remote sensing

MDPI

Article

SAMKD: A Hybrid Lightweight Algorithm Based on Selective Activation and Masked Knowledge Distillation for Multimodal Object Detection

Ruitao Lu [1], Zhanhong Zhuo [1], Siyu Wang [2], Jiwei Fan [1], Tong Shen [1] and Xiaogang Yang [1,*]

[1] College of Missile Engineering, Rocket Force University of Engineering, Xi'an 710025, China; lrt19880220@163.com (R.L.); zhanhongzhuo@yeah.net (Z.Z.); fjw19900619@163.com (J.F.); shentong_1999@163.com (T.S.)
[2] School of Electrical and Control Engineering, Xuzhou University of Technology, Xuzhou 221018, China; wsy960328@163.com
* Correspondence: doctoryxg@163.com

Highlights

What are the main findings?

- A hybrid lightweight framework (SAMKD) is proposed for multimodal UAV object detection, which integrates selective activation pruning with task-aware knowledge distillation to achieve a stable balance between computational efficiency and detection accuracy.
- A new multimodal UAV aerial imagery dataset, M2UD-18K, is constructed, containing 18,294 aligned visible–infrared image pairs with nine target categories and 148,514 labeled bounding boxes.

What are the implications of the main findings?

- Extensive experimental results demonstrate that the proposed SAMKD achieves a more stable trade-off between model compactness and detection performance through a staged hybrid compression strategy, outperforming independent pruning or distillation methods.
- The newly constructed M2UD-18K dataset serves as a benchmark for evaluating lightweight multimodal detectors, providing reliable data support for assessing algorithm robustness and generalization in real-world UAV scenarios.

Abstract

Multimodal object detection is currently a research hotspot in computer vision. However, the fusion of visible and infrared modalities inevitably increases computational complexity, making most high-performance detection models difficult to deploy on resource-constrained UAV edge devices. Although pruning and knowledge distillation are widely used for model compression, applying them independently often leads to an unstable accuracy–efficiency trade-off. Therefore, this paper proposes a hybrid lightweight algorithm named SAMKD, which combines selective activation pruning with masked knowledge distillation in a staged manner to improve efficiency while maintaining detection performance. Specifically, the selective activation network pruning model (SAPM) first reduces redundant computation by dynamically adjusting network weights and the activation state of input data to generate a lightweight student network. Then, the mask binary classification knowledge distillation (MBKD) strategy is introduced to compensate for this degradation by guiding the student network to recover missing representation patterns under masked feature learning. Moreover, MBKD reformulates classification

Check for updates

Academic Editors: Jianming Hu, Xiyang Zhi, Yongqiang Mao and Longfei Ren

Received: 28 December 2025
Revised: 23 January 2026
Accepted: 27 January 2026
Published: 1 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

logits into multiple foreground–background binary mappings, effectively alleviating the severe foreground–background imbalance commonly observed in UAV aerial imagery. This paper constructs a multimodal UAV aerial imagery object detection dataset, M2UD-18K, which includes 9 types of targets and over 18,000 pairs. Extensive experiments show that SAMKD performs well on the self-constructed M2UD-18K dataset, as well as the public DroneVehicle dataset, achieving a favorable trade-off between detection accuracy and detection speed.

Keywords: multimodal object detection; lightweight; selective activation; knowledge distillation; edge device deployment

1. Introduction

Deep learning has achieved remarkable progress in computer vision and related fields, largely driven by increasingly deep and wide neural network architectures [1–4]. By stacking more layers and parameters, deep neural networks (DNNs) are able to extract richer hierarchical representations, leading to significant accuracy improvements. However, such performance gains are accompanied by a dramatic increase in computational cost and memory consumption, resulting in severe computation–storage bottlenecks [5,6]. Due to strict constraints on processing power, memory capacity, and energy consumption, embedded edge devices often fail to support the deployment of large-scale deep learning models [7,8].

Multimodal UAV aerial imagery object detection aims to accurately localize and recognize targets captured by unmanned aerial vehicles (UAVs), such as transport vehicles, ships, and airplanes [9,10]. By fusing complementary information from visible and infrared sensors mounted on UAV platforms [11], multimodal detection can significantly improve robustness under challenging conditions such as low illumination, background clutter, and occlusion. Consequently, visible–infrared fusion–based UAV detection has become a research hotspot [12], and a series of representative methods, including MROD-YOLO [13], SuperYOLO [14], DEYOLO [15], CFT [16], and TPDTNet [17], have demonstrated impressive detection performance. However, multimodal fusion inevitably introduces additional computational overhead, since parallel feature extraction and cross-modal interaction substantially increase network complexity. This makes most state-of-the-art multimodal detection models difficult to deploy on resource-constrained UAV edge devices.

To alleviate the conflict between detection performance and deployment efficiency, lightweight strategies have been extensively studied for multimodal UAV object detection [18,19]. Achieving an effective balance between large-scale neural models and limited edge-device resources has become a significant challenge. Existing lightweight approaches mainly include knowledge distillation [20–22], model quantization [23,24], network pruning [25,26], and lightweight architecture design [27,28]. Although these techniques have been extensively studied and even combined in prior work, their integration in multimodal UAV object detection is often driven by generic compression objectives, with limited emphasis on how pruning-induced information loss should be compensated during subsequent distillation.

Knowledge distillation (KD) [29,30] transfers knowledge from a high-capacity teacher network to a compact student network, enabling the student to approach the teacher's performance with fewer parameters. However, KD-based methods face notable challenges in distillation structure design and cross-modality adaptation, especially for object detection tasks with complex heads and multi-branch outputs. Moreover, distillation alone

does not fundamentally reduce the computational burden of the student network if its structure remains unchanged, which limits its effectiveness for real-time edge deployment. Network pruning [31,32], on the other hand, directly reduces model size and computation by removing low-importance parameters, channels, or neurons. While pruning is effective in eliminating redundant computation, it inevitably causes partial semantic degradation and performance loss due to information removal.

Recent surveys have systematically summarized hybrid compression strategies that combine pruning with other techniques, including knowledge distillation, indicating rapid progress in this direction. However, for multimodal UAV object detection, most existing hybrid methods focus on general compression goals, while the task-specific challenges—such as multimodal feature redundancy, activation-level sparsity, and severe foreground–background imbalance—are rarely jointly considered in the compression design.

Therefore, this paper proposes a novel hybrid lightweight algorithm named SAMKD. SAMKD integrates a selective activation network pruning model (SAPM) and a mask binary classification knowledge distillation (MBKD) module into a unified two-stage compression framework, aiming to explicitly exploit the complementary strengths of pruning and distillation for resource-constrained UAV edge deployment. In the first stage, SAPM reduces the computational burden of multimodal detection models by selectively suppressing low-contribution activations and eliminating redundant computations, thereby producing an efficient, lightweight student network. However, such activation-level pruning may inevitably introduce partial semantic degradation due to information removal. To compensate for this performance loss, the second stage adopts MBKD to guide the pruned student model to recover missing representation patterns via masked feature learning. Moreover, MBKD reformulates the classification logits into multiple foreground–background binary mappings and employs a binary cross-entropy distillation objective, which effectively alleviates the severe foreground–background imbalance in UAV aerial imagery and improves robustness. Through this staged design, SAMKD forms a self-compensatory optimization process, enabling a more stable trade-off between computational reduction and accuracy retention than applying pruning or distillation alone.

The main contributions of this paper are summarized as follows:

1. We propose a novel hybrid lightweight algorithm, SAMKD, which achieves an effective trade-off between detection accuracy and computational efficiency through a two-stage compression framework consisting of pruning followed by distillation.
2. To reduce the computational complexity and model size, a selective activation network pruning model (SAPM) is proposed. By dynamically regulating the activation state of neurons, channels, or weights, it maximally preserves the performance of multimodal detection models.
3. A new mask binary classification knowledge distillation (MBKD) is introduced. By designing random masks and the binary classification distillation loss function, MBKD addresses the foreground-background sample imbalance problem faced by multimodal object detection and mitigates the semantic degradation caused by pruning.
4. We construct a new multimodal UAV aerial imagery object detection dataset, M2UD-18K, which includes 9 types of objects, 18,294 pairs of visible and infrared images, and 148,514 labeled boxes. The results on M2UD-18K indicate that the performance of SAMKD surpasses that of SOTA models.

The rest of this article is arranged as follows. The related works are briefly reviewed in Section 2. The details of the proposed method are described in Section 3. Experiments and evaluations are provided in Section 4. A discussion of the experimental results and limitations is provided in Section 5. A conclusion about this article is given in Section 6.

2. Related Work

To reduce the computational complexity and storage overhead of detection models while maintaining their performance as much as possible, researchers have proposed a variety of lightweight neural network approaches. These include knowledge distillation–based methods, model quantization–based methods, network pruning–based methods, and lightweight architecture–based methods.

2.1. Distillation-Based Methods

In 2015, Hinton et al. [33] introduced the widely recognized concept of knowledge distillation. Its core idea is to transfer the knowledge from a pre-trained large model (the teacher model) to a lightweight small model (the student model). The teacher model guides the training process of the student model by generating soft labels, which represent output probability distributions, enabling the student model to learn more generalized features with fewer parameters. Romero et al. [18] proposed a feature-based knowledge distillation method, which aims to enhance performance by enabling the student model to mimic the intermediate-layer features of the teacher model. In ref. [21], knowledge distillation is conducted through an attention mechanism, allowing the student model to mimic the attention feature map of the teacher model. Meng et al. [22] transferred knowledge from a complex teacher to a lightweight student via multi-stage distillation while explicitly injecting geometric shape priors and curvature-guided constraints to achieve efficient and accurate edge detection for satellite targets. The ref. [34] proposed a self-distillation method that does not require an independent teacher model, which enhances performance through the model's own knowledge. The methods of CWD [35], FGD [36], and MGD [37] significantly enhance the performance of the student model by allowing it to mimic different intermediate feature representations of the teacher model. However, there are significant difficulties in structural design for such methods. Moreover, there are significant challenges in reducing the computational complexity of student networks while maintaining high performance, as well as in selecting distillation types.

2.2. Model Quantization-Based Methods

The method based on model quantization compresses models by reducing the numerical precision of model parameters. It converts floating-point numbers (e.g., 32-bit or 64-bit) in the model to lower-precision numbers (e.g., 8-bit or lower), thereby reducing computational complexity and memory overhead. The targets of quantization include weights, activation functions, and gradients. Rastegari et al. [23] proposed a binary neural network by quantizing weights and activations to +1 or -1. To better adapt to parameter distributions and reduce quantization errors, the model was compressed in [24] via clustering and Huffman coding. Jacob et al. [38] quantized the parameters to 8-bit integers, achieving a good balance between precision and efficiency. HAQ [39] employed different quantization precisions for different layers or parameters to enable flexible adjustment of quantization precision. However, the efficient inference of quantization models relies on the computing engines, and the efficiency of low-precision operations heavily depends on hardware support. Different hardware platforms may require different quantization strategies.

2.3. Network Pruning-Based Methods

The goal of pruning is to reduce the model parameters and computational complexity by removing unimportant weights, neurons, or channels in neural networks, while retaining model performance as much as possible. Guo et al. [40] proposed a network compression method called dynamic network surgery, which can effectively reduce network complexity through dynamic connection pruning. In ref. [41], a network slimming

algorithm for convolutional neural networks was presented, and the results demonstrated the effectiveness of the method using several state-of-the-art CNN models, including VGGNet, ResNet, and DenseNet. To explore weights that minimize loss while satisfying pruning cost conditions, a "Learning to Compress" algorithm is introduced in ref. [42] for neural network pruning. To remove entire neurons and their connections, Lin et al. [25] proposed an adaptive pruning method based on neuron activations, dynamically adjusting the pruning ratio. Hu et al. [26] used the number of zero activations in filters as a pruning metric. Lin et al. [43] used the ranking of mappings to determine whether to prune filters. Literature [44] combined filter pruning with low-rank decomposition to solve the compression problem of the last convolutional layer in ResBlock. These methods may result in the loss of semantically important information by pruning weights or channels, thereby affecting overall model performance. Additionally, selecting an appropriate pruning ratio and pruning location remains a critical issue.

2.4. Lightweight Structure-Based Methods

The core idea of these methods is to design efficient convolution operations and modules to reduce the parameters and computational complexity. MobileNet [27] first proposed depth-wise separable convolution, decomposing standard convolution into depth-wise convolution and point-wise convolution to reduce computational load. Based on the work in ref. [27], MobileNetV2 [28] introduced an inverted residual structure and linear bottleneck, further enhancing performance and efficiency. ShuffleNet [45] introduced channel shuffle operations to address the information isolation issue in group convolutions. EfficientNet [46] achieved a balance between model accuracy and efficiency through a compound scaling strategy. EdgeNeXt [47] combined the advantages of CNNs and Transformers to propose a lightweight VIT. CSPNet [48] reduced computation and improved propagation efficiency through cross-stage partial connections, while maintaining high performance. Such methods often require a deep understanding of network architecture and task characteristics, with a complex and time-consuming design process. Furthermore, lightweight networks suffer from performance bottlenecks, resulting in insufficient model expressiveness.

3. Methodology

3.1. Overview of Our Method

To address the challenges of DNNs on edge devices and the trade-off between computational complexity and model performance, we propose a hybrid lightweight algorithm named SAMKD, as illustrated in Figure 1.

Firstly, this paper proposes a selective activation network pruning model (SAPM), designing two selective activation mechanisms: selective activation with primary weights (SAPW) and selective activation with primary inputs (SAPI). For the unactivated neurons or channels, pruning operations are performed to prune the relatively complex pre-trained teacher network into a lightweight student network. Then, a mask binary classification knowledge distillation (MBKD) method is designed, introducing random masks into the student network to enable it to generate features using only partial information, thereby enhancing its learning ability. MBKD formulates multi-class problems as multiple binary classification problems, effectively addressing the imbalance between target and background classes during object detection and improving detection accuracy.

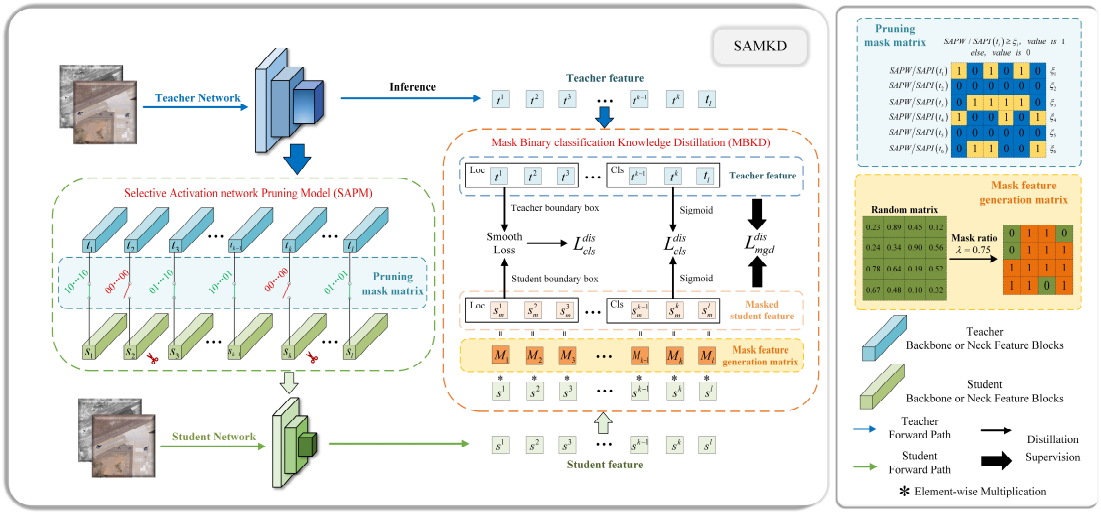


Figure 1. Overall framework of the SAMKD.

3.2. Selective Activation Network Pruning Model

The selective activation network pruning model (SAPM) is primarily composed of a batch normalization (BN) layer, a selective activation layer, and an activation management unit (AMU), as illustrated in Figure 2. AMU is used to independently manage the activation state of neurons in each selectively activated layer. Specifically, neurons in each layer are first screened through an AMU to predict which neurons need to be activated. Subsequently, activated neurons perform weighted operations and nonlinear mapping processing, while non-activated neurons directly output zero. Since the computation of neural networks mainly focuses on weighted operations and nonlinear mappings, this design can significantly reduce the computational load of the network. We propose two selective activation mechanisms: one is selective activation with primary weights (SAPW), and the other is selective activation with primary inputs (SAPI). Additionally, selective activation is implemented using mask general matrix multiplication.

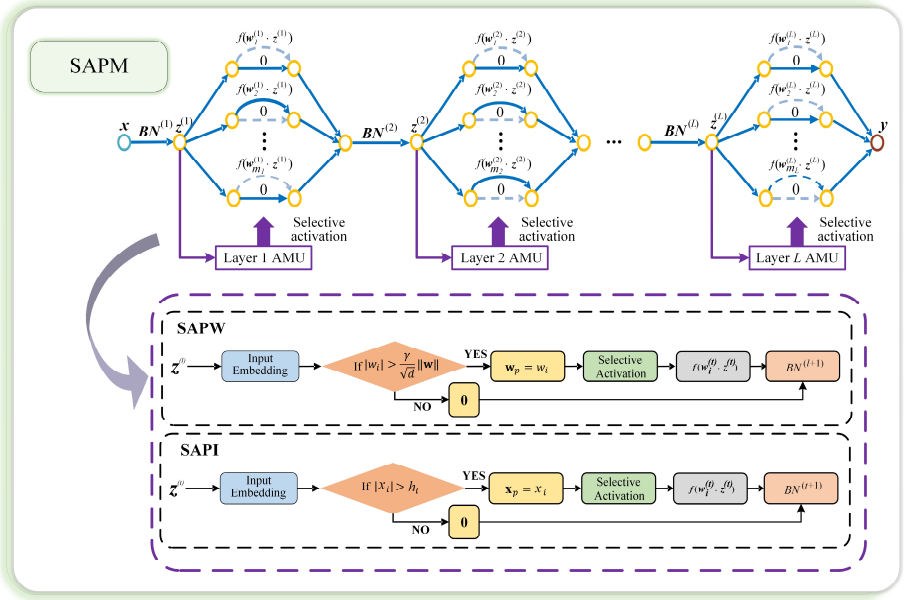


Figure 2. The schematic of SAPM.

3.2.1. Mask General Matrix Multiplication

In neural networks, a large number of computational operations involve frequent use of multiplication and addition. If inactive neurons can be identified and determined in advance, their corresponding positions in the output matrix can be marked, thereby avoiding redundant computations in the related matrix multiplication. This strategy not only reduces the amount of multiplication and addition operations but also decreases the computational complexity of matrix multiplication. Considering matrix multiplication can be denoted by:

$$\boldsymbol{C}_{n\times m} = \boldsymbol{A}_{n\times k}\boldsymbol{B}_{k\times m} \tag{1}$$

where $\boldsymbol{C}_{n\times m}$, $\boldsymbol{A}_{n\times k}$, and $\boldsymbol{B}_{k\times m}$ represent an $n \times m$-dimensional matrix, an $n \times k$-dimensional matrix, and a $k \times m$-dimensional matrix, respectively.

Expand (1) into:

$$\boldsymbol{C}_{n\times m} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n][\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m] \tag{2}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are k-dimensional vectors.

We introduce a binary mask function in the selective activation layer to control the activation state of each neuron or channel. For each neuron or channel, the binary mask function outputs a binary mask matrix, where the elements are 0 or 1, indicating whether the neuron or channel is activated or not. Unactivated neurons or channels will be pruned and will not participate in subsequent neural network computations, thereby reducing the computational load.

The activation state of a neuron is defined by the output value processed by the ReLU activation function. Specifically, the output of a single neuron under the ReLU activation function can be denoted as:

$$\widetilde{\kappa} = \mathrm{ReLU}(\kappa) = \begin{cases} \kappa, \ \kappa > 0 \\ 0, \ \text{otherwise} \end{cases} \tag{3}$$

where $\widetilde{\kappa}$ represents the activation value of the neuron, and κ denotes the input value of the neuron.

As indicated by the output activations, $\widetilde{\kappa}$ typically exhibits sparsity, where only a portion of neurons have non-zero activations. Zero ratio refers to the proportion of activation values that are zero among all activation values. Effective activation ratio refers to the proportion of activation values that are greater than zero. The relationship between them can be represented as:

$$\theta = 1 - \varpi \tag{4}$$

where θ is zero ratio, and ϖ is effective activation ratio. The activation threshold T can be obtained through the zero ratio, in order to determine which neuron activation values are valid. The core of the algorithm lies in utilizing the distribution characteristics of activation values to find an appropriate threshold, such that activation values below the threshold are treated as zero.

The thresholds T can be obtained by the following steps:

Step1: Collect data. During the training or inference phase, record the activation values of each layer's neurons across multiple input samples. Suppose the set of activation values for a certain layer is denoted as $\widetilde{K} = [\widetilde{\kappa}_1, \widetilde{\kappa}_2, \ldots, \widetilde{\kappa}_i]$, where i represents the number of input samples.

Step2: Calculate the zero ratio. During network pruning, an effective activation ratio ϖ is provided, and the zero ratio θ is computed according to (4).

Step3: Determine the threshold T. In the activation value set $\widetilde{K}$, the threshold T is set such that the proportion of activation values less than b equals θ.

This method is applicable to the distribution of activation values in each layer of a multilayer neural network. Once the activation threshold is determined, the values of the binary mask matrix can be identified by comparing the activation values with the activation threshold.

3.2.2. Selective Activation with Primary Weights (SAPW)

After batch normalization, the input of each channel exhibits statistical independence, characterized by normal distribution features with similar mean and variance:

$$x_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{5}$$

We propose an activation determination mechanism based on primary weights. All primary weights of neurons are integrated into a sparse vector $\boldsymbol{\varphi}_p$, which has the same dimension as $\boldsymbol{\varphi}$. The elements corresponding to its primary weights are non-zero, while the remaining weights are classified as secondary weights, denoted by $\boldsymbol{\varphi}_r = \boldsymbol{\varphi} - \boldsymbol{\varphi}_p$. The net activation value of a neuron for an input signal x satisfies:

$$\kappa = \kappa_p + \kappa_r = {\boldsymbol{\varphi}_p}^{\mathrm{T}}\boldsymbol{x} + {\boldsymbol{\varphi}_r}^{\mathrm{T}}\boldsymbol{x} \tag{6}$$

where κ_r is the dot product of the secondary weights and the input, and κ_p represents the dot product of the primary weights and the input. Based on κ_p, the probability of the output being less than zero can be calculated. SAPW determines that the neuron is activated based on whether this probability exceeds a preset threshold. Expanding κ_r, we obtain:

$$P\{\kappa_p + \kappa_r < 0\} = \int_{y_r < -y_p} p(\kappa_p + \kappa_r) dy_r \tag{7}$$

Assuming that the inputs are independent of each other, we can obtain from (5) that:

$$\kappa_r = {\boldsymbol{\varphi}_r}^{\mathrm{T}}\boldsymbol{x} \sim N\left({\boldsymbol{\varphi}_r}^{\mathrm{T}}\boldsymbol{\mu}, \left(\boldsymbol{\varphi}_r^2\right)^{\mathrm{T}}\boldsymbol{\varepsilon}^2\right) \tag{8}$$

where $\boldsymbol{\mu}$ is the expectation of $\boldsymbol{x}$, and $\boldsymbol{\varepsilon}^2$ is a vector consisting of the variance of each element $\boldsymbol{x}$.

Substituting (8) into (7) yields the integral form of the standard normal distribution density:

$$P\{\kappa_p + \kappa_r < 0\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{\kappa_p + {\boldsymbol{\varphi}_r}^{\mathrm{T}}\boldsymbol{\mu}}{\sqrt{\left(\varphi_r^2\right)^{\mathrm{T}}\boldsymbol{\varepsilon}^2}}} e^{-\frac{s^2}{2}} ds \tag{9}$$

Therefore, when the probability is greater than the threshold A, the corresponding coefficient β can be obtained by looking up the normal distribution function table, that is:

$$\frac{\kappa_p + {\boldsymbol{\varphi}_r}^{\mathrm{T}}\boldsymbol{\mu}}{\sqrt{\left(\boldsymbol{\varphi}_r^2\right)^{\mathrm{T}}\boldsymbol{\varepsilon}^2}} < \beta \tag{10}$$

The condition for $P\{\kappa < 0\} > \xi$ is that κ_p is satisfied:

$$\kappa_p + {\boldsymbol{\varphi}_r}^{\mathrm{T}}\boldsymbol{\mu} < \beta\sqrt{\left(\boldsymbol{\varphi}_r^2\right)^{\mathrm{T}}\boldsymbol{\varepsilon}^2} \tag{11}$$

SAPW is often applied to prune fully connected layers and self-attention layers to reduce redundant parameters and optimize computational efficiency.

During the weight training process of SAPW, for specific input samples, only the gradients of activated neurons need to be calculated, thereby reducing the computational overhead of training. We introduce the statistical method of batch normalization, which achieves more efficient training optimization by setting an average factor to perform weighted processing on the mean and variance of each batch of input:

$$\begin{cases} \mu \leftarrow (1-\alpha)\mu + \alpha \mathrm{E_B}[x] \\ \sigma^2 \leftarrow (1-\alpha)\sigma^2 + \alpha \mathrm{Var_B}[x] \end{cases} \tag{12}$$

where $\mathrm{E_B}[x]$ and $\mathrm{Var_B}[x]$ represent the mean and variance of the current batch x, respectively. Two strategies can be adopted for updating rate α: one is the iterative mode based on the inverse ratio of the number of samples, and the other is the iterative mode using a fixed momentum factor.

The distribution of weights will be changed during the updating process, therefore requiring a re-screening of the primary weights.

Since weights with larger absolute values or higher energy proportions are always distributed within the range of $[\|\boldsymbol{\varphi}\|/\sqrt{d}, \|\boldsymbol{\varphi}\|]$, the selection criteria for the main weights can be expressed as:

$$\varphi_p = \left\{ w_i, \text{if } |\varphi_i| > \frac{\gamma}{\sqrt{d}} \|\boldsymbol{\varphi}\|, \text{otherwise } 0 \right\} \tag{13}$$

where φ_i is the i-th element of vector $\boldsymbol{\varphi}$, $|\varphi_i|$ is the absolute value of φ_i, and $\gamma \in (0.3, 1]$ is the scaling factor that controls the proportion of primary weights used for activation prediction.

In practice, γ is not fixed analytically but determined empirically. We perform a grid search over a small set of candidate γ values on the validation set and select the value that achieves the best balance between detection accuracy and computational efficiency. This strategy ensures that the selected γ adapts to both the network architecture and the data distribution, while avoiding excessive performance degradation caused by overly aggressive pruning.

3.2.3. Selective Activation with Primary Inputs (SAPI)

As a dual method of SAPW, SAPI predicts the activation state of neurons by utilizing the response of a high-energy input subset. In the prediction phase, the main weight mask in SAPW does not need to be updated. However, in SAPI, due to the time-varying nature of the input, the main input mask requires online estimation.

Based on the preset threshold h, the main input components are filtered from the current input $\boldsymbol{x}$ to generate the corresponding sparse vector, as shown below:

$$\boldsymbol{x}_p = \{x_i, \text{if } |x_i| > h_i, \text{otherwise } 0\} \tag{14}$$

where x_i and h_i are the i-th elements of x and h, respectively.

Subsequently, the activation of each neuron is determined by comparing it with a preset threshold. In the inference phase, the main input of SAPI dynamically changes with the input data. Therefore, SAPI cannot directly use the input function, but instead employs an activation threshold.

Based on the remaining input being $\boldsymbol{x}_r = \boldsymbol{x} - \boldsymbol{x}_p$, after several batches of sampling statistics, the mean and variance are $\mathrm{E}[\boldsymbol{x}_r] = \boldsymbol{\mu}_r$ and $\mathrm{Var}[\boldsymbol{x}_r] = \boldsymbol{\varepsilon}_r^2$, respectively. According

to the central limit theorem, when the dimension of the input $\boldsymbol{x}$ is high, the response of the remaining inputs approximately satisfies:

$$\kappa_r = \boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{x}_r \ \sim \ N\left(\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{\mu}_r, \left(\boldsymbol{\varphi}^2\right)^{\mathrm{T}}\boldsymbol{\varepsilon}_r^2\right) \tag{15}$$

where $\boldsymbol{\mu}_r$ is the expectation of $\boldsymbol{x}_r$, $\boldsymbol{\varphi}^2$ is the vector consisting of the squares of all elements in $\boldsymbol{\varphi}$, and $\boldsymbol{\varepsilon}_r^2$ is the vector consisting of the variances of all elements in $\boldsymbol{x}_r$.

When the probability threshold is ξ, the coefficient β corresponding to this probability can be obtained based on the standard normal distribution function table. When the condition $P\{\kappa < 0\} > \xi$ holds, κ_p satisfies:

$$\kappa_p + \boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{\mu}_r < \beta\sqrt{(\boldsymbol{\varphi}^2)^{\mathrm{T}}\boldsymbol{\varepsilon}_r^2} \tag{16}$$

SAPI can reduce the number of convolutional operations while maintaining image recognition accuracy, and it is thus applied to optimize the pruning of convolutional layers.

During the training process, SAPI only performs gradient computation on the activated neurons in the current input, effectively reducing the computational overhead of weight updates. The training of SAPI requires online statistics of the mean and variance of the input, which can be noted by:

$$\begin{cases} \mu_r \leftarrow (1-\alpha)\mu_r + \alpha \mathrm{E}_{\mathrm{B}}[x_r] \\ \sigma_r^2 \leftarrow (1-\alpha)\sigma_r^2 + \alpha \mathrm{Var}_{\mathrm{B}}[x_r] \end{cases} \tag{17}$$

$$Q^2 \leftarrow (1-\alpha)Q^2 + \alpha \mathrm{RMS}_{\mathrm{B}}[x] \tag{18}$$

where $\mathrm{E}_{\mathrm{B}}[x_r]$ and $\mathrm{Var}_{\mathrm{B}}[x_r]$ represent the mean and variance of the current batch x_r, respectively, and $\mathrm{RMS}_{\mathrm{B}}[x]$ denotes the root mean square of the current batch x. The update rate α can be used in a fixed momentum factor.

Based on the design of weight selection thresholds in SAPW, the main input thresholds for each input channel in SAPI are designed as follows:

$$h_i = \gamma Q_i \tag{19}$$

where Q_i represents the root mean square of the i-th input channel, and $\gamma > 0$ is a scaling factor satisfying $(0.3, 1]$. γ determines the threshold for selecting primary inputs based on their energy distribution. By scaling the RMS of each input channel, γ controls the sparsity level of the retained input components.

The value of γ is selected empirically using the validation set. We evaluate several candidate γ values and choose the one that provides a favorable trade-off between activation sparsity and detection performance.

3.3. Mask Binary Classification Knowledge Distillation

Based on the student network generated by SAPM, we propose a mask binary classification knowledge distillation (MBKD). Based on the teacher network guiding the feature recovery of the student network, MBKD enhances the representational ability of the student network and achieves neural network quantization. The flowchart is shown in Figure 3. Firstly, a random mask is introduced during the feature generation process, enabling the student network to utilize partial information for feature generation and enhancing its learning capability. Secondly, a binary classification distillation loss function is constructed, which formulates the multi-classification problem as multiple binary classification problems. This method effectively addresses the imbalance between foreground and back-

ground categories while resolving the inconsistency between the softmax and sigmoid protocols. It should be emphasized that the proposed masking operation is only introduced during training for knowledge distillation. No masks or additional modules are retained during inference, and thus MBKD does not introduce any extra computational overhead or structural modification at deployment time.

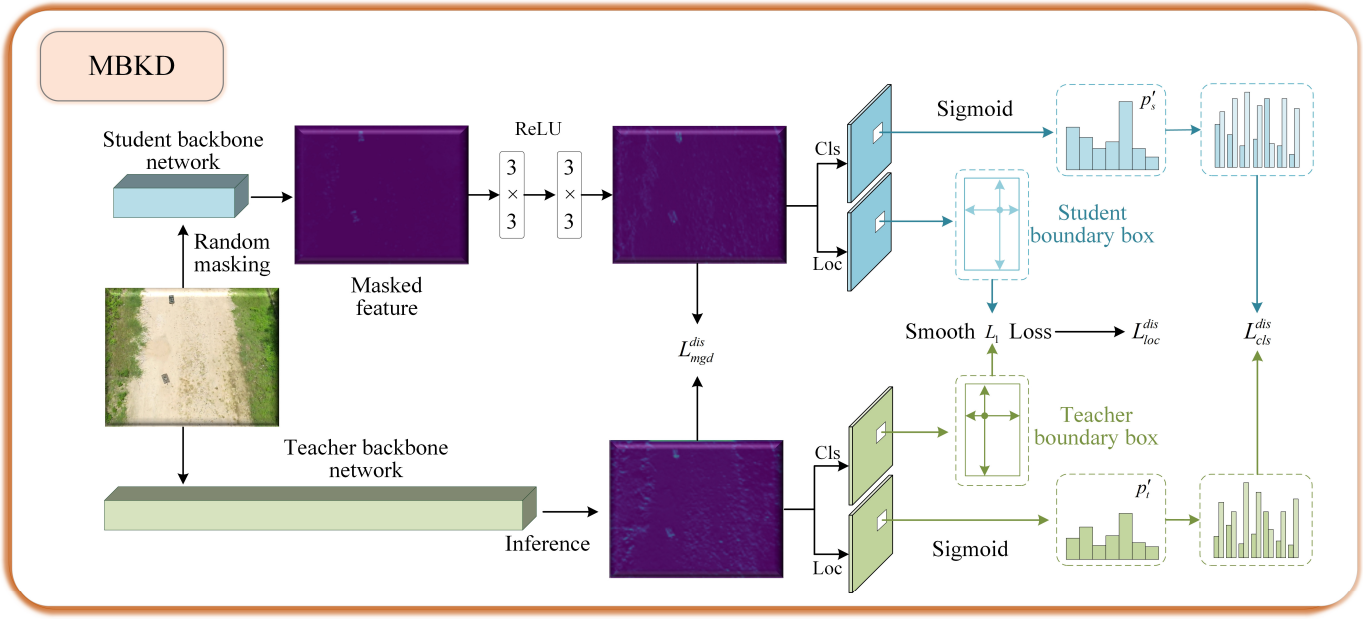


Figure 3. Schematic diagram of MBKD.

3.3.1. Mask Feature Generation

For convolutional neural networks, deeper features have a larger receptive field. Since the pixels of the feature map contain information from adjacent pixels, partial pixels can be used to recover the complete feature mapping. This paper proposes a mask-based feature generation method, which generates the features of the teacher network by partially masking the features of the student network. This method can help student networks achieve better performance.

Suppose $\boldsymbol{T}^l \in \mathbb{R}^{C\times H\times W}$ and $\boldsymbol{S}^l \in \mathbb{R}^{C\times H\times W} (l = 1, \ldots, L)$ represent the l-th feature map of the teacher and student, respectively. Firstly, we utilize an alignment layer f_{align} to align student features with teacher features. Specifically, a 1×1 convolutional layer is commonly used as an alignment layer, which can be expressed as:

$$f_{align}\left(\boldsymbol{S}^l\right) \rightarrow \boldsymbol{S}_{align}^l \tag{20}$$

where $\boldsymbol{S}_{align}^l$ denotes the aligned student features.

Then, we perform a random masking operation on student features. We apply the l-th random mask to cover the l-th feature map of the student network, generating a random mask matrix $\boldsymbol{M}^l$ with a masking ratio of λ. This can be formulated as

$$\boldsymbol{M}_{i,j}^l = \begin{cases} 0, & \text{if } R_{i,j}^l < \lambda \\ 1, & \text{otherwise} \end{cases} \tag{21}$$

where $R_{i,j}^l$ is a random number within the range (0, 1), and i and j are the horizontal and vertical coordinates of the feature map, respectively.

Next, the corresponding mask matrix is used to cover the feature map of the student network. We explore a simple generation model G to generate the feature map of teacher network by the masked feature map of the student network. It can be expressed as:

$$G\left(f_{align}\left(\boldsymbol{S}^{l}\right)\cdot\boldsymbol{M}^{l}\right)\rightarrow\boldsymbol{T}^{l} \tag{22}$$

where G is a module consisting of two convolutional layers and one activation layer, which can be denoted by

$$G(\cdot)=W_{l2}(\mathrm{ReLU}(W_{l1}(\cdot))) \tag{23}$$

where the convolutional layers W_{l1} and W_{l2} both use 3×3 convolutional layers, and the activation layer uses the ReLU activation function.

Based on the above analysis, we design mask generation distillation loss function:

$$L_{\mathrm{m}}^{\mathrm{dis}}(\boldsymbol{S},\boldsymbol{T})=\sum_{l=1}^{L}\sum_{k=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(\boldsymbol{T}_{k,i,j}^{l}-G\left(f_{align}\left(\boldsymbol{S}_{k,i,j}^{l}\right)\cdot\boldsymbol{M}_{i,j}^{l}\right)\right)^{2} \tag{24}$$

where L represents the total number of distillation layers, while C, H, W denote the dimensions of the feature map, $\boldsymbol{S}^{l}$ and $\boldsymbol{T}^{l}$ represent the features of the l-th layer in the student network and teacher network, respectively.

3.3.2. Binary Classification Distillation Loss

In UAV aerial images, targets (such as vehicles, aircraft, ships, etc.) occupy only a small number of pixels, while background pixels occupy the majority of the image. This results in a severe imbalance between target and background samples in target detection. During the training phase of multi-objective classification, the softmax protocol converts classification logical values into classification scores, assigning samples to k probabilities (where k represents the number of target categories, and the remainder denotes background). This method assigns a higher probability to the background, thereby affecting detection accuracy. This paper formulates the multi-classification problem as multiple binary classification problems to address the issue of imbalance between target and background categories.

Specifically, the training process generates classification maps of varying sizes. Labels are assigned to each position on the classification map, where positive samples are labeled as a single hot tensor and negative samples as a zero tensor. Suppose x represents the sample, $l\in\mathbb{R}^{n\times K}$ represents its classification logic, where K denotes the number of categories. To obtain the classification score for each position, the Sigmoid function is used to calculate $P=Prot_{\mathrm{sigmoid}}(l)$. For a sample x, there is a corresponding label tensor Y. Therefore, the cross-entropy loss between the classification score and the label can be calculated:

$$L_{\mathrm{cls}}(x)=\sum_{i=1}^{n}\sum_{j=1}^{K}L_{\mathrm{CE}}\left(P_{i,j},Y_{i,j}\right) \tag{25}$$

where $L_{\mathrm{CE}}\left(P_{i,j},Y_{i,j}\right)$ represents the cross-entropy loss at the i-th position and for the j-th class, which can be defined as follows:

$$L_{\mathrm{CE}}(P_{i,j},Y_{i,j})=\begin{cases}-log(P_{i,j}) & Y_{.,j}=1,\\ -log(1-P_{i,j}) & Y_{.,j}=0.\end{cases} \tag{26}$$

Let l^{t} and l^{s} represent the classification logic values from the teacher network and student network, respectively. The classification scores can be obtained by the softmax protocol, namely $P^{t}=Prot_{\mathrm{softmax}}\left(l^{t}\right)$ and $P^{s}=Prot_{\mathrm{softmax}}(l^{s})$. Calculate the classification distillation loss between P^{t} and P^{s} to encourage the student model to mimic the output of

the teacher model. This loss is typically defined as the KL divergence between the teacher scores and student scores:

$$L_{\text{cls}}^{\text{kl}}(x) = L_{\text{kl}}(P^t, P^s) \tag{27}$$

where $L_{cls}^{kl}(\cdot)$ represents the classification distillation loss, and $L_{kl}(\cdot,\cdot)$ represents the KL divergence.

The softmax protocol and sigmoid protocol are, respectively, defined as:

$$Prot_{\text{softmax}}(l^t) = \frac{e^{l^t}}{\sum_{i=1}^{K} e^{l_i^t}} \tag{28}$$

$$Prot_{\text{sigmoid}}(l^t) = \frac{1}{1+e^{-l^t}} \tag{29}$$

When n is a constant tensor of the same shape as l^t and satisfies $l^s = l^t + n$, we can obtain:

$$Prot_{\text{softmax}}(l^s) = \frac{e^{l^t+n}}{\sum_{i=1}^{K} e^{l_i^t+n}} = \frac{e^{l^t}\cdot e^n}{\sum_{i=1}^{K}\left(e^{l_i^t}\cdot e^n\right)} = \frac{e^{l^t}}{\sum_{i=1}^{K} e^{l_i^t}} = Prot_{\text{softmax}}(l^t) \tag{30}$$

When the distillation loss equals zero, the teacher model no longer transfers localized knowledge to the student model. Due to $Prot_{\text{sigmoid}}(l^s) \neq Prot_{\text{sigmoid}}(l^t)$, there is a gap between the classification scores of the teacher model and the student model during the inference process. When the score obtained by the student model is lower than that of the teacher model, it may predict incorrect class relationships. Therefore, the student model cannot inherit the correct predictive ability from the teacher model.

To address this issue, we formulate the classification logic map as multiple binary classification map. Based on $\widetilde{P}^t = Prot_{\text{sigmoid}}(l^t)$ and $\widetilde{P}^s = Prot_{\text{sigmoid}}(l^s)$, the binary classification scores of $\widetilde{P}^t$ and $\widetilde{P}^s$ can be calculated, with a size of $n \times K$. The classification distillation loss can be calculated based on these binary classification scores:

$$L_{\text{BCE}}\left(\widetilde{P}_{i,j}^s, \widetilde{P}_{i,j}^t\right) = -\left(\left(1-\widetilde{P}_{i,j}^t\right)\cdot\log\left(1-\widetilde{P}_{i,j}^s\right) + \widetilde{P}_{i,j}^t\cdot\log\left(\widetilde{P}_{i,j}^s\right)\right) \tag{31}$$

$$L_{\text{cls}}^{\text{dis}}(x) = \sum_{i=1}^{n}\sum_{j=1}^{K} L_{\text{BCE}}\left(\widetilde{P}_{i,j}^s, \widetilde{P}_{i,j}^t\right) \tag{32}$$

where $L_{\text{BCE}}(\cdot,\cdot)$ represents binary cross-entropy loss, $L_{\text{cls}}^{\text{dis}}(\cdot)$ represents classification distillation loss. $\widetilde{P}_{i,j}^s$ and $\widetilde{P}_{i,j}^t$ represent the i-th position and j-th category of $\widetilde{P}^s$ and $\widetilde{P}^t$, respectively.

We adopt a loss-weighted strategy to focus on distilling important samples, the calculation formula is as follows:

$$\omega = \left|\widetilde{P}^s - \widetilde{P}^t\right| \tag{33}$$

where $\omega = n \times K$. Each element in ω is weighted based on the classification distillation loss of sample x. Therefore, the binary classification distillation loss formula is expressed as:

$$L_{\text{cls}}^{\text{dis}}(x) = \sum_{i=1}^{n}\sum_{j=1}^{K} \omega_{i,j} L_{\text{BCE}}\left(\widetilde{P}_{i,j}^s, \widetilde{P}_{i,j}^t\right) \tag{34}$$

3.3.3. Total Distillation Loss Function

Object detection tasks typically involve two key tasks: classification and localization. In knowledge distillation, the training objective of the student network is to enhance

performance by learning from the outputs and intermediate features of the teacher network. Therefore, the total loss of the student network consists of three parts: classification loss, localization loss, and distillation loss. The classification loss $L_{\mathrm{m}}^{\mathrm{dis}}$ and distillation loss $L_{\mathrm{cls}}^{\mathrm{dis}}$ can be calculated by (24) and (34). The localization loss is used to optimize the prediction of the target bounding box. The localization loss employs the smooth L_1 loss, with the calculation formula as follows:

$$L_{\mathrm{loc}}^{\mathrm{dis}} = \sum_{i=1}^{N} \sum_{j \in \{x,y,w,h\}} smooth_{L1}\left(t_{i,j} - t_{i,j}^{*}\right) \tag{35}$$

where N is the number of samples, $t_{i,j}$ represents the bounding box coordinate offset predicted by the model, $t_{i,j}^{*}$ denotes the actual bounding box coordinate offset of the target. $smooth_{L1}$ is the smooth L_1 loss function, which can be calculated by

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{36}$$

The advantage of smooth L_1 loss lies in its sensitivity to small errors but insensitivity to large errors, thus avoiding over-optimization for outliers during training.

In summary, the total loss function $L_{\mathrm{total}}^{\mathrm{dis}}$ is the weighted sum of the mask generation loss $L_{\mathrm{m}}^{\mathrm{dis}}$, the binary classification loss $L_{\mathrm{cls}}^{\mathrm{dis}}$, and the smooth L_1 localization loss $L_{\mathrm{loc}}^{\mathrm{dis}}$. Balance these three parts of loss by introducing hyperparameters A, B, and C, the total loss function is denoted:

$$L_{\mathrm{total}}^{\mathrm{dis}} = \alpha L_{\mathrm{m}}^{\mathrm{dis}} + \beta L_{\mathrm{cls}}^{\mathrm{dis}} + \gamma L_{\mathrm{loc}}^{\mathrm{dis}} \tag{37}$$

4. Experiment

4.1. Settings

- **Dataset.** We have constructed a novel multimodal UAV aerial imagery object detection dataset, M2UD-18K, for evaluating algorithms. M2UD-18K is a comprehensive dataset encompassing multiple scenes and perspectives, containing 18,294 pairs of optical and infrared images. It includes 9 types of targets (fighter, patrol vessel, tracked vehicle, car, truck, transport vehicle, airplane, ship and off-road truck), with a total of 148,514 labeled bounding boxes. The statistical information of the M2UD-18K is presented in Table 1, and the sample images are shown in Figure 4. Each pair of optical and infrared images undergoes rigorous alignment and annotation to ensure consistency in the location and category of the target.

Table 1. Description of the M2UD-18K datasets.

Name	Scope	Number of Images	Target Types	Modal Types	Main Challenges
M2UD-18K	Multimodal UAV aerial imagery object detection	18,294 pairs of images, 148,514 annotation boxes	9 types	Optics, infrared	Background interference, small target scale

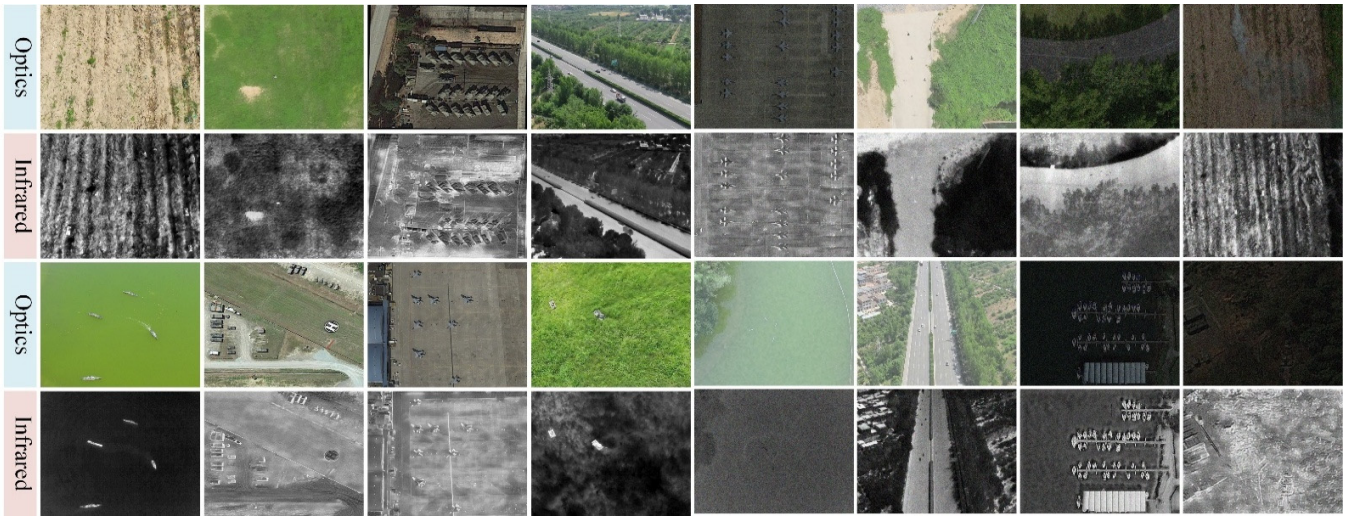
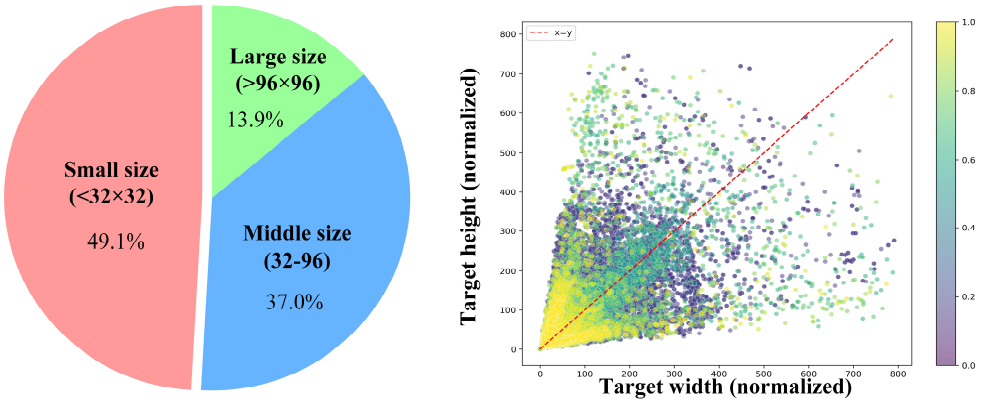


Figure 4. The sample images of M2UD-18K.

The M2UD-18K provides high-quality data support for time-sensitive object detection in complex environments, effectively evaluating and enhancing the robustness and generalization capabilities of algorithms. The following analysis is conducted on the M2UD-18K from three dimensions: target scale distribution, target species quantity, and target center point distribution.

The targets with an area less than 32 × 32 pixels are defined as small targets, those with an area between 32 × 32 and 96 × 96 pixels are considered medium targets, and those with an area greater than 96 × 96 pixels are classified as large targets. The distribution of target sizes in M2UD-18K is shown in Figure 5a. Small targets account for the highest proportion, reaching 49.1%, medium targets account for 37.0%, while large targets have the smallest proportion, only 13.9%. Small targets have a significant numerical advantage in the dataset, leading to an imbalance in the distribution of target-background samples. The normalized distribution of targets is shown in Figure 5b. Most targets have small widths and heights. As the width and height of the targets increase, the distribution range gradually expands, exhibiting diverse target scale characteristics.



(a) Target size distribution **(b) Normalized distribution of targets**

Figure 5. Analysis of the distribution of target scales in the M2UD-18K.

The distribution of target species quantity is shown in Figure 6. The dataset exhibits a class imbalance feature, where the number of ship targets is significantly higher than that of other classes. This imbalance facilitates testing the robustness of the model and evaluating its generalization performance across different categories.

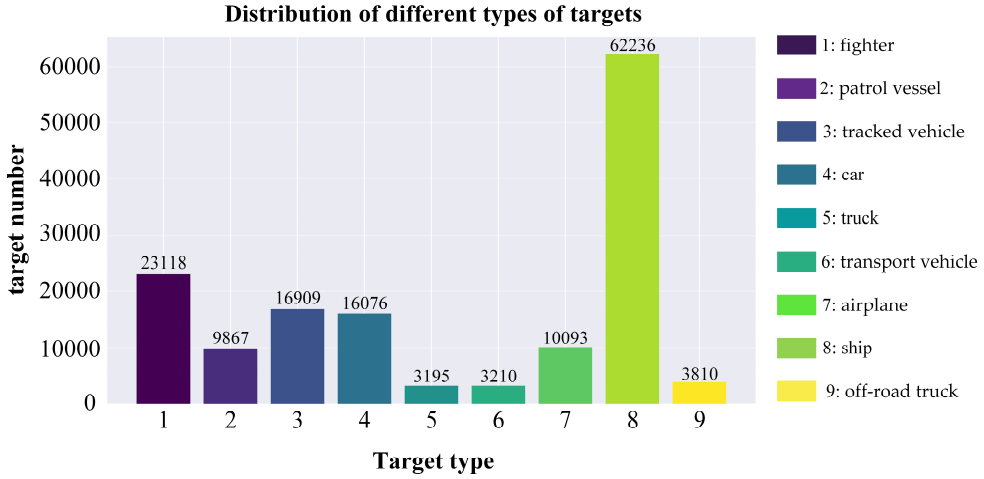


Figure 6. Distribution of the number of targets by type in the M2UD-18K.

- **Implementation Details.** The experiments of the study are conducted on a server with an NVIDIA RTX 4090 GPU, 24 GB of memory, and Python code running on Ubuntu 24.04. With CUDA11.6, PyTorch1.12, Python 3.8 and other widely used deep learning and image processing libraries installed, our development environment is PyCharm2022. Images from the M2UD-18K were randomly divided into training, validation and testing sets at a 6:2:2 ratio. We train the network via the stochastic gradient descent (SGD) optimizer with some hyperparameters set as follows. The initial learning rate is 0.01, the weight decay rate is 0.0005, the optimization momentum is 0.937, the number of iterations is 300, and the number of early stopping patience rounds is set to 100; i.e., the number of rounds to be waited for after the effect is no longer boosted.
- **Evaluation Metrics.** In this work, we conducted a comprehensive analysis of the algorithms using evaluation metrics such as Precision (Pre), Recall (Rec), F1 score (F1), Average Precision (AP), mean Average Precision (mAP), Frames Per Second (FPS), and Parameters (Params). They are used to assess the performance, real-time capability, and efficiency of the model, as defined below:

$$\text{Pre} = \frac{TP}{TP + FP} \tag{38}$$

$$\text{Rec} = \frac{TP}{TP + FN} \tag{39}$$

$$\text{F1} = 2 \times \frac{\text{Pre} \times \text{Rec}}{\text{Pre} + \text{Rec}} \tag{40}$$

$$\text{AP}_c = \int_0^1 P(r)dr \tag{41}$$

$$\text{mAP} = \frac{1}{N} \sum_{c=1}^{N} \text{AP}_c \tag{42}$$

where TP, FP, and FN denote true positive, false positive, and false negative, respectively. AP_c represents average precision for class c. $P(r)$ is the precision at the recall r and N is the number of classes. And mAP_{50} represents the mAP at an IoU = 0.5, while $\text{mAP}_{50:95}$ denotes the mAP across IoU thresholds ranging from 0.5 to 0.95, with a step size of 0.05.

4.2. Ablation Study

We conducted ablation analysis on the activation threshold in the pruning module and the masking ratio in the distillation module. To verify the effectiveness of the modules, we compared the model performance under different combinations of lightweight modules. The baseline model adopted the MROD-YOLO [13] algorithm.

4.2.1. Comparison Experiment of Activation Thresholds

In the process of model pruning, the activation threshold serves as a criterion for determining the importance of channels, which is calculated based on the activation values of channels or neurons. If the activation value of a certain channel or neuron is below the activation threshold, the channel or neuron will be pruned. Firstly, the activation values of the channels are sorted, and the top k channels with the highest activation values are retained, where k is determined by the pruning retention ratio (PRR). Secondly, the activation threshold is equal to the activation value of the k-th channel, which is dynamically calculated based on the PRR. Therefore, the comparative experiment of activation thresholds in the pruning module is converted into a comparative experiment of PRRs.

By comparing the performance under different PRRs, we investigate the impact on model accuracy and inference speed, aiming to find the optimal PRR. Set the value range of the PRR to 0.5 to 1, with a step size of 0.1. SAPM is performed on the MROD-YOLO model according to different PRRs. Then, the pruned model is fine-tuned, and its performance is evaluated on the validation set.

The comparative experimental results of different PRRs are shown in Table 2. As the PRR decreases, the number of model parameters significantly reduces, leading to a gradual improvement in computational efficiency. When the PRR decreases from 1 to 0.5, the parameters from 45.35 M to 22.68 M, and the FPS increases from 46.5 to 72.9. This demonstrates the excellent performance of SAPW in model lightweighting. A decrease in the PRR will lead to a reduction in model accuracy, accompanied by a decrease in the number of parameters and an improvement in computational efficiency.

Table 2. The comparative experimental results of different PRRs.

PRR	Pre	Rec	F1	mAP_{50}	$mAP_{50:95}$	Params	FPS
baseline	**91.5**	**89.6**	**90.5**	**93.9**	**74.8**	45.35 M	46.5
0.9	90.4	88.6	89.5	93.2	73.7	40.82 M	49.3
0.8	88.9	86.8	87.8	91.8	71.2	36.28 M	53.6
0.7	87.2	84.9	86.0	90.3	68.9	31.75 M	58.4
0.6	85.2	82.1	83.6	85.2	64.7	27.21 M	64.7
0.5	80.1	76.8	78.4	82.5	58.2	**22.68 M**	**72.9**

Note: Bold values indicate the best results. Green values correspond to a PRR of 0.7, which achieves a balanced trade-off between model accuracy and computational efficiency.

The PRR is 0.9, resulting in a small loss in accuracy but limited improvement in computational efficiency. The PRR is set at 0.7, achieving a trade-off between accuracy and efficiency, making it suitable for resource-constrained applications that require a balance between performance and efficiency. The PRR is reduced to 0.5, leading to a decrease in accuracy, which is not conducive to high-precision time-sensitive target detection scenarios. In this paper, the PRR is set at 0.7 to moderately enhance efficiency while maintaining accuracy.

4.2.2. Comparative Experiment on Mask Ratio

The masking ratio (MR) is a crucial hyperparameter in MBKD, determining the proportion of masking during the distillation process and influencing the learning performance

of the student model. By comparing the performance of student models under different MRs, we investigate the impact of MR on model accuracy and distillation effect, aiming to find the optimal MR. Set the value range of the MR to 0.1 to 0.9, with a step size of 0.2. We utilize the MROD-YOLO model as the teacher network, and the student network is the pruned version of the MROD-YOLO model, obtained using the SAPM algorithm. We perform distillation training on the student network according to different MRs and evaluate its performance on the validation set.

The experimental results comparing different MRs are shown in Table 3. When the MR is 0.5, the student model performs optimally in terms of accuracy metrics, with Pre (90.8%), Rec (88.6%), F1 (89.7%), mAP_{50} (93.6%), and $mAP_{50:95}$ (73.8%) all reaching peak values. MR controls the degree of feature occlusion during distillation and thus regulates the difficulty of knowledge transfer. A small MR leads to weak regularization and limited compensation for pruning-induced information loss, while an excessively large MR introduces overly sparse supervision and destabilizes training. An intermediate MR value (MR = 0.5) provides a balanced setting, enabling the student model to recover missing semantic patterns while maintaining stable convergence, which explains the observed peak performance.

Table 3. The comparative experimental results with different MRs.

MR	Pre	Rec	F1	mAP_{50}	$mAP_{50:95}$	Params	FPS
Teacher model	**91.5**	**89.6**	**90.5**	**93.9**	**74.8**	45.35 M	46.5
Student model	87.2	84.9	86.0	90.3	68.9	31.75 M	58.4
0.1	90.1	87.9	89.0	92.9	73.0	31.75 M	58.4
0.3	90.3	88.1	89.2	93.1	73.2	31.75 M	58.4
0.5	90.8	88.6	89.7	93.6	73.8	31.75 M	58.4
0.7	89.7	87.5	88.6	92.5	72.5	31.75 M	58.4
0.9	88.9	86.6	87.7	91.7	71.6	31.75 M	58.4

Note: Bold values indicate the best results. Green values represent the optimal masking ratio (MR = 0.5), at which the student model achieves the highest overall performance.

It is worth noting that varying the MR does not affect the number of parameters or inference speed of the student network. This is because MR is exclusively applied during the training stage of the distillation process, where random masks are introduced to perturb intermediate features for representation learning. During inference, all masking operations are removed, and the student network retains the same architecture and computational graph regardless of the MR setting. Therefore, Params and FPS remain identical across all MR configurations.

4.2.3. Ablation Experiment of Key Strategies

By comparing the performance of the baseline model, pruning model, and the combined model of pruning and distillation, we evaluate the contribution of each module to the detection results. Table 4 shows the results of the key strategy ablation experiment. The pruning strategy reduced the model parameters (from 45.35 M to 31.75 M) and improved computational efficiency (FPS increased from 46.5 to 53.6), but it also led to a decrease in model accuracy (Pre, Rec, mAP_{50}, etc.). After embedding the distillation strategy, the model accuracy has been significantly restored (mAP_{50} increased from 90.3% to 93.6%, and $mAP_{50:95}$ increased from 68.9% to 73.8%), while maintaining the advantages of fewer parameters and faster running efficiency. The MBKD proposed in this paper can effectively mitigate the accuracy loss caused by SAPM, achieving a better trade-off between model compression and performance optimization.

Table 4. The ablation experiment results of key strategies.

Model	Pre	Rec	F1	mAP_{50}	$mAP_{50:95}$	Params	FPS
Baseline (MROD-YOLO)	91.5	89.6	90.5	93.9	74.8	45.35 M	46.5
Baseline + SAPM	87.2	84.9	86.0	90.3	68.9	31.75 M	58.4
Baseline + SAPM + MBKD	90.8	88.6	89.7	93.6	73.8	31.75 M	53.6

4.3. Comparative Experimental Results and Analysis

4.3.1. Comparison with Lightweight Architectures and Structured Pruning

In this section, we compare SAMKD with lightweight backbone substitution and structured pruning baselines to evaluate its effectiveness under different model compression strategies. Specifically, we consider two lightweight architecture designs, MobileNetV4 [49] and GhostNetV2 [50], where the original backbone of the baseline detector is replaced with a lightweight backbone while keeping the remaining detection framework unchanged. In addition, we include two representative structured pruning methods: L1-norm filter pruning [51] and Network Slimming [41], which reduce model complexity by removing channels or filters based on static importance metrics. All comparison models are derived under the same baseline (YOLOv10X [52]) setting to ensure fairness. For all pruning-based methods, the PRR is fixed at 0.7. Under this setting, SAPM reduces the parameter size of YOLOv10X from 31.72 M to 22.34 M, while Network Slimming and L1-norm filter pruning yield parameter sizes of 24.65 M and 25.97 M, respectively, resulting in comparable compression strength. After pruning, all pruned models are subjected to a fine-tuning stage to restore performance. The fine-tuning process follows the same optimization configuration as the baseline training, while the learning rate is reduced to 0.001 and the number of epochs is 30.

As shown in Table 5, after substituting the baseline backbone with MobileNetV4, the GFLOPs is reduced from 171.1 to 106.2, and the inference speed increases to 75.3 FPS. Nevertheless, the detection accuracy drops markedly, with mAP50 decreasing from 90.5 to 78.8 and $mAP_{50:95}$ declining from 69.8 to 53.9, accompanied by a sharp reduction in recall to 70.0. A similar trend can be observed for GhostNetV2, further demonstrating that directly replacing the backbone with lightweight architectures can significantly reduce computational cost and improve inference speed. However, this strategy often results in substantial detection performance degradation, while the total parameter does not necessarily decrease. Lightweight architectural design primarily reduces multiply–accumulate operations through cheap operations and depthwise separable convolutions, rather than guaranteeing a more compact parameter scale. In addition, the reconstruction of the feature extraction hierarchy introduced by such architectures tends to weaken semantic representation capacity, resulting in more severe accuracy degradation in UAV scenarios characterized by dense small targets and complex backgrounds.

In contrast, structured pruning methods are more effective in reducing parameter redundancy, but their ability to preserve detection accuracy is inherently limited. L1-norm filter pruning reduces the parameter count to 25.97 M. However, mAP_{50} drops sharply to 75.4 and $mAP_{50:95}$ to 52.4. Network Slimming reduces the parameter count to 24.65 M and partially recovers accuracy, achieving an mAP_{50} of 81.0 and an $mAP_{50:95}$ of 55.4. This suggests that pruning strategies based on static importance metrics, such as filter norms or batch normalization scaling factors, are prone to removing channels that are critical for cross-modal semantic alignment in multimodal detection tasks. As a result, the fusion feature expression is unstable, especially in the recall dimension.

Table 5. Comparative experimental results of lightweight architectures and structured pruning methods.

Model	Pre	Rec	mAP_{50}	$mAP_{50:95}$	Params	GFLOPs	FPS
YOLOv10X (baseline)	88.6	86.3	90.5	69.8	31.72 M	171.1	65.1
YOLOv10X + MobileNetV4	80.3	70.0	78.8	53.9	33.19 M	**106.2**	75.3
YOLOv10X + GhostNetV2	79.2	69.5	78.2	53.2	36.03 M	106.9	**81.6**
YOLOv10X + L1-norm filter pruning	79.6	67.5	75.4	52.4	25.97 M	151.7	66.3
YOLOv10X + Network Slimming	81.0	74.8	81.0	55.4	24.65 M	145.6	68.4
YOLOv10X + SAPM	**82.7**	**78.8**	**82.4**	**58.4**	**22.34 M**	154.6	69.7

Note: Bold values indicate the best results except for the baseline.

Unlike the aforementioned approaches, SAPM employs an input-dependent selective activation mechanism to achieve more fine-grained redundancy suppression. After pruning, the model parameters are reduced to 22.34 M. At the same time, SAPM maintains competitive detection performance, achieving an mAP_{50} of 82.4 and an $mAP_{50:95}$ of 58.4, with recall reaching 78.8. These results demonstrate that SAPM consistently outperforms conventional structured pruning baselines by better preserving discriminative features essential for multimodal object detection.

4.3.2. Comparison with Knowledge Distillation Methods

To further verify the effectiveness of the proposed method from the perspective of training-level optimization, we compared three knowledge distillation methods, namely CWD [35], FGD [36], and MGD [37]. In the experiments, four models are used as teacher models (YOLOv10X [52], MROD-YOLO [13], CFT [16] and TPDTNet [17]), and the student models are the pruned version of their respective teacher models, obtained using the SAPM algorithm. By comparing the performance of teacher models and different lightweight methods, the effectiveness of SAMKD proposed in this paper in terms of model lightweighting and performance improvement is demonstrated. As shown in Table 6, SAMKD achieves optimal performance in four teacher models, demonstrating superiority in key metrics such as Pre, Rec, F1, as well as mAP_{50} and $mAP_{50:95}$. Especially in complex conditions, the performance of SAMKD is even more significant, which demonstrates its robustness and adaptability across different task scenarios.

As a knowledge distillation method, MGD exhibits excellent performance in feature alignment and knowledge transfer. Its optimization strategy outperforms CWD and FGD in all teacher models, particularly in terms of F1 and mAP_{50}, which are close to those of the teacher models. On the basis of further optimization, the method proposed in this paper maintains higher detection performance while maintaining lightweight through a more efficient distillation strategy.

For the teacher model MROD-YOLO, the F1 score (89.7%) and mAP_{50} (93.6%) of our method are close to the teacher model and surpass MGD (F1 score is 87.5% and mAP_{50} is 91.5%). For the teacher model TPDTNet, the $mAP_{50:95}$ (42.2%) of our method is higher than that of the student model and MGD, indicating our method can effectively mitigate performance degradation in complex tasks. The comprehensive results indicate that the algorithm proposed in this paper not only outperforms traditional lightweight methods but also demonstrates stronger competitiveness in terms of efficiency and adaptability.

Table 6. Comparative experimental results of knowledge distillation methods on pruned student model.

Model	Pre	Rec	F1	mAP_{50}	$mAP_{50:95}$	Params	FPS
Teacher (YOLOv10X)	88.6	86.3	87.4	90.5	69.8	31.72 M	65.1
Student (SAPM)	82.7	78.8	80.6	82.4	58.4	22.34 M	69.7
CWD	84.6	79.7	81.8	84.5	60.8	22.34 M	69.7
FGD	84.4	80.5	81.4	83.2	61.0	22.34 M	69.7
MGD	85.9	81.8	82.7	84.9	61.4	22.34 M	69.7
SAMKD (Ours)	86.8	82.5	83.4	85.8	64.7	22.34 M	69.7
Teacher (MROD-YOLO)	91.5	89.6	90.5	93.9	74.8	45.35 M	46.5
Student (SAPM)	87.2	84.9	86.0	90.3	68.9	31.75 M	58.4
CWD	88.5	86.0	87.2	91.3	70.0	31.75 M	58.4
FGD	88.4	85.9	87.1	91.2	69.9	31.75 M	58.4
MGD	88.8	86.3	87.5	91.5	70.3	31.75 M	58.4
SAMKD (Ours)	90.8	88.6	89.7	93.6	73.8	31.75 M	58.4
Teacher (CFT)	90.6	89.0	89.8	90.8	62.8	206.24 M	39.4
Student (SAPM)	84.3	84.2	85.1	82.3	53.9	123.64 M	44.2
CWD	86.4	86.5	87.6	85.9	58.4	123.64 M	44.2
FGD	85.9	85.3	86.5	83.8	56.7	123.64 M	44.2
MGD	87.2	86.7	87.9	86.6	59.5	123.64 M	44.2
SAMKD (Ours)	88.4	87.3	88.2	87.4	60.3	123.64 M	44.2
Teacher (TPDTNet)	77.2	56.0	64.9	63.2	44.2	48.16 M	57.5
Student (SAPM)	64.8	47.8	55.0	54.6	39.4	33.71 M	65.8
CWD	66.2	49.2	56.4	56.8	40.2	33.71 M	65.8
FGD	65.8	50.5	57.1	55.7	40.9	33.71 M	65.8
MGD	67.5	49.8	57.3	57.0	41.5	33.71 M	65.8
SAMKD (Ours)	68.5	51.5	58.5	58.0	42.2	33.71 M	65.8

Note: Red, blue, and green values represent the best, second-best, and third-best performances among the lightweight methods, respectively, excluding the teacher models.

4.4. Generalization Experiment

To evaluate the generalization of the algorithm proposed in this paper, a general experiment was conducted on the publicly available multimodal dataset DroneVehicle for verification. The DroneVehicle dataset is a large-scale RGB-IR (visible-infrared) multimodal dataset for vehicle detection in UAV imagery. The dataset contains a variety of shooting conditions and scenes covering different viewing angles, heights, and lighting conditions. We selected 4200, 1000, and 900 image pairs from this dataset as the training, validation, and test sets, respectively. Five vehicle types—cars, trucks, buses, vans, and freight cars—were considered.

The comparison results are presented in Table 7. The method proposed in this paper demonstrates significant advantages across different teacher model architectures. The performance of the student model in restoring the teacher model can reach 95.3% to 97.4%, which is 1.5% to 2.3% higher than methods such as CWD and MGD. For high-performance teacher models such as MFCGNet (mAP_{50} = 73.0), the SAMKD algorithm focuses on feature alignment to maintain accuracy (achieving an mAP_{50} of 71.5).

For teacher models in complex scenarios (such as MROD-YOLO), SAMKD mitigates performance degradation through loss function design. The generalization experiment results indicate that the generalization ability and efficiency balance characteristics of SAMKD are superior to those of existing baseline models.

Table 7. General experimental results and analysis.

Model	Pre	Rec	F1	mAP_{50}	$mAP_{50:95}$	Params	FPS
Teacher (YOLOv10X)	66.1	58.5	62.1	63.2	40.5	31.72 M	65.1
Student (SAPM)	61.8	53.6	57.8	58.2	36.5	22.34 M	69.7
CWD	63.7	54.5	58.4	59.8	38.2	22.34 M	69.7
FGD	62.2	54.1	57.8	58.5	37.8	22.34 M	69.7
MGD	64.1	55.6	59.2	60.5	37.4	22.34 M	69.7
SAMKD (Ours)	64.5	56.7	60.3	61.4	38.6	22.34 M	69.7
Teacher (MROD-YOLO)	68.7	68.6	68.6	70.4	52.7	45.35 M	46.5
Student (SAPM)	64.2	63.8	64.0	66.5	48.1	31.75 M	58.4
CWD	65.5	65.0	65.2	67.8	49.7	31.75 M	58.4
FGD	65.3	64.9	65.1	67.6	49.0	31.75 M	58.4
MGD	66.0	65.4	65.7	68.2	49.3	31.75 M	58.4
SAMKD (Ours)	67.1	66.5	66.8	69.5	51.2	31.75 M	58.4
Teacher (CFT)	76.2	68.2	72.0	69.4	48.4	206.24 M	39.4
Student (SAPM)	71.4	63.7	67.3	64.0	44.1	123.64 M	44.2
CWD	72.9	65.8	68.4	66.2	45.7	123.64 M	44.2
FGD	72.5	64.5	69.2	65.4	45.3	123.64 M	44.2
MGD	73.4	66.2	69.5	66.9	46.2	123.64 M	44.2
SAMKD (Ours)	73.9	66.7	70.2	67.8	46.7	123.64 M	44.2
Teacher (TPDTNet)	59.9	58.7	59.3	60.3	48.2	48.16 M	57.5
Student (SAPM)	54.5	53.2	53.8	54.8	43.0	33.71 M	65.8
CWD	55.8	54.5	55.1	56.2	44.1	33.71 M	65.8
FGD	55.5	54.9	55.2	55.9	44.3	33.71 M	65.8
MGD	56.5	55.0	55.7	56.8	44.9	33.71 M	65.8
SAMKD (Ours)	57.4	56.2	56.8	57.9	45.8	33.71 M	65.8

Note: Red, blue, and green values represent the best, second-best, and third-best performances among the lightweight methods, respectively, excluding the teacher models.

5. Discussion

From a theoretical perspective, the stability of SAMKD in complex multimodal UAV scenarios originates from the complementary effects of SAPM and MBKD on model representation and optimization. SAPM performs selective activation pruning guided by activation sparsity, which effectively eliminates low-contribution responses and improves computational efficiency while maintaining high-energy feature pathways. However, this operation inevitably weakens inter-modal semantic coupling due to information removal. MBKD, in contrast, introduces masked feature recovery and binary distillation, which enhance the student model's ability to infer missing information from incomplete cues. The random masking acts as an implicit regularizer, enforcing robustness and promoting better generalization under cross-modal uncertainty. Thus, the combination of SAPM's activation-level sparsity and MBKD's reconstruction-oriented distillation establishes a self-compensatory learning cycle enabling SAMKD to sustain a favorable balance between efficiency and accuracy even in complex scenarios.

While SAMKD effectively reduces inference time computation and model size, its two-stage training pipeline inevitably introduces additional offline training cost compared with standard single-stage training or distillation-only schemes. From our empirical observations under the same training configuration (300 training epochs on an RTX 4090 GPU), the SAPM typically runs for approximately 60–65% of the baseline training epochs, because activation statistics must stabilize before pruning decisions become reliable, and a short fine-tuning process is required after pruning. The subsequent MBKD is conducted as a fine-tuning procedure on the pruned student network and usually requires an additional 35–40% of

baseline epochs, depending on the masking ratio and the difficulty of teacher–student feature alignment.

Importantly, these epoch ratios are not directly additive in terms of training time, because the per-epoch computational cost decreases after pruning and selective activation. In other words, SAPM produces a smaller student model, and both SAPM and MBKD are executed on this compressed network, resulting in a lower per-iteration FLOPs and faster training steps. In the MROD-YOLO case, SAPM reduces the parameter count from 45.35 M to 31.75 M and improves inference throughput from 46.5 FPS to 58.4 FPS (Table 4), indicating a substantially reduced computational footprint for later optimization. Meanwhile, MBKD introduces additional overhead from the teacher forward pass and feature alignment modules, but this overhead is partially offset by the reduced student-side computation.

As a result, the overall SAMKD pipeline leads to a moderate increase in offline training time, typically around 40–60% compared with standard baseline training, rather than a near-doubling of training time. This additional cost is incurred only once during training, whereas the deployment phase retains the benefits of pruning, achieving 30–50% parameter reduction (e.g., from 45.35 M to 31.75 M for MROD-YOLO) and 20–50% inference speed improvement (e.g., FPS from 46.5 to 58.4 for MROD-YOLO). Therefore, in practical UAV applications where models are trained offline but deployed repeatedly, this one-time training overhead is acceptable relative to long-term onboard efficiency gains.

MBKD adopts binary supervision to reformulate multi-class logits into foreground–background mappings. Although this setting deviates from traditional softmax-based distillation, the observed improvements suggest that binary reformulation helps alleviate the foreground–background imbalance commonly found in UAV scenes characterized by sparse object distributions. Nevertheless, this approach also introduces instability when the target density varies significantly across domains, indicating that MBKD may not generalize equally well to dense detection tasks. To address this limitation, future work will explore several potential mitigation strategies, including adaptive masking, where the masking ratio is dynamically conditioned on scene complexity or feature entropy rather than being fixed; density-aware distillation, which assigns proportionally higher supervision to regions containing multiple objects through dynamic loss reweighting; hybrid logits formulation, which integrates binary mapping with softened multi-class targets to better preserve inter-category correlations; and curriculum-based training, where the masking difficulty is progressively increased throughout training to enhance model robustness and stability across diverse detection densities.

6. Conclusions

This paper proposes a hybrid neural network lightweight method, SAMKD, combining a selective activation network pruning model (SAPM) and mask binary classification knowledge distillation (MBKD). By utilizing SAPW to reduce the teacher network into a lightweight student network, a binary classification distillation method with masking is designed. This method formulates the multi-classification problem as multiple binary classification problems, effectively addressing the imbalance between target and background categories during object detection. We have constructed a novel multimodal UAV aerial imagery object detection dataset, M2UD-18K, and the testing results demonstrate that the proposed model, SAMKD, significantly reduces the number of model parameters and the computational complexity of the model, while enhancing its performance and generalization ability. In the future, we will explore the deployment methods of SAMKD on embedded devices, validate it on UAV platforms, and promote the engineering application of the algorithm.

Author Contributions: Conceptualization, R.L. and Z.Z.; methodology, Z.Z.; software, S.W. and J.F.; validation, R.L., Z.Z. and S.W.; investigation, R.L.; resources, Z.Z.; writing—original draft preparation, R.L.; writing—review and editing, R.L. and Z.Z.; visualization, T.S.; supervision, R.L. and X.Y.; project administration, X.Y.; funding acquisition, X.Y., R.L. and J.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Natural Science Foundation of China under grant numbers 62276274 and 62573423; in part by the Key R&D Program of Shaanxi Province 2024CY2-GJHX-42; in part by the Shannxi Sanqin Elite Special Support Program 2024-SQ-001; and in part by the Shaanxi Natural Science Basic Research Program-Youth Project 2025JC-YBQN-957.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author. Due to data management and institutional constraints, the full M2UD-18K dataset is not yet publicly available. The complete dataset can be made available for academic research purposes upon reasonable request to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned Aerial Vehicle
DNN	Deep Neural Network
CNN	Convolutional Neural Network
KD	Knowledge Distillation
SAPM	Selective Activation Network Pruning Model
MBKD	Mask Binary classification Knowledge Distillation
SAPW	Selective Activation with Primary Weights
SAPI	Selective Activation with Primary Inputs
BN	Batch Normalization
AMU	Activation Management Unit

References

1. Qu, J.; Zhao, J.; Dong, W.; Xiao, S.; Li, Y.; Du, Q. Feature Mutual Representation-Based Graph Domain Adaptive Network for Unsupervised Hyperspectral Change Detection. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5501214. [CrossRef]
2. Li, L.; Zhang, W.; Barni, M. Covert Task Embedding: Turning a DNN into an Insider Agent Leaking out Private Information. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 10159–10166. [CrossRef]
3. Soyyigit, A.; Yao, S.; Yun, H. VALO: A Versatile Anytime Framework for LiDAR-Based Object Detection Deep Neural Networks. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2024**, *43*, 4045–4056. [CrossRef]
4. Li, N.; Yang, X.; Zhao, H.; Wang, X. Spectral Parameters Selection Based on Multi-Agent Deep Reinforcement Learning for AOTF Infrared Multispectral Imaging System. *IEEE Trans. Geosci. Remote Sens.* **2026**, *64*, 5400720. [CrossRef]
5. Madan, N.; Ristea, N.C.; Ionescu, R.T.; Nasrollahi, K.; Khan, F.S.; Moeslund, T.B.; Shah, M. Self-Supervised Masked Convolutional Transformer Block for Anomaly Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 525–542. [CrossRef] [PubMed]
6. Zhang, L.; Zhou, W.; Fan, H.; Luo, T.; Ling, H. Robust Domain Adaptive Object Detection with Unified Multi-Granularity Alignment. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 9161–9178. [CrossRef] [PubMed]
7. Zhai, J.; Du, J.; Ai, Y.; Hu, T. Edge Deployment of Deep Networks for Visual Detection: A Review. *IEEE Sens. J.* **2025**, *25*, 18662–18683. [CrossRef]
8. Mittal, P. A Comprehensive Survey of Deep Learning-Based Lightweight Object Detection Models for Edge Devices. *Artif. Intell. Rev.* **2024**, *57*, 242. [CrossRef]
9. Wang, S.; Yang, X.; Lu, R.; Zhu, Z.; Lian, F.; Li, Q.-G.; Fan, J. MTTSNet: Military Time-Sensitive Targets Stealth Network via Real-Time Mask Generation. *Def. Technol.* **2024**, *33*, 601–612. [CrossRef]
10. Lu, W.; Gao, W.; Liu, B.; Niu, W.; Wang, D.; Li, Y.; Peng, X.; Yang, Z. Reinforcement Learning Driven Time-Sensitive Moving Target Tracking of Intelligent Agile Satellite. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 9085–9101. [CrossRef]
11. Li, N.; Wang, H.; Zhao, H.; Ou, W. Cross-Modal Visible-to-Infrared Image Translation in Remote Sensing Guided by Thermal Features. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 5628916. [CrossRef]

12. Khiabani, P.J.; Zubiaga, A. Few-Shot Learning for Cross-Target Stance Detection by Aggregating Multimodal Embeddings. *IEEE Trans. Comput. Soc. Syst.* **2024**, *11*, 2081–2090. [CrossRef]
13. Wang, S.; Yang, X.; Lu, R.; Zhang, D.; Xie, W.; Su, S.; Zhang, Z. MROD-YOLO: Multimodal Joint Representation for Small Object Detection in Remote Sensing Imagery via Multiscale Iterative Aggregation. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 4706314. [CrossRef]
14. Zhang, J.; Lei, J.; Xie, W.; Fang, Z.; Li, Y.; Du, Q. SuperYOLO: Super Resolution Assisted Object Detection in Multimodal Remote Sensing Imagery. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5605415. [CrossRef]
15. Chen, Y.; Wang, B.; Guo, X.; Zhu, W.; He, J.; Liu, X.; Yuan, J. DEYOLO: Dual-Feature-Enhancement YOLO for Cross-Modality Object Detection. In Proceedings of the International Conference on Pattern Recognition (ICPR), Kolkata, India, 1–5 December 2024; pp. 236–252.
16. Feng, Q.; Wang, Z. Cross-Modality Attentive Feature Fusion for Object Detection in Multispectral Remote Sensing Imagery. *Pattern Recognit.* **2022**, *130*, 108786. [CrossRef]
17. Wang, S.; Yang, X.; Lu, R.; Su, S.; Tang, B.; Zhang, T.; Zhu, Z. TPDTNet: Two-Phase Distillation Training for Visible-to-Infrared Unsupervised Domain Adaptive Object Detection. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2025**, *18*, 4255–4272. [CrossRef]
18. Romero, A.; Ballas, N.; Kahou, S.E.; Chassang, A.; Gatta, C.; Bengio, Y. FitNets: Hints for Thin Deep Nets. *arXiv* **2014**, arXiv:1412.6550.
19. Liu, D.; Bai, X.; Zhao, R.; Deng, X.; Yang, L.T. Dual-Grained Lightweight Strategy. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 10228–10245. [CrossRef] [PubMed]
20. Park, W.; Kim, D.; Lu, Y.; Cho, M. Relational Knowledge Distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 3967–3976.
21. Zagoruyko, S.; Komodakis, N. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. *arXiv* **2016**, arXiv:1612.03928.
22. Meng, Y.; Zhang, L.; Zhang, Y.; Hu, M.; Zhao, F.; Shen, X. Satellite Optical Target Edge Detection Based on Knowledge Distillation. *Remote Sens.* **2025**, *17*, 3008. [CrossRef]
23. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 525–542.
24. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2015**, arXiv:1510.00149.
25. Lin, J.; Rao, Y.; Lu, J.; Zhou, J. Runtime Neural Pruning. In *Advances in Neural Information Processing Systems 30*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 2178–2188.
26. Hu, H.; Peng, R.; Tai, Y.W.; Tang, C.K. Network Trimming: A Data-Driven Neuron Pruning Approach Towards Efficient Deep Architectures. *arXiv* **2016**, arXiv:1607.03250. [CrossRef]
27. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]
28. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
29. Luo, H.; Ren, T.; Zhang, Y.; Zhang, L. A Lightweight and Small Sample Bearing Fault Diagnosis Algorithm Based on Probabilistic Decoupling Knowledge Distillation and Meta-Learning. *Sensors* **2024**, *24*, 8157. [CrossRef]
30. Zhang, D.; Dong, Y. MAE-MACD: The Masked Adversarial Contrastive Distillation Algorithm Grounded in Masked Autoencoders. *IEEE Trans. Ind. Inform.* **2025**, *21*, 337–346. [CrossRef]
31. Hu, Z.; Gong, M.; Zhao, Y.; Zhang, M.; Lu, Y.; Li, J.; Pu, Y.; Wang, Z. Scale-Aware Pruning Framework for Remote Sensing Object Detection via Multifeature Representation. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 5614215. [CrossRef]
32. Cheng, H.; Zhang, M.; Shi, J.Q. Influence Function Based Second-Order Channel Pruning: Evaluating True Loss Changes for Pruning is Possible Without Retraining. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 9023–9037. [CrossRef]
33. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531. [CrossRef]
34. Zhang, L.; Song, J.; Gao, A.; Chen, J.; Bao, C.; Ma, K. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3712–3721.
35. Shu, C.; Liu, Y.; Gao, J.; Yan, Z.; Shen, C. Channel-Wise Knowledge Distillation for Dense Prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 5291–5300.
36. Yang, Z.; Li, Z.; Jiang, X.; Gong, Y.; Yuan, Z.; Zhao, D.; Yuan, C. Focal and Global Knowledge Distillation for Detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 4633–4642.

37. Yang, Z.; Li, Z.; Shao, M.; Shi, D.; Yuan, Z.; Yuan, C. Masked Generative Distillation. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2022; pp. 53–69.
38. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 2704–2713.
39. Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; Han, S. HAQ: Hardware-Aware Automated Quantization with Mixed Precision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 8604–8612.
40. Guo, Y.; Yao, A.; Chen, Y. Dynamic Network Surgery for Efficient DNNs. In *Advances in Neural Information Processing Systems 29*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 1387–1395.
41. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning Efficient Convolutional Networks through Network Slimming. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2755–2763.
42. Carreira-Perpinán, M.A.; Idelbayev, Y. "Learning-Compression" Algorithms for Neural Net Pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8532–8541.
43. Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; Shao, L. HRank: Filter Pruning Using High-Rank Feature Map. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 1526–1535.
44. Li, Y.; Gu, S.; Mayer, C.; Van Gool, L.; Timofte, R. Timofte Group Sparsity: The Hinge Between Filter Pruning and Decomposition for Network Compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 8015–8024.
45. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
46. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
47. Maaz, M.; Shaker, A.; Cholakkal, H.; Khan, S.; Zamir, S.W.; Anwer, R.M.; Khan, F.S. EdgeNeXt: Efficiently Amalgamated CNN-Transformer Architecture for Mobile Vision Applications. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2022; pp. 3–20.
48. Wang, C.-Y.; Liao, H.-Y.M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 1571–1580.
49. Qin, D.; Leichner, C.; Delakis, M.; Fornoni, M.; Luo, S.; Yang, F.; Wang, W.; Banbury, C.; Ye, C.; Akin, B.; et al. MobileNetV4: Universal Models for the Mobile Ecosystem. In Proceedings of the European Conference on Computer Vision (ECCV), Milan, Italy, 29 September–4 October 2024; pp. 78–96.
50. Tang, Y.; Han, K.; Guo, J.; Xu, C.; Xu, C.; Wang, Y. GhostNetV2: Enhance Cheap Operation with Long-Range Attention. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 9969–9982.
51. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning Filters for Efficient ConvNets. *arXiv* **2016**, arXiv:1608.08710.
52. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. Yolov10: Real-Time End-to-End Object Detection. In Proceedings of the 2024 Advances in Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 9–15 December 2024; Volume 37, pp. 107984–108011.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.