# A Generative Adversarial Network-Based Method for Intelligent Detection of Military Targets with Few Training Samples for Embedded Edge Devices

**Xin Li, Xiaoli Tang, Hanyu Shi**
School of Electronic and Optical Engineering, Nanjing University of Science and Technology Zijin College, Nanjing 210023, China
Corresponding author: Xin Li (e-mail: kdlixin@163.com).

**ABSTRACT** With the development of artificial intelligence and intelligent algorithms, new technical methods of automatic military target recognition in complex environments have been provided. However, limited availability of labeled samples, significant scale variations in targets and computing resource limitations on embedded devices all still limit detection accuracy and the ability to deploy detection in real-time. In this regard, a generative adversarial network (GAN) based few-shot military target detection is presented. To enlarge the training data set, class-conditional sample generation, multi-scale feature encoding, target-region attention constraints and generated-sample quality screening are employed, and the joint training of real and generated samples is used to enhance the generalization ability of the model. Based on this, depthwise separable convolution, channel pruning, structural re-parameterization, mixed-precision quantization, knowledge distillation, and operator fusion are introduced to realize lightweight network design and optimize the network on the edge. Experimental results show that the proposed method achieves a precision of 91.2%, a recall of 88.9%, and an F1-score of 90.0%. The mAP@0.5 and mAP@0.5:0.95 reach 92.7% and 61.8%, respectively, while the model contains only 4.1 M parameters and requires 7.6 G FLOPs. On the NVIDIA Jetson Orin NX platform, the inference speed reaches 106.4 FPS. These results demonstrate that the method effectively balances few-shot detection accuracy, robustness in complex environments, and real-time inference requirements on embedded edge devices.

**INDEX TERMS** few-shot military target detection; conditional generative adversarial network; lightweight detection; knowledge distillation; embedded edge devices

## I. INTRODUCTION

With the progress of artificial intelligence and intelligent algorithms, military target recognition technology has moved from manual interpretation to data-driven recognition. The high cost of obtaining real battlefield images, however, the lack of labeled samples, the great variation in target scale, and the computation limitation of embedded devices still impose limitations on the generalization ability and real-time implementation of detecting models.Zhuang et al. combined EfficientDet with generative adversarial networks (GANs) to improve military target sample augmentation and detection accuracy [1]; Berkol used generative models to construct synthetic infrared images, providing a pathway for supplementing scarce scene data [2]; Cheong et al. improved long-range infrared target detection at sea through 3D scene augmentation and generation mechanisms [3].Zhang et al. introduced text-to-image and vision-language models to expand the data sources for military target detection with few samples [4]; Zhuang et al. implemented real-time infrared target detection on resource-constrained hardware, validating the feasibility of lightweight deployment [5]. Jiang et al. enhanced SAR image quality through a lightweight generative network [6], while Kong et al.enhanced the stability of sonar image generation using a dual-branch attention mechanism [7], while Gao et al. utilized generative adversarial networks to augment the dataset for low-sample-count SAR vehicle target detection [8]. Sample generation, target detection and edge deployment are covered by existing works separately, but the unified design of sample quality control, lightweighting detection network and optimising edge inference is still lacking. Based on this, a collaborative mechanism is set up, which includes category-conditional generation, multi-scale feature encoding, regional attention constraints and sample filtering. Through the comprehensive application of channel pruning, mixed-precision quantization, knowledge distillation and operator fusion, we provide technical support for high precision, low latency recognition in complex environments through comprehensive optimization for military target detection with few samples and embedded deployment.

## II. MODELING THE PROBLEM OF MILITARY TARGET DETECTION WITH FEW SAMPLES

Military target detection with few examples aims to achieve accurate classification and location regression of targets such as tanks, armored vehicles, ships, and military aircraft under conditions of limited labeled samples [9]. Let the training set be $D = \{(x_i, y_i, b_i)\}_{i=1}^{N}$, where $x_i$ represents the $i$th input image, $y_i$ is the target class label, $b_i = (u_i, v_i, w_i, h_i)$ is the true bounding box of the target, and $N$ is the total number of labeled samples [10]. Due to the small number of available samples per class and their uneven distribution, the detection task can be formulated as:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left[ L_{\text{cls}}(f_\theta(x_i), y_i) + \lambda L_{\text{loc}}(f_\theta(x_i), b_i) \right] \tag{1}$$

where $f_\theta$ is the object detection model with parameters $\theta$, $L_{\text{cls}}$ and $L_{\text{loc}}$ represent the classification loss and bounding box regression loss, respectively, and $\lambda$ is the balancing coefficient for the two losses [11]. A generative model $G(z, c)$ is introduced to expand the training distribution, where $z$ represents random noise and $c$ represents the category condition; the generated sample set is denoted as $D_g$. The joint training set $D_u = D_r \cup D_g$ is formed by combining the real sample set $D_r$ with the high-quality generated sample set $D_g$, and edge inference is performed using a lightweight detection network. The overall problem modeling framework is shown in Figure 1.

## III. GENERATIVE ADVERSARIAL NETWORK-DRIVEN METHOD FOR FEW-SAMPLE DETECTION

### A. CONDITIONAL GENERATIVE ADVERSARIAL NETWORK ARCHITECTURE

The Conditional Generative Adversarial Network (CAAN) consists of a generator $G$ and a discriminator $D$, and incorporates military target class labels as conditional information into the adversarial learning process. The generator takes random noise $z$ and class conditions $c$ as inputs and outputs synthetic samples of the corresponding class:

$$\hat{x} = G(z, c) \tag{2}$$

where $\hat{x}$ is the generated sample, $z$ is a random vector following the prior distribution, and $c$ represents the category encoding for tanks, ships, or aircraft, etc. [12]. The discriminator takes both the sample and the category condition as inputs to judge the authenticity of the sample and its consistency with the category. Its adversarial objective is expressed as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x,c}[\log D(x, c)] + \mathbb{E}_{z,c}[\log(1 - D(G(z, c), c))] \tag{3}$$

where $x$ is a real sample, $D(x, c)$ is the probability of authenticity output by the discriminator, and $\mathbb{E}$ denotes the mathematical expectation [13]. Through alternating optimization of the generator and discriminator, the category controllability and visual realism of the synthesized targets can be enhanced, providing effective expanded samples for subsequent detection models.

### B. MULTISCALE TARGET FEATURE ENCODING

In the field of remote sensing images or complex battlefield environment, the characters of military targets, such as wide variation in scale, weak contour information, and strong background interference, simultaneously high-precision positioning of small targets and high-level semantic expression is difficult to achieve [14]. To improve the model's ability to represent targets of different size, we propose hierarchical extraction of shallow-level texture information, mid-level contour information, and deep-level semantic information and then fusing the features across different layers by adding a multi-scale target feature encoding structure in the feature extraction stage shared by the generator and detector [15].

Let the feature map of layer $i$ be $F_i$; its fused features can be expressed as:

$$\widetilde{F}_i = \phi\left(F_i, U_p(F_{i+1}), \text{Down}(F_{i-1})\right) \tag{4}$$

where $\widetilde{F}_i$ represents the multiscale features after fusion in layer $i$, $\phi(\cdot)$ denotes the concatenation and convolution fusion operations, and $U_p(\cdot)$ and $\text{Down}(\cdot)$ denote upsampling and downsampling, respectively [16]. This architecture introduces high-level contextual semantics while preserving fine-grained local information, thereby improving the model's ability to identify small, weak, and occluded targets under low-data-sample conditions. The multiscale target feature encoding architecture is shown in Figure 2.

### C. CATEGORY-CONDITIONAL SAMPLE GENERATION MECHANISM

The category-conditional sample generation mechanism embeds military target category priors into the generator, enabling the model to synthesize target samples with differentiated structural features according to specified categories [17]. Let the category label $c$ be transformed into a conditional vector via the embedding function $E(\cdot)$ and concatenated with random noise $z$; the generation process is then represented as:

$$\hat{x} = G\left([z, E(c)]\right) \tag{5}$$

where $\hat{x}$ is the generated sample, $G(\cdot)$ is the generator, and $[\cdot]$ represents feature concatenation [18]. To ensure that the generated result aligns with the given category, a category constraint loss is introduced:

$$L_c = -\mathbb{E}_{z,c} \log P(c \mid \hat{x}) \tag{6}$$

where $P(c \mid \hat{x})$ represents the probability predicted by the classifier that the generated sample belongs to class $c$, and $\mathbb{E}$ denotes the expected value [19].

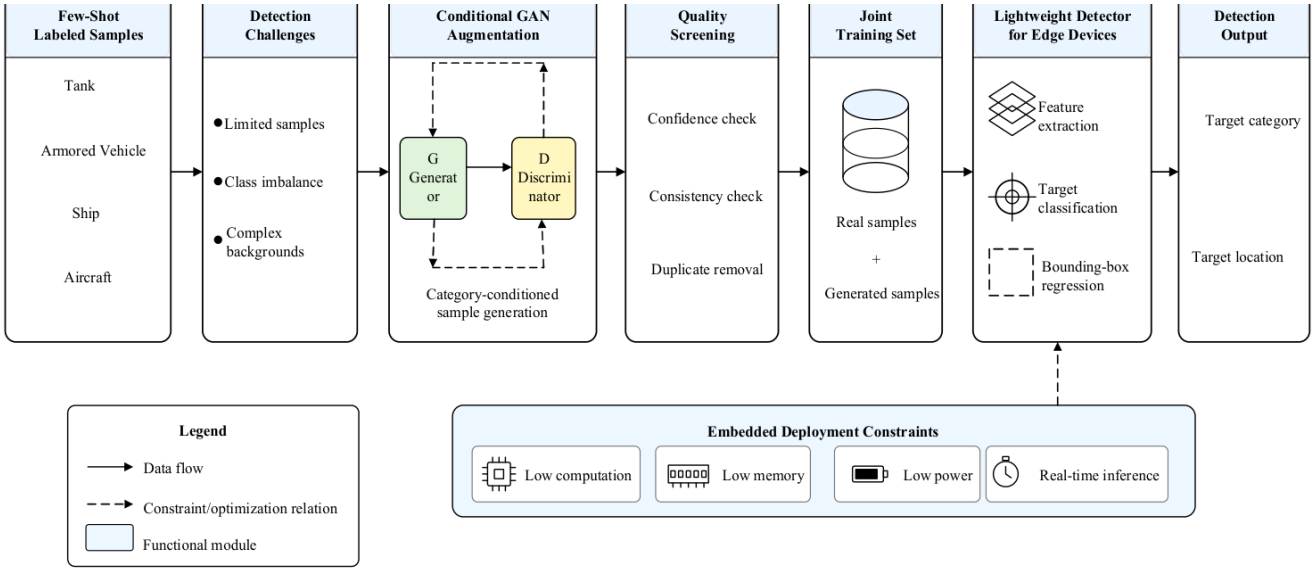


**FIGURE 1.** Modeling Framework for the Few-Sample Military Target Detection Problem

### *D. TARGET REGION ATTENTION CONSTRAINT*

Target-area attention constraints are applied to reduce the interference from the complex backgrounds, and make the generator focus on the structural characteristics of military targets. In particular, a binary target mask is first obtained from the bounding boxes of real samples, where the target area is set to 1, and the background area is set to 0. In turn, the feature maps are taken from the intermediate layers of the generator and a spatial attention map is computed through a channel compression, convolutional mapping, and Sigmoid activation function [20]. The attention map is aligned with the target mask, pixel-wise, during training, and a regional consistency error is imposed to force the attention response to the target area. The features in the target region are weighted and enhanced and the features in the background region are suppressed, and the weighted and enhanced features are sent to the subsequent upsampling and image reconstruction modules. This operation decreases the background texture replication and the blurring of the edges of the target, which is beneficial for the integrity of the generated samples in terms of their contours, local details and spatial positioning [21].

### *E. ADVERSARIAL LOSS AND CONSISTENCY LOSS*

To balance the visual reality, class correctness, and structural stability of the generated samples, during the training phase, the adversarial loss, class consistency loss, feature consistency loss, and target region constraints are all considered with the contribution of each optimization objective weighted by weight parameters [22]. Adversarial loss is used to improve the distribution of generated samples, while consistency losses are used to reduce class drift and structural distortion; specific settings are shown in Table 1.

### *F. QUALITY SCREENING OF GENERATED SAMPLES*

In every round, 3,000 samples of each of the four types of vehicles (tanks, armored vehicles, ships and aircraft) are randomly generated, then all resized to 640×640 pixels and subjected to the three-stage screening. In the first step, an auxiliary classifier is used to retain the category consistent samples, the category confidence of the sample is no less than 0.85;In the second step, the IOU of the target mask and the predicted region is calculated, and the samples whose IOU is less than 0.70 are deleted, which means that the ratio of the target area to the total area is less than 2% or the ratio of the target area to the total area is greater than 65% are deleted;In the third step, deep features are extracted and the similarity between the two is calculated by cosine similarity, and when the similarity of the two samples is greater than 0.98, the sample with the higher clarity score is deleted. Finally, a generated sample database is built, and an edge integrity, texture clarity and plausibility of the background are checked.

### *G. JOINT TRAINING OF REAL AND GENERATED SAMPLES*

In the joint training phase, the screened generated samples are dynamically mixed with real samples to make training data of 32 pieces in a batch. Both real and generated samples are combined in each batch, which will always have a fixed set of 20 samples, with the real sample dominating. Using only real samples, the first 20 training epochs are used to warm up the detector, and then the number of generated samples is gradually increased from 20% to 40% [25]. The number of samples in each class (tanks, armored vehicles, ships and aircraft) is approximately equal using the method of category-balanced sampling during the training phase. To reduce the influence of the synthetic noise, generated

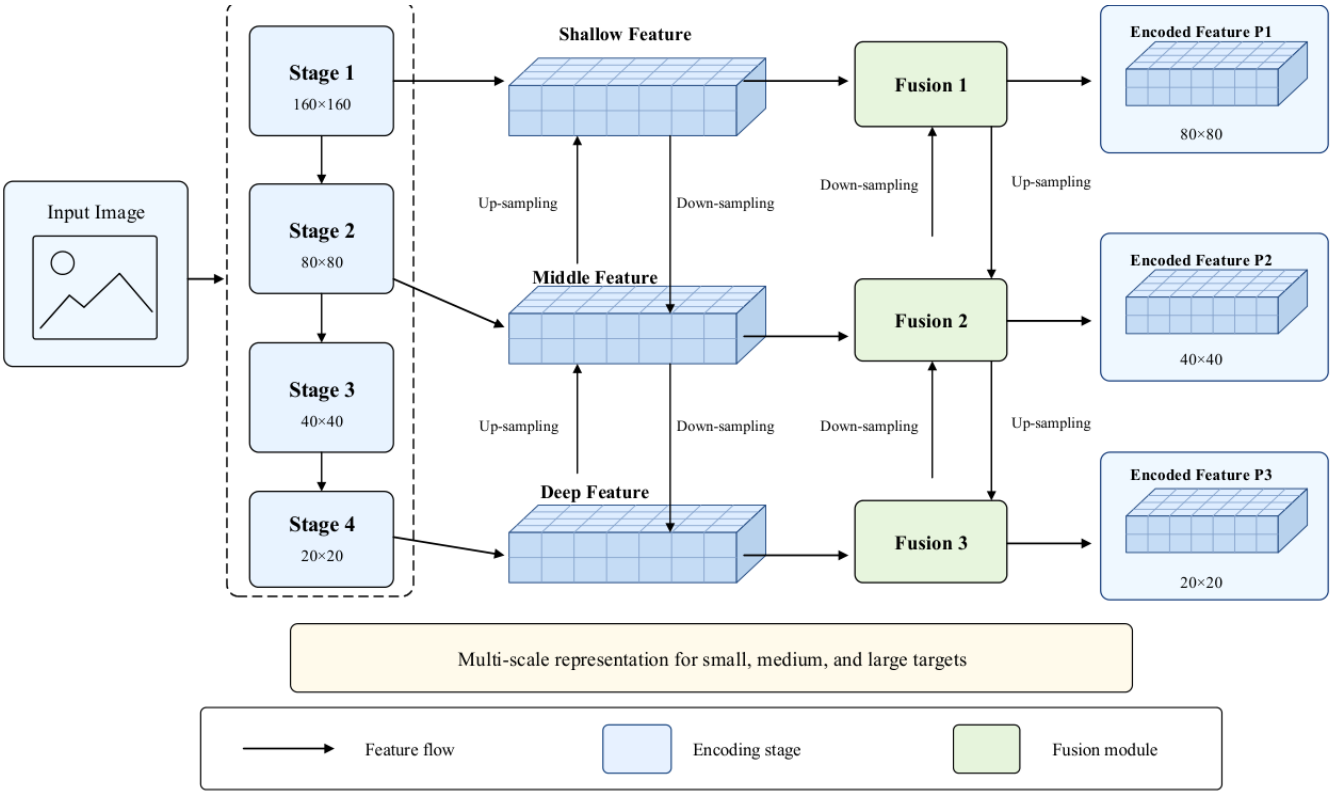


**FIGURE 2.** Multiscale Object Feature Encoding Architecture

**TABLE 1.** Model loss functions and parameter descriptions.

| Loss Type | Parameter Symbol | Primary Function | Constraint Object | Weight |
|---|---|---|---|---|
| Adversarial Loss | $\lambda_{\mathrm{adv}}$ | Improves the realism of generated samples | Generator and discriminator outputs | 1.00 |
| Class Consistency Loss | $\lambda_{\mathrm{cls}}$ | Preserves the correctness of generated categories | Class prediction results | 0.80 |
| Feature Consistency Loss | $\lambda_{\mathrm{fea}}$ | Reduces semantic feature deviation | Multi-scale feature representations | 0.50 |
| Region Attention Loss | $\lambda_{\mathrm{att}}$ | Enhances responses within target regions | Attention maps and target masks | 0.30 |
| Reconstruction Consistency Loss | $\lambda_{\mathrm{rec}}$ | Preserves target contours and texture structures | Generated and reference samples | 0.20 |

samples are given a weight of 0.6, whereas the real samples keep a weight of 1.0; the confidence of generated samples is re-computed every 10 rounds and samples having a confidence score below 0.80 are not used in further training [26].

### *H. LIGHTWEIGHT OBJECT DETECTION NETWORK*

The lightweight object detection network is composed of backbone feature extraction module, multi-scale feature fusion module and decoupled detection head. The backbone network adopts deep separable convolutions to replace the traditional convolutional network, and adopts the reverse residual structure to reduce the number of parameters and calculation quantity; the neck network adopts bidirectional feature fusion, and passes the information of shallow layer detail to the deep layer semantic, which improves the representation ability of small target images [27]. The detection heads are respectively responsible for the classification task and the bounding box regression task, which outputs the object class and confidence and the location parameters respectively, so as to reduce the feature interference between the different tasks. All the network input sizes are the same, $640\times640$, and the network outputs detection feature maps in three scales, which are used for the recognition of small, medium and large military targets, respectively [28]. The number of the high channel convolutional layers in the network is decreased and the size of activation feature cache is controlled to suit embedded devices. The structure of the lightweight object detection network is presented in Figure 3.

### *I. MODEL TRAINING AND OPTIMIZATION PROCESS*

There are four stages of model training: generative model pre-training, sample screening, detector warm-up, and joint

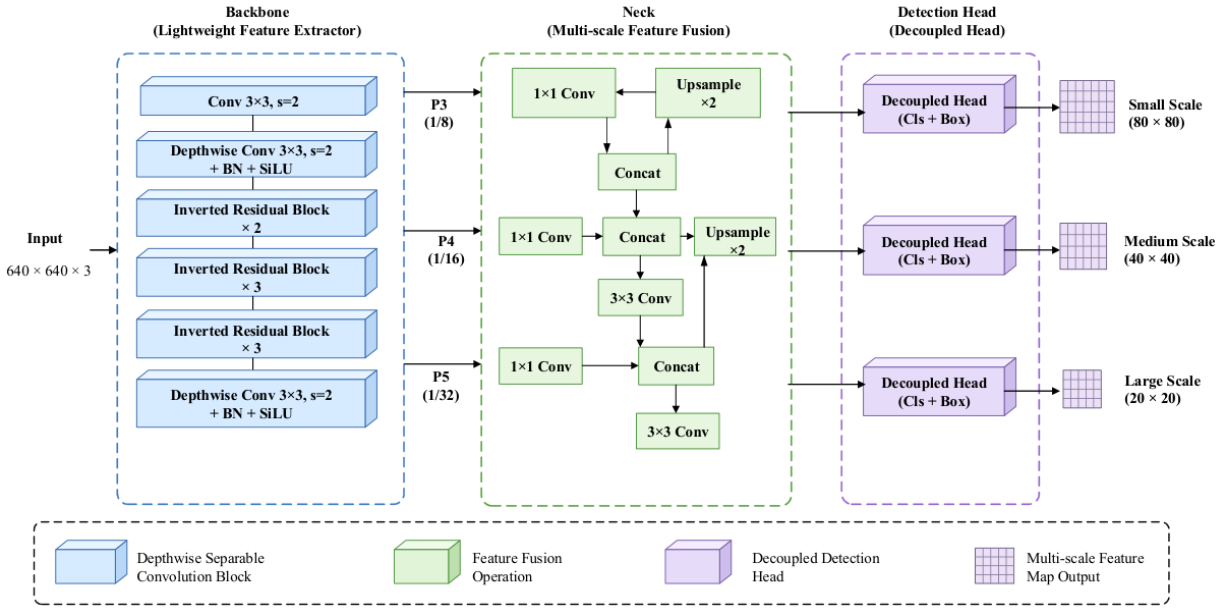


**FIGURE 3.** Architecture of the lightweight object detection network

optimization. First, a conditional generative adversarial network (GAN) is trained with real military target samples to let the generator sufficiently learn the appearance and structure distribution of various target categories; second, according to the category confidence, region integrity and similarity of features, low-quality samples are filtered out [29]. The detection model is first trained, and introduced 20 epochs of real samples to the detection model, then gradually added the generated samples, and network parameters are updated by AdamW. The learning rate is dynamically changed according to the validation set loss and training early stopping is performed if the validation loss does not decrease after 15 consecutive epochs. The main parameter settings of generative and detection models are listed in Table 2.

## IV. MODEL OPTIMIZATION FOR EMBEDDED EDGE DEVICES

### A. EDGE INFERENCE SYSTEM ARCHITECTURE

The embedded edge inference system is composed of five modules: image acquisition, data pre-processing, model inference, result post-processing and task output. The quantized light-weight detection model is deployed to the GPU / NPU and resizing, normalization and format conversion of image and video streams of military targets are done at the CPU side by the front-end camera devices. Following the confidence filtering and non-maximum suppression, the results of the inference are outputted as a target category, a location, and an alarm information [30]. Shared memory and asynchronous data transfer are used to minimize inter-module communication overhead and hence to achieve pipelined parallelism for acquisition, inference and display. The architecture of the embedded edge inference system is shown in Figure 4.

### B. LIGHTWEIGHT DESIGN OF THE BACKBONE NETWORK

The backbone network employs a deep separable convolutional and reverse residual structure to replace standard convolutions, thereby reducing the parameter size and computational overhead on embedded devices. The computational complexity of a standard convolution can be expressed as:

$$C_{\text{std}} = HWOK^2I \tag{7}$$

The computational complexity of deep separable convolution is:

$$C_{\text{ds}} = HWK^2I + HWIO \tag{8}$$

where $H$ and $W$ are the dimensions of the output feature maps, $K$ is the size of the convolution kernel, and $I$ and $O$ represent the number of input and output channels, respectively [31]. From this, the computational complexity compression ratio can be derived as:

$$r = \frac{C_{\text{ds}}}{C_{\text{std}}} = \frac{1}{O} + \frac{1}{K^2} \tag{9}$$

To preserve high-resolution features in the shallow layers to better model the edge and texture of small military targets, the network uses “channel expansion—deep convolution—channel compression” inverted residual units, while cross-layer shunts are used to reduce the gradient vanishing problem in the middle and deep layers. A number of 32, 64, 128 and 256 channels is planned to be used for each stage,

**TABLE 2.** Main parameters of the generative and detection models.

| Model | Parameter | Setting | Description |
|---|---|---|---|
| Generative Model | Input image size | $256 \times 256$ | Resolution of generated samples |
| | Noise dimension | 128 | Dimension of the latent random vector |
| | Batch size | 32 | Number of samples per iteration |
| | Initial learning rate | 0.0002 | Learning rate for adversarial training |
| | Optimizer | Adam | Optimizer for generator and discriminator |
| | Training epochs | 200 | Maximum adversarial training epochs |
| | Label embedding dimension | 64 | Dimension of category condition vectors |
| Detection Model | Input image size | $640 \times 640$ | Resolution of detector input |
| | Batch size | 32 | Number of images per training batch |
| | Initial learning rate | 0.001 | Initial learning rate of the detector |
| | Optimizer | AdamW | Parameter optimization method |
| | Weight decay | 0.0005 | Regularization coefficient |
| | Training epochs | 150 | Maximum detector training epochs |
| | Learning-rate scheduler | Cosine decay | Dynamic learning-rate adjustment |
| | Early-stopping patience | 15 epochs | Termination threshold without improvement |

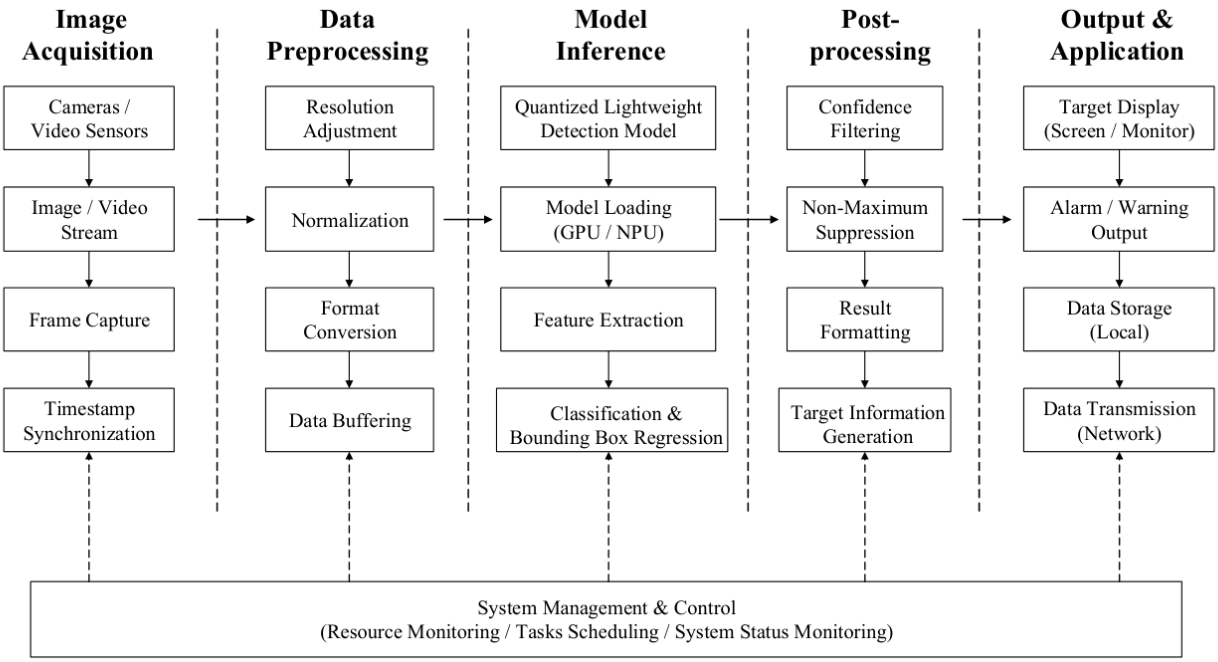


**FIGURE 4.** System Architecture Flowchart

respectively. The feature downsampling is done by using the technique of stepwise convolution, while the SiLU activation function and batch normalization are used to stabilize the training process. This design can effectively reduce the redundant channels and memory access overhead and retain the multi-scale semantic information of the targets.

### *C. CHANNEL PRUNING AND STRUCTURAL REPARAMETERIZATION*

Channel pruning uses a " global ranking, hierarchical constraints and iterative fine-tuning " method to eliminate redundant channels. Channel importance scores are built after training by taking the absolute values of the scaling coefficients of the batch normalization layer, and the L1 norm of the convolutional kernels, and computing the average activations response on 3, 200 training images. The pruning rate is initially 10% and is increased by 5% each 20 training rounds, up to a maximum of 40%, for the scores which are sorted in ascending order. The channel retention rate is kept at 80% at least in the first two layers, and at 60% at least in the middle and deep layers to avoid losing texture information in the shallow layer. Finetuning is done for 15 rounds after every pruning round with a 5:3 ratio of real and generated samples. In the training stage, the $3\times3$ convolution branch, $1\times1$ convolution branch and identity mapping branch are weight fused into a $3\times3$ convolution branch, and batch normalization parameters are also merged, so as to avoid branch scheduling and intermediate feature

caching in the stage of inference.

### D. HYBRID PRECISION QUANTIZATION METHOD

Different quantization bit widths are set for different layers of the network to balance out detection accuracy and the efficiency of using the network for inference on the edge. Shallow layers of the main branch and the detection output layer are kept as 16-bit floating point in order to minimize the loss of texture and boundary information of small objects, while the intermediate layers of the convolutional layers are compressed into 8-bit integers, and the layers with stable activations and high redundancy are compressed into 6 bits. The linear quantization process for weights or activation values $x$ is expressed as:

$$q = \text{clip}\left(\text{round}\left(\frac{x}{s}\right) + z, q_{\min}, q_{\max}\right) \tag{10}$$

where $q$ is the quantized integer, $s$ is the scaling factor, $z$ is the zero point, and $q_{\min}$ and $q_{\max}$ are the value boundaries corresponding to the bit width [32]. The dequantization value is:

$$\hat{x} = s(q - z) \tag{11}$$

where $\hat{x}$ is the restored approximate value. The bit widths for each layer are determined jointly based on quantization error and computational cost:

$$b_i^* = \arg\min_{b_i} (E_i + \alpha C_i) \tag{12}$$

where $b_i$ is the bit width of layer $i$, $E_i$ is the quantization error for that layer, $C_i$ is the computational and storage cost, and $\alpha$ is the balancing coefficient. Before quantization, the activation range is estimated using 512 calibration samples, and channel-wise weighted quantization and layer-wise activation quantization are employed to minimize the interference of outliers on the quantization scale.

### E. KNOWLEDGE DISTILLATION AND FEATURE TRANSFER

Knowledge distillation uses a teacher network to instruct a light-weight student network. Teacher network is chosen as a complete detection model, whose parameters are frozen, and an input image is fed to the student network, which learns from the teacher the classification outputs, the bounding box regressions and the multi-scale feature layers simultaneously. In the training phase, feature response differences of corresponding layers in the teacher network and the student network are computed after channel-aligning the corresponding layers and aligning the feature dimensions using $1\times1$ convolutions. The classification branch uses soft label constraints, and the localization branch passes information about the object center and the distribution of object boundaries to improve the distillation weights in the small object feature layer. The distillation loss is optimized along with the original detection loss, allowing the student network to preserve the semantic representation and the ability to localize objects, even when the parameters of the student network are limited. The knowledge distillation architecture is shown in Figure 5.

### F. OPERATOR FUSION AND MEMORY OPTIMIZATION

To reduce the overhead of operator scheduling and memory access during embedded inference, convolution, batch normalization, and activation operations are fused into a single computational node. The equivalent weights and biases of the convolution layer and the batch normalization layer are expressed as follows:

$$W' = \frac{\gamma W}{\sqrt{\sigma^2 + \varepsilon}} \tag{13}$$

$$b' = \frac{\gamma(b - \mu)}{\sqrt{\sigma^2 + \varepsilon}} + \beta \tag{14}$$

where $W$ and $b$ are the original convolution weights and biases, $\mu$ and $\sigma^2$ are the batch normalization statistics, $\gamma$ and $\beta$ are the scaling and translation parameters, and $\varepsilon$ is the constant term. After fusion, the convolution output can be mapped directly, reducing the need to write intermediate tensors back to memory. Memory optimization employs lifecycle analysis and buffer reuse strategies, and its peak memory consumption can be expressed as:

$$M_{\text{peak}} = \max_t \sum_{i \in A(t)} m_i \tag{15}$$

where $A(t)$ is the set of tensors that are still active at time $t$, and $m_i$ is the memory size of the $i$th tensor. During inference, feature maps with non-overlapping lifetimes are allocated to the same buffer, and in-place activation, contiguous memory layout, and asynchronous data copying are employed to reduce the number of dynamic allocations and memory fragmentation, thereby improving execution stability on edge devices.

### G. IMPLEMENTATION OF THE EMBEDDED INFERENCE WORKFLOW

The embedded inference workflow consists of the following steps: “Acquisition — Preprocessing — Inference — Postprocessing—Output.” The camera captures images at 25 fps. The CPU resizes and converts the colour space and normalizes the frames and saves each frame in a double buffer queue. GPU or NPU asynchronously reads data from the queue, loads the quantized lightweight model and does forward computation. The total latency per frame is expressed as:

$$T_{\text{all}} = T_{\text{cap}} + T_{\text{pre}} + T_{\text{inf}} + T_{\text{post}} + T_{\text{out}} \tag{16}$$

where $T_{\text{cap}}$, $T_{\text{pre}}$, $T_{\text{inf}}$, $T_{\text{post}}$, and $T_{\text{out}}$ represent the acquisition, preprocessing, inference, postprocessing, and output delays, respectively. The system throughput is defined as:

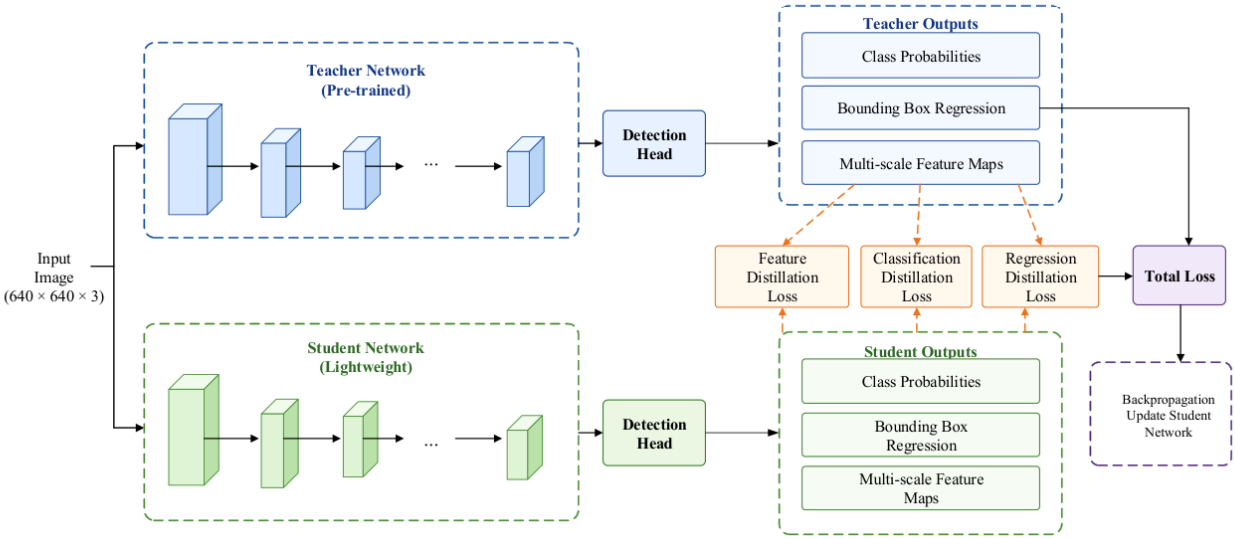


**FIGURE 5.** Knowledge Distillation Architecture

$$F = \frac{N}{\sum_{i=1}^{N} T_i} \tag{17}$$

where $N$ is the number of consecutively processed frames, and $T_i$ is the processing time for the $i$th frame. In the post-processing stage, candidate boxes are first filtered based on a confidence threshold of 0.35, and then duplicate detections are removed using non-maximum suppression (NMS); the NMS threshold is set to 0.5, and the overlap ratio is:

$$\mathrm{IoU} = \frac{B_p \cap B_g}{B_p \cup B_g} \tag{18}$$

where $B_p$ and $B_g$ represent the predicted bounding boxes and candidate bounding boxes, respectively. The final output includes class, position, confidence, and timestamp, which are displayed locally or transmitted to the task terminal via a communication interface.

## V. EXPERIMENTAL RESULTS AND ANALYSIS

### A. EXPERIMENTAL ENVIRONMENT CONFIGURATION

The experiments were conducted on an Ubuntu 20.04 system with a hardware platform configured with an Intel Core i9- 12900K processor, 64 GB of RAM, and an NVIDIA RTX 3090 GPU. The model was implemented using Python 3.10 and PyTorch 2.0, with training accelerated by CUDA 11.8 and cuDNN 8.7.Input images were uniformly resized to 640× 640 pixels, with a batch size of 32 and a maximum of 150 training iterations. The AdamW optimizer was used with an initial learning rate of 0.001, dynamically adjusted using a cosine annealing strategy. To ensure experimental stability, all models used the same data split and were trained independently five times under different random seeds. The main experimental configurations are shown in Table 3.

**TABLE 3.** Software and hardware environment and training parameters.

| Category | Item | Configuration |
|---|---|---|
| Hardware | CPU | Intel Core i9-12900K |
| | GPU | NVIDIA GeForce RTX 3090, 24 GB |
| | Memory | 64 GB DDR4 |
| Software | Operating system | Ubuntu 20.04 LTS |
| | Programming language | Python 3.10 |
| | Deep-learning framework | PyTorch 2.0 |
| | Acceleration libraries | CUDA 11.8, cuDNN 8.7 |
| Training | Input image size | 640 × 640 |
| | Batch size | 32 |
| | Maximum epochs | 150 |
| | Optimizer | AdamW |
| | Initial learning rate | 0.001 |
| | Weight decay | 0.0005 |
| | Learning-rate scheduler | Cosine annealing |
| | Random seeds | 5 independent runs |

### B. DATASET AND EVALUATION METRICS

The experimental dataset consisted of publicly available remote-sensing images and a self-constructed military target dataset, comprising 12,480 images in total. The dataset covered four target categories: tanks, armored vehicles, ships, and military aircraft. The data were divided into training, validation, and test sets at a ratio of 7:1:2. Horizontal flipping, random cropping, brightness perturbation, and scale transformation were applied to enhance sample diversity. Detection performance was evaluated using precision, recall, F1-score, mAP@0.5, and mAP@0.5:0.95. Deployment performance was assessed in terms of parameter count, computational complexity, model size, single-frame latency, frames per second, peak memory usage, and average power consumption.

### C. ANALYSIS OF GENERATED SAMPLE QUALITY

To evaluate the ability of different generative models to preserve the appearance, category, and structural information

of military targets, we selected FID, IS, LPIPS, category consistency rate, and valid sample rate as evaluation metrics and conducted comparisons under the same number of training iterations and data scale. The specific results are shown in Table 4.

Table 4 shows that the FID of the proposed method decreased to 39.68, a 21.8% reduction compared to AC-GAN' s 50.76, indicating that the generated distribution is closer to the real samples.Its IS reaches 2.91 and LPIPS is 0.263, indicating improvements in both sample clarity and diversity. The category consistency rate and effective sample rate reach 94.8% and 91.7%, respectively — an increase of 15.2 and 20.9 percentage points compared to DCGAN—proving that category-specific constraints, regional attention, and quality filtering can effectively reduce category confusion, structural distortion, and low-quality samples.

### D. COMPARISON OF DIFFERENT DETECTION METHODS

To evaluate the model's overall performance in the small-sample military target detection task, we selected Faster R-CNN, SSD-MobileNetV3, YOLOv5n, YOLOv7-tiny, and YOLOv8n as comparison methods, ensuring uniform training conditions and data splitting. The results are shown in Table 5.

As presented in Table 5, the proposed method achieved a precision of 91.2%, a recall of 88.9%, and an F1-score of 90.0%, outperforming all comparison models. Its mAP@0.5 reached 92.7%, which was 1.9 percentage points higher than that of YOLOv8n, while its mAP@0.5:0.95 increased to 61.8%. Meanwhile, the model contained only 4.1 M parameters and required 7.6 G FLOPs, substantially lower than the 41.3 M parameters and 180.2 G FLOPs of Faster R-CNN. These results demonstrate that the proposed method maintains high detection accuracy while satisfying lightweight deployment requirements.

### E. DETECTION RESULTS UNDER DIFFERENT SAMPLE SIZES

To investigate the impact of labeled sample size on detection performance, the number of real training images per class was set to 10, 20, 30, 50, and 100, respectively. The detection results of the methods without augmentation, traditional augmentation, CGAN augmentation, and the proposed method were compared; the results are shown in Table 6.

Table 6 shows that when only 10 labeled images were available for each class, the proposed method achieved an mAP@0.5 of 66.9%, which was 17.1 percentage points higher than that obtained without augmentation. This result indicates that generated samples provide substantial compensation in extremely limited-data scenarios. When the number of samples increased to 30 per class, the detection accuracy reached 81.4%, remaining 4.9 percentage points higher than that of CGAN augmentation. With 100 samples per class, the proposed method achieved an mAP@0.5 of 91.3% and an mAP@0.5:0.95 of 59.0%, demonstrating stable performance gains across different sample scales.

### F. ABLATION STUDIES AND PARAMETER ANALYSIS

To investigate the actual contributions of modules such as multiscale encoding, conditional sample generation, regional attention, quality filtering, and knowledge distillation, we progressively stacked these functional modules under the same training settings and further analyzed the proportion of generated samples and the channel pruning rate. The results are shown in Table 7.

As shown in Table 7, the baseline model achieved an mAP@0.5 of 88.2%. After introducing multi-scale encoding, the value increased to 89.4%. With the further addition of conditional generation, region attention, and quality screening, the mAP@0.5 rose to 90.6%, 91.4%, and 92.1%, respectively. The complete model achieved an mAP@0.5 of 92.7% and an mAP@0.5:0.95 of 61.8%. The best performance was obtained when generated samples accounted for 40% of the training data. However, increasing the pruning rate from 35% to 45% reduced the mAP@0.5 by 1.1 percentage points, indicating that excessive compression may remove effective feature channels.

### G. ROBUSTNESS ANALYSIS IN COMPLEX ENVIRONMENTS

To evaluate stability in non-ideal battlefield scenarios, we constructed subsets of the test set featuring low illumination, partial occlusion, complex backgrounds, small-scale targets, and motion blur, and compared them with YOLOv5n and YOLOv8n. The results are shown in Table 8.

**TABLE 8.** Detection results under different complex environments.

| Test Condition | YOLOv5n mAP@0.5/% | YOLOv8n mAP@0.5/% | Proposed Method mAP@0.5/% | Proposed Method Recall/% |
|---|---|---|---|---|
| Normal Condition | 88.2 | 90.8 | 92.7 | 88.9 |
| Low Illumination | 70.9 | 75.4 | 82.6 | 78.1 |
| Partial Occlusion | 68.7 | 72.8 | 80.3 | 75.6 |
| Complex Background | 72.6 | 77.9 | 84.7 | 80.5 |
| Small Targets | 64.8 | 69.6 | 78.1 | 73.9 |
| Motion Blur | 67.5 | 73.2 | 80.9 | 76.4 |

Table 8 indicates that the proposed method achieved an mAP@0.5 of 92.7% under normal conditions. Under low illumination, partial occlusion, and complex background conditions, the corresponding values remained at 82.6%, 80.3%, and 84.7%, respectively. In the more challenging small-target scenario, the mAP@0.5 still reached 78.1%, exceeding YOLOv8n by 8.5 percentage points. Under motion blur, the proposed method achieved 80.9%, which was 13.4 percentage points higher than YOLOv5n. These findings demonstrate that multi-scale feature fusion and target-region attention can effectively alleviate performance degradation caused by missing details, background interference, and partial occlusion (see Figure 6).

**TABLE 4.** Sample quality evaluation results.

| Generation Method | FID ↓ | IS ↑ | LPIPS ↑ | Class Consistency/% ↑ | Valid Sample Rate/% ↑ |
|---|---|---|---|---|---|
| DCGAN | $72.84 \pm 2.11$ | $2.18 \pm 0.06$ | $0.206 \pm 0.008$ | $79.6 \pm 1.2$ | $70.8 \pm 1.5$ |
| WGAN-GP | $58.31 \pm 1.74$ | $2.46 \pm 0.05$ | $0.229 \pm 0.007$ | $84.1 \pm 1.0$ | $78.5 \pm 1.2$ |
| ACGAN | $50.76 \pm 1.43$ | $2.63 \pm 0.04$ | $0.237 \pm 0.006$ | $89.7 \pm 0.8$ | $84.6 \pm 1.0$ |
| CGAN with Multi-scale Encoding | $44.92 \pm 1.31$ | $2.78 \pm 0.05$ | $0.251 \pm 0.007$ | $92.4 \pm 0.7$ | $88.9 \pm 0.8$ |
| Proposed Method | $39.68 \pm 1.17$ | $2.91 \pm 0.04$ | $0.263 \pm 0.006$ | $94.8 \pm 0.6$ | $91.7 \pm 0.7$ |

**TABLE 5.** Performance comparison of different object detection methods.

| Detection Method | Precision/% | Recall/% | F1-score/% | mAP@0.5/% | mAP@0.5:0.95/% | Parameters/M | FLOPs/G |
|---|---|---|---|---|---|---|---|
| Faster R-CNN | 88.2 | 84.9 | 86.5 | 89.1 | 58.6 | 41.3 | 180.2 |
| SSD-MobileNetV3 | 83.7 | 80.5 | 82.1 | 85.4 | 49.8 | 5.1 | 4.8 |
| YOLOv5n | 87.1 | 83.6 | 85.3 | 88.2 | 55.1 | 1.9 | 4.5 |
| YOLOv7-tiny | 88.4 | 85.2 | 86.8 | 89.7 | 57.6 | 6.2 | 13.2 |
| YOLOv8n | 89.5 | 86.7 | 88.1 | 90.8 | 59.4 | 3.2 | 8.7 |
| Proposed Method | 91.2 | 88.9 | 90.0 | 92.7 | 61.8 | 4.1 | 7.6 |

**TABLE 6.** Comparison of detection performance.

| Labeled Images per Class | No Augmentation mAP@0.5/% | Traditional Augmentation mAP@0.5/% | CGAN Augmentation mAP@0.5/% | Proposed Method mAP@0.5/% | Proposed Method mAP@0.5:0.95/% |
|---|---|---|---|---|---|
| 10 | 49.8 | 55.9 | 61.7 | 66.9 | 33.7 |
| 20 | 59.7 | 65.1 | 70.8 | 75.8 | 40.9 |
| 30 | 66.4 | 71.9 | 76.5 | 81.4 | 46.6 |
| 50 | 74.6 | 79.2 | 83.5 | 87.2 | 53.1 |
| 100 | 82.7 | 85.8 | 88.9 | 91.3 | 59.0 |

**TABLE 7.** Comparison of ablation results.

| Study Type | Configuration | mAP@0.5/% | mAP@0.5:0.95/% | F1-score/% | Parameters/M | FPS |
|---|---|---|---|---|---|---|
| Ablation | Baseline Detector | 88.2 | 55.1 | 85.3 | 4.8 | 65.2 |
| | + Multi-scale Encoding | 89.4 | 56.8 | 86.5 | 5.0 | 63.7 |
| | + Conditional Generation | 90.6 | 58.5 | 87.8 | 5.0 | 63.7 |
| | + Region Attention | 91.4 | 59.7 | 88.7 | 5.1 | 62.9 |
| | + Quality Screening | 92.1 | 60.9 | 89.4 | 5.1 | 62.9 |
| | Full Model | 92.7 | 61.8 | 90.0 | 4.1 | 72.5 |
| Generated-sample Ratio | 0.20 | 91.8 | 60.5 | 89.1 | 4.1 | 72.5 |
| | 0.40 | 92.7 | 61.8 | 90.0 | 4.1 | 72.5 |
| | 0.60 | 91.9 | 60.8 | 89.2 | 4.1 | 72.5 |
| Channel-pruning Rate | 0.25 | 92.9 | 62.0 | 90.1 | 4.8 | 65.2 |
| | 0.35 | 92.7 | 61.8 | 90.0 | 4.1 | 72.5 |
| | 0.45 | 91.6 | 60.2 | 88.9 | 3.5 | 79.1 |

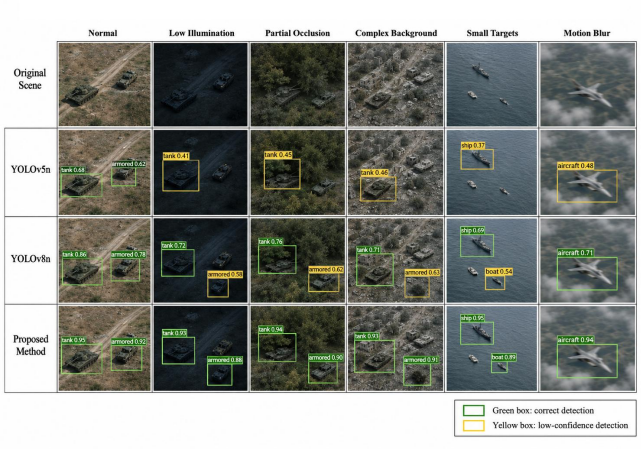


**FIGURE 6.** Visualization of Detection in Complex Environments

### *H. PERFORMANCE IN EMBEDDED DEVICE DEPLOYMENT*

To evaluate deployment adaptability across different edge computing platforms, the pruned and quantized models were deployed on Jetson Nano, Jetson Xavier NX, Jetson Orin Nano, Jetson Orin NX, and RK3588. The results are shown in Table 9.

As shown in Table 9, the model achieved a single-frame latency of 9.4 ms and an inference speed of 106.4 FPS on the Jetson Orin NX, while maintaining an mAP@0.5 of 92.1%, thereby satisfying high real-time processing requirements. On the Jetson Orin Nano, the model reached 71.9 FPS with an average power consumption of 14.9 W, providing a favorable balance between speed and energy consumption. The RK3588 consumed only 7.6 W on average and required 1,184 MB of peak memory while still achieving 33.8 FPS. The accuracy reduction across all platforms remained

**TABLE 9.** Comparison of deployment performance on different embedded devices.

| Embedded Device | Inference Precision | Latency/ms | FPS | Peak Memory/MB | Average Power/W | mAP@0.5/% |
|---|---|---|---|---|---|---|
| NVIDIA Jetson Nano | INT8 | 54.7 | 18.3 | 1328 | 8.9 | 91.6 |
| NVIDIA Jetson Xavier NX | INT8 | 25.3 | 39.5 | 1476 | 12.8 | 91.9 |
| NVIDIA Jetson Orin Nano | INT8 | 13.9 | 71.9 | 1538 | 14.9 | 92.0 |
| NVIDIA Jetson Orin NX | INT8 | 9.4 | 106.4 | 1712 | 22.1 | 92.1 |
| Rockchip RK3588 | INT8 | 29.6 | 33.8 | 1184 | 7.6 | 91.4 |

within 1.3 percentage points, confirming the strong hardware adaptability of mixed-precision quantization and operator fusion (see Figure 7).

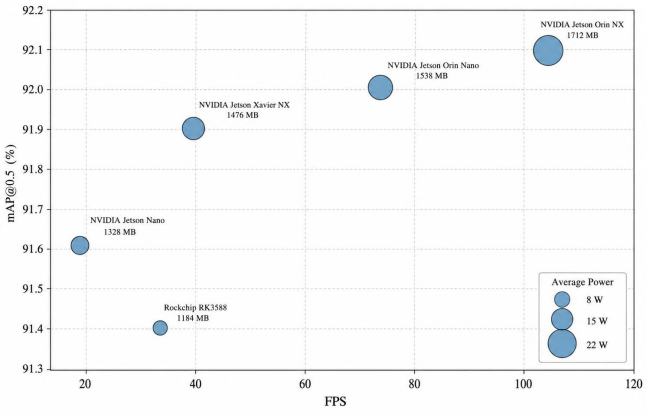


**FIGURE 7.** Comparison of Device Deployment Performance

## VI. CONCLUSIONS

Based on the fact that the military target detection data on embedded edge devices are limited, we built a military target detection framework, which consists of sample generation based on the condition of the target category, multi-scale feature encoding, target region attention constraint, generated sample filtering, and joint training of real and generated samples. We realized edge deployment by synergistically applying backbone network lightweighting, channel pruning, structural reparameterization, mixed-precision quantization, knowledge distillation, and operator fusion in unison. Experimental results show that this method has a better category representation and target localization ability in few sample conditions, has good stability in low light, occlusion, complex background and small target scenarios, and has a good balance between detection performance, inference speed, power consumption, and memory usage. The innovation is that the generative sample quality control and lightweight detection optimization are designed collaboratively, but it still has the problem of insufficient coverage of military target categories, relying on the generative distribution, and difference in operator support between devices. In the future, more cross domain adaptation, incremental learning, and dynamic quantization techniques could be integrated into the model to increase the extent of multi-source sensor data and the number of open scene categories, further improve the model's generalization ability, and ensure its long-term operation in unconfirmed scenarios, continuous missions, and heterogeneous edge platforms.

## REFERENCES

[1] X. Zhuang, D. Li, Y. Wang, et al., "Military target detection method based on EfficientDet and generative adversarial network," *Eng. Appl. Artif. Intell.*, vol. 132, Art. no. 107896, 2024, doi: 10.1016/j.engappai.2024.107896.

[2] A. Berkol, "Exploring AI-based techniques for the generation of synthetic infrared images and their practical applications," in *Proc. 2025 9th Int. Symp. Innov. Approaches Smart Technol. (ISAS)*, 2025, pp. 1–5, doi: 10.1109/ISAS66241.2025.11101900.

[3] S. Cheong, W. Jung, Y. S. Lim, et al., "Thermal-infrared remote-target detection system for maritime rescue using 3-D game-based data augmentation with GAN," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, pp. 1–13, 2024, doi: 10.1109/TGRS.2024.3454983.

[4] Y. Zhang, J. Zhu, and K. Cao, "Few-shot military target detection with text-to-image and vision-language models," in *Proc. 2024 7th Int. Conf. Image Graph. Process.*, 2024, pp. 47–53, doi: 10.1145/3647649.3647657.

[5] T. Zhuang, X. Liang, B. Xue, et al., "An in-vehicle real-time infrared object detection system based on deep learning with resource-constrained hardware," *Intell. Robot.*, vol. 4, no. 3, pp. 276–292, 2024, doi: 10.20517/ir.2024.18.

[6] N. Jiang, W. Zhao, H. Wang, et al., "Lightweight super-resolution generative adversarial network for SAR images," *Remote Sens.*, vol. 16, no. 10, Art. no. 1788, 2024, doi: 10.3390/rs16101788.

[7] W. Kong, H. Yang, M. Jia, et al., "CDBA-GAN: A conditional dual-branch attention generative adversarial network for robust sonar image generation," *Appl. Sci.*, vol. 15, no. 13, Art. no. 7212, 2025, doi: 10.3390/app15137212.

[8] D. Gao, X. Wu, Z. Wen, et al., "Few-shot SAR vehicle target augmentation based on generative adversarial networks," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 10, pp. 83–90, 2024, doi: 10.5194/isprs-annals-X-1-2024-83-2024.

[9] M. S. Alexiou and J. S. Mertoguno, "A survey on recent advancements in lightweight generative adversarial networks, their applications and datasets," in *Proc. 2023 IEEE 35th Int. Conf. Tools Artif. Intell. (ICTAI)*, 2023, pp. 269–278, doi: 10.1109/ICTAI59109.2023.00047.

[10] Y. R. Musunuri, C. Kim, O. S. Kwon, et al., "Object detection using ESRGAN with a sequential transfer learning on remote sensing embedded systems," *IEEE Access*, vol. 12, pp. 102313–102327, 2024, doi: 10.1109/ACCESS.2024.3432532.

[11] X. Yi, H. Pan, H. Zhao, et al., "Cycle generative adversarial network based on gradient normalization for infrared image generation," *Appl. Sci.*, vol. 13, no. 1, Art. no. 635, 2023, doi: 10.3390/app13010635.

[12] Z. Wang, C. Wang, Y. Chen, et al., "Target detection algorithm based on super-resolution color remote sensing image reconstruction," *J. Meas. Eng.*, vol. 12, no. 1, pp. 83–98, 2024, doi: 10.21595/jme.2023.23510.

[13] S. Cheng, Y. Han, Z. Wang, et al., "An underwater object recognition system based on improved YOLOv11," *Electronics*, vol. 14, no. 1, Art. no. 201, 2025, doi: 10.3390/electronics14010201.

[14] T. H. Vu, S. K. Jagatheesaperumal, M. D. Nguyen, et al., "Applications of generative AI (GAI) for mobile and wireless networking: A survey," *IEEE Internet Things J.*, vol. 12, no. 2, pp. 1266–1290, 2024, doi: 10.1109/JIOT.2024.3487627.

[15] A. Prabagaran, S. Kalavathi, N. M. Balamurugan, et al., "Secured heterogeneous data fusion using adversarial autoencoders in internet of military things," in *Proc. 2025 Int. Conf. Data Sci. Bus. Syst. (ICDSBS)*, 2025, pp. 1–6, doi: 10.1109/ICDSBS63635.2025.11031555.

[16] M. Chemmakha, O. Habibi, and M. Lazaar, "Towards a deep learning approach for IoT attack detection based on a new generative adversarial network architecture and gated recurrent unit," *J. Netw. Syst. Manage.*, vol. 32, no. 4, Art. no. 96, 2024, doi: 10.1007/s10922-024-09873-1.

[17] A. Guesmi, M. A. Hanif, B. Ouni, et al., "Physical adversarial attacks for camera-based smart systems: Current trends, categorization, applications, research challenges, and future outlook," *IEEE Access*, vol. 11, pp. 109617–109668, 2023, doi: 10.1109/ACCESS.2023.3321118.

[18] Z. Wang, B. Wang, C. Zhang, et al., "Robust feature-guided generative adversarial network for aerial image semantic segmentation against backdoor attacks," *Remote Sens.*, vol. 15, no. 10, Art. no. 2580, 2023, doi: 10.3390/rs15102580.

[19] C. Wang, S. Cheng, Y. Hou, et al., "Research on image damage assessment testing method based on deep learning," in *Proc. 2025 8th Int. Conf. Adv. Algorithms Control Eng. (ICAACE)*, 2025, pp. 1255–1260, doi: 10.1109/ICAACE65325.2025.11019570.

[20] M. Andreoni, W. T. Lunardi, G. Lawton, et al., "Enhancing autonomous system security and resilience with generative AI: A comprehensive survey," *IEEE Access*, vol. 12, pp. 109470–109493, 2024, doi: 10.1109/ACCESS.2024.3439363.

[21] A. B. Rashid, A. K. Kausik, A. al Hassan Sunny, et al., "Artificial intelligence in the military: An overview of the capabilities, applications, and challenges," *Int. J. Intell. Syst.*, vol. 2023, no. 1, Art. no. 8676366, 2023, doi: 10.1155/2023/8676366.

[22] M. A. Habib, M. A. H. Wadud, M. F. K. Patwary, et al., "Exploring progress in text-to-image synthesis: An in-depth survey on the evolution of generative adversarial networks," *IEEE Access*, vol. 12, pp. 178401–178440, 2024, doi: 10.1109/ACCESS.2024.3435541.

[23] N. K. Son, A. K. Sangaiah, D. V. Medhane, et al., "Enhancing resilience in edge IoT devices against adversarial attacks," *IEEE Consum. Electron. Mag.*, vol. 14, no. 4, pp. 48–56, 2024, doi: 10.1109/MCE.2024.3522524.

[24] C. Liu, X. Fu, Y. Wang, et al., "Overcoming data limitations: A few-shot specific emitter identification method using self-supervised learning and adversarial augmentation," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 500–513, 2023, doi: 10.1109/TIFS.2023.3324394.

[25] X. Liu, G. Li, Z. Zhao, et al., "EAF-WGAN: Enhanced alignment fusion-Wasserstein generative adversarial network for turbulent image restoration," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 10, pp. 5605–5616, 2023, doi: 10.1109/TCSVT.2023.3262685.

[26] J. Yang, H. Chu, L. Guo, et al., "A weighted-transfer domain-adaptation network applied to unmanned aerial vehicle fault diagnosis," *Sensors*, vol. 25, no. 6, Art. no. 1924, 2025, doi: 10.3390/s25061924.

[27] Y. Li, Q. Fan, H. Huang, et al., "A modified YOLOv8 detection network for UAV aerial image recognition," *Drones*, vol. 7, no. 5, Art. no. 304, 2023, doi: 10.3390/drones7050304.

[28] Y. Zou and G. Fang, "Underwater image enhancement based on generative adversarial network with multi-attention," in *Proc. 2024 5th Int. Conf. Big Data Artif. Intell. Internet Things Eng. (ICBAIE)*, 2024, pp. 351–354, doi: 10.1109/ICBAIE63306.2024.11116943.

[29] J. Zhang, Z. Liu, W. Jiang, et al., "Application of deep generative networks for SAR/ISAR: A review," *Artif. Intell. Rev.*, vol. 56, no. 10, pp. 11905–11983, 2023, doi: 10.1007/s10462-023-10469-5.

[30] X. Zhao, H. Zhang, C. Li, et al., "DVIF-Net: A small-target detection network for UAV aerial images based on visible and infrared fusion," *Remote Sens.*, vol. 17, no. 20, Art. no. 3411, 2025, doi: 10.3390/rs17203411.

[31] S. N. Khonina, N. L. Kazanskiy, I. V. Oseledets, et al., "Eyes of the future: Decoding the world through machine vision," *Technologies*, vol. 13, no. 11, Art. no. 507, 2025, doi: 10.3390/technologies13110507.

[32] N. Manakitsa, G. S. Maraslidis, L. Moysis, et al., "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision," *Technologies*, vol. 12, no. 2, Art. no. 15, 2024, doi: 10.3390/technologies12020015.