

YOLO-OpenCV: A Lightweight Object Detection Algorithm for Military Soldiers Safety & Weapon Detection

Abstract

Images of military targets typically exhibit characteristics such as camouflage, varying scales, and uneven distribution, making the task of target detection complex and challenging. Furthermore, the limited computing resources of unmanned platforms, such as drones and ground unmanned vehicles, make it difficult to deploy detectors with large parameter counts on these systems. In this study, we enhanced the components of YOLOv8n and proposed a new lightweight military target detection algorithm named YOLO-E. We constructed a military target dataset composed of armed personnel holding various weapons to facilitate the verification of different algorithms. In our designed algorithm, we applied an efficient multi-scale convolution module in the feature extraction network to improve the detection speed of military targets. Additionally, we designed a head network based on weight sharing, significantly reducing the model parameters. We also proposed a novel bounding box loss function, the Normalized Corner Distance IoU, to further enhance the detection accuracy of military targets. We tested YOLO-E on a self-developed military target dataset. Experimental results showed that compared to the original YOLOv8n algorithm, YOLO-E improved detection accuracy by approximately 1.30%, increased detection speed by 1.68%, reduced parameter count by 30.87%, and decreased computational complexity by 37.33%. Furthermore, we compared our method with several advanced object detection algorithms. The results demonstrated that YOLO-E also outperformed in terms of comprehensive parameters, real-time performance, and accuracy. The proposed network model provides effective auxiliary support for analyzing battlefield situations.

Keywords: Object detection, military targets, lightweight, YOLOv8, EMSConv, weight sharing, IoU loss.

1 Introduction

Object detection is a straightforward task for humans, but over a decade ago, enabling computers to perform it was highly challenging. However, with the advent of deep learning, computer vision technology has found widespread application across various fields, including intelligent security, autonomous driving, remote sensing detection, medical and pharmaceutical domains, agriculture, intelligent transportation, and information security. In the military sector, modern warfare is increasingly driven by information technology, making real-time battlefield perception critical for decision-making and deployment. The ability to identify and position military targets is a key technology that enhances battlefield situational awareness. Currently, nations around the world are intensifying the development of these technologies. Therefore, researching military target detection technology in complex environments is vital for effective battlefield analysis.

The intelligent detection of military targets is a crucial aspect of national defense applications. Scholars from various countries have initiated extensive studies in this field. For instance, Putatunda et al. proposed an intelligent detection method for camouflaged targets. Park et al. created a military target dataset and developed a few-shot object detection method capable of identifying objects with a limited number of examples. Jafarzadeh et al. achieved intelligent tank target detection using an improved YOLOv5 method. Zeng et al. proposed a hybrid detection model based on convolutional neural networks and Transformer architecture for low-altitude unmanned aerial vehicles. By incorporating a multi-gradient path network, they captured local feature information from images, thereby enhancing detection accuracy.

Object detection can be divided into two main categories: traditional object detection algorithms and deep learning-based object detection algorithms. Traditional object detection algorithms primarily rely on feature extractors and use sliding windows to generate target candidate regions. Examples include the Viola-Jones object detector, which combines Haar-like features, integral images, Adaboost, and cascaded classifiers, and the HOG detector, which extracts gradients and their orientations from edges to create a feature map. During inference, parts of the object are detected separately, and their possible configurations are marked as detected deformable part models (DPM). Deep learning-based object detection algorithms can be divided into two categories: two-stage and one-stage methods. Two-stage object detection methods, such as R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, FPN, R-FCN and Mask RCNN, first generate sample candidate boxes, then use deep convolutional neural networks to encode the extracted feature vectors, and finally regress the category and position of the target object within the candidate boxes. By adopting a two-stage operation, these algorithms achieve high detection accuracy at the cost of slower speed and non-real-time detection. One-stage object detection algorithms, such as SSD and the YOLO series abandon the stage of generating candidate bounding boxes and directly output the position and category of the target through regression, thereby improving detection speed. The YOLO series is a classic example of single-stage object detection algorithms, representing a significant breakthrough in real-time object detection. However, the training of each component in YOLO must be conducted separately, resulting in slower inference speed. A

strategy for jointly training components has been proposed, improving YOLO's detection performance while increasing inference speed and reducing network training complexity. YOLOv3 utilizes Darknet-53 as the backbone network, fusing sampled feature maps with shallow feature maps to preserve the semantic information of small objects and improve their detection performance. YOLOv4 uses CSPMarket53 as the backbone network and introduces Spatial Pyramid Pooling (SPP) to optimize the receptive field of deep feature maps, thereby further improving detection accuracy. Ultralytics released YOLOv5, which uses CSPNet as its backbone network. The neck component uses a Feature Pyramid Network (FPN) to achieve top-down semantic information transmission while utilizing low-level features with high resolution and high-level features with rich semantic information. Additionally, the algorithm employs Path Aggregation Networks (PAN) for bottom-up localization transmission, helping propagate low-level information to higher levels. YOLOv8 proposes five types based on differences in network structure depth and width - YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. As the model parameters and performance increase sequentially, detection speed gradually decreases. The YOLOv8n/s models have a small backbone feature extraction network and are lightweight; however, their target bounding box regression needs to be more accurate for practical applications. With increased network depth and width, YOLOv8m/l/x models exhibit better detection and recognition performance, though they struggle to meet the real-time detection requirements of embedded devices with limited hardware.

To address this issue, this study proposes a lightweight object detection algorithm called YOLO-E based on the YOLOv8n network. YOLO-E integrates the GhostConv module, EMsConv (or E-EMsConv) module, lightweight shared detection head, and NCDIoU bounding box regression loss function to enhance both the detection accuracy and speed of the model. The main contributions and innovations of the proposed model are summarized as follows:

- **Enhanced Feature Extraction Network:** In the feature extraction network, we applied the GhostConv module to replace the convolution operations in the CBS modules of layers 1, 3, 5, and 7 in the backbone. Additionally, inspired by group convolution, we propose two lightweight convolution modules: the Efficient Multi-Scale Convolution module (EMSC) and the Extended-Efficient Multi-Scale Convolution module (EEMSC). These modules are combined in the C2f module to reduce model parameters and improve detection accuracy;
- **Lightweight Shared Convolutional Detection Head:** A novel Lightweight Shared Convolutional Detection (LSCD) head is introduced, which utilizes Group Normalization instead of the Batch Normalization (BN) layer. By sharing a convolutional module across different detection heads, the model parameters are significantly reduced. For different detection heads handling various scale targets, we use a learnable parameter, Scale, to adjust the features, thereby improving detection accuracy;
- **Novel Loss Function:** A new loss function, Normalized Corner Distance IoU (NCDIoU), for bounding box regression is proposed. This results in higher quality prediction boxes, faster model convergence, and improved localization performance;
- **Military Targets Dataset:** A dataset containing military targets has been created and categorized into three groups based on the weapons held by soldiers: soldiers with RPGs, soldiers with guns, and soldiers without weapons;

The remainder of this paper is organized as follows. Section 2 reviews relevant work on lightweight neural networks and bounding box regression losses. Section 3 provides detailed information on the network model proposed in this article. Section 4 introduces the implementation details of the experiment, and presents the results. Section 4 discusses ablation and comparative experiments. Finally, Section 6 presents our conclusions and outlook.

2 Related Works

2.1 Lightweight neural network

The single-stage detector reduces the model size; however, due to the large number of parameters, it still cannot achieve real-time detection of military images on the platform. Many researchers have proposed lightweight networks to address the challenges posed by large-scale neural network models. In 2017, Howard et al. introduced Depthwise Separable Convolution (DSC) in MobileNetV1, which decomposes standard convolution into depthwise convolution and pointwise convolution, significantly reducing the number of operations and parameters. In 2018, Sandler et al. proposed MobileNetV2, based on MobileNetV1, which added an inverted residual structure and a linear bottleneck to improve model efficiency. In 2019, Howard et al. introduced MobileNetV3, which incorporated an attention mechanism SE module and updated the activation function to achieve greater accuracy and efficiency. Qin et al. proposed MobileNetV4, which introduced the Universal Inverted Bottleneck (UIB) search block and an attention block, Mobile MQA, specifically designed for mobile accelerators. ShuffleNetV1 added group convolution

and channel shuffling modules based on DSC, offering better computational complexity and detection time compared to MobileNetV1. In 2018, ShuffleNetV2 proposed four principles for designing lightweight networks and introduced channel segmentation. The splitting structure replaces the addition operation with a concatenation operation, thereby reducing the number of model parameters.

2.2 Bounding box regression loss function

The bounding box regression loss function is essential in object detection. So far, most bounding box regression loss functions can be divided into two categories: ℓ_n loss functions and Intersection over Union (IoU)-based loss functions. ℓ_1 loss, also known as Mean Absolute Error (MAE), has a constant derivative, which can cause oscillations and convergence issues at the minimum value of the function. On the other hand, ℓ_2 loss, known as Mean Squared Error (MSE), can result in significant derivative variation and instability, especially in the early stages of training when the loss is large, leading to a larger derivative. As the derivative decreases, the training speed also decreases. The Smooth ℓ_1 loss used in Faster R-CNN balances the advantages and disadvantages of both ℓ_1 and ℓ_2 losses: in the early stages, it uses the ℓ_1 gradient to stabilize and converge quickly, and in the later stages, it utilizes ℓ_2 to gradually converge to the optimal solution. However, these loss functions calculate the coordinates separately, treating x, y, w , and h as independent objects. The four parts of the bounding box should be considered as a whole, but they have been treated independently. Moreover, the ℓ_n loss does not exhibit scale invariance and tends to favor larger objects during training. To address these issues, IoU loss [was proposed. However, due to the problem of gradient vanishing in the actual regression process caused by IoU loss, many IoU-based improvements have been proposed, which achieve more accurate and faster regression by adding a penalty term to the IoU loss function. GIoU[introduces the smallest enclosing bounding box of the predicted and ground truth bounding boxes based on IoU. When the two boxes completely overlap, the minimum value is 0. When the edges of the two boxes are tangential, the loss function value is 1. When the two boxes are separated and far apart, the loss function value is 2. DIoU introduces the Euclidean distance between the center point coordinates of the predicted bounding box and the ground truth box. CIoU incorporates the aspect ratio of the two boxes based on DIoU. EIoU calculates the length and width of the true value box and the prediction box separately, based on the penalty term of CIoU, breaking down the influence factor of the aspect ratio to achieve faster convergence. SIoU redefines the angle penalty metric, which allows the prediction box to quickly align to the nearest axis, thereby requiring regression of only one coordinate (X or Y), effectively reducing the total number of degrees of freedom.

3 Methods

3.1 YOLOv8 network structure

In YOLOv8, there are five versions: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. The performance of these models on the MSCOCO dataset is listed in Table 1. YOLOv8n has the fewest parameters and is the fastest among these models in terms of detection speed; however, it has the lowest detection accuracy based on the performance comparison. On the other hand, YOLOv8x achieves the highest detection accuracy but uses the most parameters and has a slower scanning speed. All of these models share the same network structure, but their network depth and feature map width differ, allowing the network model to scale accordingly. Considering the specific characteristics of these models, the improved YOLOv8 is more suitable for deployment on mobile platforms. This study proposes a target detection algorithm suitable for military images that strikes a balance between network speed and detection accuracy. From the perspective of network structure, YOLOv8n can be divided into three parts: the backbone network, the neck network, and the detection head.

Table 1: Performance comparison of different models of YOLOv8 on the COCO dataset.

Model	mAP50-95	Paramters	GFLOPs
YOLOv8n	37.3	3.2	8.7
YOLOv8s	44.9	11.2	28.6
YOLOv8m	50.2	25.9	78.9
YOLOv8l	52.9	43.7	165.2
YOLOv8x	53.9	68.2	257.8

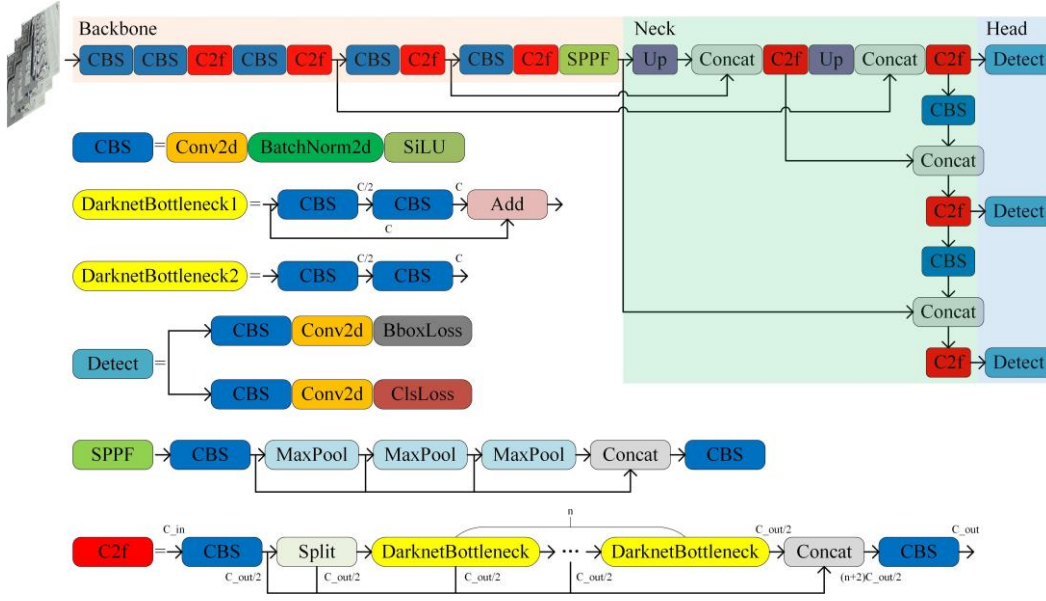


Figure 1: The network structure of YOLOv8 and the detailed construction of each module.

As shown in Figure 1, YOLOv8 employs an improved CSPDarknet53 as the backbone network to downsample the input features five times, sequentially obtaining five features of different scales. The Cross Stage Partial (CSP) module in the original backbone network has been replaced by the C2f module, where "n" in the structure of the C2f module represents the number of bottlenecks. The C2f module adopts a gradient diversion connection, which enriches the information flow of the feature extraction network while maintaining a lightweight design. The CBS module performs convolution on the input information, followed by batch normalization, and finally uses the SiLU activation function to obtain the output result. Ultimately, the backbone network utilizes the Spatial Pyramid Pooling Fast (SPPF) module to pool the input feature map into a fixed-size graph for adaptive output. Compared to the Spatial Pyramid Pooling (SPP) structure, SPPF reduces the computational workload and has lower latency by sequentially connecting three max pooling layers.

Inspired by PANet, YOLOv8 designed a PAN-FPN structure for the neck. Compared with the neck structures of the YOLOv5 and YOLOv7 models, YOLOv8 removes the convolution operation after upsampling in the PAN structure, achieving a lightweight design while maintaining the original performance. The traditional Feature Pyramid Network (FPN) uses a top-down approach to deliver deep semantic information, enhancing the semantic information of features by fusing features of different scales. However, this process can result in the loss of some object positioning information. To alleviate this problem, PAN-FPN adds a Path Aggregation Network (PAN) to the FPN to enhance the learning of location information and achieve path enhancement in a top-down manner. The PAN-FPN constructs a top-down and bottom-up network structure, realizing the complementarity of shallow location information and deep semantic information through feature fusion, thereby achieving feature diversity and completeness.

The head of the YOLOv8 model uses a decoupled structure. The decoupled head structure employs two separate branches for object classification and predicted bounding box regression, using different loss functions for these two types of tasks. Binary Cross-Entropy Loss (BCE Loss) is used for the classification task, while Distribution Focal Loss (DFL) and Complete Intersection over Union (CIoU) loss are used for the predicted bounding box regression task. This detection structure can improve detection accuracy and accelerate model convergence.

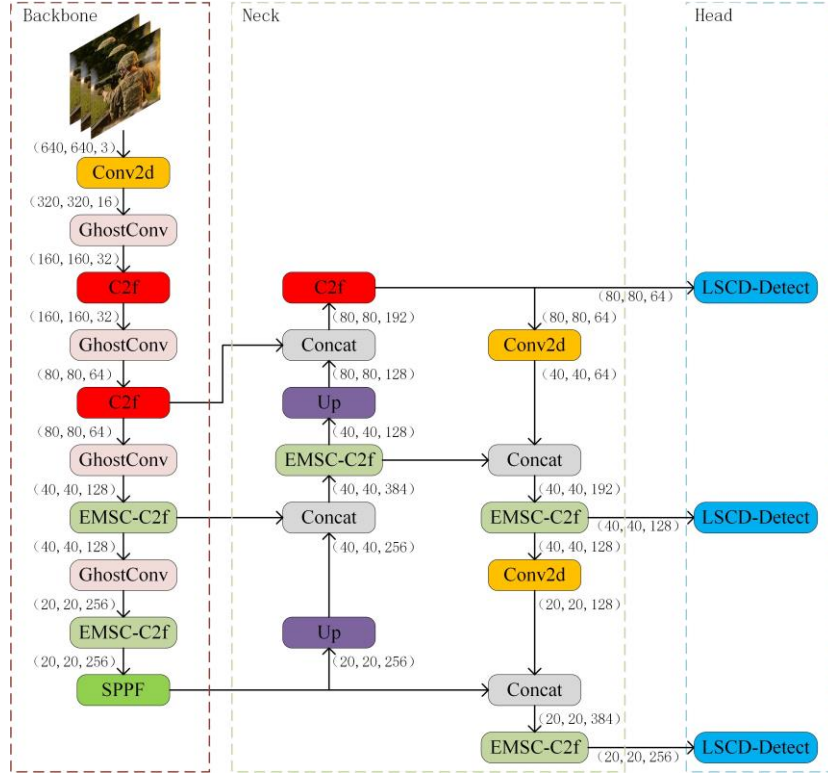


Figure 2: YOLO-E structure. The numbers in brackets represent the length, width, and number of channels of the output feature map of each module.

3.2 YOLO-E network structure

The backbone network in YOLOv8n consists of CBS modules and C2f modules, which effectively extract features from images but introduce a large number of model parameters, resulting in an increased computational workload. Therefore, the improved backbone incorporates GhostConv and EMSC-C2f modules. The GhostConv and EMSC-C2f modules effectively reduce the number of parameters in the model, thereby reducing its FLOPs. Although the number of model parameters in the backbone network has decreased, a significant number of parameters remain in the feature fusion process of the neck network, which affects the speed of feature fusion. Considering the characteristics of feature fusion, we also incorporated the EMSC-C2f module into the neck network to reduce the number of parameters. In the detection head of the network, we used a lightweight shared convolutional detection (LSCD) head instead of the original detection head, reducing the parameters and FLOPs. Additionally, we used the proposed NCDIoU loss function instead of the CIoU loss function in the head network to improve the accuracy of the bounding box regression. The resulting YOLO-E network structure is shown in Figure 2, and the main modules and parameters are summarized in Table 2.

3.2.1 Ghost convolution

Currently, most convolution operations involve point-wise convolution for dimensionality reduction, followed by depth-wise convolution for feature extraction, as shown in Figure 3. To extract more feature information from the input image data, neural network models trained with vanilla convolution (CNN) often produce redundant feature maps after training. While this approach can achieve better performance, it requires many convolutional layer calculations, which increases the consumption of computing resources and memory access. The purpose of some lightweight networks is to utilize the redundancy of features and achieve a lighter model by eliminating some redundant features. GhostNet combines a linear operation with ordinary convolution. Some redundant feature maps can be linearly transformed from the generated ordinary convolution feature maps to obtain similar feature maps, thus achieving a high-dimensional convolution effect while reducing model parameters and computational workload.

We make the following assumptions regarding the input and output feature maps, biases, and convolution kernels:

1. The channel, height and width of the input feature map are C_{in} , W_{in} , H_{in} ;

2. The channel, height and width of the output feature map generated by ordinary convolution are C_{out} , W_{out} , H_{out} ;

Table 2: Main modules, parameters and FLOPs of the YOLO-E.

Number	Module	Output	Repeat Times	Parameters	FLOPs
	Input	$3 \times 640 \times 640$			
0	CBS	$16 \times 320 \times 320$	1	464	88.474M
1	GhostConv	$32 \times 160 \times 160$	1	2768	138.445M
2	C2f	$32 \times 160 \times 160$	1	7360	367.002M
3	GhostConv	$64 \times 80 \times 80$	1	10144	128.205M
4	C2f	$64 \times 80 \times 80$	2	49644	629.146M
5	GhostConv	$128 \times 40 \times 40$	1	38720	123.085M
6	EMSC-C2f	$128 \times 40 \times 40$	2	149632	475.136M
7	GhostConv	$256 \times 20 \times 20$	1	151168	120.525M
8	EMSC-C2f	$256 \times 20 \times 20$	1	364160	289.997M
9	SPPF	$256 \times 20 \times 20$	1	164608	131.072M
10	Upsample	$256 \times 40 \times 40$	1	0	819.200K
11	Concat	$384 \times 40 \times 40$	1	0	0
12	EMSC-C2f	$128 \times 40 \times 40$	1	124224	394.854M
13	Upsample	$128 \times 80 \times 80$	1	0	1.638M
14	Concat	$192 \times 80 \times 80$	1	0	0
15	C2f	$64 \times 80 \times 80$	1	37248	471.859M
16	CBS	$64 \times 40 \times 40$	1	36992	117.965M
17	Concat	$192 \times 40 \times 40$	1	0	0
18	EMSC-C2f	$128 \times 40 \times 40$	1	99648	316.211M
19	CBS	$128 \times 20 \times 20$	1	147712	117.965M
20	Concat	$384 \times 20 \times 20$	1	0	0
21	EMSC-C2f	$256 \times 20 \times 20$	1	396928	316.211M
22	LSCD-Detect	3	1	107414	1.403G
	Total			1888854	5.632G

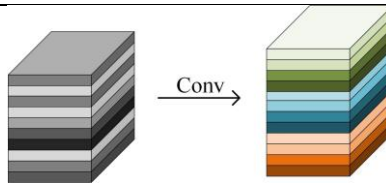


Figure 3: Standard convolution module.

$$\begin{aligned}
r_s &= \frac{C_{out} \cdot W_{out} \cdot H_{out} \cdot C_{in} \cdot K_w \cdot K_h}{\frac{C_{out}}{s} \cdot W_{out} \cdot H_{out} \cdot C_{in} \cdot K_w \cdot K_h + \frac{C_{out}}{s} \cdot W_{out} \cdot (s-1) \cdot W_{out} \cdot H_{out} \cdot D_w \cdot D_h} \\
&= \frac{C_{out} \cdot K_w \cdot K_h}{\frac{1}{s} \cdot C_{out} \cdot K_w \cdot K_h + \frac{s-1}{s} \cdot D_w \cdot D_h} \approx \frac{s \cdot C_{out}}{s + C_{out} - 1} \approx s
\end{aligned} \tag{1}$$

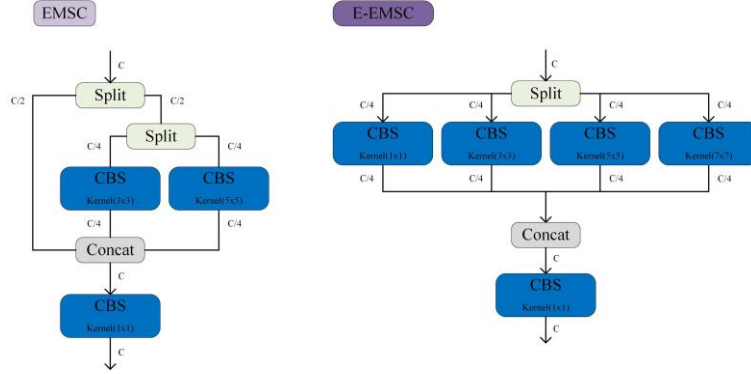


Figure 6: The structure of Efficient Multi-Scale Convolution(EMSC) and Extended-EMSC modules.

3.2.2 Efficient Multi-Scale Convolution (EMSC) and Extended-EMSC

Traditional convolution uses a single-scale convolution kernel, which may not effectively capture features of objects or scenes at different scales. Multi-scale convolution, on the other hand, uses multiple-scale convolution kernels to better capture features at different scales and extract richer feature information. Taking the Feature Pyramid Network (FPN) as an example, it performs two processes: bottom-up and top-down, fusing features obtained at different scales through horizontal connections. Inspired by this idea, we aim to achieve multiscale information fusion within a convolutional module. To this end, we proposed the Efficient Multi-Scale Convolution (EMSC) module and the Extended Efficient Multi-Scale Convolution (E-EMSC) module.

In the EMSC module, the input obtained from convolution is divided into two groups based on the number of channels. The first group divides the original image into "cheap" features and unprocessed features. The second group further divides the unprocessed features into two subsets and performs convolution operations using 3×3 and 5×5 convolution kernels, respectively. The results are then concatenated with the cheap features obtained from the first group. The concatenated features remain independent of each other within each channel. To achieve point-to-point convolution, a 1×1 convolution is added after concatenating the outputs to complete the fusion of the channel information. This approach also allows for latitude elevation and dimensionality reductions. Similarly, to fuse multi-scale features into a single module while maximizing efficiency, we developed the E-EMSC module. In this module, we grouped the cheap features and obtained two sets of features through 1×1 and 7×7 convolutions, enabling more detailed feature extraction. This method retains multi-scale feature information and reduces the number of parameters compared to traditional convolution methods. As shown in Figure 6, we follow the parameter assumptions stated in Section 3.2.1. Based on this, it is assumed that the convolution kernel size used in traditional convolutions is 3×3 . This balanced approach allows for effective feature extraction while maintaining computational efficiency.

In the EMSC module, the ratio of the FLOPs of the standard convolution to it is given by (2).

$$r_{EMSC} = \frac{C_{out} \cdot W_{out} \cdot H_{out} \cdot C_{in} \cdot 3 \cdot 3}{W_{out} \cdot H_{out} \cdot \left(\frac{C_{out}}{4} \cdot \frac{C_{in}}{4} \cdot (3 \cdot 3 + 5 \cdot 5) + C_{out} \cdot C_{in} \cdot 1 \cdot 1 \right)} = \frac{72}{25} \approx 3 \tag{2}$$

In the E-EMSC module, the ratio of the FLOPs of the standard convolution to it is given by (3).

From the results, it can be seen that the FLOPs of the above two modules are $\frac{1}{3}$ and $\frac{2}{3}$ of the standard convolution. Therefore, the standard convolution in the C2f module of YOLOv8n can be replaced, and the replaced bottleneck and C2f are shown in Figure 7 and Figure 8, respectively. The use of these two modules not only reduces the number of parameters in the network model but also fully extracts feature information of different scales, effectively improving the accuracy of the network model in object detection.

$$r_{E-EMSC} = \frac{C_{out} \cdot W_{out} \cdot H_{out} \cdot C_{in} \cdot 3 \cdot 3}{W_{out} \cdot H_{out} \cdot \left(\frac{C_{out}}{4} \cdot \frac{C_{in}}{4} \cdot (1 \cdot 1 + 3 \cdot 3 + 5 \cdot 5 + 7 \cdot 7) + C_{out} \cdot C_{in} \cdot 1 \cdot 1 \right)} = \frac{72}{50} \approx \frac{3}{2} \tag{3}$$

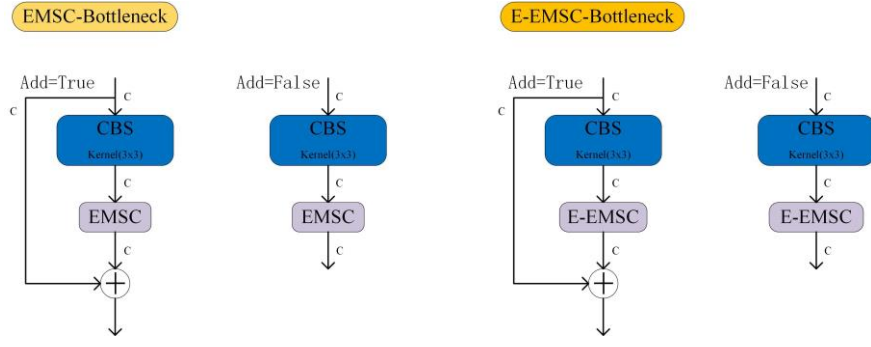


Figure 7: The structure of Efficient Multi-Scale Convolution(EMSC)-Bottleneck and Extended-EMSC Bottleneck modules.

3.2.3 Lightweight Shared Convolutional Detection (LSCD) head

The detection head of YOLOv8 utilizes a decoupling structure, where two independent branches are employed for target classification and bounding box regression. While this structure improves the accuracy of target detection, applying such operations to outputs at large, medium, and small scales results in a significant number of parameters and increased computational complexity. To maintain accuracy while reducing parameters and GFLOPs, we have designed a lightweight shared convolutional detection head.

As illustrated in Figure 9, we first replace the traditional CBS module with the ConvGN module to unify the number of channels for targets from different layers and scales. Unlike the Batch Normalization layer, Group Normalization divides the channels into several groups and calculates the mean and variance within each group. GN does not depend on the batch size, as it is calculated independently on each sample. This makes GN more stable when the batch size is small or variable, and it has been shown to improve the performance of detection head localization and classification. Next, we introduce a weight contribution module to convolve the outputs of different layers. This module significantly reduces the number of parameters, making the model more lightweight, particularly suitable for resource-limited devices. To address the issue of inconsistent target scales detected by each detection head, learnable parameters are used to scale the features appropriately.

Through these optimizations, our designed detection head achieves a reduction in parameters and computational requirements while minimizing accuracy loss as much as possible. This design ensures that the model remains efficient and effective, even when deployed on devices with limited computational resources.

3.2.4 Normalized Corner Distance IoU (NCDIoU) loss function

The loss function of YOLOv8 mainly consists of confidence loss, classification loss, and bounding box regression loss. In the original YOLOv8, regression prediction of the bounding box by using the CIoU function is defined as follows:

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(B^{gt}, B^{pred})}{\rho_C^2} + \alpha V \quad (4)$$

$$V = \frac{4}{\pi^2} \left(\arctan \frac{w_{gt}}{h_{gt}} - \arctan \frac{w_{pred}}{h_{pred}} \right)^2 \quad (5)$$

$$\alpha = \begin{cases} 0, & \text{if } IoU < 0.5, \\ \frac{V}{(1-IoU)+V}, & \text{if } IoU \geq 0.5. \end{cases} \quad (6)$$

where B^{gt} and B^{pred} denote the ground truth and predicted bounding boxes, respectively. $\rho(B^{gt}, B^{pred})$ represent the Euclidean distance between B^{gt} and B^{pred} , and ρ_C represents the diagonal length of the smallest enclosing bounding box of the predicted box and ground truth bounding box. α denotes the trade-off parameter and V is used to measure the consistency of the aspect ratio between B^{gt} and B^{pred} . w_{gt} and h_{gt} represent the

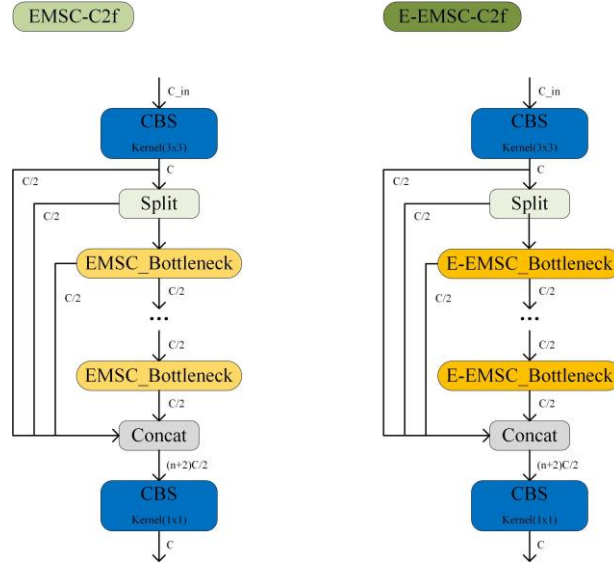


Figure 8: The structure of Efficient Multi-Scale Convolution(EMSC)-C2f model and Extended-EMSC-C2f modules.

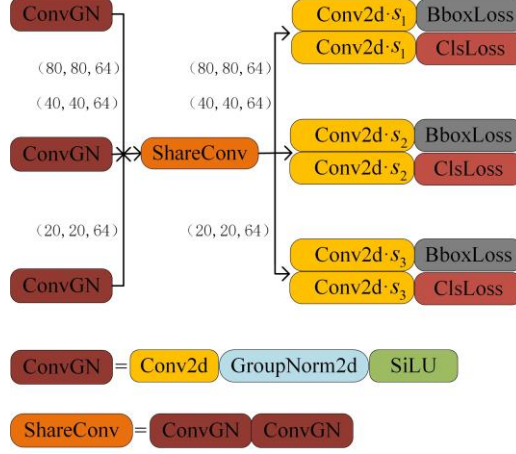


Figure 9: Lightweight shared convolutional detection head.

width and height of the ground truth box, respectively. w_{pred} and h_{pred} represent the width and height of the predicted box, respectively.

In the penalty term of the Clou loss function, it is considered that, on the one hand, the center point of the predicted bounding box and the ground truth bounding box should be approached as soon as possible. On the other hand, the aspect ratio of the two boxes should also tend to be consistent during this process. In order to make the predicted bounding box converge to the ground truth bounding box accurately, it can be achieved by minimizing the distance between the two points corresponding to the diagonals of the two boxes, which can be established:

$$\begin{aligned} d_1 &= \sqrt{(x_1^{pred} - x_1^{gt})^2 + (y_1^{pred} - y_1^{gt})^2} \\ d_2 &= \sqrt{(x_2^{pred} - x_2^{gt})^2 + (y_2^{pred} - y_2^{gt})^2} \end{aligned} \quad (7)$$

$$\mathcal{L}_{NCIoU} = 1 - IoU + \frac{d_1 + d_2}{\rho_C} \quad (8)$$

where d_1 and d_2 represent the distances between the top left and bottom right corner points of the predicted bounding box and the ground truth bounding box, respectively. Owing to the significant difference in pixel difference between large-scale and small-scale targets in the image, simply adding these two distances as a penalty term will make the loss focus more on the large target and ignore the small target. We standardize this

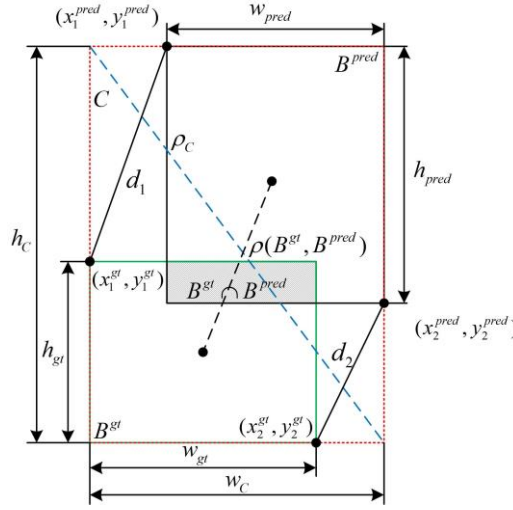


Figure 10: The relationship between ground truth bounding box and predicted bounding box, and the relevant parameters required to calculate the CIoU and NCDIoU loss.

distance by comparing it with the length of the diagonal of box C . From this, we can also derive the range of values for this penalty term. When two boxes overlap, the lengths of d_1 and d_2 are 0; therefore, the value of the penalty term is 0. When the two boxes are far apart, the lengths of d_1 and d_2 are approximately equal to ρ_C , and the value of the penalty term is 2.

3.3 Military dataset

After conducting extensive research, we found that there are very few public datasets available for military targets, particularly those involving soldiers. To facilitate the study of intelligent detection of military targets, we created a soldier target monitoring dataset using images sourced from internet search engines, war-themed movies, and simulations of soldiers in military uniforms. The LabelImage tool was utilized to annotate these targets. In modern warfare, soldiers operate in complex environments such as forests, deserts, and urban areas. Factors like weather, lighting, and angles can interfere with the assessment of soldiers carrying weapons, making data annotation and detection particularly challenging. To address these difficulties, we categorized the images into three classes: S for soldiers without weapons, G for soldiers with guns, and R for soldiers with rocket launchers. As shown in Figure 11, we collected a total of 7,347 images. The dataset was divided into training, test, and validation sets in a 7:2:1 ratio. The distribution of the dataset is depicted in Figure 12.

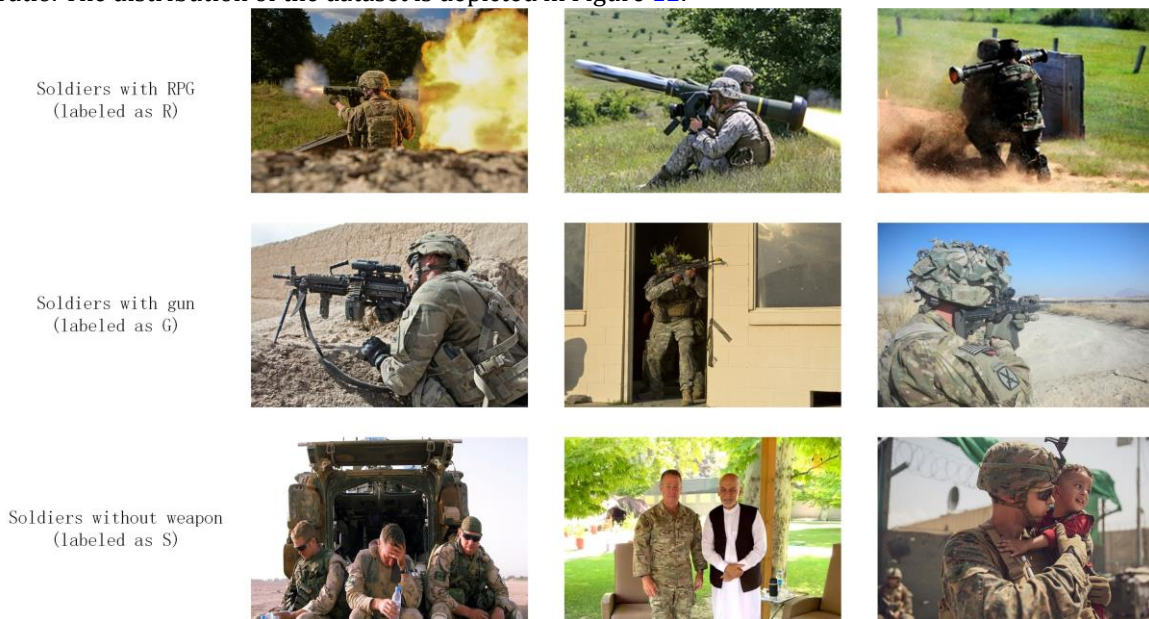


Figure 11: Examples for all categories of soldiers in the dataset.

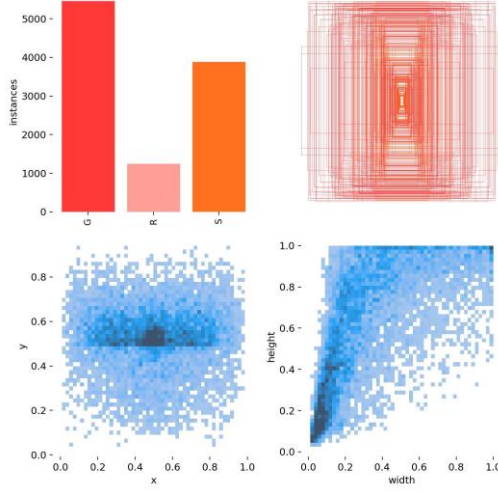


Figure 12: The distribution of the dataset, the upper left represents the number distribution of different types of targets, the upper right represents the distribution of bounding boxes, the lower left represents the distribution of center points of bounding boxes, and the lower right represents the distribution of bounding box sizes.

4 Experiment Studies

4.1 Training Protocol

All the experiments in this work are carried out on a workstation running the Ubuntu20.04 operating system. The graphics processor is GeForce RTX 4090, and the memory is 32 GB. The neural network takes Pytorch2.1.2 as the basic framework and is realized by Python programming. In this paper, all the model training batchsize is set to 32, the input image size is set to 640×640, and epoch is set to 200. The training, test, and validation data are the military target dataset constructed in this work.

4.2 Evaluation Metrics

The Mean Average Precision (mAP) serves as a crucial metric in gauging the accuracy of object detection methodologies, by plotting a curve against accuracy and recall, and the area of the curve against the coordinate axis is the AP value. Precision, Recall, AP, and mAP are calculated as follows:

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

$$AP = \int_0^1 P(r)dr \quad (11)$$

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i \quad (12)$$

where True Positive (TP) signifies the count of positive samples accurately classified, False Positive (FP) denotes the count of negative samples erroneously identified as positive, and False Negative (FN) indicates the count of positive samples inaccurately categorized as negative. The variable n represents the total number of detection categories. The mAP, serving as the average of AP values across all categories, provides a holistic assessment of model performance across diverse object classes.

4.3 Experimental Results

To verify the effectiveness of the YOLO-E model in military target detection, we conducted three sets of experiments using the established dataset:

1. YOLO-E Ablation Experiments: The proposed model has undergone numerous optimizations and improvements. To evaluate the contributions of various strategies to detection performance and assess the impact of the improved modules, we conducted ablation experiments on the proposed algorithm.;
2. Comparative Experiments Using Different Loss Functions: These experiments aimed to compare the detection performance between the proposed loss function and mainstream loss functions within military target datasets;
3. Comparative Experiments Between YOLO-E and Mainstream Object Detection Algorithms: To further verify the performance of our proposed algorithm on military target datasets, we selected several advanced object detection algorithms for comparative testing.

These experiments provide a comprehensive evaluation of the YOLO-E model's effectiveness and its enhancements in the context of military target detection.

4.3.1 YOLO-E Ablation Experiments

Table 3: Ablation experiments of the YOLO-E.

	YOLOv8n		M1	M2	M3	M4	M5	M6	M7	M8
GhostConv			✓					✓	✓	✓
EMSCConv				✓				✓	✓	✓
E-EMSCConv					✓					
LSCD						✓			✓	✓
NCDIoU							✓			✓
Layers	168	180	204	228	157	168	216	205	205	
Parameters	3006233	2816393	2717721	2871577	2362518	3006233	2527881	1884166	1884166	
P	0.815	0.846	0.848	0.851	0.831	0.838	0.814	0.805	0.868	
R	0.807	0.778	0.79	0.793	0.789	0.809	0.822	0.811	0.779	
mAP50	0.858	0.862	0.866	0.867	0.857	0.874	0.877	0.864	0.871	
mAP50-95	0.676	0.676	0.679	0.676	0.668	0.68	0.681	0.670	0.674	
GFLOPs	8.1	7.7	7.6	7.9	6.5	8.1	7.2	5.6	5.6	
FPS	2555	2458	2491	2398	2755	2552	2415	2592	2598	

As shown in Table 3. The YOLO-E network proposed in this work introduces the GhostConv convolution module and our proposed EMSC and E-EMSC modules into the feature extraction network. At the head of the network, we replace the original decoupled detection head with the proposed LSCD module and redesign a new loss function. To verify the effectiveness of each improvement and optimize the mutual influence between modules, we conduct ablation experiments on a self-built dataset to complete the evaluation. To ensure the fairness of the experiment, we set the same parameters for each variable in the training process. To ensure the accuracy of the obtained FPS, we set the warmup parameter to 200 when testing the inference time. We set the single validation of the epoch to 200. Set batchsize to 16. In M1, we only replaced the convolutions in the CBS modules of the first, third, fifth, and seventh layers in the backbone network with the improved GhostConv module, resulting in a parameter reduction of 6.33% in the replaced network. In terms of computational complexity, it decreased by 4.94% and improved by 0.4% in the mAP50 indicator. In M2 and M3, we applied the EMSC and E-EMSC modules to the feature extraction network, respectively, resulting in a reduction of 9.60% and 4.5% in parameter count and 6.2% and 2.5% in computational complexity. And there is an improvement of 0.8% and 0.9% in accuracy. However, all three improvements mentioned above show a slight decrease in FPS parameters. In M4, we tested the application of our self-developed LSCD detection head in the head network, which reduced the number of parameters by 21.41% and the computational cost by 19.76%. FPS increased by 7.8%, while the result of mAP50 only decreased by 0.1%. Next, we tested the proposed loss function on the M5 model and achieved a 1.6% improvement in mAP50 results,

effectively improving the detection accuracy of the network with minimal FPS variation. In the M6, the combination of the GhostConv module and EMSC module resulted in a further reduction in the number of parameters and computational complexity compared to M1 and M2 and an increase of 1.9% in the mAP50 result. Subsequently, we applied the LSCD detection head to test the M7 on this basis. Compared with the baseline, the parameter count was reduced by 37.33%, the computational complexity was reduced by 30.87%, the mAP50 test results were improved by 0.6%, and the Table 4: Comparison of different loss functions

Loss Function	P	R	mAP50	mAP50-95
CIOU	0.805	0.811	0.864	0.670
DIOU	0.851	0.769	0.86	0.664
EIOU	0.832	0.784	0.861	0.667
SIOU	0.814	0.798	0.854	0.668
GIoU	0.82	0.813	0.86	0.668
NCDIoU(Ours)	0.868	0.779	0.871	0.674

FPS was also improved by 1.4%. Finally, we comprehensively tested all optimization modules on the M8, and the accuracy was improved by 0.6% on the basis of the M7 model.

In order to visually demonstrate the detection capabilities of different modules, we randomly selected 4 images from the dataset, and the detection results are shown in Figure 13. Compared with the baseline, our proposed YOLO-E detection bounding box is closer to the ground truth in the first image. In the second image, we successfully detected the soldier without a handheld weapon. We detected all targets in the third image and also detected targets obscured by flags in the fourth image. Overall, our proposed algorithm achieved better detection results.

4.3.2 Comparison of different loss functions

As mentioned earlier, since the IoU loss function was proposed, many improved loss functions have been based on IoU loss. We conducted comparative experiments between our proposed NCDIoU loss function and the CIoU loss used in the baseline, as well as GIoU, DIoU, EIou, and SIou losses. We deployed the loss function in the improved network YOLO-E and replaced NCDIoU with different loss functions. Without changing other parameters and training conditions, the results obtained are shown in Table 4. NCDIoU performed better than other loss functions in mAP and exceeded the second place by 0.7% in mAP50 results. Exceeding the second place by 0.4% in the parameters of mAP50-95.

In order to more intuitively demonstrate the detection results of different loss functions, we randomly selected four images from the dataset, and the detection results are shown in Figure 14. The GIoU and EIou models mistakenly detected soldiers holding guns and RPGs in the first image, while the GIoU, DIoU, and EIou models missed the target in the second image. In the third image detection of Baseline, redundant boxes appeared, while in the fourth image detection, the GIoU model mistakenly detected the rightmost soldier as a soldier holding an RPG. The above results indicate that using NCDIoU as the detection network's bounding box regression loss function achieves higher detection accuracy.

4.3.3 Comparison of different models

To further confirm the overall detection performance of the proposed algorithm in military target datasets, YOLO-E was compared with six advanced object detection algorithms, including the original YOLOv8n, YOLOv6n, YOLOv5n, YOLOv3Tiny, Replacing the YOLOv8n backbone network with the Mobilenetv4 model, as well as RT-DETR, ensures that the training environment and dataset distribution of the four algorithms are completely consistent. In order to illustrate the performance of each algorithm in the dataset more intuitively, we randomly selected four images from the dataset, and the detection results are shown in Figure 15. Figure 16 shows the mAP curve generated by the above algorithm after training in the dataset. It can be seen that our proposed YOLO-E has more obvious advantages and higher accuracy. The results obtained in the validation set are shown in Table 5. Although our proposed model performs slightly worse than YOLOv5n and YOLOv6n in FPS, it reduces the number of parameters by 55.4%, reduces computational complexity by 52.6%, and improves accuracy by 1.9% compared to YOLOv6n. Compared with YOLOv5n, our proposed algorithm reduces parameter count by 24.74%, computational complexity by 21.13%, and accuracy by 2.9%. In addition, compared with the RT-DETR with the largest parameter count, our algorithm reduces parameter count by 90.53%, computational complexity by 90.16%, accuracy by 3.7%, and FPS index by 11.34 times. In the detection of the second image, YOLOv8n and YOLOv3Tiny incorrectly identified the category of the leftmost target in the image. In the detection of the third image, YOLOv8n, YOLOv6n, and RT-DETR mistakenly detected the rock on the left side of the image. YOLOv5n and Mobilenetv4 generate redundant boxes to detect other targets in the image. In the fourth image, YOLOv8n

generated missed and false detection, while YOLOv3Tiny also has cases of target classification errors, and redundant boxes appear in the detection results of Mobilenetv4. In summary, our proposed detection algorithm performs better overall than other algorithms.

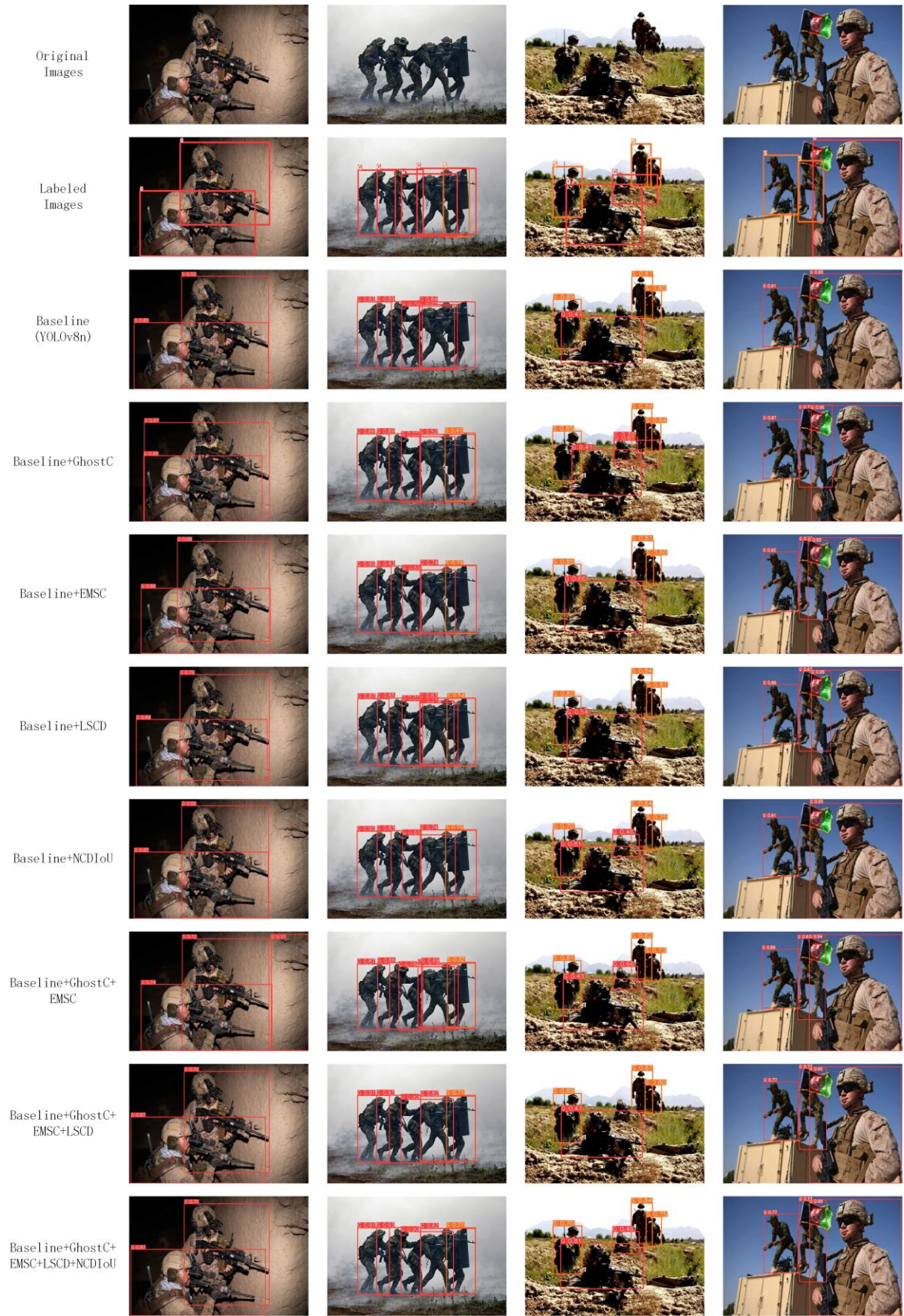


Figure 13: Ablation experiment of the YOLO-E.

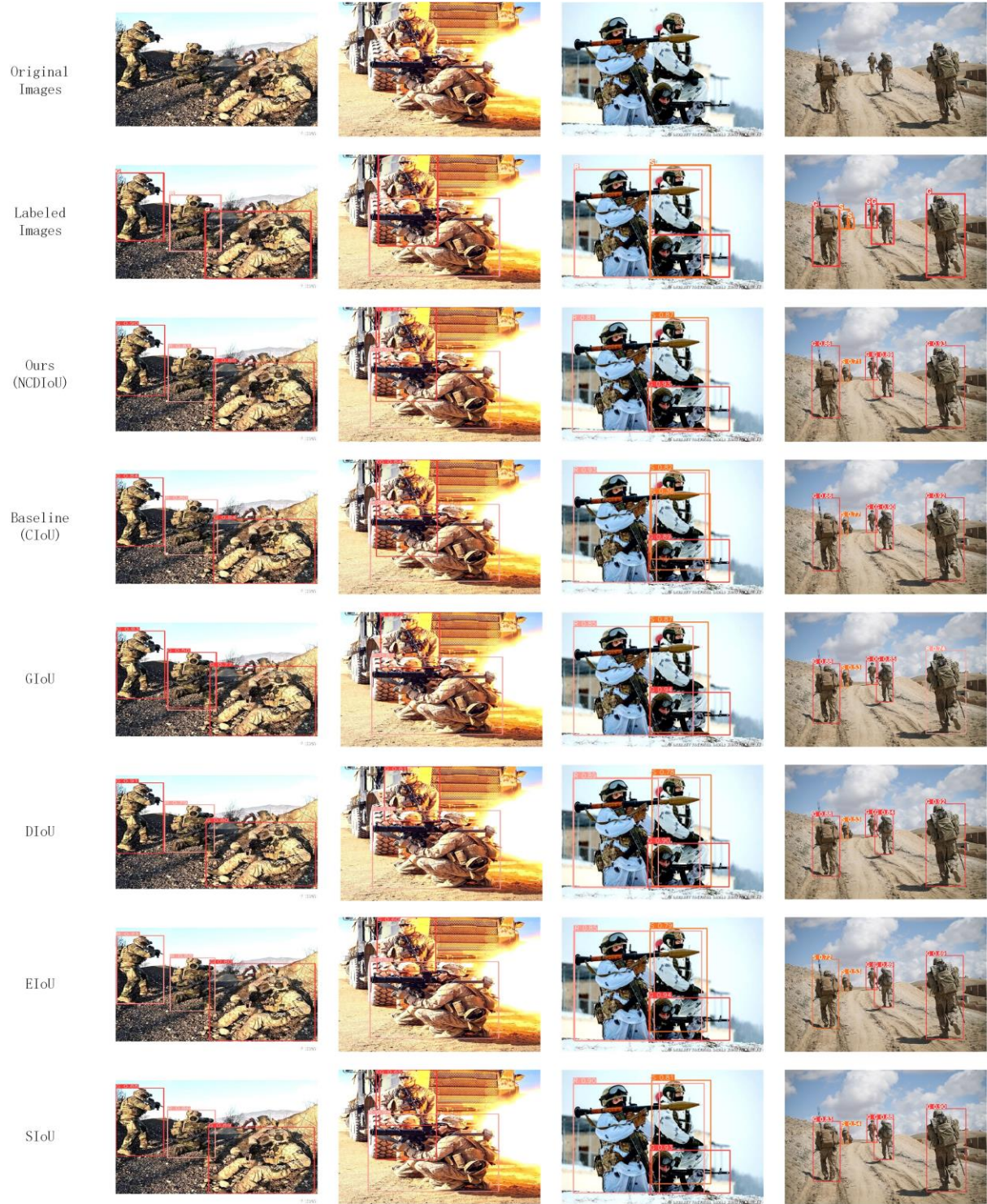


Figure 14: Comparison of different loss functions.

Table 5: Comparison of different models

Model	Layers	Parameters	P	R	mAP50	mAP50-95	GFLOPs	FPS
YOLOv6n	142	4234041	0.82	0.813	0.852	0.674	11.8	2720
YOLOv5n	193	2503529	0.853	0.753	0.842	0.66	7.1	2724
YOLOv3tiny	63	12129206	0.801	0.794	0.833	0.6	18.9	1468
MobileNetv4	292	5700473	0.85	0.783	0.853	0.662	22.5	1799





































RT-DETR	299	19875612	0.881	0.815	0.834	0.653	56.9	229
YOLO-E(Ours)	205	1884166	0.868	0.779	0.871	0.674	5.6	2598
Original Images								
Labeled Images								
Ours (YOLO-E)								
Baseline (YOLOv8n)								
YOLOv6n								
YOLOv5n								
YOLOv3Tiny								
Mobilenetv4								
RT-DETR								

Figure 15: Comparison of different models.

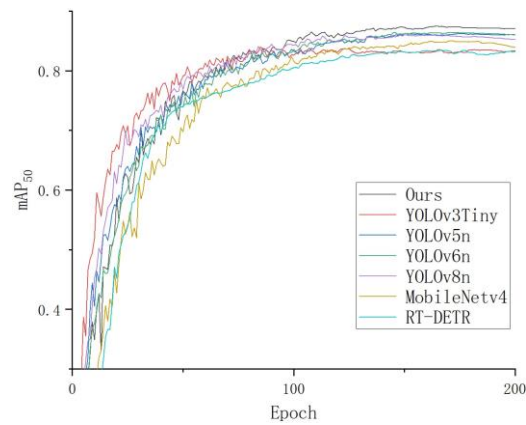


Figure 16: mAP curves of each model.

5 Code

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load Haar cascade for upper body (simulate soldier detection)
upper_body_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_upperbody.xml')

# Load the image
image = cv2.imread("C:/Users/Admin/Downloads/SOLD.jpg")

if image is None:
    print("Error loading image!")
    exit()

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect upper bodies (simulating soldiers)
soldiers = upper_body_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

# Simulate weapon detection using contour detection
def detect_weapon(contour):
    area = cv2.contourArea(contour)

    if 1000 < area < 10000: # Adjusted area range
        return True

    return False
```

```

# Convert image to binary for contour-based shape detection
_, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Loop through detected "soldiers"
for (x, y, w, h) in soldiers:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2) # Draw rectangle around soldier
    label = "Unarmed"

# Debugging: Print soldier positions
print(f"Soldier detected at: x={x}, y={y}, w={w}, h={h}")

# Search for weapon-like shapes in ROI
for cnt in contours:
    cx, cy, cw, ch = cv2.boundingRect(cnt)
    if x < cx < x+w and y < cy < y+h:
        if detect_weapon(cnt):
            label = "Armed"
            cv2.rectangle(image, (cx, cy), (cx+cw, cy+ch), (0, 0, 255), 2)
            break

    cv2.putText(image, label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                (255, 0, 0) if label == "Unarmed" else (0, 0, 255), 2)

# Save result
cv2.imwrite("output_opencv_basic.jpg", image)

# Show result using matplotlib (this works without GUI support)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert BGR to RGB
plt.imshow(image_rgb)
plt.axis('off') # Hide axes
plt.show()

```

6 Output



7 Conclusions

In this work, a multi-scene military target data set is constructed for the military target detection task in complex scenes, and an improved target detection method based on YOLOv8n is proposed. In this method, the GhostConv module and EMSC module are introduced into the feature extraction network, and the proposed LSCD detection head is used to replace the original detection head in the head network, thus reducing the network parameters and computation, the improved NCDIOU loss function is used to improve the detection accuracy of the algorithm. According to the ablation experiment and the comparison experiment between the algorithms, YOLO-E has good detection performance for military targets such as armed personnel and can meet the detection requirements of complex battlefield scenes.

Due to the disciplinary background, the data set created in this paper is only for soldier categories. In the future, we will add military targets such as tanks and armored vehicles to optimize our data set. We will also try to improve the new algorithm with our designed components to improve the detection of military targets further. Finally, we will attempt to deploy our algorithm on an embedded platform to verify the model's performance on a resource-limited device. THE PROJECT IS DONE BY -

S.SHYAM SHANKAR - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)

G.SIVA KUMAR - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)

SB.GOKUL - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)

P.DHANUSH - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)