InsectDSOT: A Neural Network for Insect Detection in Olive Trees

Lotfi Souifi[1], Afef Mdhaffar[1,2], Ismael Bouassida Rodriguez[1], Mohamed Jmaiel[1,2] and Bernd Freisleben[3]

[1] *ReDCAD Laboratory, ENIS, University of Sfax, B.P. 1173 Sfax, Tunisia*

[2] *Digital Research Center of Sfax, 3021 Sfax, Tunisia*

[3] *Dept. of Math. & Comp. Sci., Philipps-Universität Marburg, Germany*

Keywords: Deep Learning, Object Detection, Insect Detection, Yolov5, RepVGG.

Abstract: Controlling insect pests in agricultural fields is a major concern. Despite technological developments, most farm management methods and technologies still rely on experts for management and do not yet match the criteria required for precise insect pest control. In this paper, we present a neural network approach for detecting and counting insects. Using the Yolov5n 6.1 version as a baseline model, this paper proposes replacing the Conv layers in the original model's backbone and neck with the RepVGG layer. We use transfer learning to improve performance by training our proposal on the MS COCO dataset and then use the output model of this training as the input of our new training. Our proposal is validated using the DIRT (Dacus Image Recognition Toolkit) dataset. The obtained results demonstrate that our approach, based on an improved Yolov5, achieves 86.1% of precision. It outperforms four versions of the original yolov5 and yolov5-based versions with modified backbones based on lightweight models.

1 INTRODUCTION

Insects, pests, diseases, and weed infestations are estimated to harm 40% of agricultural productivity (Carvajal-Yepes et al., 2019). At least $220 billion is spent annually on the global economy by plant diseases, and at least $70 billion is spent annually on invasive insects (FAO, 2021). Therefore, it is crucial to detect insects and precisely count their numbers to propose an adequate spraying strategy. For small targets like insect pests, the detection ability is currently inadequate for three reasons. First, there is a lack of information. Indeed, minor items take up fewer pixels in the image and have less information, making it difficult to extract important characteristics without being influenced by the underlying surroundings. Second, the positional precision requirements are stringent. The offset of the bounding box is high for the inaccuracy of tiny target identification, whether in the training or prediction phases. Third, there is the object aggregation difficulty. When this happens, targets are aggregated into one point after being presented to the deep feature map via repeated downsampling, resulting in the inability to differentiate between separate objects. Furthermore, it will make regressing the bounding boxes and converging the model harder. Moreover, the accuracy of the lightweight model is typically low when compared to other models.

Some studies use one of the two following approaches to tackle the problem of low accuracy in the lightweight model. The first approach consists of using databases (Yang et al., 2021), which includes photos of insects at enormously large sizes. The disadvantage of this approach is that the output model performs poorly, particularly under real-world settings. The second approach consists of adding more layers to existing models and/or combining them with a new feature extractor from another technique. The downside of this approach is that the output model size generally increases. The second approach is used by (Li et al., 2021b; He et al., 2019; Cardoso et al., 2022). (Li et al., 2021b) integrated EfficientNet and Yolo. The precision improvement is around 5%, but the output model size is nearly 60 MB, which is considered to be large for edge deployment. (He et al., 2019), employed three meta-architectures (Faster R-CNN (Girshick, 2015), R-FCN (Dai et al., 2016), and SSD (Liu et al., 2016)) in conjunction with feature extractors (ResNet (He et al., 2016), Inception (Szegedy et al., 2017), and MobileNet (Howard et al., 2017)) to select the best model while taking the mean average precision (mAP) into account. Their technique has the drawback of having a low mAP (77%). (Cardoso et al., 2022) evaluated four basic models: Faster R-CNN, Yolov4 (Bochkovskiy et al., 2020), and Yolov5 (in small and large versions). The average precision (AP) in three of the four selected models is low (i.e., approximately 80%) and the size of the final model is too large for edge devices (i.e., 167 MB).

In this paper, we adopt the second approach to develop a neural network, called InsectDSOT, for detecting and counting insects. Using the Yolov5n 6.1 version as a baseline model, we replace the Conv layers in the original model's backbone and neck with a RepVGG layer. We use transfer learning to improve performance by training our proposal on the MS COCO dataset and then use the output model of this training as the input of our new training. The model's accuracy is improved significantly. The size of the resulting model is increased by 0.5 MB, which is reasonable for edge deployment. We test our hypothesis using a variety of backbones, including mobileNet series, ShuffleNet series, Yolov3, Yolo5Face, EfficientLite, PPLC, GhostNet, Global Context, on the DIRT dataset of olive insects. Our experimental results demonstrate the usefulness of our proposal in terms of precision, without increasing significantly the model size.

The remainder of the paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes our proposal. Section 4 describes our experiments. Section 5 concludes this paper and outlines areas for future research.

2 RELATED WORK

(Chen et al., 2022) used MobileNetv3 (Howard et al., 2019) with Yolov5 to detect airplanes in real time. The difficulty with their technique is that the suggested model accuracy is very weak, falling below 60%. (Hou et al., 2022) proposed M-YOLO, an infrared object detector created by combining MobileNetv2 (Sandler et al., 2018) with Yolov4. The accuracy of the proposed model is low, indicating a flaw in their technique. (Qi et al., 2021) proposed YOLO5Face, a face detector based on YOLOv5. This technique achieves good results and outperforms other models when dealing with face identification, in terms of size and performance. This method has not been applied in the identification of small objects, particularly insects. (Liu et al., 2021) proposed combining the Global Context Network with Yolov3 (Redmon and Farhadi, 2018). Six out of 24 types of insects respond well to this approach (i.e., the same characteristic of those six is that their size is big compared with the others). Their model's mean average precision is insufficient. (Qian et al., 2022) proposed using EffientNet-Lite (Tan and Le, 2019) as the backbone of their method. The authors validated their technique by testing it with five different backbones: MobileNetv3Small, GhostNet (Han et al., 2020), ShuffleNetV2 (Ma et al., 2018), and PP-LCNet (Cui et al., 2021). The findings demonstrate that all the other models produce inferior outcomes when compared to their suggestion.

Yolov3 is used by (Li et al., 2021a) with data gathered from the Internet. The flaw in this method is that the resolution of the image characteristics, such as light and insect size, may differ from the real environment. In contrast to (Li et al., 2021a), (Bjerge et al., 2022; Tresson et al., 2019) implemented their method in the real world while using expensive materials. (Takimoto et al., 2021) combines two detection and classification approaches which are Yolov4 alongside with EfficientNet in their study. (Mamdouh and Khattab, 2021) proposed a system for detecting and counting olive flies using an enhanced version of Yolov4. The suggested model's drawback is that it has a poor accuracy (84%) that can be improved. (Cabrera and Villanueva, 2022) used Yolov5 in its default configuration without any additional changes. The size of the insects in the images used by (Yuan et al., 2021) was at least 30%, which is much larger than what would be found in the real world (in some cases, the percentage of insects is less than 10%). In our article, we focus on the YOLO (Yolov5) detector, a one-stage anchor-based detector. Each deep learning model typically has benefits and drawbacks as feature extractors. Some models were created using feature extractors from already-built models, like SSD (Single Shot multi box Detector), which was suggested using the VGG16 feature extractor. Numerous studies have been conducted about combining model advantages to overcome other models' disadvantages. ResNet (and its variations), MobileNet (and its variations), and InceptionNet were the three models that were most often used as backbones. The target models, such as SSD (Patel and Bhatt, 2019) (SSD with Inception and MobileNet) and Yolo (Zha et al., 2021) (uses Yolo with MobileNet), were both single-stage detectors. Additionally, there are two stage detectors like Faster R-CNN (Teng et al., 2022) (uses Faster-RCNN with ResNet50). To summarize, most extant one-stage object identification methods use feature fusion to increase the accuracy of tiny objects. The main explanation for this is that low-level feature map semantic information is sparse but correct in presenting object position, while high-level feature semantic information is abundant but inaccurate.

3 InsectDSOT

This section presents the original architecture of Yolov5 and the core principles of the RepVGG block. Also, we present InsectDSOT, our improved Yolov5 for insect detection.

3.1 Architecture of Yolo

Yolov5 has undergone various modifications since its initial release in 2020. For example, the adoption of the C3 layer was the most significant enhancement in version 4 (Jocher et al., 2021a). Also in version 6 (Jocher et al., 2021b), the nano version suggestion, which is the basis of our approach. Furthermore, some other enhancements were recently incorporated in versions 6.1 (Jocher et al., 2022a) and 6.2 (Jocher et al., 2022b). The version used in our experiment is 6.1 since the last version (6.2) included classification and we just focus on detection. The Yolov5 architecture is shown in Figure 1.

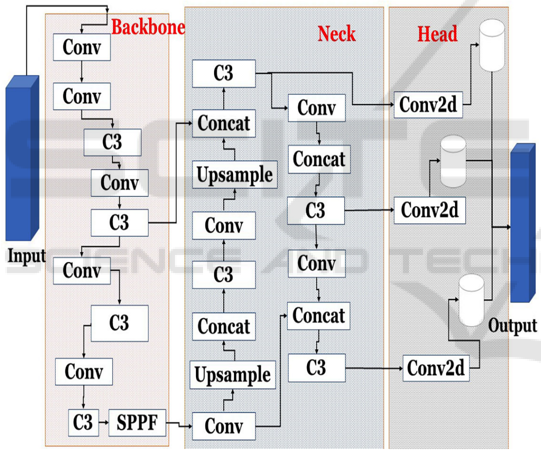


Figure 1: Yolov5 v6.1 architecture.

YOLOv5 is available in five versions: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The only difference between the variants is the model depth and layer width. We chose YOLOv5N because it has the smallest depth and feature map width, as well as the quickest training time. Also, the result of our proposal is a lightweight model that can be deployed on both edge devices and servers. The other three are expanded and deeper variants of YOLOv5s. Table 1 lists the features of each Yolov5 version.

3.2 RepVGG Model

According to (Ding et al., 2021), many advanced ConvNets outperform basic ConvNets in terms of accuracy, despite having significant limitations, such as:

Table 1: Yolo's versions: depth and width.

Yolov5	model depth	layer width
N	0.33	0.25
S	0.33	0.5
M	0.67	0.75
L	1.0	1.0
X	1.33	1.25

(i) complex multi-branch designs (for example, residual addition in ResNet) make the model harder to develop and adapt, slow inference, and decrease memory consumption; (ii) some modules increase memory access costs and do not support a variety of devices (e.g., depthwise conv in MobileNets). The authors suggest RepVGG as a solution to these restrictions, RepVGG is a strong convolutional neural network architecture with a VGG-like inference-time body (Simonyan and Zisserman, 2014) made up of basically a stack of 3×3 convolution and ReLU. While it can reach the accuracy and speed of multi-branch complicated networks, we chose RepVGG-A0 since it has the fewest network parameters among the RepVGG configurations. Nonetheless, the RepVGG-A0 network has a considerable network size and high recognition accuracy (it can identify 1000 images classes in the ImageNet-2012 test set with a Top-1 accuracy of 72.41%). RepVGG provides the following benefits:

- The model features a VGG-like plain (also known as feed-forward) topology 1 (i.e., each layer takes the output of the previous layer as input and feeds it into the next layer).
- The model's body mainly makes use of 3×3 conv and ReLU.
- No automatic search (Zoph et al., 2018), manual refinement (Radosavovic et al., 2020), compound scaling (Tan and Le, 2019), or other heavy designs are used to instantiate the concrete architecture (including the particular depth and layer widths).

The differences between the ResNet and RepVGG designs, as well as the RepVGG inference, are depicted in Figure 2:

- (A) ResNet: It has a multi-path topology during training and inference, which causes it to be extremely slow and memory-inefficient.
- (B) RepVGG Training: It only acquired multi-path topology during training.
- (C) RepVGG Inference: It only has a single-path topology during inference, allowing a quick inference time.

RepVGG employs ResNet-like identity and 1×1 branches to ensure that a building block's training-time information flow is $y = x + g(x) + f(x)$, as in (B).

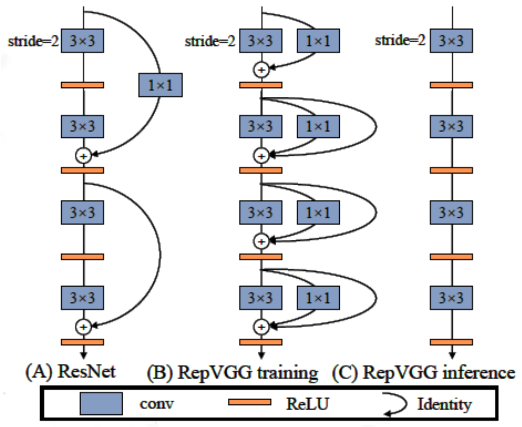


Figure 2: Difference between RepVGG and ResNet architecture.

From the same point of view as (Veit et al., 2016), the model is transformed into an aggregation of 3^n members with n such blocks.

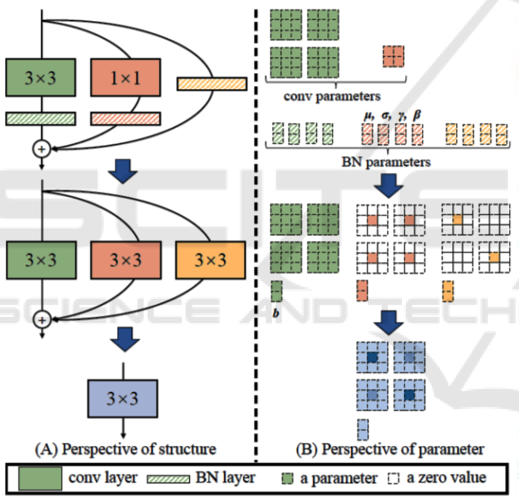


Figure 3: A RepVGG block's hierarchical re-parameterization.

To move from (B) to (C), RepVGG employs the concept of re-parameterization to turn a learned block into a single 3×3 convolutional layer for inference. It is important to note that batch normalization (BN). (Ioffe and Szegedy, 2015) is used before the addition for each branch. The concept of re-parameterization employs $W^{(3)} \in \mathbb{R}^{C_1\times C_2\times3\times3}$ to represent the kernel of a 3×3 conv layer with C_1 input and C_2 output channels, and $W^{(1)} \in \mathbb{R}^{C_1\times C_2}$ for the kernel of 1×1. We utilise $\alpha^{(3)},\sigma^{(3)},\gamma^{(3)},\beta^{(3)}$ as the accumulated mean, standard deviation and learned scaling factor and bias of the BN layer following 3×3 conv, $\alpha^{(1)},\sigma^{(1)},\gamma^{(1)},\beta^{(1)}$ for the BN following 1×1 conv, and $\alpha^{(0)},\sigma^{(0)},\gamma^{(0)},\beta^{(0)}$ for the identity branch. Let $M^{(1)} \in \mathbb{R}^{C_1\times N\times H_1\times W_1}$, $M^{(2)} \in \mathbb{R}^{C_2\times N\times H_2\times W_2}$, be the source and result, and $*$ be the convolution operator, if $C_1 = C_2, H_1 = H_2$, and $W_1 = W_2$ we have:

$$\begin{aligned}M^{(2)} &= bn(M^{(1)} * W^{(3)}, \alpha^{(3)}, \sigma^{(3)}, \gamma^{(3)}, \beta^{(3)}\\ &+bn(M^{(1)} * W^{(1)}, \alpha^{(1)}, \sigma^{(1)}, \gamma^{(1)}, \beta^{(1)}) \qquad (1)\\ &+bn(M^{(1)}, \alpha^{(0)}, \sigma^{(0)}, \gamma^{(0)}, \beta^{(0)})).\end{aligned}$$

where bn in is the inference-time BN function:

$$bn(M,\lambda,\sigma,\gamma,\beta)_{:,i,:,:} = (M_{:,i,:,:} - \mu_i)\frac{\gamma_i}{\gamma_i} + \beta_i. \qquad (2)$$

The final bias is obtained by summing the three bias vectors, and the final 3×3 kernel is obtained by putting the 1*1 kernels into the central point of the 3×3 kernel, which is readily achieved by first zero-padding the two 1×1 kernels to 3×3 and adding the three kernels up, as illustrated in Figure 3. As shown in Figure 3, we assume $C_2 = C_1 = 2$ for clarity, therefore the layer contains four 3×3 matrices and the kernel of the 1×1 layer is a 2×2 matrix.

3.3 The Architecture of InsectDSOT

We used the Yolov5 model as the base model and used the RepVGG as a feature extractor. Since the primary purpose of the RepVGG block is to overcome some of the drawbacks of ConvNets, we replace the Conv layers in the first part (backbone) of the Yolo, as well as the final two Conv layers in the second part (Neck) of the Yolov5 with RepVGG block. We only maintained three original Conv layers: one in the initial input of the model, and the others two are in the are in the model's Neck and have no direct influence on the model output. We opted to retrain them. Figure 4 illustrates the original Yolo's architecture, with the changed blocks of our InsectDSOT approach highlighted in red.

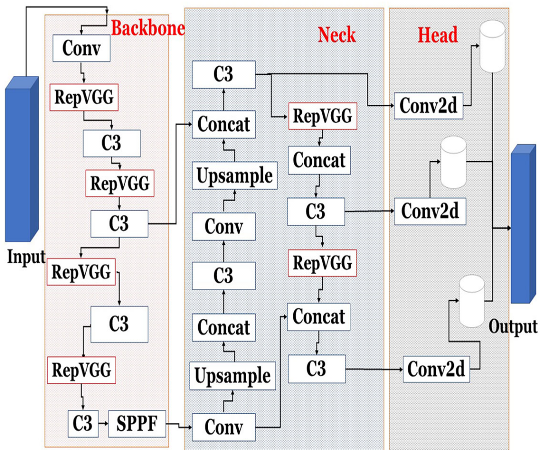


Figure 4: Modified Yolo version based on RepVGG block.

4 EXPERIMENTS

This section describes the used dataset, the conducted experiments, and the obtained results.

4.1 Dataset

Our experiments are conducted on the DIRT (i.e., Dacus Image Recognition Toolkit) dataset (Kalamatianos et al., 2018). The liquid-based pheromone McPhail traps are used to collect the DIRT images. Soup images are produced when olive fruit flies are trapped in liquid pheromone. There are 848 images in the dataset. At least one olive fruit fly is shown in every picture in DIRT. Originally, the dataset had three classes: dacus, dacus male, and dacus female. In our situation, we only consider the first class. Figures 5 and 6 depict two samples of the original DIRT dataset and two samples of the augmented dataset, respectively.

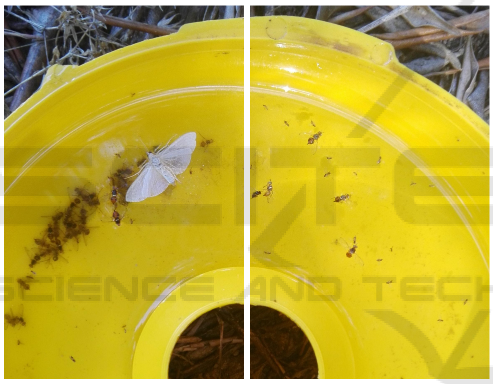


Figure 5: Original images of the DIRT dataset.

We removed several images from the augmented dataset since they are not compatible with our criteria (i.e., the size of insects in photos is bigger than 10% of the entire image size). The augmented dataset has 3877 images.

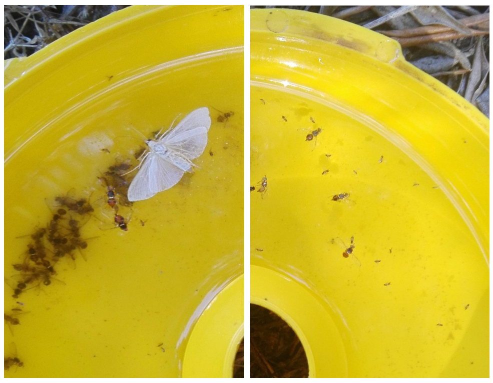


Figure 6: Augmented dataset: examples images.

Deep learning models often require a large amount of training data to generate effective results. Therefore, we applied data augmentation techniques. The original 848 images were separated into three groups: the training set (552 photos), the validation set (170 photos), and the final test set (126 photos). Then, we enhanced all the sets by combining them with zooming (from 10% to 40%), and rotation (from 10% to 60%) on each photo to get extra training and testing examples. After the augmentation, we have around 3877 pictures divided as follows: 2520 images for training, 775 images for validating, and 582 images for testing.

Table 2: Number of images in DIRT dataset.

	Training	**Validation**	**Final Test**	**Total**
DIRT	**552**	**170**	**126**	**848**
After DA	**2520**	**775**	**582**	**3877**

In this paper, we used Google Colab to train each deep learning model for 100 epochs. We adopted the standard hyperparameters, considering our study is not focused on them. The standard hyperparameters are: learning rate (lr) = 0.01; IoU training threshold = 0.2; obj loss gain = 1.0; class loss gain = 0.5.

4.2 Experimental Results

According to (Fawcett, 2006), the evaluation matrix of the object detection matrix is based on: (i) **True Positives (TP):** is the number of detected insects that are detected as target insects; (ii) **False Negatives (FN):** is the number of target insects that are not detected; (iii) **True Negatives (TN)**: is the number of non-target insect that are not detected as target insects; (vi) **False Positives (FP):** is the number of non-target insect that are detected as target insects.

The above criteria need a definition in terms of what constitutes a right detection and a wrongly detection. One frequent method is to use the intersection over union (IOU). In the object detection scope, the IOU divides the area of union between the predicted bounding box B_p and the ground-truth bounding box B_{gt} by the overlapping area between them.

$$IOU = \frac{area(B_p \cup B_{gt})}{area(B_p \cap B_{gt}} \quad (3)$$

We can characterize a detection as correct or wrong by comparing the IOU to a defined threshold t. If $IOU \geq t$, the detection is considered to be correct. If $IOU < t$, the detection is considered mistaken.

The remaining elements of the confusion matrix are as follows: **True Positive Rate (TPR):**(also known as sensitivity) is the measure of correctly detected insect.

$$TPR = \frac{TP}{(TP+FN)} \tag{4}$$

True Negative Rate (TNR): (also known as specificity) is the measure of non insects that were correctly not detected and is expressed as follows:

$$TNR = \frac{TN}{(TN+FP)} \tag{5}$$

Positive Predictive Value (PPV): (or precision) describes how much of the correctly detected labels are truly positive and is expressed as follows:

$$PPV = \frac{TP}{(TP+FP)} \tag{6}$$

Accuracy (ACC): is defined as the number of accurately identified insects or non-insects divided by the total number of insects and is expressed as follows:

$$ACC(\%) = \frac{(TN+TP)}{(TN+TP+FN+FP)} \times 100 \tag{7}$$

4.2.1 Yolov5 Performance

First, we trained the original version of Yolov5 on the DIRT dataset, in order to compare it to our proposal. Table 3 shows obtained results in terms of precision, recall, mAP, and model's size.

Table 3: Original Yolov5: Augmented DIRT Dataset.

Models_Name	Pr%	R %	mAP %	Size (MO)
Yolo_N	82.9	81.4	84.4	3.9
Yolo_S	83.9	83.7	85.2	14.4
Yolo_M	85.1	84.2	85.7	42.3
Yolo_L	86.0	84.6	86.2	92.9
Yolo_X	86.4	85.5	87.7	173.3

As shown in Table 3, the main difference between all versions is their size in comparison to their output performance. The difference in accuracy between versions X and N is 3.5%, but the size difference is over 45 times greater.

4.2.2 Evaluation of InsectDSOT

We focused on selecting lightweight feature extraction models with a limited set of trainable parameters. This is primarily done to support a real-world application by increasing the effectiveness of the feature extraction step and assisting in lowering the amount of data needed for fine-tuning. Deeper networks may also have difficulties during training as a result of issues like vanishing gradients brought on by the increased network depth. When it comes to the employment of backbones, we choose one of two approaches:

- First: remove the previous backbone and replace it with the new backbone. In contrast to our method and the Global Context block, most of the used backbones adopt this strategy.
- Second: replace only a section of the backbone with the new block from the new model. This method is used for both the Global Context and our proposal.

In the case of RepVGG, we use both methodologies in order to compare their outcomes. The confusion matrix is shown in Figure 7.

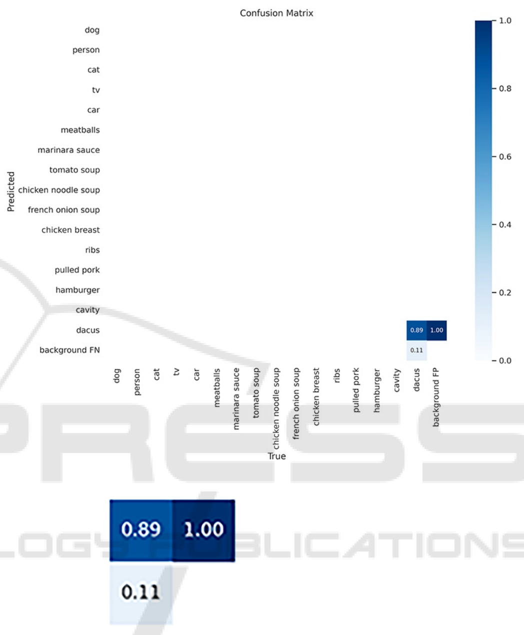


Figure 7: Confusion Matrix of Yolov5 N with TR.

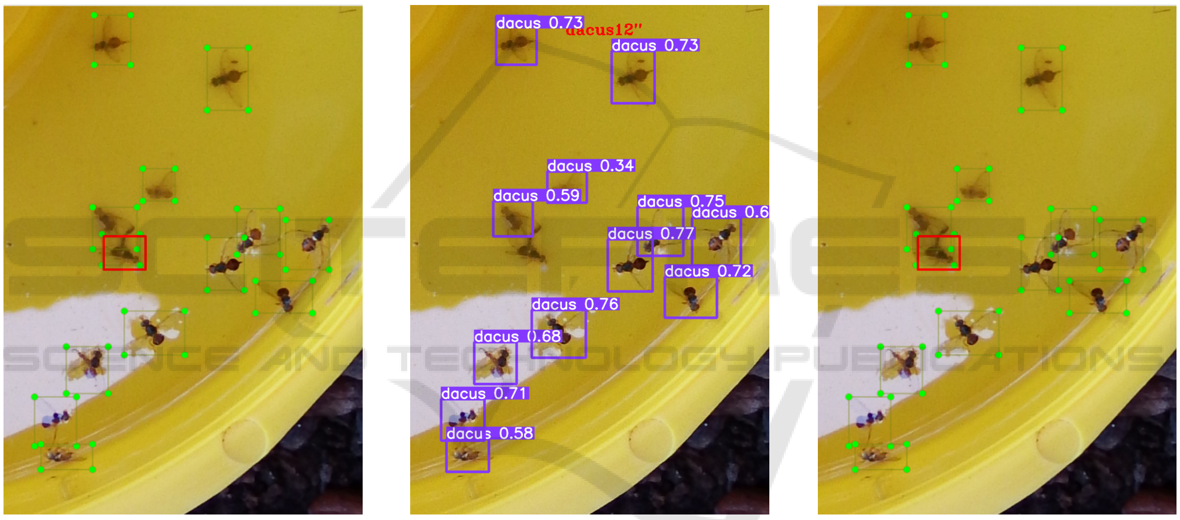
Table 4 displays the outcomes of both approaches.

We now compare our proposal to Yolov5's original versions and other Yolov5 models. We adhere to the principle of transfer learning (TL). In our case, we used the MS COCO (Microsoft Common Object in Context) benchmark dataset (Lin et al., 2014) to train our model and then use the output model as a pre-trained weight for further training. As a disadvantage of using TL, only the precision of the model is raised; the recall of our model is decreased by 1.2%, which seems reasonable when compared to the 2.6% gain in precision. Table 4 demonstrates that not all models perform well on tiny object detection. Some models outperform others in terms of performance. ShuffleNet and his two variants perform poorly in contrast to Yolo5Face, even though his output size, which is less than 1 MB, performs well enough in our dataset.

We included InsectDSOT_S, a larger configuration of our proposal, to outperform all Yolov5 ver-

Table 4: Comparison between used backbones.

Model	Backbone Version	Precision %	Recall %	mAP0.5 %	mAP0.5-0.95 %
MobileNet	V2	79.2	78.7	82.4	31.7
	V3 Small	78.3	77.3	80.1	29.3
	V3 Large	81.4	79.2	84.5	31.1
ShuffleNet	V1	74	72.4	77.3	26.7
	V2	74.5	75.3	78.3	26.8
Yolo5Face	BlazeFace	79.9	82.5	83.8	30.1
	BlazeFPN	77	77.5	78.7	28.8
EfficientNEt	EfficientNEt Lite	79.4	81.3	84.2	32.4
PPLC	PPLC	80.1	78.8	82	30.3
RepVGG	Entire RepVGG	83.6	82.9	86.2	34
GhostNet	GhostNet	82.8	82.5	85.2	31.7
Yolov3	Yolov3	83.7	82.5	86.2	32.6
Global Context	GC Net	80.5	84.1	85.2	30.8
InsectDSOT	RepVGG	83.5	85.1	86.4	33.7
InsectDSOT+TL	RepVGG	86.1	83.9	86.4	33.7



(a) Image in labeling program. (b) Output of our model. (c) Difference between two images.

Figure 8: Detected Image.

Table 5: Our proposal compared with original Yolov5.

Model_Name	Size (Mo)	Precision (%)
Original_N	3.9	82.9
Original_S	14.4	83.3
Original_M	42.3	85.1
Original_L	92.9	86.0
InsectDSOT_N	4.4	86.1
Original_X	173.3	86.4
InsectDSOT_S	15.4	87.0

sions. In terms of accuracy, InsectDSOT_S outperforms Yolov5x by 0.6%. Table 5 displays the size and precision of all models (Yolov5 and our InsectDSOT), sorted from worst to best in terms of precision.

The image in Figure 8 was randomly selected from the test dataset. 13 target insects were identified in the original image by experts in the labeling process. We executed the detection procedure with our proposal. The second image displays the output image, which shows that our model successfully identified 12 out of 13 insects, which is a very good result. The unidentified insect in the previous image has a red rectangle around it to draw attention to it. We built auto-counting into the model's head. As a consequence, the insect's name and the number of detected insects are added on top of the image, in addition to the regular detection.

5 CONCLUSION

We presented InsectDSOT, an improved version of Yolov5 for insect detection. The proposed approach replaces the Conv layers in the original model's backbone and neck with the RepVGG layer. We use transfer learning to improve performance by (i) training our proposal on the MS COCO dataset and then (ii) using the output model of this training as the input of our new training. Our proposal has been validated, using the DIRT (Dacus Image Recognition Toolkit) dataset. The obtained results demonstrate that our approach, based on an InsectDSOT, achieved 86.1% precision. It outperforms four versions of the original yolov5 and yolov5-based versions with modified backbones, based on lightweight models. InsectDSOT achieved good results in terms of precision and recall, when compared to the other approaches.

In the future, we plan to collect a dataset that includes many types of olive insects. Also, we will deploy our model on edge devices. Furthermore, since the classifications approach is included in the most recent Yolov5 release, we plan to include this function into our model, as nature does not only have one type of insect-pest that might damage olive trees.

ACKNOWLEDGEMENTS

This work is supported by the German Academic Exchange Service (DAAD), Germany (Transformation Partnership: OLIVIA Project).

REFERENCES

Bjerge, K., Mann, H. M., and Høye, T. T. (2022). Real-time insect tracking and monitoring with computer vision and deep learning. *Remote Sensing in Ecology and Conservation*, 8(3):315–327.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Cabrera, J. and Villanueva, E. (2022). Investigating generative neural-network models for building pest insect detectors in sticky trap images for the peruvian horticulture. In *Annual International Conference on Information Management and Big Data*, pages 356–369. Springer.

Cardoso, B., Silva, C., Costa, J., and Ribeiro, B. (2022). Internet of things meets computer vision to make an intelligent pest monitoring network. *Applied Sciences*, 12(18):9397.

Carvajal-Yepes, M., Cardwell, K., Nelson, A., Garrett, K. A., Giovani, B., Saunders, D. G., Kamoun, S., Legg, J., Verdier, V., Lessel, J., et al. (2019). A global surveillance system for crop diseases. *Science*, 364(6447):1237–1239.

Chen, Y., Yang, J., Wang, J., Zhou, X., Zou, J., and Li, Y. (2022). An improved yolov5 real-time detection method for aircraft target detection. In *2022 27th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE.

Cui, C., Gao, T., Wei, S., Du, Y., Guo, R., Dong, S., Lu, B., Zhou, Y., Lv, X., Liu, Q., et al. (2021). Pp-lcnet: A lightweight cpu convolutional neural network. *arXiv preprint arXiv:2109.15099*.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*, 29.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. (2021). Repvgg: Making vgg-style convnets great again. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13728–13737.

FAO (2021). Food and agriculture organization of the united nations. https://www.fao.org/news/story/en/item/1402920/icode/. Accessed: 2022-10-25.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Girshick, R. B. (2015). Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448.

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., and Xu, C. (2020). Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1580–1589.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. IEEE.

He, Y., Zeng, H., Fan, Y., Ji, S., and Wu, J. (2019). Application of deep learning in integrated pest management: A real-time system for detection and diagnosis of oilseed rape pests. *Mobile Information Systems*, 2019.

Hou, Z., Sun, Y., Guo, H., Li, J., Ma, S., and Fan, J. (2022). M-yolo: an object detector based on global context information for infrared images. *Journal of Real-Time Image Processing*, pages 1–14.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Jocher, G. R., Stoken, A., and Chaurasia, A. (2021a). ultralytics/yolov5: v4.0 - nn.silu() activations. https://zenodo.org/record/4418161. Accessed: 2022-11-10.

Jocher, G. R., Stoken, A., and Chaurasia, A. (2021b). ultralytics/yolov5: v6.0 - yolov5n 'nano' models. https://zenodo.org/record/5563715. Accessed: 2022-11-10.

Jocher, G. R., Stoken, A., and Chaurasia, A. (2022a). ultralytics/yolov5: v6.1 - tensorrt, tensorflow edge tpu and openvino export and inference — zenodo. https://zenodo.org/record/6222936.Y3t3Q3bP3IU. Accessed: 2022-11-10.

Jocher, G. R., Stoken, A., and Chaurasia, A. (2022b). Yolov5 classification models, apple m1, reproducibility, clearml and deci.ai integrations. https://zenodo.org/record/7002879.Y3t3P3b7TIV. Accessed: 2022-11-10.

Kalamatianos, R., Karydis, I., Doukakis, D., and Avlonitis, M. (2018). Dirt: The dacus image recognition toolkit. *Journal of Imaging*, 4(11):129.

Li, K., Zhu, J., and Li, N. (2021a). Insect detection and counting based on yolov3 model. In *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, pages 1229–1233. IEEE.

Li, K., Zhu, J., and Li, N. (2021b). Lightweight automatic identification and location detection model of farmland pests. *Wireless Communications and Mobile Computing*, 2021.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, Q., Yan, Z., Wang, F., and Ding, C. (2021). Research on object detection algorithm for small object of pests based on yolov3. In *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*, pages 14–18.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Ma, N., Zhang, X., Zheng, H., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, volume 11218, pages 122–138. Springer.

Mamdouh, N. and Khattab, A. (2021). Yolo-based deep learning framework for olive fruit fly detection and counting. *IEEE Access*, 9:84252–84262.

Patel, D. J. and Bhatt, N. (2019). Insect identification among deep learning's meta-architectures using tensorflow. *Int. J. Eng. Adv. Technol*, 9(1):1910–1914.

Qi, D., Tan, W., Yao, Q., and Liu, J. (2021). Yolo5face: Why reinventing a face detector. *ArXiv preprint ArXiv:2105.12931*.

Qian, Y., Miao, Y., Huang, S., Qiao, X., Wang, M., Li, Y., Luo, L., Zhao, X., and Cao, L. (2022). Real-time detection of eichhornia crassipes based on efficient yolov5. *Machines*, 10(9):754.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *ArXiv*, abs/1804.02767.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, pages 4278—-4284. ACM.

Takimoto, H., Sato, Y., Nagano, A. J., Shimizu, K. K., and Kanagawa, A. (2021). Using a two-stage convolutional neural network to rapidly identify tiny herbivorous beetles in the field. *Ecological Informatics*, 66:101466.

Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

Teng, Y., Zhang, J., Dong, S., Zheng, S., and Liu, L. (2022). Msr-rcnn: A multi-class crop pest detection network based on a multi-scale super-resolution feature enhancement module. *Frontiers in Plant Science*, 13.

Tresson, P., Tixier, P., Puech, W., and Carval, D. (2019). Insect interaction analysis based on object detection and cnn. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE.

Veit, A., Wilber, M. J., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29.

Yang, Z., Yang, X., Li, M., and Li, W. (2021). Automated garden-insect recognition using improved lightweight convolution network. *Information Processing in Agriculture*.

Yuan, Z., Fang, W., Zhao, Y., and Sheng, V. S. (2021). Research of insect recognition based on improved yolov5. *Journal of Artificial Intelligence*, 3(4):145.

Zha, M., Qian, W., Yi, W., and Hua, J. (2021). A lightweight yolov4-based forestry pest detection method using coordinate attention and feature fusion. *Entropy*, 23(12):1587.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.