Journal of Real-Time Image Processing (2021) 18:1343–1356
https://doi.org/10.1007/s11554-021-01136-5

SPECIAL ISSUE PAPER

AFTD-Net: real-time anchor-free detection network of threat objects for X-ray baggage screening

Yiru Wei[1] · Zhiliang Zhu[1] · Hai Yu[1] · Wei Zhang[1]

Received: 9 July 2020 / Accepted: 18 May 2021 / Published online: 31 May 2021

Abstract
X-ray baggage screening is a vitally important task to detect all kinds of threat objects at controlled access positions, which can prevent crime and guard personal safety. It is generally performed by screeners to visually determine whether a bag contains threat objects. However, manual detection shows several limitations, from the detection errors to different detection results produced by different screeners. To address these issues, several automated detection approaches have been proposed; nevertheless, none of the methods can achieve end-to-end detection and the results have only classification information without positional information. In this paper, we propose a real-time anchor-free detector of threat objects that can recognize threat objects without using pre-designed anchor boxes. We employ a lightweight but strong backbone network: MobileNetV2 to extract the multi-level information. The backbone network is followed by a deformation layer which aims at handling the nonrigid deformation of threat objects in X-ray images. To further strengthen the proposed network, we design a context enhancement module to aggregate the multi-scale features and generate the enhanced features. We name the network as anchor-free detection network of threat objects (AFTD-Net). We demonstrate the effectiveness of the proposed method against other object detection algorithms on the GDXray database. Our AFTD-Net is a fully convolutional network which does not need any pre-designed anchors and achieves a real-time computation speed of 44.8 FPS.

Keywords X-ray baggage screening · Detection of threat objects · Anchor free · Deep learning

1 Introduction

X-ray baggage screening is an important task in public space and security checkpoint, which can prevent crime and guard personal safety [1]. X-ray scanning can capture inner details of bags by emitting a series of narrow beam of X-ray. The X-ray absorption rate differs much across different types of items in the bags. This provides a way for screeners to recognize threat objects. The detection of threat objects is a complicated work since it is very difficult to recognize threat objects that are placed in cluttered bags or be occluded by other objects [2–5].

The detection of threat objects is usually performed by screeners who visually examine X-ray images and judge whether a bag has threat objects or not. Safety screeners are required to consume a lot of concentration to identify threat objects with various shapes, sizes and materials (metals, organic and inorganic substances). Since screeners have only a few seconds to recognize the threat objects in a bag, the likelihood of missed and incorrect inspection is high over a long period of time. Although manual method has been widely used in the field of X-ray security screening, it mainly relies on the experience of screeners, by which the detection accuracy is unable to be fully assured and the detection results always vary between screeners. Some literatures show that the detection performance is only 80–90% [6]. For these reasons, automated detection of threat objects can act as a complementary tool to assist screeners in improving the detection performance of threat object.

In the past, automated detection of threat objects mainly uses manual method to extract the features and feed the features into a classifier to detect the threat objects. For the detection of threat objects in mono-energy X-ray images, methods like the adapted implicit shape model (ISM) based on visual codebooks [7], and adapted sparse representations

✉ Zhiliang Zhu
zzl_neu@yeah.net

Yiru Wei
weiyiru0228@yeah.net

[1] Software College, Northeastern University, Shenyang 110819, China

(XASR+) detection model [8] have been proposed. For the detection of threat objects in dual-energy X-ray images, gabor texture features [9], bag of words (Bows) [10, 11], and support vector machines (SVM) classifiers [12] have been used. Detection methods based on 3-D features for 3-D objects recognition have been developed. For instance, rotation invariant feature transform and scale-invariant feature transform (SIFT) descriptors [13], 3-D visual cortex modeling 3-D Zernike descriptors and histogram of shape index [14]. In these methods, threat objects are represented and detected separately. The detection of threat objects is designed as a classification problem. The detection results include only classification information without location information. In short, previous detection methods cannot realize end-to-end detection, which is difficult to meet the automation demands of X-ray baggage inspection. Automated detection of threat objects is far from our satisfaction. There are still potentials for further improvement in terms of detection accuracy and recognition efficiency.

With the tremendous success of deep learning in image processing, deep convolutional neural networks (CNNs) based detection methods of threat objects were proposed in the last few years. The method [15] uses a pre-trained model as generic feature extractor, which automates the process of extracting discriminative features. In the testing stage, a simple nearest neighbor classifier is applied, which means that the label of a test image is the label of its nearest neighbor in the training stage. This method can only predict the classification information of threat objects but not location information. The method [16] uses the existing CNN-based models to exhaustively explore the use of CNN in the tasks of classification and detection within X-ray baggage imagery. Despite the existing CNN-based models achieve the good performances, these models contain a large number of parameters and cannot meet the real-time requirement of detection of threat objects for X-ray baggage screening. Moreover, previous methods did not train an ad-hoc detection network with the X-ray baggage images. We hope to train an ad hoc object detection network with the GDXray database [17], which can predict the categories and locations of threat objects simultaneously.

Nevertheless, we cannot deny that CNNs are favorable for detection of threat objects. Recently, CNN-based object detector YOLOv3 [18] dominates the real-time object detection task. Compared with manual approaches and traditional approaches, it can learn rich features automatically and predict categories and locations of objects simultaneously. According to the dataset in process, it adopts the anchor mechanism to enumerate all possible bounding boxes templates with different scales and aspect ratios. Though the anchor mechanism can improve recall, it also causes a large number of false positives. These false positives can put huge pressure on non-maximum suppression and make the inference slow. Moreover, the anchor configuration must adopt to the characteristics of the dataset. Otherwise, the degradation in performance will occur.

To address these issues caused by anchor mechanism, the "Anchor-Free" methods [19–21] have received great attention recently. There is no need to find the best anchor settings for different dataset and objects are represented as keypoints. For instance, CornerNet [19] and CornerNet-Squeeze [20] denote an object as a pair of keypoints (top-left and bottom-right corners). Moreover, removing the anchor can reduce the parameters of network, thus fielding fewer highly overlapped bounding boxes during training, which can potentially accelerate the detection speed. The above-mentioned anchor-free methods mainly focus on directly predict the categories and bounding boxes at one scale. The backbone network hourglass [22] in anchor-free methods [19–21] employ multiple up-sampling and down-sampling operations in CNNs. This causes the loss of finer images details which is bad for keypoint-based detection task. We, in a different way, replace the hourglass network [22] with lightweight backbone network: MobileNetV2 [23] to generate the robust multi-level features maps. The feature maps are fed into the context enhancement model (CEM) to aggregate the multi-level features and output one scale enhanced feature maps. To handle the deformation of threat object in X-ray images, we introduce a deformation layer. Besides, we design two lightweight prediction modules to reduce the redundancy operations and speed up the computation time.

Our main contributions are summarized as follows:

- We present a novel anchor-free detector of threat objects for X-ray baggage screening. Similar to [19], we represent object as a pair of points; yet replace the time-consuming hourglass network [22] with the lightweight backbone network: MobileNetV2 [23] to achieve real-time response.
- We employ a deformation layer to enhance the robustness of our network to nonrigid deformation. In addition, we design a CEM that is located behind the backbone to receive the multi-scale feature maps and output one scale enhanced feature maps. Finally, we achieve two lightweight prediction modules to reduce the redundancy operations and speed up the computation time.
- In order to evaluate the performance of our proposed network, a series of experiments are conducted on the GDXray database [17]. The experiment results show that our detector is superior to anchor-free methods: CornerNet [19], CenterNet [21] and CornerNet-Squeeze [20] and anchor-based method: YOLOv3 [18] with respect to accuracy and computation time.

2 Related work

2.1 Detection of threat object

Detection of threat objects requires both of the categories and boundaries of threat objects in X-ray screening images. Generally speaking, existing works can be divided into manual detection and automated detection. The manual detection mainly depends on the experience of screeners, which causes that the detection results always vary between screeners and the detection accuracy is unable to be fully assured. The traditional automated detection uses image processing methods or manual method to extract the features and feed the features into a classifier to detect the threat objects. The methods cannot achieve end-to-end detection. With the advent of deep learning, CNN-based object detector YOLOv3 [18] has been widely applied for real-time object detection. It uses pre-defined anchors for predicting spatial bounding boxes and classification scores. However, the anchor mechanism produces a large number of overlapped proposals, which consumes a lot of time on the post-processing step (Non-Maximum Suppression). To the best of our knowledge, our model is the first one to apply the anchor-free method to automated detection of threat objects for X-ray baggage screening. We believe that the anchor-free design is a very effective solution in our task in term of speed and accuracy.

2.2 Anchor-free detectors

While most of object detectors employ pre-designed anchors, anchor-free object detectors [19–21] have received considerable attention in recent years based on their good adaptability towards different datasets. The anchor-free methods can address the issues caused by pre-designed anchors, which have shown superior performance for object detection. Representative approach is CornerNet [19]. The problems are, on the one hand, threat objects are often distorted in X-ray screening images. On the other hand, detection of threat objects for X-ray baggage screening requires high real-time performance. These limitations make the CornerNet [19] fail to detect threat objects in X-ray images in real time. In CornerNet, objects are defined as pairs of keypoints: top-left corners points and bottom-right corners points. We follow this idea in our work, but we replace the complicated hourglass [22] in CornerNet [19] with a small, sophisticated network: MobileNetV2 [23]. We employ a deformation layer to handle the deformation of threat objects in X-ray images. Moreover, we use the depthwise separable convolution in the prediction module, which has fewer parameters and lower operational costs compared to traditional convolution.

2.3 MobileNetV2

MobileNetV2 [23] is a lightweight convolutional neural network, which applies depthwise separable convolution to inverted residual block. Depthwise separable convolution consists of depthwise convolution and pointwise convolution. The depthwise convolution convolves the input feature maps by channel-wise fashion and then the pointwise convolution convolves the new feature maps with a 1×1 convolution kernel. The depthwise separable convolution can significantly reduce the number of parameters compared to traditional convolution. For instance, a convolution kernel is 3×3, the input and output channels as 3 and 256 respectively. The number of parameters in the traditional convolution is $3 \times 3 \times 3 \times 256 = 6912$. The number of parameters in depthwise separable convolution is $3 \times 3 \times 3 + 3 \times 1 \times 1 \times 256 = 795$. Besides, MobileNetV2 employs inverted residual block, the principle of which is to "expand" the input feature channels and then performs depthwise separable convolution. It is essential to obtain more features in the depthwise separable convolution, which can enhance the expression ability of network. Therefore, MobileNetV2 has the advantages of strong expression ability and fast calculation speed, which is extremely well suited to our detection task.

2.4 Multi-scale features enhancement

In our network, we predict the threat objects based on one scale feature maps. Generally, detection methods based on one scale feature maps deploy a fully connected layer on top of the last convolutional layer to capture the global contextual information. However, the issue coming with the fully connected layer is that a large number of calculating parameters in the fully connected layer need a lot of GPU memory. Another way is to employ the feature pyramid network (FPN) [24] to obtain the multi-scale features. It applies the pyramid structure before the final prediction layers to form a multi-scale prediction network. However, the multi-scale predictions put tremendous pressure on the post-processing and make the inference time far slower than real-time. We employ the CEM to aggregate the multi-scale local features. This module can generate the enhanced global features for detection of threat objects. The CEM has relatively fewer parameters compared to FPN. Meantime, it retains the multi-level information but does not employ multi-scale predictions.

3 Detection network of threat objects

3.1 Overview of AFTD-Net

We address the detection of threat objects by employing CNNs in a fully convolutional fashion. We denote our network as AFTD-Net and describe the overall architecture in Fig. 1. In our network, threat objects are represented as pairs of top-left corner and bottom-right corner similarly to CornerNet [19]. In CornerNet, Hourglass-104 is made up by residual blocks that consist of two 3×3 convolutional layers and a skip connection, which consumes most of the computational resources. To reduce the complexity of the network architecture, our network uses a lightweight MobileNetV2 [23] as the backbone network. The original MobileNetV2 [23] is trained for image classification so the output feature maps have low resolution relative to the input image. To obtain fine feature maps suitable for detection of threat objects, we make some modifications: we discard the global average pooling layer and the last fully connected layer on top of MobileNetV2 [23] as they are designed for image classification problems. In addition, we incorporate a deformation layer [25] in our backbone, which aims at handling large and unknown orientation and scale changes without complex model parameters.

The backbone network is followed by the CEM, which aims at obtaining sufficient context information from multi-scale feature maps outputted by the backbone network. The enhanced feature maps generated by the CEM are fed into the prediction branch containing two prediction modules that have same outputs as the CornerNet [19].

The prediction module outputs the heatmaps, embedding vectors and offsets. The heatmaps record the location of different object categories of corners and save the confidence score for each corner. The embedding vectors are used to pull together corners belonging to the same objects and to push away corners not belonging to the same group. The offsets can adjust the location of the corners to produce more accurate bounding box. For obtaining the final bounding

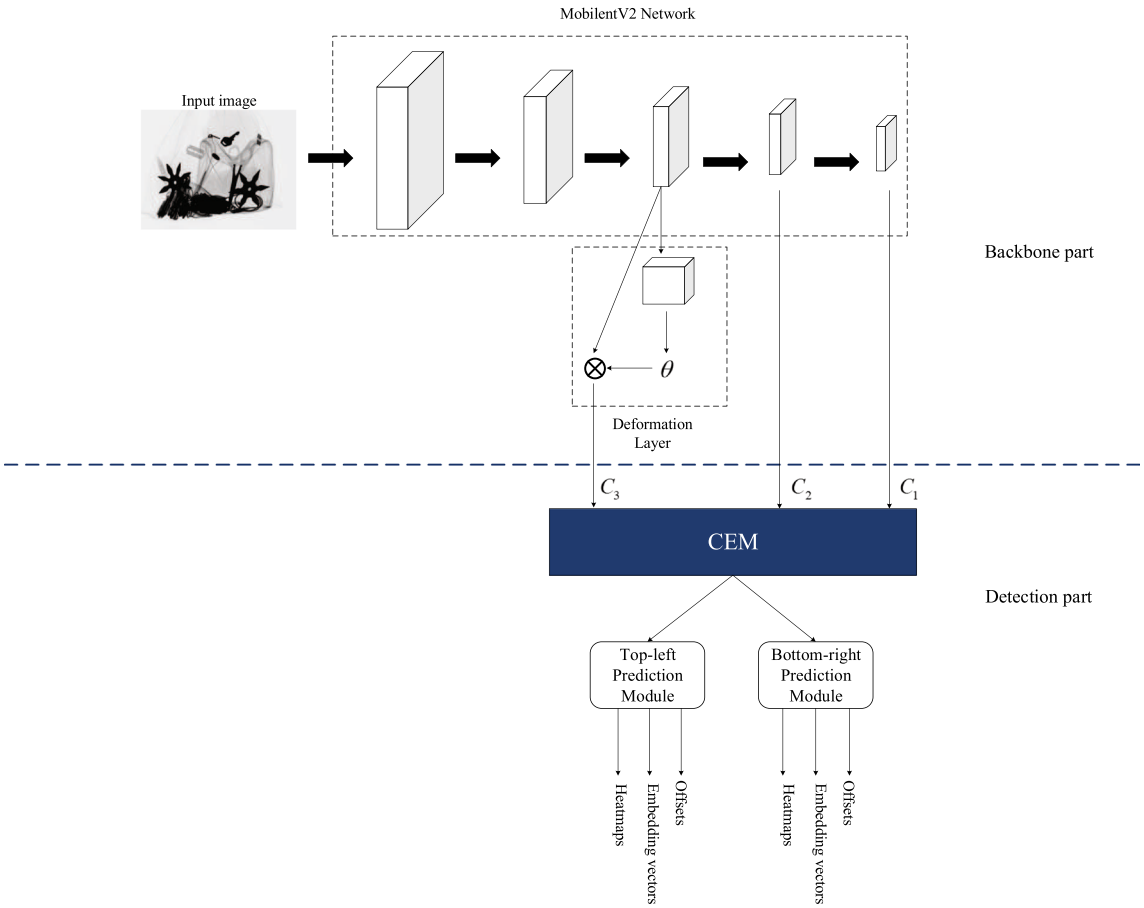

Fig. 1 The overall architecture of AFTD-Net: combination of the revised MobileNetV2 network, a deformation layer, the CEM and two lightweight prediction modules. The revised MobileNetV2 network is used to extract multi-scale features from input image. A deformation layer is applied to feature maps in order to obtain invariance to affine warping. The CEM can aggregate multi-scale feature maps to generate more discriminative feature maps. Two lightweight prediction modules output the heatmaps, embedding vectors and offsets

boxes, top-k left-top corners and bottom-right corners are chosen from the heatmaps according to their scores. If the distance of the embedding vectors of a pair of corners is less than the threshold, a bounding box is generated by this pair corners. The bounding box has a confidence score that is equal to the average score of the pair corners. Moreover, our prediction module is lightweight relative to the one in the CornerNet [19], which can speed up the prediction process and decrease the model parameters. Our detection network can quickly generate the accurate detection results through our feature enhancement strategy and lightweight prediction modules.

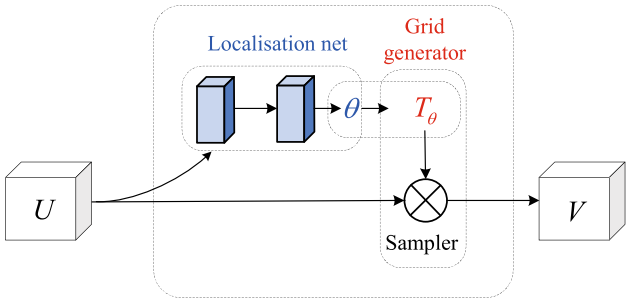

Fig. 2 The architecture of a deformation layer. It includes three parts: the localisation network, the grid generator and the sampler module

3.2 Backbone part

Input resolution The backbone network uses the input resolution of 512×512 pixels and outputs three scales feature maps of 64×64, 32×32 and 16×16 pixels.

Feature extraction network A common approach for deep learning is to use a pre-trained network as initial configuration and "fine-tune" it on the dataset being processed. This technique helps prevent overfitting and speeds up the training. As off-the-shelf lightweight convolutional neural network we fine-tuned three common networks on the GDXray database: ShuffleNetV2 [26], MobileNetV2 [23] and SqueezeNet [27]. Details are discussed in Sect. 4.4.1, from which we observe that MobileNetV2 achieves the best performance and thus is employed as the feature extraction network.

A deformation layer To enhance the robustness to geometric transformations, a deformation layer [25] is added in our backbone part, as shown in Fig. 2. It is applied to input images or feature maps in order to obtain invariance to affine warping. This new layer learns to calculate the parameters of an affine transformation which translates, rotates, scales and crops an input image or a feature map in order to handle object nonrigid deformations.

The localisation network is a small convolutional neural network, the specific parameters of which are shown in Table 1. It outputs a 6-dimensional transformation parameter vector $\theta = \{\theta_1, \ldots, \theta_6\}$. The grid generator determines mapping relationships between input and output coordinate points. $G_i = (x_i^t, y_i^t)$ is the coordinate point of output feature map V. (x_i^s, y_i^s) is the coordinate point of input feature map U. The affine transformation T_θ is defined by Eq. (1):

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = T_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}. \tag{1}$$

Finally, the sampler module employs bidirectional interpolation to compute pixel values of the output feature map. It is important to understand that the deformation layer generates the output feature maps according to the characteristics of each input. This layer can be added to our backbone network either before the first layer—serving as a pre-processing layer on the input image—or after one of the convolutional layers—to handle the deformations of high-level image features.

Table 1 Specific parameters of the localisation network in the deformation layer

Layer	Kernel size	Stride	Padding	Activate function	Feature map size
Input	–	–	–	–	$64 \times 64 \times 32$
Conv2d	3	2	1	–	$32 \times 32 \times 64$
BatchNorm2d	–	–	–	ReLU	$32 \times 32 \times 64$
Conv2d	3	2	1	–	$16 \times 16 \times 128$
BatchNorm2d	–	–	–	ReLU	$16 \times 16 \times 128$
Conv2d	3	1	1	–	$16 \times 16 \times 256$
BatchNorm2d	–	–	–	ReLU	$16 \times 16 \times 256$
Conv2d	3	2	1	–	$8 \times 8 \times 256$
BatchNorm2d	–	–	–	ReLU	$8 \times 8 \times 256$
GlobalAveragePooling2D	2	1	1	–	$1 \times 1 \times 256$
Fully connected	–	–	–	ReLU	6

It outputs a 6-dimensional transformation parameter vector

3.3 Detection part

Context enhancement module. To encode sufficient context information, a common technique is FPN [24]. However, previous FPN structures employ multiple detection branches, which inevitably increases the computational cost and causes enormous runtime latency. For this reason, we decide to discard the FPN design in our detection part.

To obtain sufficient context information, we design an efficient CEM. We aggregate multi-scale feature maps to generate more discriminative features. In our CEM, the three scales input feature maps from backbone are represented as: C_3, C_2 and C_1. We apply bilinear interpolation to upsampling the low-resolution features with factor 2. Denoting the depthwise separable convolution with 3×3 filters as DPConv_3. C_1 is first processed by a up-sampling module and a 3×3 depthwise separable convolution module, which can be written as:

$$C_{1_1} = \mathrm{DPConv}_3(\mathrm{upsampling}_2(C_1)) \tag{2}$$

where $\mathrm{upsampling}_2$ denotes the up-sampling operation with factor 2. Then, C_{1_1} and C_2 is concatenated into the feature C_{2_1} that can be written as:

$$C_{2_1} = \mathrm{Concat}(C_{1_1}, C_2) \tag{3}$$

where Concat represents the concatenation operation. C_{2_1} and C_{1_1} are also processed by a up-sampling module and a 3×3 depthwise separable convolution, which can be written as:

$$C_{2_2} = \mathrm{DPConv}_3(\mathrm{upsampling}_2(C_{2_1})) \tag{4}$$

$$C_{1_2} = \mathrm{DPConv}_3(\mathrm{upsampling}_2(C_{1_1})). \tag{5}$$

Then, C_{2_2} and C_3 is concatenated into the feature C_{3_1} that can be written as:

$$C_{3_1} = \mathrm{Concat}(C_{2_2}, C_3). \tag{6}$$

Afterwards, C_{1_2}, C_{2_2} and C_{3_1} are processed by a 1×1 convolution to squeeze the number of channels to 512, which can be represented as:

$$C_1' = \mathrm{Conv}_1(C_{1_2}) \tag{7}$$

$$C_2' = \mathrm{Conv}_1(C_{2_2}) \tag{8}$$

$$C_3' = \mathrm{Conv}_1(C_{3_1}) \tag{9}$$

Conv_1 denotes the 1×1 convolution. Finally, the three enhanced feature maps are aggregated by element-wise addition, which can be represented as:

$$C' = C_1' \oplus C_2' \oplus C_3' \tag{10}$$

where $\oplus$ represents the element-wise addition. By utilizing the CEM module, we can enlarge the receptive field and obtain representation ability of thin feature map. Compared with FPN structures, our CEM contains only a detection branch, which is more friendly for subsequent prediction module. Figure 3 describes the structure of this module.

Lightweight prediction modules In CornerNet [19], the backbone is followed by two prediction modules. One is top-left corners prediction module and the other one is the bottom-right corners prediction module, which are too heavy when combined with lightweight backbone network and causes imbalance between the backbone and detection part. Therefore, to address this problem, we compress the original prediction module in CornerNet [19] by replacing the first 3×3 Conv-BN-ReLU module with a 5×5 DepthConv-BN module and a 1×1 Conv-BN-ReLU module. The Depth-Conv-BN consists of a depthwise convolutional layer and a batch normalization layer. We enlarge the receptive field by increasing the kernel size and obtain more context information for the corner pooling layer. We remove the 1×1 Conv-BN module on the shortcut to retain the discriminating features from CEM. At last, the second 3×3 Conv-BN-ReLU

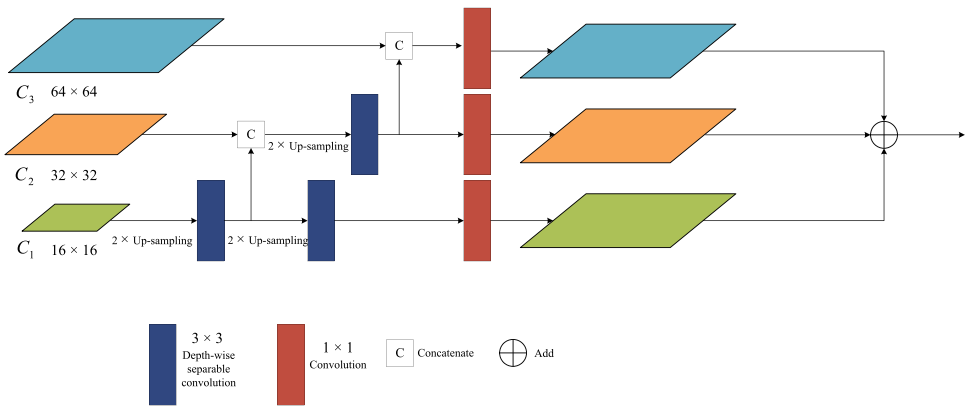

Fig. 3 The architecture of CEM. It can aggregate multi-scale feature maps to generate more discriminative feature maps

module that follows the residual module is replaced with a 3×3 DepthConv-BN module and a 1×1 Conv-BN-ReLU module, which further reduce the computational cost. Figure 4 shows the architecture of the prediction module in detection network. We demonstrate effectiveness of our lightweight prediction modules in Sect. 4.4.2. Experimental result shows that our prediction modules perform faster than the original prediction modules in CornerNet [19].

3.4 Training and inference

In our model, prediction modules output heatmaps, embedding vectors and offsets. Heatmaps record the location of different object categories of corners and save the confidence score for each corner. The size of heatmaps is $64 \times 64 \times 3 \times 2$, in which 64 is the size of feature map, 3 is number of categories in the GDXray database and 2 represent top-left and bottom-right corners. Each feature map is a binary mask that indicates the confidence score of corners. Embedding vectors are used to pull together corners belonging to the same objects and to push away corners not belonging to the same group. The size of embedding vectors is same to the size of heatmaps. Offsets can adjust the location of the corners to produce accurate bounding box. The size of Offsets is $64 \times 64 \times 2$, in which 64 is the size of feature map and 2 represents the two offsets of x and y.

Three outputs of prediction modules correspond to three losses: heatmaps loss, embedding vectors loss and offsets loss.

Heatmaps loss can be represented as:

$$L_{\text{det}} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1 - p_{cij})^{\alpha} \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1 - y_{cij})^{\beta} (p_{cij})^{\alpha} \log(1 - p_{cij}) & \text{otherwise} \end{cases} \tag{11}$$

where p_{cij} denotes the predicted probability of channel c at coordinates (i, j), y_{cij} is ground truth label of channel c at coordinates (i, j), C is the number of channels, H and W are height and width, N denotes the number of objects. α and β are set to 2 and 4.

Embedding vectors losses include two parts: L_{pull} and L_{push}. L_{pull} is used to reduce the distance between embedding vectors of two corners belonging to the same object. L_{push} is used to enlarge the distance between embedding vectors of two corners that do not belong to the same object. L_{pull} and L_{push} can be represented as:

$$L_{\text{pull}} = \frac{1}{N} \sum_{k=1}^{N} [(e_{t_k} - e_k)^2 - (e_{b_k} - e_k)^2] \tag{12}$$

$$L_{\text{push}} = \frac{1}{N(N-1)} \sum_{k=1}^{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \max(0, \triangle - |e_k - e_j|) \tag{13}$$

where e_{t_k} represent the embedding vectors of top-left corners belongint to object-k, e_{b_k} represent the embedding vectors of bottom-right corners belongint to object-k, e_k is the average value of e_{t_k} and e_{b_k}, $\triangle$ is 1.

Offsets loss can be represented as

$$L_{\text{off}} = \frac{1}{N} \sum_{k=1}^{N} \text{SmoothL1Loss}(o_k, \hat{o}_k) \tag{14}$$

$$o_k = \left(\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right) \tag{15}$$

where o_k is offset, x_k, y_k are the x and y coordinate for corner k. SmoothL1Loss is smooth L1 loss [28] at the ground truth corner locations.

Therefore, the total loss can be represented as

$$L = L_{\text{det}} + \alpha L_{\text{pull}} + \beta L_{\text{push}} + \gamma L_{\text{off}} \tag{16}$$

where α, β and γ represent the weights for the corresponding losses, value of which are 0.1, 0.1 and 1, respectively.

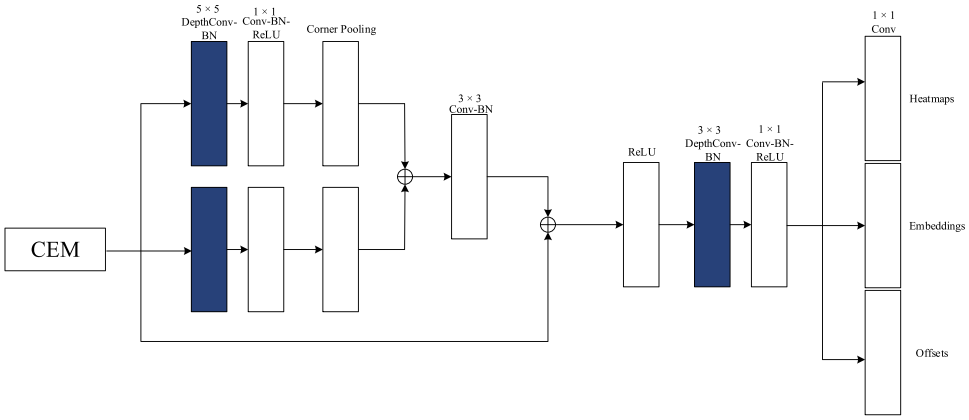

Fig. 4 The architecture of the lightweight prediction module. It employs DepthConv-BN module to enlarges the receptive field and reduce the model parameters. Finally, it outputs heatmaps, embedding vectors and offsets

During testing, after obtaining the predicted corners, we choose the first 50 top-left corners and the first 50 bottom-right corners according to their scores. The offsets can adjust the location of the corresponding corners. The L1 norm is used to calculate the distance between the embedding vectors of top-left corners and bottom-right corners. If the distance of a pair of corners is greater than 0.5 or two corners belong to different categories, these corners will be removed. By doing this, we generate the bounding boxes. The scores of bounding boxes are average scores of top-left corners and bottom-right corners. Soft-nms [29] is applied to remove the redundant bounding boxes. Finally, we choose the first 50 bounding boxes according to their scores as the final bounding boxes.

CornerNet [18] chooses the first 100 top-left corners and the first 100 bottom-right corners according to their scores. Then, the distances between the embedding vectors of top-left corners and bottom-right corners are calculated in the post-processing. Obviously, it is very time consuming in post-processing operation. This strategy cannot meet the real-time requirement of detection of threat objects. Meanwhile, these are only three kinds of threat objects in the GDXray database. Therefore, we choose the first 50 top-left corners and the first 50 bottom-right corners according to their scores to generate the final bounding boxes, which can achieve high accuracy as well as fast computation time.

4 Experiments

4.1 Implementation details

We train the network on a computer with a 3.7 GHz CPU, 12 GB RAM and an Nvidia Titan X GPU. Experimental codes are implemented in Pytorch [30]. The batch sizes are set to 32. The maximum number of iterations is 80K. We use a learning rate of 0.1×10^{-2} for the first 75K iterations and then continue training 5K iterations with a rate of 0.1×10^{-3}. The Adam [31] is employed as optimizer.

4.2 Dataset

We conduct our experiments on the GDXray database. It contains three kinds of threat objects: handgun, shuriken and razor blade, as shown in Fig. 5. The large amount of distortion make it a very challenging dataset. We apply translation, random rotation, cutting and reversal on the X-ray images containing threat objects for the data augmentation. The threat objects in the X-ray images are manually labeled as bounding boxes by the labelImg tool. Then we converted the annotated dataset into the format of the COCO dataset.

In the end, we obtain 15,336 images for training, 1916 images for validation. Meanwhile, we have 2200 images for testing set to verify the generalization detection ability of our proposed model.

4.3 Evaluation criteria

In our approach, we use the precision (P), recall (R), F1 scores (F_1), average precision (AP) and mean average precision (mAP) as the main evaluation metrics. The relevant terms are describe in the following:

True positive (TP): When the predicted bounding box contains a threat object, it will be viewed as a True Positive detection.

False positive (FP): When the predicted bounding box does not contain a threat object, it will count as a False Positive.

False negative (FN): When a threat object is not detected by the detector, it will be viewed as a False Negative.

The precision (P), recall (R), F1 scores (F_1) are defined as:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{17}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{18}$$

$$F_1 = \frac{2 \times P \times R}{P + R}. \tag{19}$$

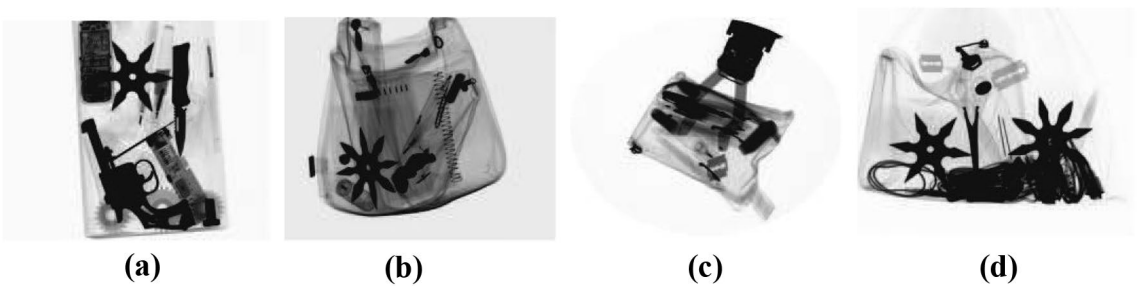

Fig. 5 X-ray images containing threat objects. **a** X-ray image containing one handgun and one shuriken. **b** X-ray image containing one shuriken and one razor blades. **c** X-ray image containing one razor blade. **d** X-ray image containing two shurikens and two razor blades

A set of recall thresholds $[0, 0.1, 0.2, \ldots, 0.9, 1]$ is set, average precision (AP) can be calculated as

$$\mathrm{AP} = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 0.9, 1\}} P_m(R), \tag{20}$$

$$P_m(R) = \max_{\tilde{R}, \tilde{R} \geq R} P(\tilde{R}), \tag{21}$$

where $P_m(R)$ is the maximum precision when the recall meets $\tilde{R} \geq R$, and $\tilde{R}$ denotes the recall corresponding to the maximum precision. Mean average precision (mAP) can be defined as

$$\mathrm{mAP} = \frac{\sum_{i=1}^{\mathrm{T}} \mathrm{AP}_i}{T}, \tag{22}$$

where T is the number of threat objects categories. In addition, we will compare computation speed (frame per second) of different models. In general, when the computation speed reaches to 30 FPS, it could be considered as real-time.

4.4 Ablation study

In this section, we conduct experiments on the GDXray database to analyze the effectiveness of the four components in our architecture, i.e. the backbone network, the deformation layer, CEM and the lightweight prediction modules.

4.4.1 Backbone part

We compare the detection results of multiple detection networks of threat object that are different in backbone parts. Three pre-trained networks on ImageNet [32] (namely ShuffleNetV2 [26], SqueezeNet [27] and MobileNetV2 [23]) are used as backbone for feature extraction and are followed by two prediction modules. These three networks do not include the deformation layer and CEM. In these three networks, C_2 and C_1 from backbone part are up-sampled to same size to C_3 and then C_3, C_2 and C_1 are combined by element-wised addition and passed to the prediction modules. The three networks are indicated as ShuffleTD-Net, SqueezeTD-Net and MobileTD-Net. We train and test three models on the GDXray database [17]. The results are shown in Table 2, from which we can see that MobileTD-Net achieves the best performance.

We introduce a deformation layer to provide invariance to affine warping. To demonstrate the effectiveness and determine position of the deformation layer, we implement several experiments with different configurations, i.e. (1) applying the deformation layer after the input X-ray images to process nonrigid deformation of the whole image (experiment represented as a "Yes" in Table 3); (2) placing the deformation layer after the output layer of the feature extraction network to deal with deformation of small regions (represented as "Yes: X" in Table 3, after the Xth output layer).

Table 3 shows the experimental results corresponding to the deformation layers in different positions. We find that the performance is best when the deformation layer is located after the third output layer. However, the performance degrades when the deformation layer directly acts on input images. That seems to be for two reasons. First, the deformation layer that is located after the input images can eliminate some discriminative features for subsequent convolutional layer. Second, for object detection, the small regions are more important than the whole image. Therefore, the deformation layer is placed after the output of the feature extraction network, which can process the nonrigid deformation of small regions and improve the detection performance.

4.4.2 Detection part

As discussed in Sect. 3.3, we design an efficient CEM, which can aggregate multi-scale feature maps to generate more discriminative feature maps. The CEM is located after the backbone to receive the multi-scale outputs from the backbone. To demonstrate the effectiveness of the CEM, we add CEM in MobileTD-Net, in which CEM receive the multi-scale feature maps from backbone network to produce the enhanced feature maps for prediction modules. This network is denoted as MobileTD-Net + CEM. We compare the performance of MobileTD-Net and MobileTD-Net + CEM in Table 4. Compared to MobileTD-Net, MobileTD-Net + CEM achieves a

Table 2 The performance comparisons among three detection networks of threat objects that have different backbone networks

Model	P (%)	R (%)	F_1 (%)	AP			mAP (%)
				Handgun (%)	Shuriken (%)	Razor blade (%)	
SqueezeTD-Net	96.8	91.61	94.15	95.06	92.7	91.33	92.77
ShuffleTD-Net	97.28	91.9	94.52	95.47	94.11	92.21	93.93
MobileTD-Net	97.64	93.25	95.4	96.38	94.15	92.59	94.37

Three pre-trained networks on ImageNet [32] (namely ShuffleNetV2 [26], SqueezeNet [27] and MobileNetV2 [23]) are used as backbone for feature extraction and are followed by two prediction modules. The three networks are indicated as ShuffleTD-Net, SqueezeTD-Net and MobileTD-Net. P, R, F_1, AP and mAP represent precision, recall, F1 score, average precision and mean average precision, respectively

significant performance improvement. The combination of the three-scale feature maps in CEM introduces different levels of semantics and details information, which can greatly strengthen the representation ability of output feature maps.

We achieve lightweight prediction modules by compressing the original prediction modules in CornerNet [19]. Our prediction modules employ depthwise convolution to enlarge the receptive field, which is conductive to obtain more context information. Meanwhile, the prediction modules speed up the computation speed by reducing the redundancy operation. To demonstrate the effectiveness of our lightweight prediction modules, we implement another network that employ the same backbone part as our AFTD-Net and the same prediction modules as CornerNet [19], which can be denoted as AFCorner-Net.We compare the performance of the two networks in Table 5. Our AFTD-Net achieves high accuracy while reaching the computation speed of 44.8 FPS, which is greatly superior to the another network.

4.5 Overall performance

To evaluate the performance of our proposed method, we compare our proposed detection network with the existing object detection networks on the GDXray database [17]. Table 6 shows the comparison results of different methods in the same experimental configuration. Our proposed detection network achieves the best result in terms of precision, recall, F1 score, average precision and mean average precision.

Table 3 Performance comparisons of different configurations for AFTD-Net

Deform	P (%)	R (%)	F_1 (%)	AP			mAP (%)
				Handgun (%)	Shuriken (%)	Razor blade (%)	
No	98.28	95.37	96.8	97.5	95.24	94.98	95.9
Yes	97.87	95.23	96.53	97.13	94.95	94.55	95.55
Yes: 1	98.46	95.6	97	97.86	95.69	95.42	96.32
Yes: 2	98.84	95.77	97.28	97.45	95.93	97.25	96.88
Yes: 3	**99.21**	**96.49**	**97.83**	**98.36**	**96.76**	**95.8**	**96.97**

The "Deform" column indicates whether the deformation layer exists and the position of the deformation layer in the detection network: "No" represents that these is no the deformation layer in our detection network; "Yes" indicates that the deformation layer directly processes the input X-ray images; "Yes: X" represents that the deformation layer is located after a specific output layer. In all experiments, the backbone part uses MobileNetV2 [23] as the feature extraction network and the detection part includes CEM and lightweight prediction modules. P, R, F_1, AP and mAP represent precision, recall, F1 score, average precision and mean average precision, respectively

Table 4 Performance comparisons of two detection networks of threat objects

Model	P (%)	R (%)	F_1 (%)	AP			mAP (%)
				Handgun (%)	Shuriken (%)	Razor blade (%)	
MobileTD-Net	97.64	93.25	95.4	96.38	94.15	92.59	94.37
MobileTD-Net + CEM	**98.28**	**95.37**	**96.8**	**97.5**	**95.24**	**94.98**	**95.9**

The MobileTD-Net denotes the network that consists of the backbone network: MobileNetV2 [23] and the lightweight prediction modules. The MobileTD-Net + CEM denotes the network that consists of the backbone network: MobileNetV2 [23], the CEM and the lightweight prediction modules. P, R, F_1, AP and mAP represent precision, recall, F1 score, average precision and mean average precision, respectively

Table 5 Effectiveness of lightweight prediction module

Experiment	P (%)	R (%)	F_1 (%)	AP			mAP (%)	Speed (FPS)
				Handgun	Shuriken	Razor blade		
AFCorner-Net	99.15	96.3	97.8	98.51	96.39	95.42	96.78	33.5
AFTD-Net (ours)	**99.21**	**96.49**	**97.83**	**98.36**	**96.76**	**95.8**	**96.97**	**44.8**

AFCorner-Net is the network that employs the same backbone part as our AFTD-Net and the same prediction modules as CornerNet [19]. AFTD-Net is our detection network of threat objects that uses the lightweight prediction module. P, R, F_1, AP and mAP represent precision, recall, F1 score, average precision and mean average precision, respectively

Table 6 The detection performance comparisons between the AFTD-Net and existing object detectors: CornerNet [19], CenterNet [21] and CornerNet-Squeeze [20] and YOLOv3 [18]

Model	P (%)	R (%)	F_1 (%)	AP			mAP (%)	Speed (FPS)
				Handgun (%)	Shuriken (%)	Razor blade (%)		
CornerNet	98.56	95.93	97.23	97.65	95.78	93.52	95.63	6.5
CenterNet-104	98.6	96.23	97.4	97.93	96.5	94.43	96.29	7.8
CornerNet-Squeeze	97.43	93.33	95.34	96.85	95.8	92.78	95.14	34.7
YOLOv3	97.37	95.19	96.27	96.7	94.4	93.65	94.92	28.5
AFTD-Net (ours)	**99.21**	**96.49**	**97.83**	**98.36**	**96.76**	**95.8**	**96.97**	**44.8**

P, R, F_1, AP and mAP represent precision, recall, F1 score, average precision and mean average precision, respectively

To prove our AFTD-Net can run faster than other anchor-free methods, we test CornerNet [19] and CenterNet [21] on the same environment with our method and report the computation speed in Table 6. Our AFTD-Net achieves the computation speed of 44.8 FPSs, which is greatly faster than the CornerNet [19] and CenterNet [21]. This indicates that our anchor-free AFTD-Net has achieved the real-time performance and can be applied to real-time detection of threat objects and other real-time detection applications.

Meanwhile, to further demonstrate high real-time performance of our proposed method, we also give quantitative comparisons between anchor-free real-time detector: CornerNet-Squeeze [20] and our AFTD-Net in Table 6. Our AFTD-Net still performs favorably against the anchor-free real-time detector. The advantage stems from lightweight feature extraction network and lightweight prediction modules. Based on CEM, our AFTD-Net not only achieves high real-time performance, but also maintains high detection accuracy. Therefore, when applied to detection of threat objects in X-ray baggage images, our AFTD-Net can detect threat objects accurately and quickly.

We also compare our method with the anchor-based real-time detector: YOLOv3 [18]. As can be observed from the Table 6, although our AFTD-Net runs faster than the YOLOv3 [18] thanks to the anchor-free design strategy, the overall performance of our method is also more competitive against the anchor-based YOLOv3 [18].

Finally, we have compared the computation time of three networks (i.e., AFTD-Net, CornerNet-Squeeze and YOLOv3) in Fig. 6. These three networks rank the top three in computation speed in Table 6. Although our networks contains four parts, the computation time of the whole network is far less than that of the other two networks. These three networks all contain feature extraction network and prediction network, among which our feature extraction network and prediction network are obviously faster than those of other two networks. This contributes to our lightweight backbone network and lightweight prediction modules. In our AFTD-Net, the deformation layer is a small CNN-based

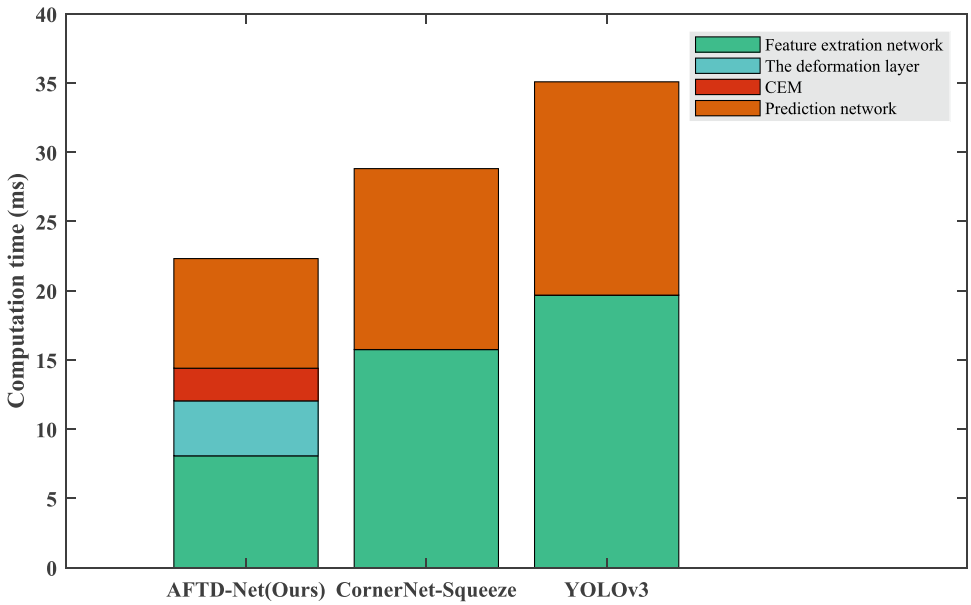

Fig. 6 The computation time comparisons of three networks

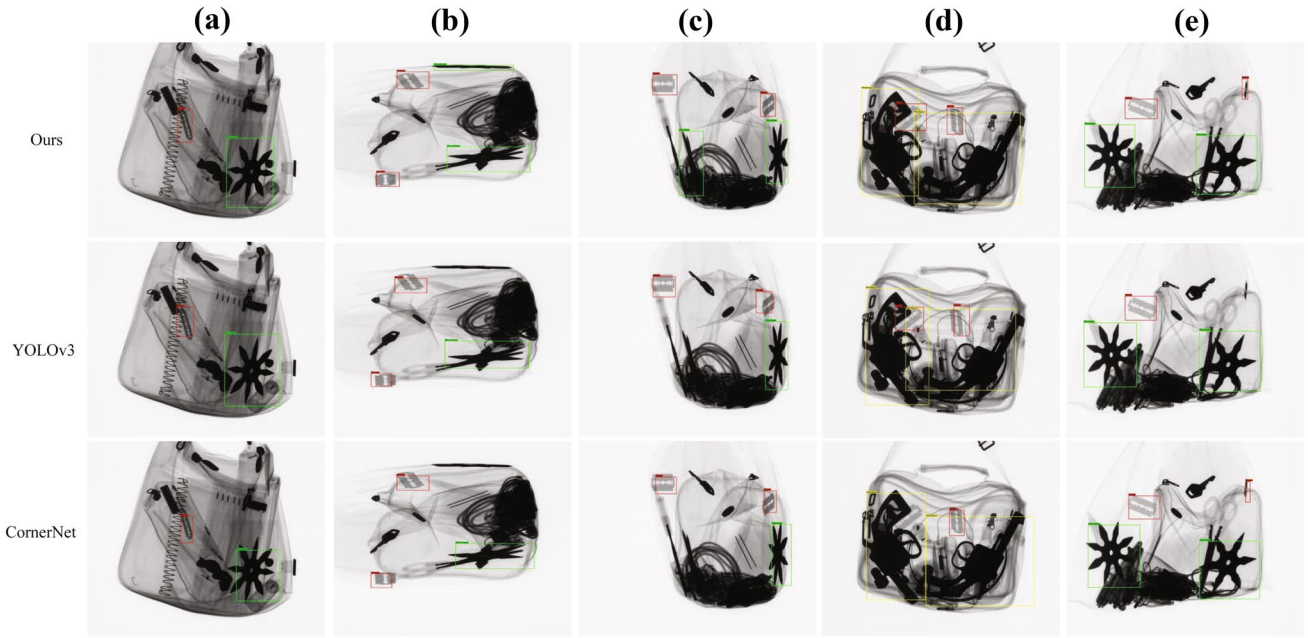

Fig. 7 The detection results of the AFTD-Net, YOLOv3 [18] and CornerNet [19]

classifier, which contains multiple convolutional layers and a fully connected layer and therefore is a bit time-consuming. However, the deformation layer can handle the nonrigid deformation of threat objects in X-ray images and therefore greatly improve the detection performance of threat objects. Even so, our detection network: AFTD-Net is greatly superior to the other networks and meets the real-time demand.

Thus, based on the comprehensive performance evaluation, as an automated detection network of threat objects, our proposed AFTD-Net can assist screeners to detect threat objects in X-ray baggage images. Meanwhile, our network can also meet high real-time requirement of the detection of threat objects for X-ray baggage screening.

4.6 Visual comparison

Figure 7 shows the visual comparisons of our approach to anchor-based YOLOv3 [18] and anchor-free CornerNet [19]. We can see that threat objects can be detected by three networks in the absence of occlusion and deformation in Fig. 7a. Figure 7b, c show that the deformed shuriken can not be detected by YOLOv3 [18] and CornerNet [19]. Figure 7d, e show that the small razor blades cannot located by the CornerNet [19] or YOLOv3 [18]. Compared with the other two networks, bounding boxes produced by our approach are closer to the real regions. Our proposed approach is robust for the images with occlusion and deformation thanks to the deformation layer and the CEM.

5 Conclusion

In this paper, we propose a real-time detection network of threat objects (AFTD-Net), which can quickly and accurately detect threat objects in X-ray baggage images. Our network employ a lightweight but strong backbone network: MobileNetV2 to extract the multi-scale features. The backbone network is followed by a deformation layer which aims at handling the nonrigid deformation of threat objects in X-ray images. To further improve the performance, we design CEM to integrate the multi-scale features and generate the enhanced features. To demonstrate the effectiveness of our proposed network, we have conducted a series of experiments to compare its performance to other object detection networks. Experimental results demonstrate that our proposed network outperforms many existing anchor-free methods and the widely used real-time YOLOv3 [18] in terms of accuracy and computation speed. We believe that our proposed network can be an effective automated aid to assist the screeners to detect threat objects for X-ray baggage screening.

In the future work, we intend to integrate different types of features and further improve the detection performance of threat object.

Acknowledgements This research was supported by the National Natural Science Foundation of China (Grant nos. 61977014, 61902056, 61603082), the Fundamental Research Funds for the Central Universities (Grant nos. N2017011, N2017016).

References

1. Gaus, Y.F.A., Bhowmik, N., Breckon, T.P.: On the use of deep learning for the detection of firearms in X-ray baggage security imagery. In: 2019 IEEE International Symposium on Technologies for Homeland Security (HST), pp. 1–7 (2019). https://doi.org/10.1109/HST47167.2019.9032917
2. Bayat, A., Nand, A.K., Koh, D.H., Pereira, M., Pomplun, M.: Scene grammar in human and machine recognition of objects and scenes. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2073–20737 (2018). https://doi.org/10.1109/CVPRW.2018.00268
3. Khazaie, V.R., AkhavanPour, A., Ebrahimpour, R.: Occluded visual object recognition using deep conditional generative adversarial nets and feedforward convolutional neural networks. In: 2020 International Conference on Machine Vision and Image Processing (MVIP), pp. 1–6 (2020). https://doi.org/10.1109/MVIP49855.2020.9116887
4. Tang, S., Roberts, D., Golparvar-Fard, M.: Human-object interaction recognition for automatic construction site safety inspection. Autom. Constr. **120**, 103356 (2020). https://doi.org/10.1016/j.autcon.2020.103356
5. Bicanski, A., Burgess, N.: A computational model of visual recognition memory via grid cells. Curr. Biol. **29**(6), 979-990.e4 (2019). https://doi.org/10.1016/j.cub.2019.01.077
6. Michel, S., Ruiter, J.C.D., Hogervorst, M., Koller, S.M., Schwaninger, A.: Computer-based training increases efficiency in X-ray image interpretation by aviation security screeners. In: IEEE International Carnahan Conference on Security Technology, pp. 201–206 (2007)
7. Riffo, V., Mery, D.: Automated detection of threat objects using adapted implicit shape model. IEEE Trans. Syst. Man Cybern. Syst. **46**(4), 472–482 (2017)
8. Mery, D., Svec, E., Arias, M.: Object recognition in baggage inspection using adaptive sparse representations of X-ray images. In: Pacific-rim Symposium on Image and Video Technology, pp. 709–720 (2015)
9. Uroukov, I., Speller, R.: A preliminary approach to intelligent X-ray imaging for baggage inspection at airports. Signal Process. Res. **4**(5), 1–11 (2015)
10. Turcsany, D., Mouton, A., Breckon, T.P.: Improving feature-based object recognition for X-ray baggage security screening using primed visual words. In: IEEE International Conference on Industrial Technology, pp. 1140–1145 (2013)
11. Bastan, M., Yousefi, M.R., Breuel, T.M.: Visual Words on Baggage X-ray Images. Springer (2011)
12. Franzel, T., Schmidt, U., Roth, S.: Object Detection in Multi-view X-Ray Images. Springer (2012)
13. Flitton, G., Breckon, T.P., Megherbi, N.: A comparison of 3D interest point descriptors with application to airport baggage object detection in complex CT imagery. Pattern Recognit. **46**(9), 2420–2436 (2013)
14. Megherbi, N., Han, J., Breckon, T.P., Flitton, G.T.: A comparison of classification approaches for threat detection in CT based baggage screening. In: IEEE International Conference on Image Processing, pp. 3109–3112 (2013)
15. Mery, D., Svec, E., Arias, M., Riffo, V., Saavedra, J.M., Banerjee, S.: Modern computer vision techniques for X-ray testing in baggage inspection. IEEE Trans. Syst. Man Cybern. Syst. **47**(4), 682–692 (2017)
16. Akcay, S., Kundegorski, M.E., Willcocks, C.G., Breckon, T.P.: Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery. IEEE Trans. Inf. Forensics Secur. **13**(9), 2203–2215 (2018). https://doi.org/10.1109/TIFS.2018.2812196
17. Mery, D., Riffo, V., Zscherpel, U., Mondragón, G., Lillo, I., Zuccar, I., Lobel, H., Carrasco, M.: GDXray: the database of X-ray images for nondestructive testing. J. Nondestruct. Eval. **34**(4), 42 (2015)
18. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv:1804.02767 (2018)
19. Law, H., Deng, J.: CornerNet: detecting objects as paired keypoints. Int. J. Comput. Vis. **128**, 642–656 (2020)
20. Law, H., Teng, Y., Russakovsky, O., Deng, J.: CornerNet-Lite: efficient keypoint based object detection. arXiv:1904.08900 (2019)
21. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: keypoint triplets for object detection. arXiv:1904.08189 (2019)
22. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision, pp. 483–499 (2016)
23. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.: Inverted residuals and linear bottlenecks: mobile networks for classification, detection and segmentation. arXiv:1801.04381v2 (2018)
24. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125 (2017)
25. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Advances in Neural Information Precessing Systems, pp. 2017–2025 (2015)
26. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: European Conference on Computer Vision, pp. 122–138 (2018)
27. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360 (2016)
28. Girshick, R.: Fast R-CNN. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1440–1448 (2015)
29. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-NMS-improving object detection with one line of code. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5561–5569 (2017)
30. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: 31st Conference on Neural Information Processing Systems (NIPS) (2017)
31. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2014)

Yiru Wei received the B.E degree in software engineering from Wuhan Institute of Technology, China, in 2010, and the M.S. degree in computer system structure from North China Electric Power University, China, in 2013. She is currently pursuing the Ph.D. degree with the Software College, Northeastern University, China. Her main research interests include object detection, salient object detection and machine learning.

Zhiliang Zhu received the M.S. degree in computer applications and the Ph.D. degree in computer science from Northeastern University, China. He is currently a Professor with the Software College, Northeastern University. He is also a Fellow of the China Institute of Communications. His main research interests include information integration, complexity software systems, network coding and communication security, chaos-based digital communications, applications of complex network theories, and cryptography. He has authored and co-authored over 130 international journal papers and 100 conference papers, and published five books. He was a recipient of nine academic awards at national, ministerial, and provincial levels. He has served in different capacities at many international journals and conferences. He is a Senior Member of the Chinese Institute of Electronics and the Teaching Guiding Committee for Software Engineering under the Ministry of Education. He currently serves as the Co-Chair of the 1st–11th International Workshop on Chaos-Fractals Theories and Applications (IWCFTA).

Hai Yu received the B.E. degree in electronic engineering from Jilin University, China, in 1993 and the Ph.D. degree in computer software and theory from Northeastern University, China, in 2006. He is currently an associate professor of software engineering with Northeastern University. His research interests include complex networks, chaotic encryption, software testing, software refactoring, and software architecture. Moreover, he has served on different roles at several international conferences, such as Associate Chair for the 7th IWCFTA in 2014, the Program committee Chair for the 4th IWCFTA in 2010, the Chair of the Best Paper Award Committee at the 9th International Conference for Young Computer Scientists in 2008, and a Program committee member for the 3rd–10th IWCFTA and the 5th Asia Pacific Workshop on Chaos Control and Synchronization. He currently serves as an Associate Editor for the International Journal of Bifurcation and Chaos and a Guest Editor for Entropy and the Journal of Applied Analysis and Computation. He was a Lead Guest Editor for Mathematical Problems in Engineering in 2013.

Wei Zhang received the Ph.D. degree in computer science and technology from Northeastern University, China, in 2013. He currently works as an associate professor with the Software College, Northeastern University. His research interests include signal processing and multimedia security.