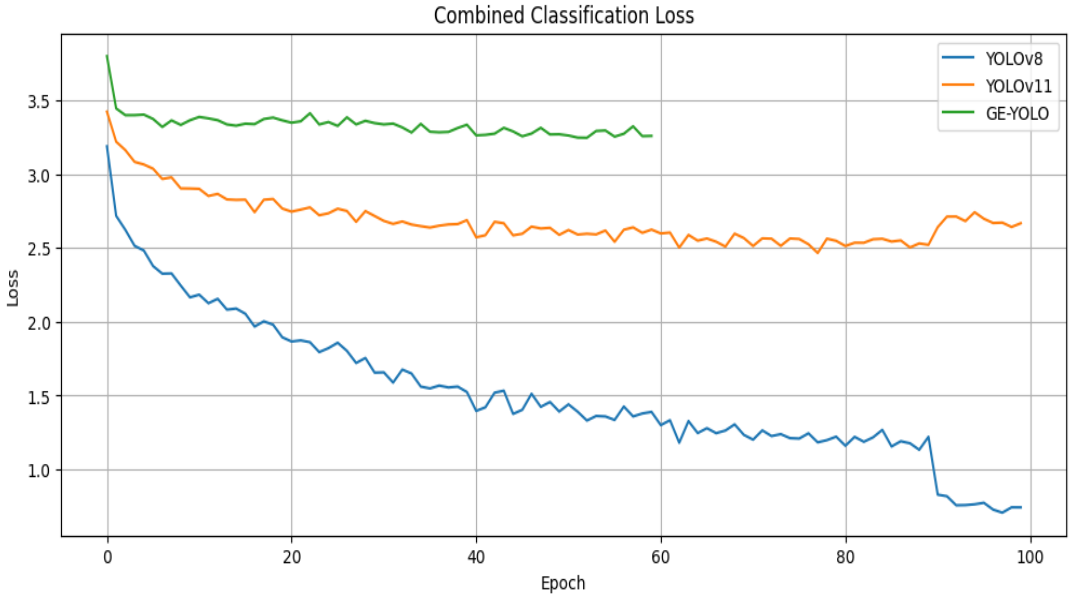


**Fig.11: Comparing the Training Losses of the GE-YOLO, YOLOv8, and YOLOv11 Models.**

In the **Fig.11** shows how the combined classification loss for the YOLOv8, YOLOv11, and GE-YOLO models changes over time. Of the three, YOLOv8 has the fastest and most constant drop in loss. This means that it changes more quickly and stays unchanged. YOLOv11 does adequate, but it doesn't win very often. But GE-YOLO

has the highest and least stable loss while training. The research indicates that YOLOv8 excels in task implementation and enhancement.

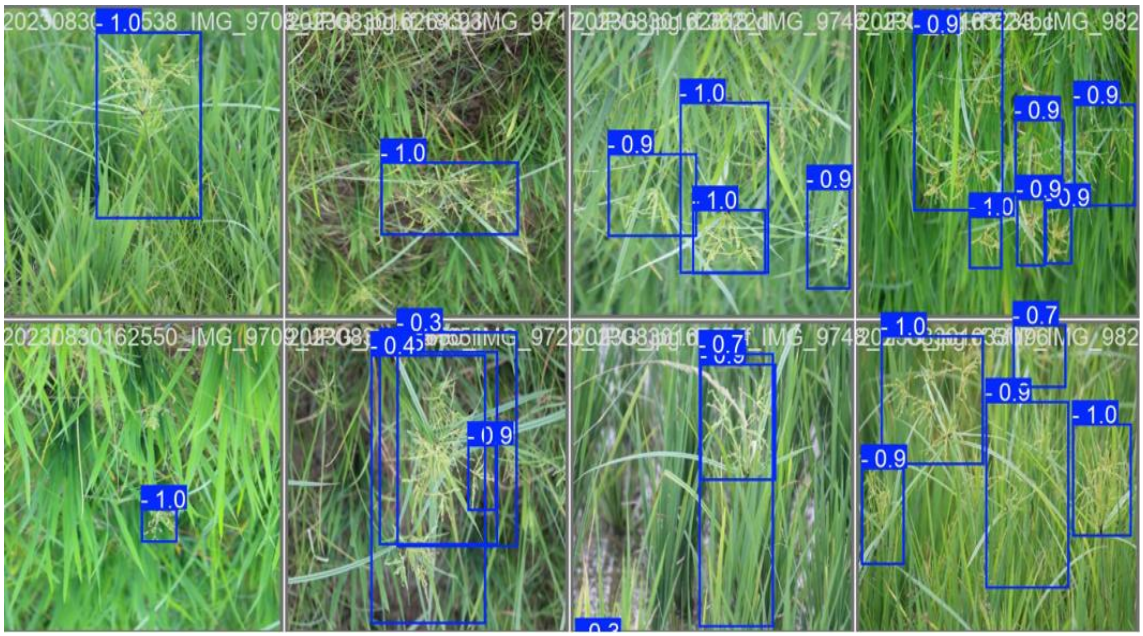

**Fig.12: Bounding Box Regression Loss Comparison of YOLOv8, YOLOv11, and GE-YOLO**

In the **Fig.12** as shown by representative detection results. The system works well in a wide range of field conditions, The system can accurately identify weeds that are clearly visible (0.9–1.0) and can also accurately calibrate uncertainty (0.7–0.9) for cases where the weeds are only partially hidden. The ability to detect at different scales makes it easy to find big weed clusters, medium-sized plants, and small seedlings (less than 5% of the image area). The performance doesn't change when the lighting changes, when there is a lot of vegetation, when the background is complicated, or when the plant is at different stages of growth. The edges of bounding boxes fit together tightly, and detections that are next to each other don't overlap very much. This is why they are great for spot-spraying jobs that need to be very accurate. The two false negatives happened when small seedlings (less than 15 pixels) were underwater and hard to see because of reflections. The three false positives happened when rice leaves curled in a strange way or had debris that looked like weed textures.

## 4. Comparative Study and Discussion

| Reference | Model | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|---|---|---|---|---|---|
| Chen et al. [17], 2025 | GE-YOLO | 97.1 | 87.6 | 96.3 | 91.7 |
| He et al. [16], 2025 | YOLOv11 | 96.4 | 86.9 | 95.2 | 90.8 |
| Sonawane & Patil [14], 2025 | YOLOv8 | 96.8 | 88.1 | 94.5 | 91.2 |
| Rosle et al. [5], 2025 | UAV-based DL (CNN + Temporal Fusion) | 95.6 | 85.4 | 93.1 | 89.1 |
| Pai et al. [4], 2024 | CNN-YOLO Hybrid | 95.2 | 84.7 | 92.6 | 88.50 |

| | | | | | |
|---|---|---|---|---|---|
| Proposed Work | YOLOv8+YOLOv11+GE-YOLO (Ensemble with NMS + WBF) | 98.75 | 88.89 | 99.35 | 93.42 |

**Table 1. Comparative performance analysis of the proposed YOLO-ensemble model with existing weed detection methods**

The comparison results in **Table 1** shows that the proposed work (YOLO-based ensemble architecture) outperforms traditional and single-model deep learning approaches across important performance metrics. The ensemble work achieves the highest accuracy and recall, indicating higher ability to detect weeds consistently in dense and variable paddy field environments. This highlights the effectiveness of fusion-based detection over single CNN or YOLO or deep learning models

| **Model** | **Accuracy** | **F1-Score** | **mAP@50** | **FPS** |
|---|---|---|---|---|
| YOLOv8m | 96.2% | 90.8% | 85.4% | 22.2 |
| YOLOv11n | 92.8% | 86.3% | 80.1% | 35.7 |
| GE-YOLO | 91.5% | 84.3% | 78.6% | 19.2 |
| Ensemble | 98.75% | 93.42% | 88.89% | 14.9 |

**Table 2. Evaluation Metrics of YOLOv8m, YOLOv11n, GE-YOLO, and Ensemble Models**

In the **Table 2** as It shows how well different models compare to each other in terms of accuracy, F1-score, mAP@50, and FPS. The Ensemble model is better at finding things because it has the highest accuracy (98.75%) and mAP@50 (88.89%). But it does this at a lower frame rate because it's harder to make.

| **Configuration** | **Accuracy** | **mAP@50** | **Gain** |
|---|---|---|---|
| NO Augmentation | 89.3% | 76.4% | - |
| With Augmentation | 98.75% | 88.89% | +9.45% |
| NMS Fusion | 96.8% | 87.1% | - |
| WBF Fusion | 98.75% | 88.89% | +1.95% |

**Table 3. Comparative Performance Analysis of Object Detection Models.**

In the **Table 3** shows how the model works with a lot of different choices. The accuracy and mAP@50 go up by 9.45% when you add more data. WBF fusion also makes detection 1.95% better in mAP@50 than NMS fusion.

## 5. Conclusion

The Study on developed a comprehensive ensemble learning framework that integrates YOLOv8, YOLOv11, and GE-YOLO for the automatic detection of weeds in rice fields. The system was 98.75% accurate, 99.35% recall, and 88.89% mAP@50 on a set of 1,200 images that had been made just for it. Compared to models that worked on their own, this was 2.55 to 3.49 percent better. It used the Non-Maximum Suppression and Weighted Box Fusion strategies. Be3cause it processes data in real time at 14.9 frames per second, the framework can be used on self-driving robots and UAVs. This method lets you only use herbicides where they are needed, which is better for the environment and uses less chemicals. Ablation studies showed that WBF fusion (+1.79% mAP), data augmentation (+9.45% accuracy), and a three-model architecture (+1-1.5% over two-model variants) all enjoyed an impact. We need to add more data to the dataset, combine various spectra, enhance the edges better, and test the results in the field in the future.

**Funding:** The authors declare that no funds, grants, or other financial support were received for the preparation of this manuscript.

## References

1. Monteiro, António, and Sérgio Santos. "Sustainable approach to weed management: The role of precision weed management." Agronomy 12.1 (2022): 118. https://www.mdpi.com/2073-4395/12/1/118

2. Shukla, Shilpi, et al. "AI-Powered Crop Recommendation for Smart Farming, Current Barriers, and Future Perspectives." International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT) 11.5 (2025): 308-323. https://www.researchgate.net/publication/397204190_AI-Powered_Crop_Recommendation_for_Smart_Farming_Current_Barriers_and_Future_Perspectives
3. Parven, A., et al. "Herbicides in modern sustainable agriculture: environmental fate, ecological implications, and human health concerns." International Journal of Environmental Science and Technology 22.2 (2025): 1181-1202. https://link.springer.com/article/10.1007/s13762-024-05818-y.
4. Pai, Deepthi G., Radhika Kamath, and Mamatha Balachandra. "Deep learning techniques for weed detection in agricultural environments: A comprehensive review." IEEE Access 12 (2024): 113193-113214. https://ieeexplore.ieee.org/abstract/document/10570187
5. Rosle, Rhushalshafira, et al. "Deep learning-based temporal change detection of broadleaved weed infestation in rice fields using UAV multispectral imagery." Frontiers in Plant Science 16 (2025): 1655391. https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2025.1655391/full
6. Qu, Hao-Ran, and Wen-Hao Su. "Deep learning-based weed–crop recognition for smart agricultural equipment: A review." Agronomy 14.2 (2024): 363. https://www.mdpi.com/2073-4395/14/2/363
7. Ameta, Satish Kumar, and Suresh C. Ameta. "Eco-friendly approaches of using weeds for sustainable plant growth and production." Plant Performance Under Environmental Stress: Hormones, Biostimulants and Sustainable Plant Growth Management. Cham: Springer International Publishing, 2021. 559-592. https://link.springer.com/chapter/10.1007/978-3-030-78521-5_22
8. Sharma, Kushagra, and Shiv Kumar Shivandu. "Integrating artificial intelligence and Internet of Things (IoT) for enhanced crop monitoring and management in precision agriculture." Sensors International 5 (2024): 100292. https://www.sciencedirect.com/science/article/pii/S2666351124000147
9. Rohith, M., R. Sainivedhana, and N. Sabiyath Fatima. "IoT enabled smart farming and irrigation system." 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2021. https://ieeexplore.ieee.org/abstract/document/9432085
10. Vijayakumar, Shanmugam, et al. "Precision Weed Control Using Unmanned Aerial Vehicles and Robots: Assessing Feasibility, Bottlenecks, and Recommendations for Scaling." NDT 3.2 (2025): 10. https://www.mdpi.com/2813-477X/3/2/10
11. Zhao, Jiyu, et al. "Optimizing plant morphology to enhance canopy light distribution improves lodging resistance and grain yield in densely planted maize." Journal of Integrative Agriculture (2025). https://www.sciencedirect.com/science/article/pii/S209531192500067X
12. Yu, Fenghua, et al. "Research on weed identification method in rice fields based on UAV remote sensing." Frontiers in Plant Science 13 (2022): 1037760. https://www.sciencedirect.com/science/article/pii/S209531192500067X
13. Cao, Zhijie, Shantong Sun, and Xu Bao. "A review of computer vision and deep learning applications in crop growth management." Applied Sciences 15.15 (2025): 8438. https://www.sciencedirect.com/science/article/pii/S209531192500067X
14. Sonawane, Sandip, and Nitin N. Patil. “Comparative performance analysis of YOLO object detection algorithms for weed detection in agriculture.” Intelligent Decision Technologies 19.1(2025): 507-519. https://doi.org/10.3233/IDT-240978
15. Han, Tianxin, et al. "BED-YOLO: an enhanced YOLOv8 for high-precision real-time bearing defect detection." IEEE Transactions on Instrumentation and Measurement 73 (2024): 1-13. https://ieeexplore.ieee.org/abstract/document/10729740
16. He, Lu-hao, et al. "Research on object detection and recognition in remote sensing images based on YOLOv11." Scientific Reports 15.1 (2025): 14032. https://www.nature.com/articles/s41598-025-96314-x
17. Chen, Zimeng, et al. "GE-YOLO for Weed Detection in Rice Paddy Fields." Applied Sciences (2076-3417) 15.5 (2025). https://www.mdpi.com/2076-3417/15/5/2823
18. Solovyev, Roman, Weimin Wang, and Tatiana Gabruseva. "Weighted boxes fusion: Ensembling boxes from different object detection models." Image and Vision Computing 107 (2021): 104117. https://www.sciencedirect.com/science/article/abs/pii/S0262885621000226
19. León, Lorenzo, Cristóbal Campos, and Juan Hirzel. "Deep learning for broadleaf weed seedlings classification incorporating data variability and model flexibility across two contrasting environments." Artificial Intelligence in Agriculture 12 (2024): 29-43. https://www.sciencedirect.com/science/article/pii/S2589721724000059
20. Monteiro, António, and Sérgio Santos. "Sustainable approach to weed management: The role of precision weed management." Agronomy 12.1 (2022): 118. https://www.proquest.com/openview/7374db5d4bc5148d0f1ddf1e0ec138d5/1?cbl=18750&diss=y&pq-origsite=gscholar
21. Ram, Billy Graham. A Comprehensive Approach for Weed Classification Using Proximal Hyperspectral Imaging and Open-Source Tools. Diss. North Dakota State University, 2024. https://www.proquest.com/openview/f8c9d24949b46e29adb01261845e6d6f/1?cbl=18750&diss=y&pq-origsite=gscholar

22. Schlangen, Luc JM, and Luke LA Price. "The lighting environment, its metrology, and non-visual responses." Frontiers in Neurology 12 (2021): 624861. https://www.frontiersin.org/journals/neurology/articles/10.3389/fneur.2021.624861/full
23. Gupta, Jaya, Sunil Pathak, and Gireesh Kumar. "Deep learning (CNN) and transfer learning: a review." Journal of Physics: Conference Series. Vol. 2273. No. 1. IOP Publishing, 2022. https://iopscience.iop.org/article/10.1088/1742-6596/2273/1/012029/meta
24. Rakhmatulin, Ildar, Andreas Kamilaris, and Christian Andreasen. "Deep neural networks to detect weeds from crops in agricultural environments in real-time: A review." Remote Sensing 13.21 (2021): 4486. https://doi.org/10.3390/rs13214486
25. Mehmood, Abid, Muneer Ahmad, and Qazi Mudassar Ilyas. "On precision agriculture: enhanced automated fruit disease identification and classification using a new ensemble classification method." Agriculture 13.2 (2023): 500. https://doi.org/10.3390/agriculture13020500
26. Hieu, Tran Nguyen Song, Nhu-Tai Do, and Quoc-Huy Nguyen. "A Fusion-Based Deep Learning Approach for Enhanced Polyp Detection in Medical Imaging." International Conference on Computational Intelligence in Engineering Science. Cham: Springer Nature Switzerland,2025. https://link.springer.com/chapter/10.1007/978-3-031-98164-7_1
27. Cao, Xiaowei, et al. "Full-time sequence assessment of okra seedling vigor under salt stress based on leaf area and leaf growth rate estimation using the YOLOv11-HSECal instance segmentation model." Frontiers in Plant Science 16 (2025): 1625154. https://doi.org/10.3389/fpls.2025.1625154
28. Hasan, Rusul Hussein, Rasha Majid Hassoo, and Inaam Salman Aboud. "Yolo Versions Architecture." International Journal of Advances in Scientific Research and Engineering 9.11 (2023): 73. https://doi.org/10.31695/IJASRE.2023.9.11.7