Master Thesis

CNN-based X-ray dangerous objects detection

畳み込みニューラルネットワークに基づく X 線危険物認識

Student Id: 48-176439

Author: Zou Lekang

Supervisor: Prof. Iba Hitoshi

Department of Information & Communication Engineering

The University of Tokyo

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 X-ray Security System

In this thesis, we focus on the topic of CNN-based X-ray dangerous objects detection. As terrorist activities around the world become more frequent, protecting people's lives and property has become an important issue for the security department. Airports, subway stations, government and large-scale event venues are high-risk places for terrorist activities. They use bombs, metal weapon and violent means to endanger people's safety. X-ray security system plays an important role in these places. Due to tourists' baggage is always compact and cluttered, checking the baggage manually is inefficient. X-ray security system solved this problem. **Fig. 1** explained how the X-ray security system works. In this system, baggage was transported to the inspection machine in first-in-first-out order. The inspection machine will scan the baggage in a short time using X-ray radiation. After several seconds, an X-ray image which contains all items in the baggage will be displayed on the screen. By this way, human operators can review the highly varying baggage without open it in a very short time.

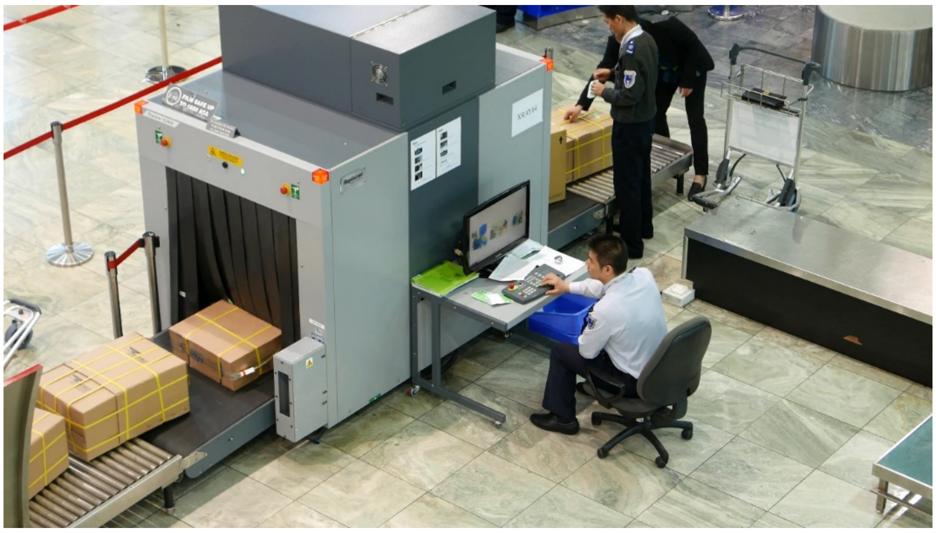


Fig. 1: An X-ray security system

However, this kind of X-ray system has several shortcomings. In the first place, the device for X-ray radiation is heavy and expensive. Second, the X-ray image processing for screen displaying is time consuming. For each X-ray image displayed on the screen, human operators need to check these images one by one. Since there is only a small amount of baggage which may include dangerous objects, the workload of checking all of them is tedious and meaningless. Last, due to there are too many types of dangerous objects, this manual work requires human operators to have accurate judgment and high ability of concentration. Even so, this checking work is still accompanied by a high error rate.

1.2 Convolution neural network

1.2.1 Principle of CNN

In recent years, convolution neural network (CNN) has shown state-of-art performance in image processing task. CNN is a mathematical algorithm which consists of multiple layers. These layers simulate the functions of human brain neural network. The structure design of CNN is based on a biological research result. In 1959, Hubel & Wiesel [1] conducted an experiment on cat's brain. They observed the response of a single neuron in the cat's brain to the image and found that neurons in the frontal area of the visual system reacted strongly to specific light signals but did not respond to any other pattern at all. This phenomenon indicates that animals' visual system always prefers to catch the low-level features of the image at first. These low-level features include edge information, color, direction and others, which are extracted by the first layer of animals' visual system called the Primary Visual Cortex (V1). Due to animals' visual system is hierarchical and progressive, the deep layers of the system will handle these low-level features and generate high-level features. As the image information flows through the visual system, the features will be processed layer by layer. Finally, the whole image will be displayed in front of our eyes.

Fig. 2 [2] explains the structure of a human visual system. This system consists of four kinds of layer. The image information flows from the retina and the V1 layer will conduct edge detection on the image. In the Secondary Visual Cortex (V2) layer, the visual system will extract low-level features of the image. Then, V4 layer will extract high-level features based on the information flowing from V2 layer. Finally, Inferior Temporal Gyrus (IT) layer will perform recognition task.

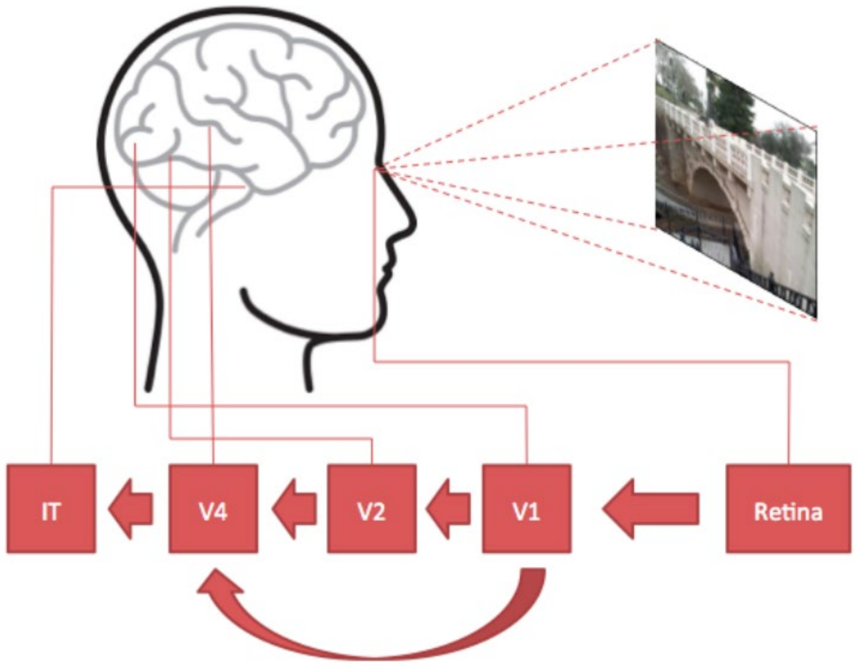


Fig. 2: A human visual system[2]

1.2.2 Basic components of CNN

Similar to this system, we introduce the function layer design of CNN now. The most important layer in CNN is Convolutional Layer. This kind of layer simulates the V1 layer in the human visual system. In the human visual system, the image information flowing from retina is 3-dimensional information. V1 layer is able to transform the 3D information to 2D information. A convolutional layer consists of a set of filters. These filters are n*n 2D square matrix, which can directly handle 2D information. To extract features of the image, each filter will convolve the input image in the convolutional layer. Depending on the size of the filter, the size of the feature images is different, so that different levels of feature images can be obtained. These feature images can be sent into deep convolutional layers for further extraction. By repeating this process, CNN is able to extract high-level features of the input image. **Fig. 3** explains how convolutional layer works. In this case, the input image size is 7*7 and the filter size is 3*3. During the convolution, the filter will scan the input image. The process multiples each number in the filter by the pixel intensity value and sums all of the products together. This value becomes the new intensity of the pixel. For example, when we conduct convolution operation in the colorized field in the **Fig. 3**, we get a new pixel value of 4 and the whole output image size will be 5*5 after we finish the convolution operation.

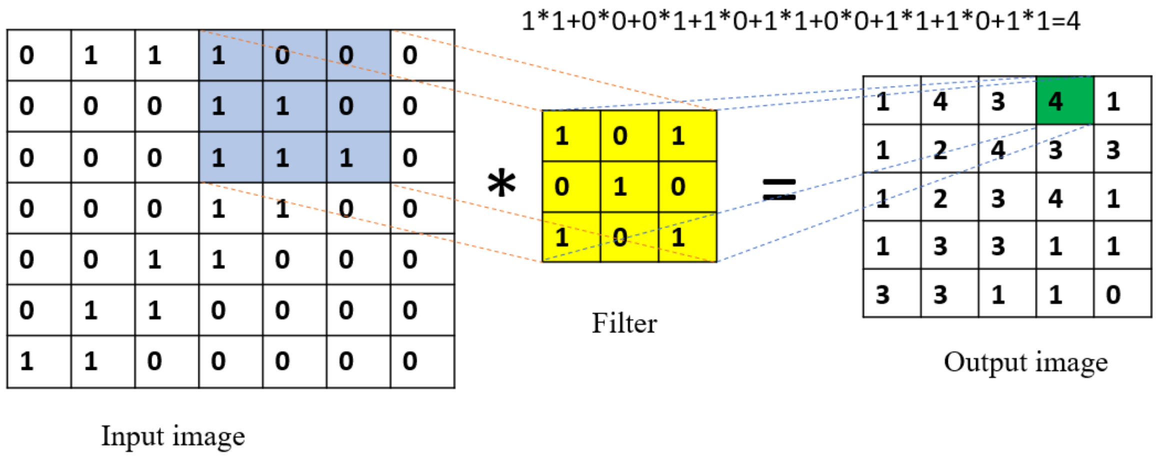
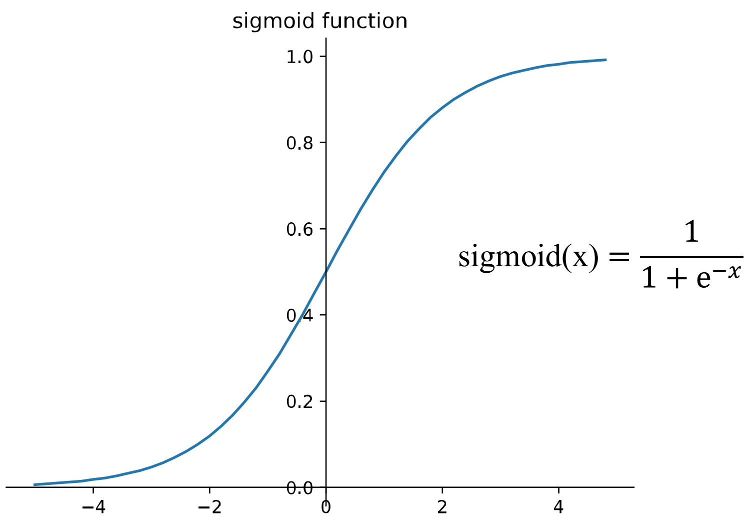


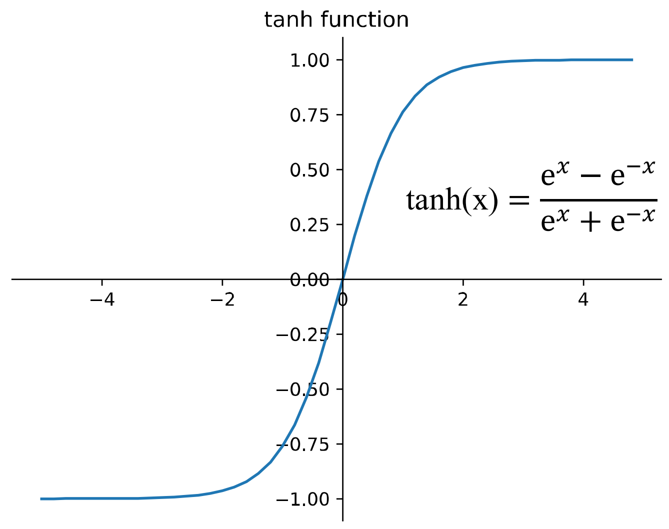
Fig. 3: A convolution operation

However, only the structure of convolutional layer is not enough. This is because convolution operation is a kind of linear operation, which indicate that this layer can only deal with linear problem. It is necessary that adding non-linear factors in CNN to enable CNN to solve any kind of problem. For this purpose, CNN takes use of Activation Function to bring in non-linearity property. The activation function simulates the process of electrical signal transmission between neurons in the animal's nervous system. We emphasize that activation functions should be nonlinear and continuously differentiable in most cases. This is because the proposed backpropagation algorithm (BP) [3] for training CNN, which we will introduce it later. There are many activation functions, here we introduce the three most commonly used activation functions.

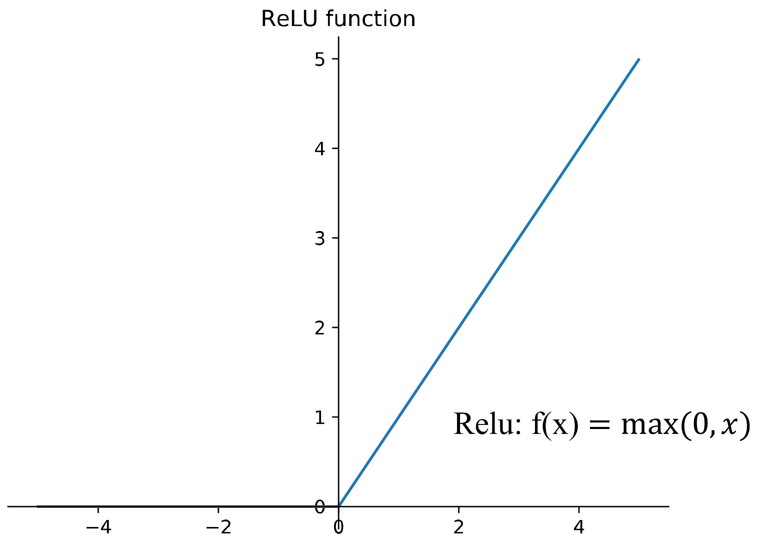
The first activation function we want to introduce is Sigmoid Function (see **Fig. 4** (a)). This function maps the real number interval to the 0~1 interval, which represents the process of neurons from inactive to active. Sigmoid function explains the principle of neuron propagation, but this function has two shortcomings. One is that the gradient of this function diagram at both ends is infinitely close to zero, which causes the gradient of CNN to disappear in the back propagation, leading to the network not converge to the global minimum. The other shortcoming is that the distribution of the function output is unbalanced in the positive and negative intervals. This will cause the gradient in the back propagation to be updated in the same direction, which is not conductive to the neural network convergence. The second activation function we want to introduce is Hyperbolic Tangent Function (see **Fig. 4** (b)). This function is the improvement based on sigmoid function. It can be seen from the function picture that the tanh function solves the problem that the output of the sigmoid function is unbalanced in the positive and negative intervals, but this function also has the problem that the gradient disappears in the back propagation. The third one we want to introduce is the most popular activation function in the recent years called Relu Function [4] (see **Fig. 4** (c)). Relu is a special activation function, whose function diagram is not full interval differentiable. Compared to the exponential operations of the previous two functions, the linearly unsaturated form of relu enables CNN to converge faster and turns a part of the parameters into zero to increase the sparseness of the network, which alleviates the problem of overfitting [5]. At the same time, relu also solved the gradient vanishing problem. Recently, researchers have proposed some improved function about relu [6][7]. Due to page limitation, we will not repeat them.



(a) sigmoid function



(b) tanh function



(c) ReLU function

Fig. 4:Three kinds of activation function

Another important component of CNN is the pooling layer. Because the convolutional layer has a lot of filters and these filters will generate multiple feature images after scanning the input image. These feature images contain a large number of parameters. If using these feature images directly for classification tasks, the GPU will face huge computational load and even memory overflow. The role of the pooling layer is to reduce the parameters of CNN and subsample the feature images to achieve the purpose of relieving overfitting. There are two common pooling layers. One is max pooling layer. In the max pooling layer, the algorithm selects the largest value in the pooling window as the sample value. The pooling window will scan the whole feature image. After the scanning, the algorithm will generate a subsampled feature image. The other layer is mean (average) pooling layer. In this layer, the sample value is calculated by averaging all the pixel values in the pooling window and the rest of the algorithm is the same as the max pooling layer. In the both two kinds of pooling layer, we can adjust the size of subsampled image by adjusting the size of the pooling window. This allows that the reducing the parameters of CNN be controllable. **Fig. 5** show how max pooling layer and mean pooling layer works.

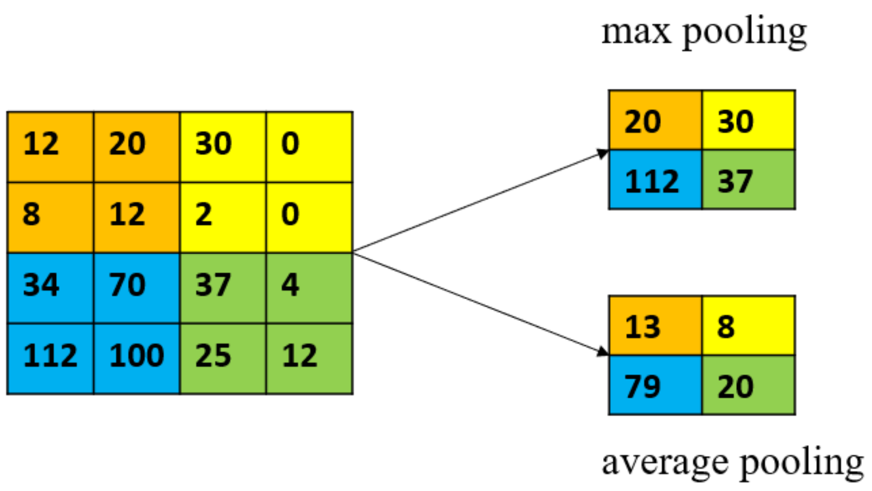


Fig. 5: Max pooling and Mean (Average) pooling

At the end of CNN, there is used to be a classifier or a detector or a fully connected layer for handling different types of tasks. Their main role is to globally classify the feature images learned by the previous layers and calculate the probability that the input image is most likely to belong to a certain category. This is the final component of CNN.

1.2.3 Gradient descent and Backpropagation algorithm

Supposing $x_1, x_2, \ldots, x_n$ are the inputs of linear the neural network and y is the real output of the linear neural network. We have the follow equation:

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n \quad , \tag{1}$$

where $w_1, w_2, \ldots, w_n$ are the weights from input neuron nodes to output neuron nodes in the linear neural network. We use variable t to represent the correct output of the neural network.

Because there is always a difference between y and t, we use the square error method to measure the difference between them:

$$E = \frac{1}{2}(y - t)^2 \tag{2}$$

The training process of the neural network is actually the process of updating a set of weights so that E is the global minimum. For a neural network of the single neuron, the function diagram of E is a parabola (see **Fig. 6**) and we can easy to find the global minimum. For a neural network of multiple neurons, the function diagram of E is a paraboloid (see **Fig. 7** (a)[8]). It is necessary to use gradient descent [10][11] method to find the minimum of the paraboloid. In multi-layer CNNs. the network structure incorporates nonlinear transformations, and the paraboloid becomes more complex (**Fig. 7** (b)[9]). In this case, we can't solve the global minimum by solving the equations. In order to avoid the E falling into a local minimum [12], we need to take use of the optimizers to improve the gradient descent algorithm. There are some published optimizers such as Momentum [13], NAG [14], Adagrad [15] and Adam [16].

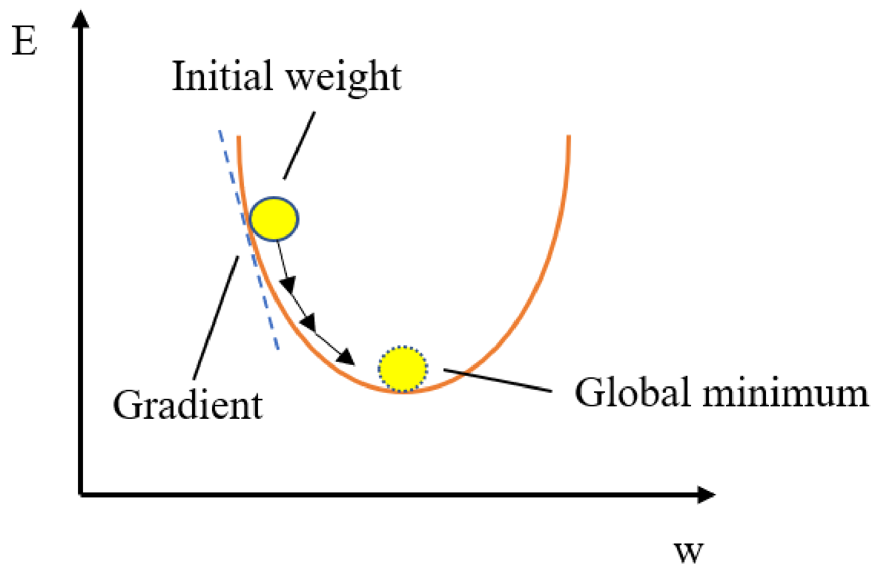
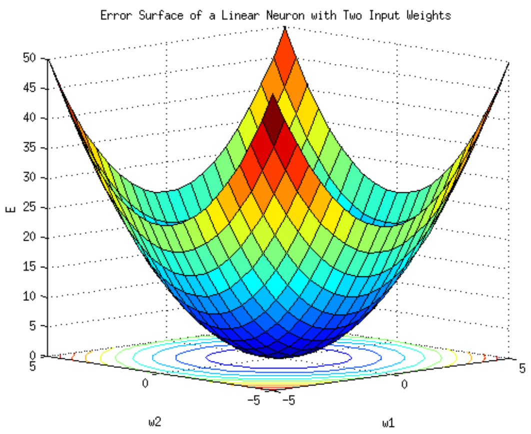
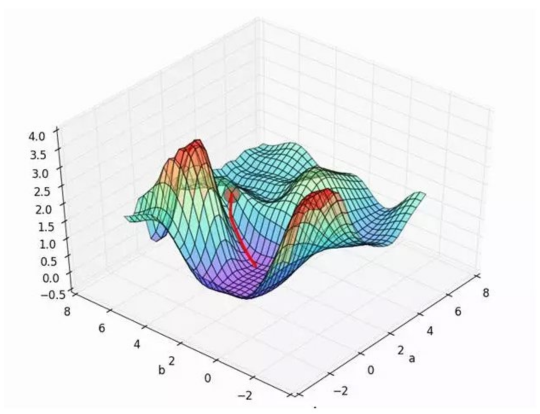


Fig. 6: Gradient descent in parabola





(a) A paraboloid diagram of a linear (b) A paraboloid diagram of a CNN

Fig. 7: The function diagram of E in linear and non-linear networks[8][9]

Gradient descent algorithm is performed in backpropagation in neural networks. The gradient descent algorithm tells us that the gradient of the previous layer of the neural network can be obtained by the loss function taking the partial derivative of the weight in the latter layer. In the process of backpropagation, the difference of the output layer will propagate forward. Each layer of the neural network uses the difference of the current layer to calculate the difference of the previous layer, so that the weights are updated in the direction of decreasing the difference. This kind of propagation process needs to set a learning rate. By adjust the learning rate, the problem of gradient disappearance and gradient explosion occurring during propagation can be effectively prevented, and the overfitting of the network can be alleviated [17].

1.3 Motivation and Methods

Although the security system has been studies for decades, there is still not enough published works on raw X-ray images baggage security. In 2003, an explosive detection system (EDS) was proposed [18], their work was mainly based on explosives in airport or post office security. However, dangerous objects are not just explosives, but also guns, metals, liquids and other items. Because most X-ray images are illegible, some researches work on X-ray image enhancement and segmentation [19][20]. This kind of work contribution is vert limited because it cannot solve the dangerous object detection problem fundamentally. Baştan M et.al proposed a method from the perspective of colorizing the objects in X-ray images according to their spectrum energy call bag of visual words (BoVW) [21]. This work is similar to the previous image enhancement method, but its performance is quite well. Still, their work has two shortcomings. One is that their work requires expensive device support. The other is that adding information on X-ray images means that it is hard to detect dangerous objects in real time. An improve method was proposed in 2013 called primed visual words [22]. Their experiment is based on firearm detection. They have achieved true positive rate of 90.07% and false positive rate of 4.31% in classification task. However, it is necessary to conduct experiments on object detection task because security systems require more accurate location detection in practical applications. Mery, Domingo, et al proposed a multiple view analysis method [23]. Their method includes three part: data association, 3D analysis and final analysis. The main idea of their method is that reconstructing the data in 3D clustering and using multiple view information and neighbor information in to classify it. But the size of their experiment dataset is small and the X-ray images in their dataset are not complicated enough. It is hard to verify the robustness of their algorithms. Jaccard, N et.al introduced a data synthesizing method of X-ray images and they conducted their experiments of small metal detection in freight containers [24][25]. Although their work is about cargo security not baggage security, their contribution also has great reference value for us. Cargo security is different from baggage security. This is because cargo always has large size. When there are dangerous objects in the cargo, they have a very small proportion of the area in the X-ray image of the entire cargo, which is extremely difficult for inspection. Baggage security is similar to it, the objects in baggage are usually irregular, which increases the difficulty of detection. Some studies use Support Vector Machine (SVM) to X-ray image classification task [26][27]. Subsequently, CNNs demonstrated strong capabilities in the field of image processing [28]. There is also a research about X-ray image detection task [29]. Their experiments are mainly focus on three kinds of dangerous objects detection: razor blades, shuriken and handguns. However, their dataset is a little small and their results are unable to be robust in practical applications.

In fact, most of previous works we mentioned are based on multi-view X-ray image dataset. When collecting images with a typical X-ray imaging device, we will get grayscale X-ray images called single-view X-ray images. But such images are often illegible. The multi-view

X-ray image is RGB colorized image in which an X-ray imaging device illuminates the same object from different angles with X-rays of different energies, and combines the images obtained by the same objects at different X-ray intensities (see **Fig. 8**[21]). In practical security systems, single-view X-ray images are often processed and converted into multi-view X-ray images to help human operators better identify objects in the baggage. This kind of multi-view X-ray device has some limitations. In the first place, the price of the device is in high level and it is cumbersome to handle. In the security area of the airport, we usually see dozens of such security devices, which take up a lot of place. Special transporters are also required to carry these devices. In the second place, the device needs to convert the single-view X-ray images of the baggage one by one and display these images to the screen, which take a lot of time. This is also why the security area is always overcrowded. In most cases, the baggage is safe. This is because the proportion of baggage containing dangerous objects is only a small percentage. The work that colorizing all the X-ray baggage images is undoubtedly meaningless. Finally, even if all the baggage objects can be displayed on the screen in the form of multi-view X-ray images, this also has a very high requirement for the professional quality of human operators. These human operators have to check thousands of images every day. This work requires rich experience and accurate judgement. At the same time, long-term work is very prone to visual fatigue. This also means that manual checks are accompanied by higher error rates, which in some cases are fatal.

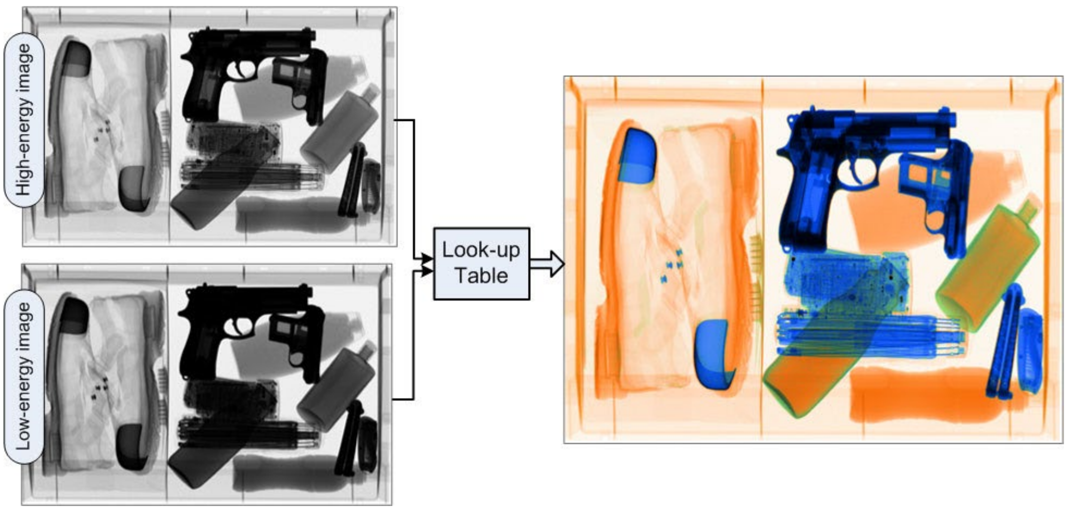


Fig. 8: Example of multi-view X-ray images[21]

Considering the many deficiencies of the multi-view X-ray device. We aim to study a portable automatic security system that can directly handle single-view X-ray images. We use CNN to train dataset including a large amount of single-view X-ray images to challenge the dangerous object detection task. This system is not only easy to handle, but also efficient. If this security system can be used for mass production, it can improve the efficiency of security inspection, reduce the cost and reduce the workload of security human operators.

1.4 Main Contributions

This thesis main has three contributions which can be listed as follows:

First, instead of using multi-view X-ray devices which are expensive and heavy, we introduce a portable X-ray device (see **Fig. 9**). In contrast to multi-view device that need to colorize the image, our device can collect single-view X-ray image in real time. This device contains three part: a laptop, an X-ray generator and an imaging device. These three parts play different roles. The left one is laptop. It has a dedicated X-ray processing software that visualizes the collected X-ray images in real time and saves the source files. The illumination time and intensity of each X-ray image will also be displayed on the screen in real time, which will facilitate the researcher's reference and adjust the illumination environment. The software also can reverse the color of the image to meet the needs of different conditions. The middle one is an X-ray generator (Product Name NS-100-L). This is a hand-held X-ray generator with a hand-held part that collects the operator's biometric information for personal authentication. This is extremely high security and prevents the possibility of being stolen. The electron acceleration voltage in the X-ray tube can output to 100kV. This intensity of illumination can meet the needs of all security areas. The one in the right place is X-ray imaging device. This is a board which is used to put the baggage. This board can remove the extra environmental background information in the X-ray image. This system is very efficient. We only need to put the baggage on the X-ray imaging device, then use the X-ray generator to illuminate the baggage. and the X-ray image of the baggage will be displayed in real time and saved in the laptop. All these three parts are lightweight, compact, safe and easy to carry.

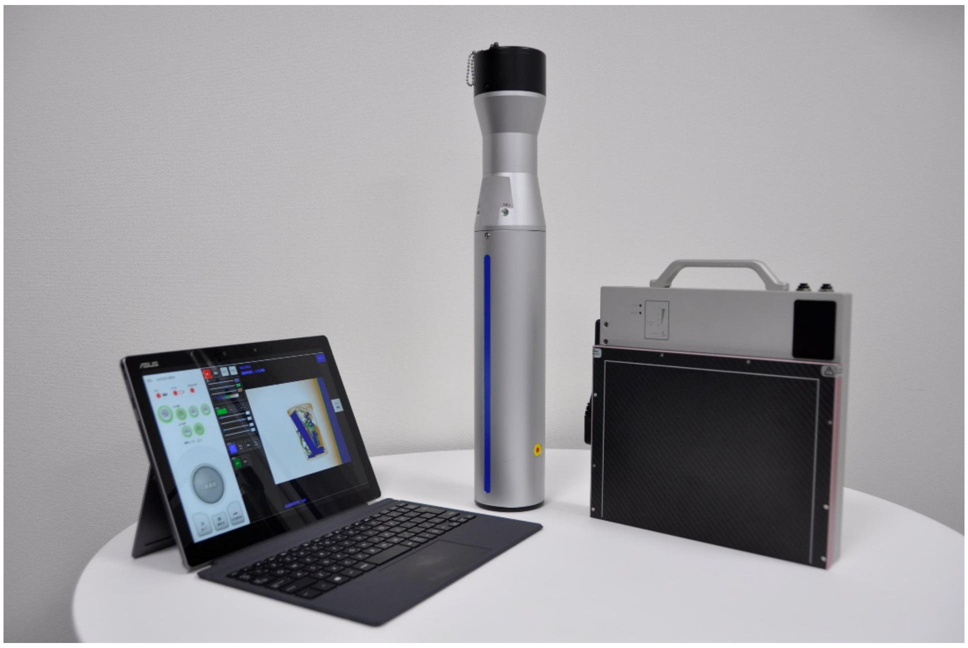


Fig. 9: A portable X-ray device

Second, we built a baggage security screening system for detecting dangerous objects. We would like to emphasize that our security system is different from previous works. As we mentioned before, most of previous works are based on multi-view X-ray images. Our system can directly handle single-view X-ray images. Our experiments are based on object detection tasks, not only to classify X-ray images but also to determine the location of dangerous objects in the X-ray image. We trained a powerful CNN for real time detection and prepared a large single-view X-ray dataset.

Third, considering that raw single-view X-ray dataset is very limited. We proposed a method to synthesize X-ray image. The images we synthesized are very close to real baggage X-ray images, and we have experimentally proved the improvement of our synthesized images. Our synthesizing algorithm can automatically label the images in the synthesizing process without adding meaningless workload. At the same time, this synthesizing method can also be used on other kings of X-ray data synthesizing.

1.5 Outline

The rest of this thesis is organized as follows. Chapter 2 discusses the related work of object detection. Chapter 3 presents some preliminary knowledge regarding Yolo for readers who are not familiar with the CNN model we use. Chapter 4 describes the method we proposed in this thesis. We show our experiment results in Chapter 5, and in this chapter, we also analysis our experiment results. In the Chapter 6, we discuss the limitations of our system and summarize the thesis and propose the future work.

Chapter 2
Related work

2.1 Overview

This chapter consists of four parts. In the first part, we introduce traditional object detection algorithm, which can be view as a basic method. In the second part, we introduce some two-stage object detection CNN models, which can be view as the development process of the object detection CNN. In the third part, we introduce some recently popular real time detection CNN and we will make a performance comparison and evaluate these CNNs in the last part.

2.2 Traditional object detection algorithm

2.2.1 Adaboost cascade object detection algorithm based on harr-like feature

The object detection task is much more complicated than the image classification task, because the object detection task not only needs to classify the object but also needs to accurately locate the object. Generally speaking, the position of an object in the image is random, and the shape of the same object observed at different angle is also very different. Therefore, for the object detection algorithm, the most important point is how to extract the features of the object.

The earliest feature extraction method is harr-like feature proposed by Viola et.al [30][31]. They used the harr-like feature for face detection. The harr-like feature is a series of black and white rectangular features (see **Fig. 10**). This kind of rectangular feature is equivalent to a sliding window, and when the sliding window is move to an area of human face, the feature of the face in the area can be calculated. The calculation is done by summing the pixel values of the white areas in the rectangular and subtracting the sum of the pixel values of the black areas. After that, Lienhart et.al proposed some extended harr-like features that made the extracted feature forms more diverse [32]. Since the size and shape of each human face are different, the size of rectangular sliding window is also not fixed. The algorithm performs translation and scaling on the sliding window and thereby effectively extracting harr-like features of different scale at different positions. The number of harr-like features is huge. For a 20*20 size image, there are about 100,000 generated harr-like feature images.

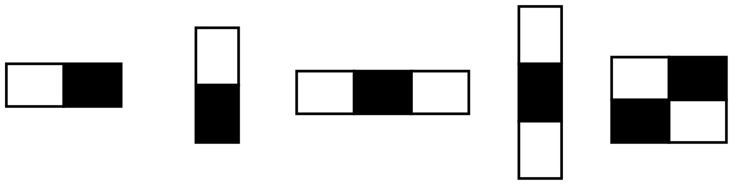


Fig. 10: Examples of harr-like feature

In 1984, Valiant et.al proposed a concept of a classifier [33]. If a classifier can obtain a slightly higher accuracy than a random guess, this kind of classifier is called weak classifier. If a classifier can significantly improve the accuracy of the random guess, this kind of classifier is said to be strong classifier. AdaBoost is an algorithm that improves the performance of weak classifiers [34]. We can get a strong classifier by training a weak classifier using AdaBoost algorithm combined with the harr-like features. However, when performing object detection tasks, a single strong classifier cannot solve high-precision complex tasks, we usually use cascaded strong classifiers to solve the problem.

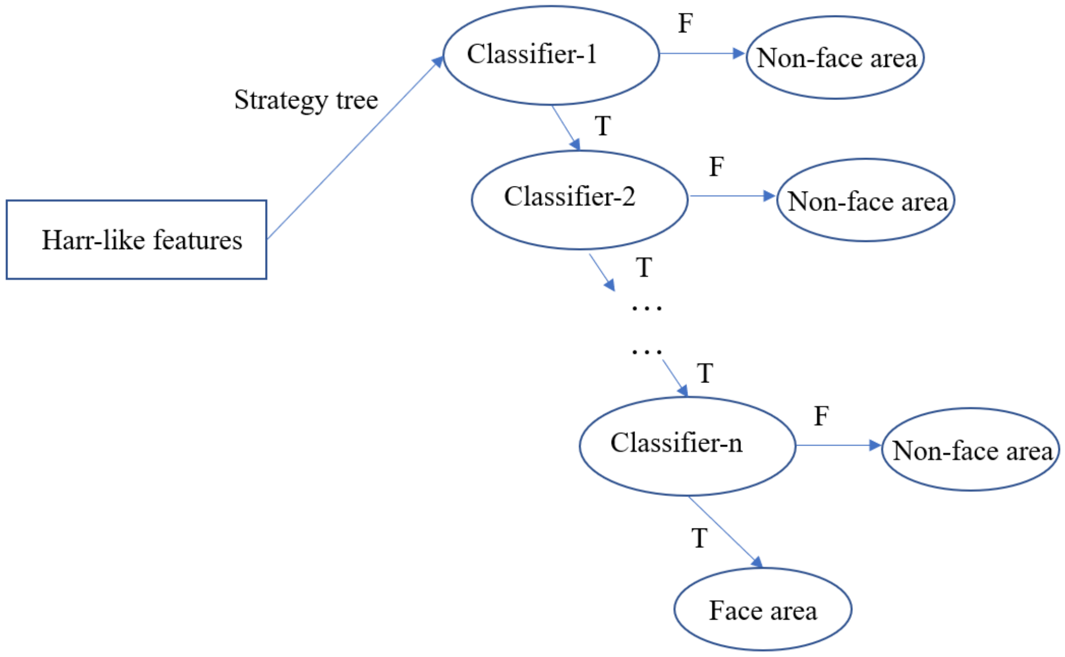


Fig. 11: Classification strategy tree based on cascade structure

Fig. 11 shows the classification process. The harr-like features will pass through a strategy tree. Each layer of the strategy tree is a strong classifier based on AdaBoost algorithm training. Each strong classifier consists of several weak classifiers. Each layer of the strategy tree filters the features that meet the criteria. Only features detected as non-negative regions can enter the next layer for further screening. The final positive region obtained by such layer screening is the target region of detection. By this way, the algorithm implements the detection task.

2.2.2 SVM detector based on HOG features

Histogram of Oriented Gradient (HOG) feature is similar to harr-like feature [35], which is a feature descriptor used for object detection. HOG calculate the histogram based on the gradient not based on color. This algorithm is based on an idea that if we use the entire image as a feature to count the gradient information, it is difficult to find the object features of the image through the gradient histogram. However, the features of the local area can be described by a gradient histogram. This is because the gradient information mainly exists at the edge position of the local object. Therefore, HOG constructs the features by calculating and counting the gradient direction histogram of the local region of the image.

Compared with harr-like feature, the HOG feature is not affected by the image lighting factor. Before constructing the HOG feature, it is necessary to perform gamma correction on the image. The effect of gamma correction is to improve the contrast between the light and dark parts of the image. The algorithm then divides the image into small connected areas called cell units. Each cell units will generate a gradient histogram. In order to make the HOG features have better illumination invariance, the gradient histogram will be normalized. At the same time, HOG features have the characteristics of rotation invariance and scale invariance, and the calculation amount is also smaller than harr-like features. These generated HOG features will be trained by SVM to achieve the purpose of detection.

2.2.3 Improvement based on HOG feature: DPM algorithm

HOG feature has a shortcoming that although it has a good detection performance on the front and back of the pedestrian, the side detection performance on the pedestrian is not so well. This is because the model based on HOG feature is not complicated enough. To solve this problem, Pedro F proposed DPM algorithm [36]. The essence of DPM is to use the HOG feature, but the HOG feature has been improved. The gradient of the original HOG feature is signed. DPM uses a method that combining signed gradient and unsigned gradient, which reduces the dimension of the HOG feature and reduces the time complexity.

Compared with the simple model of HOG, DPM proposed a multi-filter detection method and got better performance. DPM uses multiple resolution filters for detection an object. The response score is obtained by convolving the original object with these different resolution filters. The higher the response score, the closer the feature is to original object. Finally, a comprehensive response is made to detect the object. Taking human detection as an example (see **Fig. 12**[36]). **Fig. 12** (a) is a root filter, which is used for detection the whole contour of a person. **Fig. 12** (b) divides human body into six parts and each part is a filter with higher resolution than the root filter. This is used to detect the details of human body. **Fig. 12** (c) is a spatial model which shows the relationship between each part and root. The brightness of the pixel determines how relevant it is to the root. Combined with these three models, DPM can detect human more accurately.

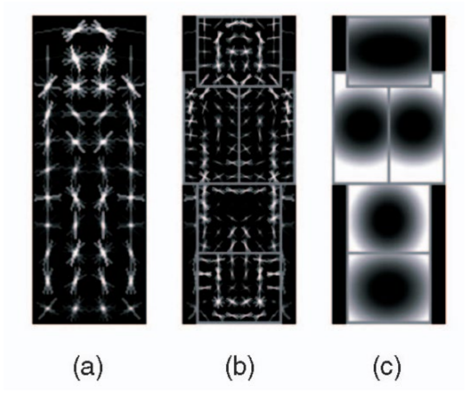


Fig. 12: A component person model for human detection[36]

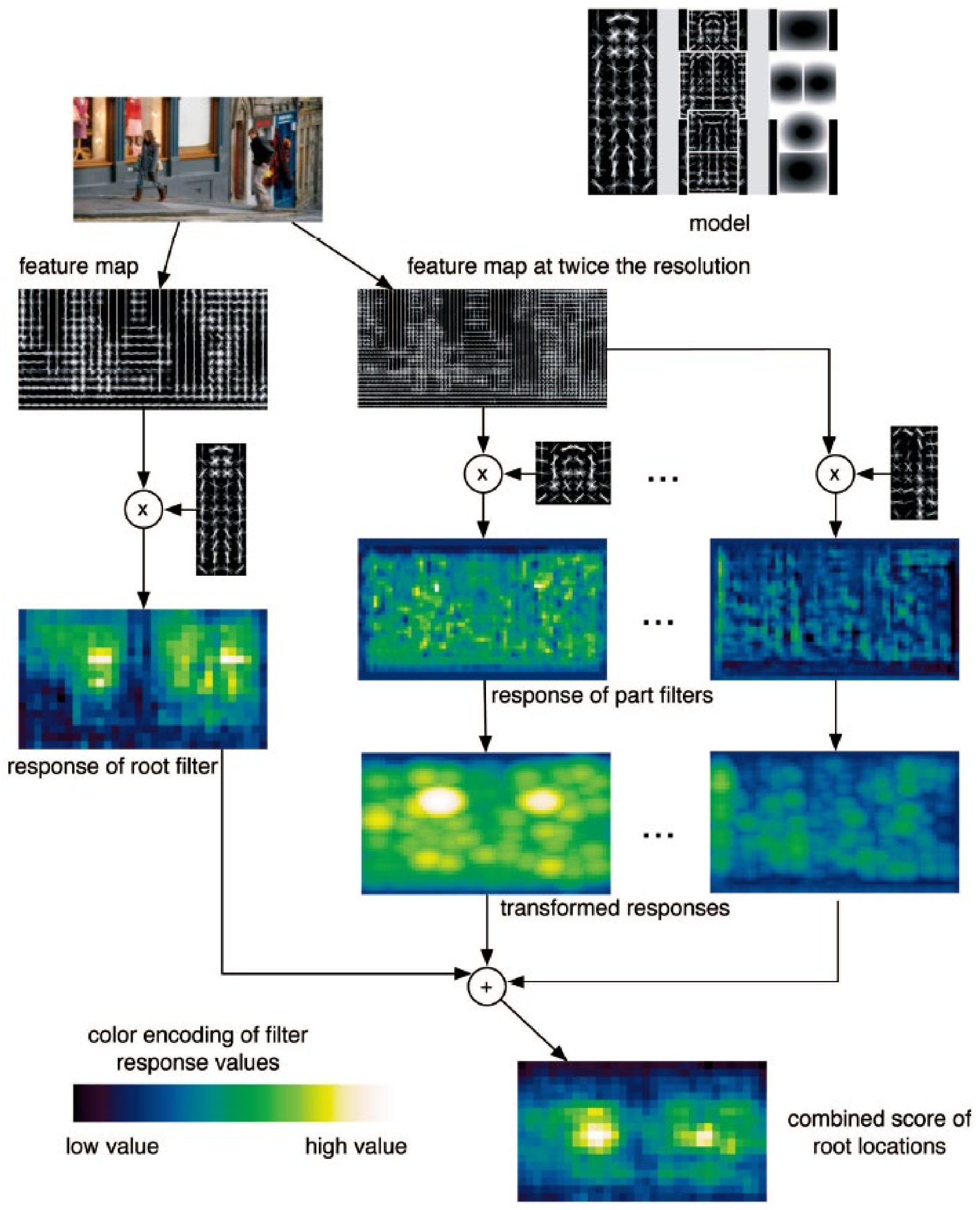


Fig. 13: The detection process of DPM[36]

Fig. 13[36] shows the detection process of DPM algorithm. DPM is also an algorithm based on sliding window. It builds resolution pyramids and uses filters of different resolution to progressively detect objects from contour to detail. This method is easy to understand and adapts well to the deformation of the object. However, this algorithm still has some shortcomings. This first is that this multi-resolution filter is manually designed. Different filters are required for different objects, and workload is huge. The second is that when the object rotates a lot, the detection performance is not good, and the adaptability needs to be improved.

2.2.4 Summary of traditional object detection algorithms

we introduced three traditional object detection algorithms in this section. it is easy to conclude their common features. The traditional object detection algorithm is mainly divided into three steps (see **Fig. 14**). The first step is region selection. Algorithms for region selection are generally based on sliding windows. Sliding windows traverse the entire image with filters of different sizes. This strategy enumerates all possibilities of object position but reduces the efficiency of the algorithm. The second step is feature extraction. Commonly used feature extraction methods are harr-like features and HOG features. Since the shape, position, environmental background and illumination changes of the object are all uncertain, it is difficult to extract robust features. The final step is to train the classifier. The commonly used classifiers are SVM and Adaboost. The process of training the classifier is actually the process of finding the true positive features of the target object. We can use the trained classifier to detect the target object.

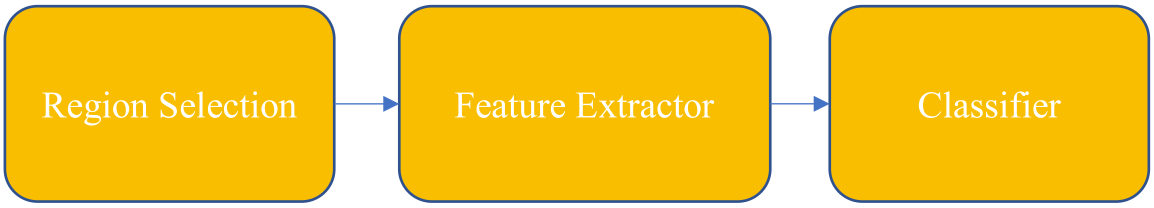


Fig. 14: Traditional object detection method

In summary, traditional object detection algorithm has two shortcomings. One is that region selection strategy is based on the sliding window algorithm with high time complexity and redundancy. The other is that the features of the manual design lack robustness for diversity changes.

2.3 Two-stage object detection CNN models

2.3.1 R-CNN (Region CNN)

In 2014, Girshick R first used CNN for object detection task and made a great progress. This CNN model is called R-CNN [37]. Compared with the huge computational complexity of sliding window extraction features, R-CNN uses a method called selective search to calculate candidate regions, which greatly improves the efficiency of extracting features [38]. Selective search uses the similarity between regions to extract object features. This algorithm splits the image into different regions and then merges regions with similar features such as texture, color, and contour (see **Fig. 15**[38]). The merged region is used as a candidate region for detection. This results in multiple candidate regions by hierarchical merging. This algorithm does not exhaust all possibilities. The selective search algorithm in R-CNN will generate approximately two thousand candidate regions, which is why it is better than the exhaustive method.

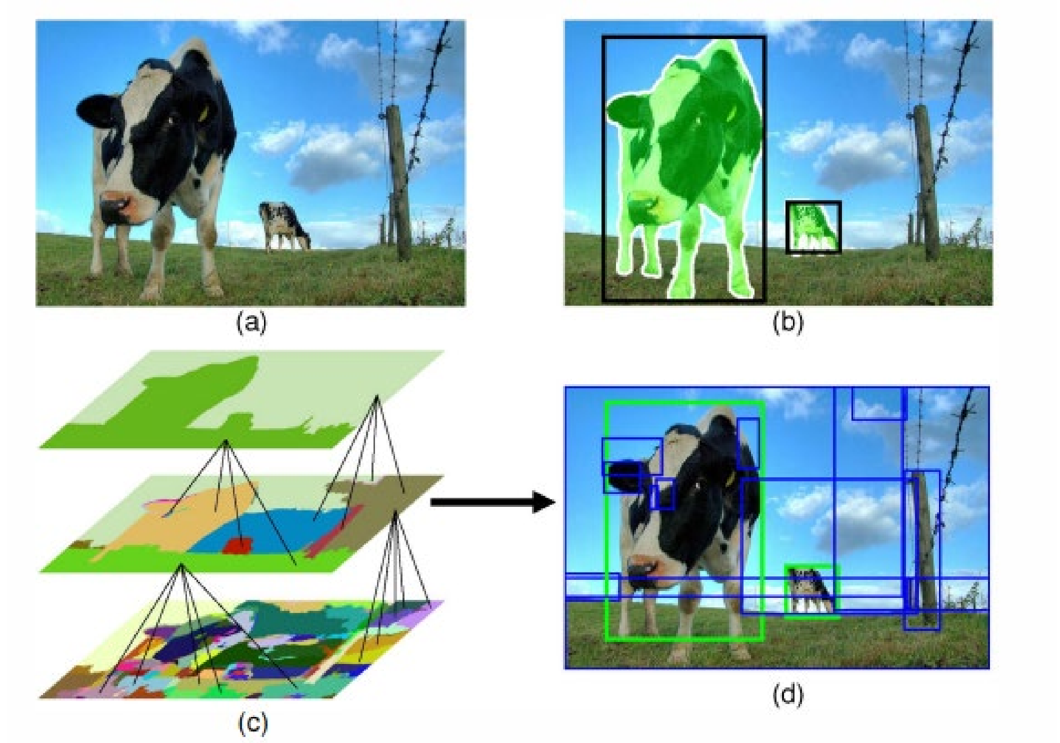


Fig. 15: Selective Search method[38]

After the candidate regions is obtained, the R-CNN algorithm will use CNN to extract feature. After the image is input to CNN, CNN will normalize the size of candidate regions and then perform feature extraction. Finally, the features of CNN output will be classified by SVM. Although R-CNN improves detection performance, it also has several shortcomings. First, the R-CNN needs to pre-read candidate regions of the image, which requires a large amount of computer storage resources. Secondly, when the R-CNN normalizes the candidate region, the object may be deformed due to normalization resulting in distortion. Finally, there is a large amount of overlap in the candidate regions generated by selective search. Since each candidate region will be extracted in the CNN, too much overlap will cause the CNN to repeatedly extract the same feature. This will increase the redundancy of the algorithm. The framework of R-CNN is shown in **Fig. 16**[37].

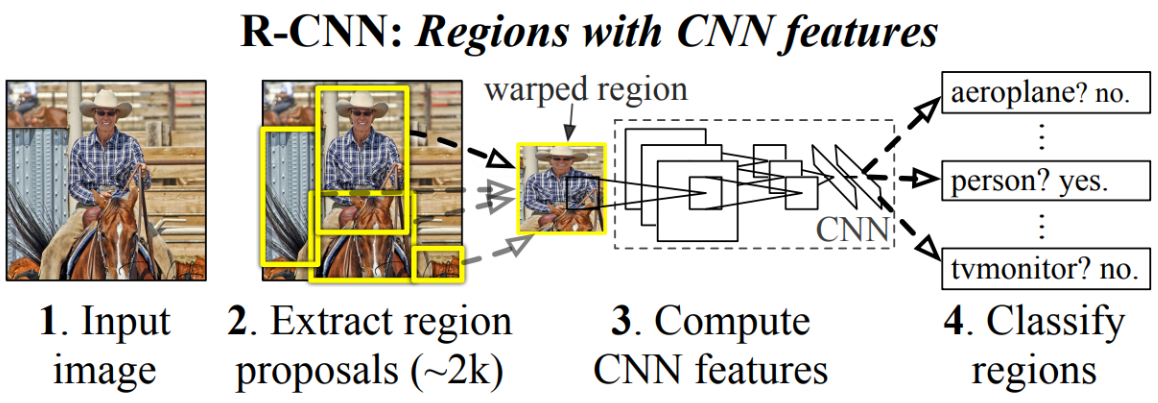


Fig. 16: The framework of R-CNN[37]

2.3.2 Spatial pyramid pooling (SPP-NET)

Like R-CNN, the input of the previous CNNs is fixed to the image size. This is because in the last fully connected layer of CNN, a fixed dimension input is required. In fact, a fixed size can cause image distortion and affect algorithm performance. From the principle of recognizing images by the human brain, the fixed size is superfluous. The human brain does not morph the image, but first recognizes the image as a whole, and then analyzes the features hierarchically to achieve the purpose of recognition. SPP-NET [39] the problem of fixed input size of CNN. By adding a spatial pyramid pooling before the fully connected layer, the features of the same dimension can be extracted from the different sizes of images output by CNN. The function of the spatial pyramid pooling is to enable the CNN to input images of ant size and provide the same dimension input the fully connected layer.

Fig. 17[39] shows how spatial pyramid pooling layer work. The input of this layer is the feature maps with different sizes. The spatial pyramid pooling layer has three sizes of pooling window, which are 4*4, 2*2, and 1*1 respectively. Each size pooling window can slip the feature map into equal parts. Then the pooling operation is performed in each grid of pooling window. So that the features of the same dimension can be obtained. SPP-NET also has several shortcomings. First, like R-CNN, SPP-NET is also trained in stages, and the features of each stage require a large amount of computer storage resources. This means that the CNN model of SPP-NET cannot be too deep. Second, the CNN model and SVM classifier are independent of each other in SPP-NET. In the backpropagation, the parameters on both sides cannot be updated at the same time, and the performance improvement cannot improve the performance of the other size. Compared with R-CNN, the advantage of SPP-NET is that the input to the network is the entire image not candidate regions, which is more flexible than R-CNN.

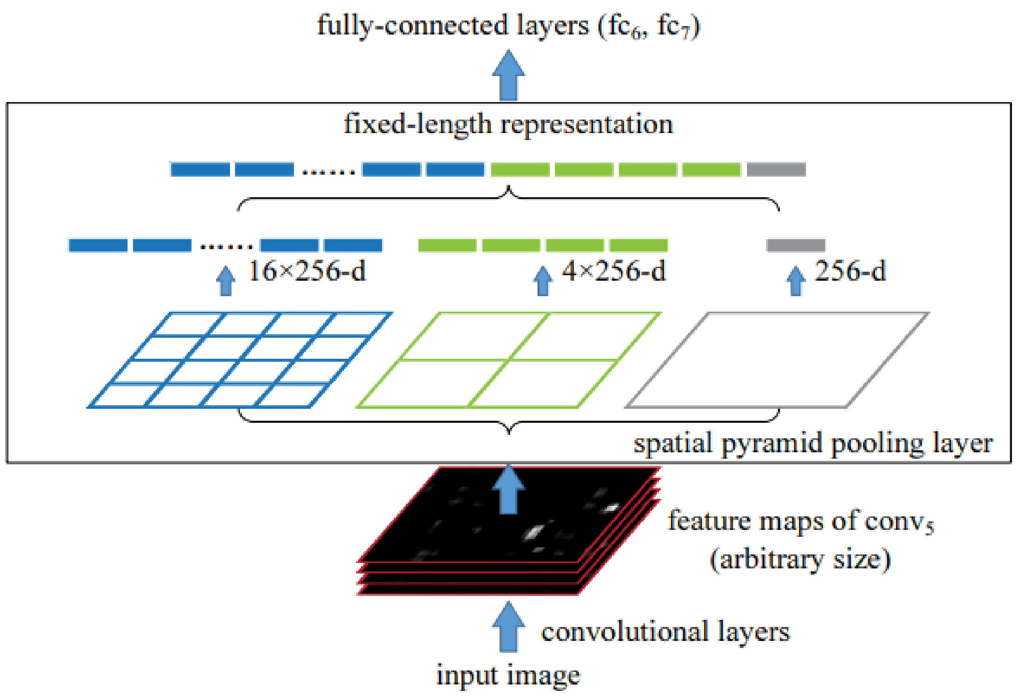

fully-connected layers (fc6, fc7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of conv5
(arbitrary size)
convolutional layers
input image

Fig. 17: Spatial pyramid pooling layer

2.3.3 Fast R-CNN

Independent training of each part is not only slow, but also unable to pass parameters in real time. To solve this problem, Girshick et.al proposed Fast R-CNN [40]. Fast R-CNN has two main contributions. First of all, Fast R-CNN has greatly improved the training and testing speed than R-CNN. Second, Fast R-CNN proposed the concept of multi-task loss. It adds the box regression directly to the CNN for training. This kind of end-to-end structure can update the weights better. **Fig. 18**[40] shows the architecture of Fast R-CNN. The input to Fast R-CNN is an image of any size. The candidate regions generated by the selective search algorithm are also taken as inputs, and these candidate regions are called region of interests (ROIs). Fast R-CNN extracts feature using a fully convolutional network. After the fully convolutional network generating feature maps of different sizes, these feature maps will be sent to the ROI pooling layer. This ROI layer has a similar function to the spatial pyramid pooling layer in SPP-NET and can fix feature maps of different sizes to the same size. The ROI pooling layer can map the ROI window to the feature maps according to the mapping relationship between the original input image and the feature maps, and the generate the same size feature maps by pooling operation. Fast R-CNN does not need to repeat extraction of features, and pooling is much simpler. Therefore, Fast R-CNN greatly increased training and testing speed.

Fast R-CNN does not use SVM as a classifier. Behind the ROI layer is the softmax classification and box regression. These two parts are trained in parallel and can be seen as a part of the neural network. This design allows the training weights of the entire network to be updated simultaneously, and this multi-task training method can improve the training efficiency. In addition, box regression does not require additional storage resources. The main shortcoming of Fast R-CNN is that although the latter half of the neural network performance is improved, the candidate regions is still calculated by selective search algorithm. This algorithm has a high time complexity, which hinders the efficiency of the entire neural network. From the results, Fast R-CNN gave us an inspiration. If we can find an algorithm instead of selective search, then we can achieve real end-to-end training. This kind of object detection can be better popularized in practical applications.

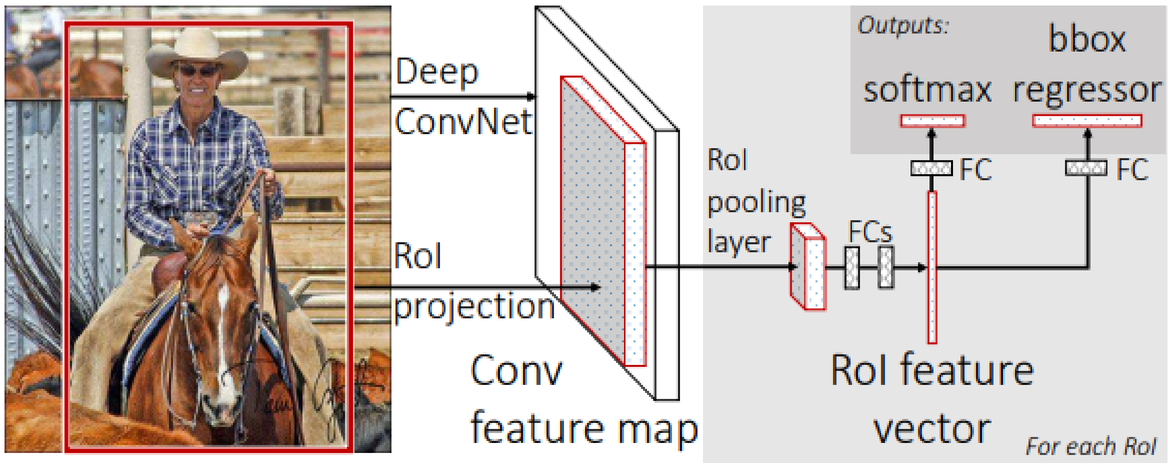


Fig. 18: Fast R-CNN architecture[40]

2.3.4 Faster R-CNN

Faster R-CNN effectively solves the problem of slow calculation of candidate regions [41]. In the network structure of Faster R-CNN, there is a sub-network to replace the selective search to calculate candidate regions. This sub-network is called region proposal network (RPN). Faster R-CNN can be seen as a network combining RPN and Fast R-CNN. After the input images enter Faster R-CNN, the features will be extracted by a CNN. The output feature maps of CNN are the same size as the input image. The feature maps generated here will be shared by the later RPN and ROI pooling layer. **Fig. 19**[41] shows the structure of RPN. In the RPN, multiple candidate region boxes will be predicted for each sliding window position on the feature map. Each candidate region box can be uniquely determined by ratio, size, and center point. These candidate boxes are called anchors. RPN define nine different sizes and ratios of anchors with sizes of 128*128, 256*256 and 512*512. The ratios are 1:1, 2:1 and 1:2 respectively. In order to determine whether there are detected objects in these candidate regions. RPN will give the region score by calculating the degree of overlap between the anchor and the ground truth area. If the region score is higher than the threshold, it will enter the ROI pooling layer as a positive candidate region. At the same time, the feature maps will also enter the ROI pooling layer. The rest structure of Faster R-CNN is the same as Fast R-CNN. Faster R-CNN needs to train two networks during the training process, one is the RPN and the other is the classification network. However, the convolutional layer parameters of the two networks are shared, so the alternate training mode can be selected to make Faster R-CNN converge faster.

Let’s discuss the shortcomings of Faster R-CNN. First of all, although Faster R-CNN achieves an end-to-end structure, the detection speed is still not ideal enough. The requirements for real time detection are still not met. Secondly, the method of calculating candidate regions by RPN does improve the performance compared with selective search, but it is still computationally intensive. Nevertheless, Faster R-CNN is also important for object detection.

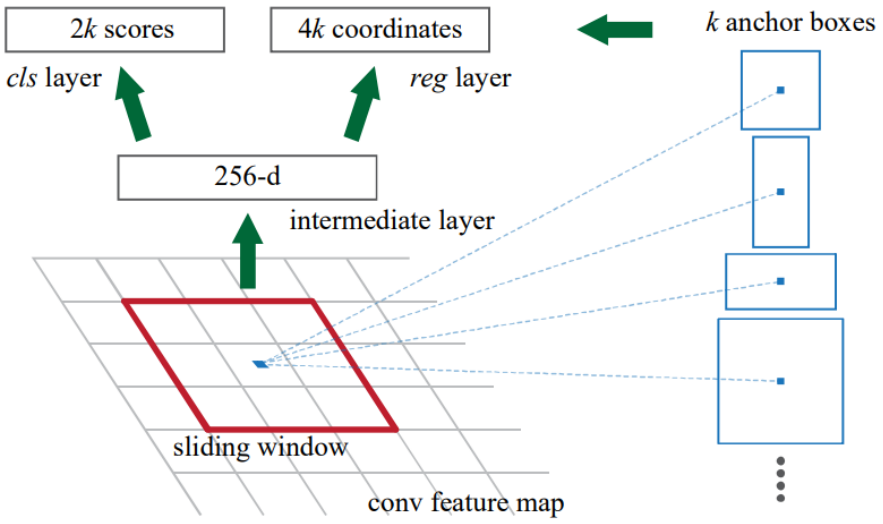


Fig. 19: Region proposal network[41]

2.3.5 Summary of two-stage object detection algorithms

A series of improvements to the two-stage algorithms greatly improves detection performance compared with traditional object detection algorithms. The two-stage algorithms mainly improve the traditional algorithms from the following two parts. First, the traditional algorithm needs to train each part of network independently, and the latest two-stage algorithm has achieved end-to-end training. Secondly, the traditional algorithm uses selective search to extract candidate regions, but two-stage algorithms propose a method based on region proposal, which reduces the computational complexity and avoids repeated extraction features. The shortcoming of the two-stage algorithms is that although the performance of the algorithm is improved, the detection speed of the algorithm is still not up to the real-time detection due to existence of the fully connected layer.

2.4 Single-shot object detection CNN models

2.4.1 Yolo

It is unreasonable to pursue the accuracy blindly and the speed cannot be improved. Therefore, in the method of improving the object detection performance, there are also some algorithms that sacrifice a part of the precision and greatly increase the speed, and Yolo is one of them [42]. Compared with two-stage object detection algorithms which are based on region proposal, Yolo turns the classification problem into a regression problem. We take use of **Fig. 20**[42] as an example to explain the system principle of Yolo.

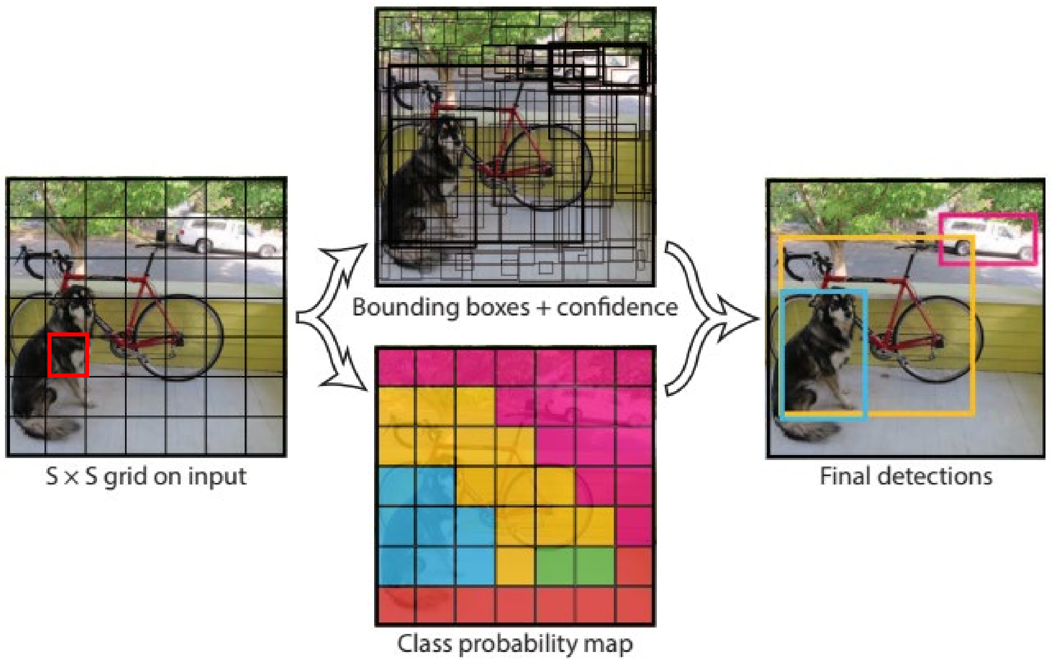


Fig. 20: The system model of Yolo[42]

Yolo first resizes the input image and turn the size of the input image into a square. The Yolo divides the image into equal-sized regions using S*S grids. In **Fig. 20**, S is 7, and the image is divided into 49 grid regions. Each grid will predict B bounding boxes and probability values for C categories. In **Fig. 20**, B is 2 and C is 20. In Yolo, each bounding box needs to be represented by five parameters: the coordinate of the center point on the x and y axes, the height h and width w of the bounding box, and the confidence value of the predicted bounding box. In Yolo, each grid only predicts one object. Only when the center point of the object is in a grid, the corresponding grid will predict that object. In **Fig. 20**, we can clearly see that there is a dog in the lower left part of the image. The dog's center point is in the red grid, so only the red grid will predict the dog category, and the grid next to the red grid will not predict the dog category. This means that Yolo has a drawback. For small objects groups such as birds and insects, Yolo is difficult to detect. This is because there may be multiple object centers in a grid, and each grid of Yolo can only predict one object, leading to prediction errors. This is why Yolo's accuracy is lower than Faster R-CNN. After the prediction, Yolo will generate a large number of prediction boxes, many of which are meaningless. In order to avoid multiple detections of the same object, Yolo uses the non-maximum suppression method to ensure that each object is detected only once [43]. Finally, Yolo selects the prediction box with the highest confidence value as the detection result.

Fig. 21[42] is the network architecture of Yolo. It is easy to find that the entire Yolo network is composed of convolutional layers and fully connected layers without any sub-network structure. This is why Yolo is faster than other two-stage models. Here, the output dimension of Yolo is 7*7*30. This is because when S=7, B=2 and C=20, each bounding box needs (x,y,w,h,c) five parameters and each grid predicts two bounding boxes in total requiring 7*7*(5*2+20) parameters. Yolo's speed and accuracy are inversely proportional. According to actual needs, we can adjust the size of Yolo's grid and the size of input image to adjust the balance of accuracy and detection speed of Yolo. When the input image size is large, Yolo's detection speed will be slower and the accuracy will increase. When the input image size is small, Yolo's detection speed will be faster and the accuracy will decrease. Therefore, Yolo can adapt to various environmental needs in practical applications.

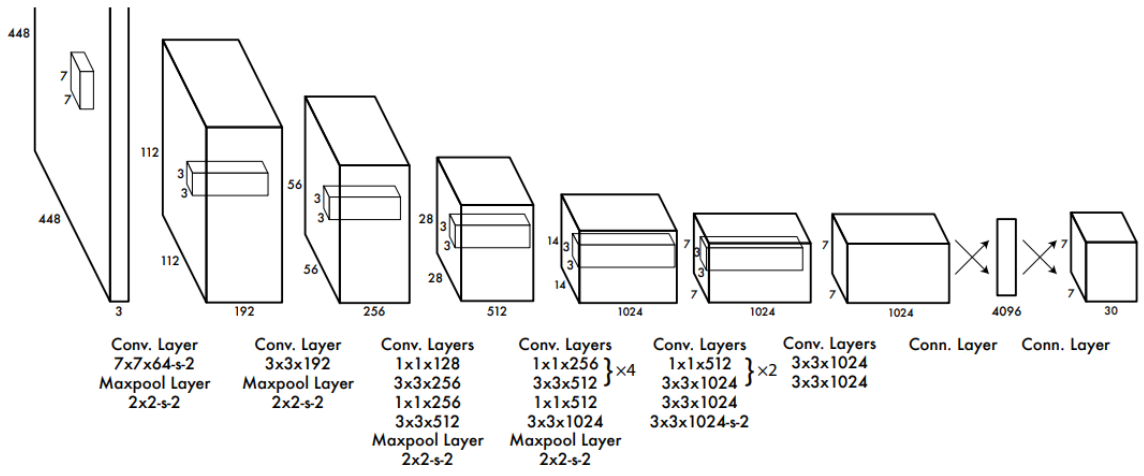
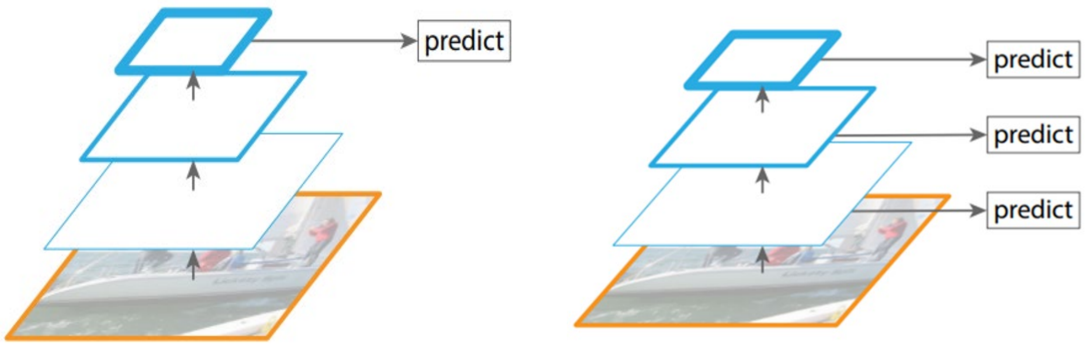


Fig. 21: The architecture of Yolo[42]

2.4.2 Single shot multibox detector （SSD）

In the real time detection field, there is a model faster than Yolo, and the performance is close to the Faster R-CNN. This model is called SSD[44]. The method of SSD feature extraction is inspired by FPN[45]. In the traditional object detection algorithm, the method of feature extraction is similar to the pyramid structure (see **Fig. 22**[45] (a)). The convolutional layers in the front part extract low-level features, and the deep convolutional layers extract high-level features. The last convolutional layer extracted features are used for category prediction. One shortcoming of this method is that it ignores the impact of low-level features on prediction accuracy. From a physiological point of view, human brain tends to pre-judicate object categories through low-level through low-level features such as contours, colors, shapes, and then recognized objects based on the features of the details. In other word, the combination of low-level features and high-level features can better recognize objects. SSD is taking this idea. The way SSD extracts features is also a pyramid structure. The difference is that the features extracted by each layer are used for prediction (see **Fig. 22**[45] (b)), which can better identify objects and improve accuracy.



(a) Traditional feature extraction method (b) Feature extraction method in SSD

Fig. 22: Different feature extraction method[45]

In the design of the anchor part, SSD takes a similar method to Faster R-CNN. Faster R-CNN takes the method of generating nine different sizes anchors at each prediction point. Then get the final bounding box through the ROI pooling layer and classifier. SSD take six anchors generated at each prediction point (see **Fig. 23**[44]). The size of the anchor is different according to the size of feature map. The larger feature map has smaller anchors, which can better detect large features. The small feature map has larger anchors, which can better detect small features. The combination of the big anchors and the small anchors finally use NMS method to determine the bounding box. Although this method can extract features better, it also has shortcomings. The anchor size of each feature map cannot be adjusted by learning. The anchors need to be pre-set manually. Since the feature map size of each layer is different, the anchor size is also different. The setting of the anchor size usually depends on experimental experience, and it is difficult to find the most suitable setting method.

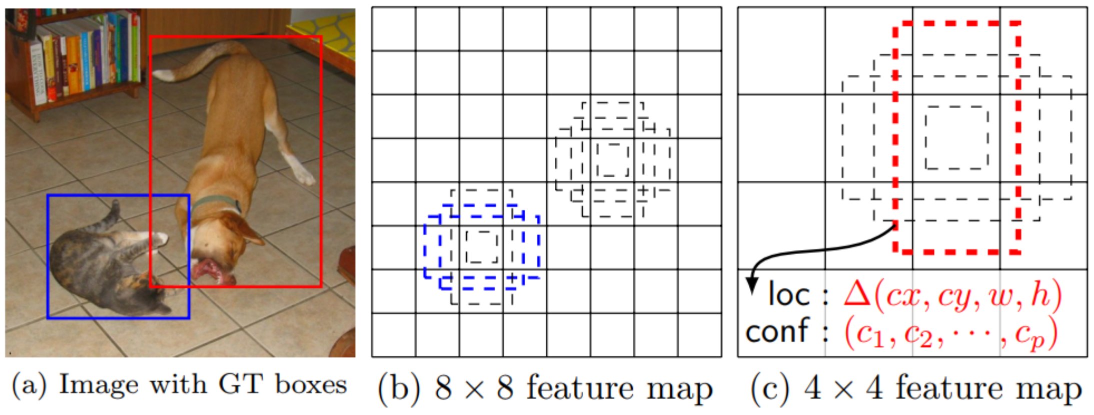

loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$
(a) Image with GT boxes
(b) 8×8 feature map
(c) 4×4 feature map

Fig. 23: SSD anchors on different sizes of feature maps[44]

2.4.3 Yolov2

Yolov2 is an improved version of Yolo[46]. Yolov2 uses a series of methods to improve the accuracy on the premise of ensuring the original speed. It is the most stable object detection model at present. Yolov2 mainly made the following improvements.

First, Yolov2 adds batch normalization after each layer to reduce the overfitting of the network [47]. This is because the essence of neural network training is to learn the data distribution. When the distribution of training dataset is different from the distribution of testing dataset, the performance of the network is bad. The distribution of datasets can be made consistent by batch normalization.

Second, Yolov2 made improvements to the image resolution adaptation. In the training network of Yolo, the input image size is 224*224, and the input image size of the detection network is 448*448. When training Yolo with large size images, the resize process will result in insufficient image resolution. This will cause Yolo's poor detection performance on high resolution images. Yolov2 divided the training process into two parts. First training images with size of 224*224 and then training images with size of 448*448. By this way, Yolov2 has better adaptability to high resolution images.

Third, Yolov2 removed the fully connection layer of Yolo and borrowed the idea of taking use of anchor for prediction. Yolov2 ensures that the height and width of the feature maps are odd by adjusting the size of the input image. The merit of this is that when the feature map of Yolov2 is divided into grids, it can be guaranteed that the feature map has only one center grid. The reason for this is that the essence of Yolo is to use the position of the center point of the object to make predictions. The large object usually occupies the center of the entire image. If the height and width of the feature map are even, there will be four center grids in the feature map. These four center grids will predict the same object, which is obvious redundant.

Fourth, in Faster R-CNN and SSD, the sizes of anchors are set manually, and Yolov2 uses k-means clustering to find the most suitable anchor size setting for the network. Yolov2 uses intersection over union to redefine the loss function of k-means. The size of the anchor does not lead to error. Yolov2 found that the network performance and error can be optimally balanced at K=5. That is to say, each grid of Yolov2 only predicts five anchors but achieves the same performance as nine anchors per point of Faster R-CNN.

Fifth, Yolov2 improves the prediction method of anchor. In Faster R-CNN and SSD, the prediction of anchors is made by using offset. Both networks adjust the position of anchors by calculating the offset value between anchors and ground truth of the object. However, all parameters are randomly initialized wen the network is just starting training and there is no limitation on offset, which may cause the anchor of the initial prediction to be far from the true position of the object. This kind of prediction is meaningless and reduces the efficiency of network training. Yolov2 uses the relative position of the anchor and the grid cell to make the anchor not deviate from the center of the object. This makes the network more stable and easier to converge.

Sixth, Yolov2 borrowed the idea of ResNet [48] to propose a passthrough layer. Yolov2's feature map size is 13*13, which is enough to detect large objects. However, the detection performance of small objects is not good. This is because the features of small objects are easily disappeared in the deep convolutional layer. Yolov2 uses the passthrough layer to concatenate the feature maps of the previous convolutional layer and the feature maps of the deep convolutional layer. This prevents features of small objects from disappearing in deep convolution.

Last, Yolov2 proposed a new network architecture which is called darknet (see **Table 1**). This network is quite faster than Yolo. Yolo's network is based on Google's VGGNet [49], which contains twenty-four convolutional layers and two fully connected layer. Because this network is very deep, the parameters of this network exceed thirty billion orders of magnitude. Since the fully connected layer has a large number of parameters, darknet removes the fully connected layer. At the same time, darknet only contains nineteen convolutional layers, the reduction of the convolutional layer greatly increases the speed of the network.

Yolov2 is faster, stronger and better than Yolo. It tells us that in order to get better performance, we must not only improve the overall structure of the network, but also improve the network details.

Table 1: Darknet-19[46]

Type	Filters	Size/Stride	Output
Convolutional	32	3*3	224*224
Maxpool		2*2/2	112*112
Convolutional	64	3*3	112*112
Maxpool		2*2/2	56*56
Convolutional*3	128,64,128	3*3,1*1,3*3	56*56,56*56,56*56
Maxpool		2*2/2	28*28
Convolutional*3	256,128,256	3*3,1*1,3*3	28*28,28*28,28*28
Maxpool		2*2/2	14*14
Convolutional*5	512,256,512, 256,512	3*3,1*1,3*3, 1*1,3*3	14*14,14*14,14*14, 14*14,14*14
Maxpool		2*2/2	7*7
Convolutional*5	1024,512,1024, 512,1024	3*3,1*1,3*3, 1*1,3*3	7*7,7*7,7*7, 7*7,7*7
Convolutional	1000	1*1	7*7
Avgpool		Global	1000
Softmax			

2.4.4 Yolov3

Yolov3 [50] is the final version of Yolo currently, with minor improvements on the basis of Yolov2. There are three main improvement. First, Yolov3 draws on the ideal of FPN to make multi scale predictions of features. Based on Yolov2, the prediction scale is change from two to three, which can extract the features of overlapping objects more accurately. Second, Yolov3 uses darknet-53 network as a classifier. This network is deeper than Yolov2's darknet-19 network. Finally, Yolov3 does not use the softmax function for classification, but instead uses the logistic function. This is because the softmax function does not apply to multi scale classification. Compared with Yolov2, Yolov3 have better performance in detecting small objects, the training time is two to three times more than Yolov2 due to the deeper network structure.

2.4.5 Summary of single-shot object detection algorithms

The single-shot object detection algorithm is currently the fastest real time detection algorithm. Compared with the two-stage algorithms, there are mainly three improvements. First, the single-shot algorithm converts the object detection task into a regression problem, simplifies the process of object detection, greatly improves the detection speed, and proposes the idea of end-to-end detection. Secondly, single-shot proposed a multi scale training method that is more flexible than two stage algorithms and can adapt the different datasets. Third, the single-shot algorithm improves the anchor design of two stage algorithms, and finds the optimal anchor setting by using clustering.

2.5 Performance comparison

We show performance comparison of previous object detection models in **Fig. 24**. In **Fig. 24**, the horizontal axis represents the detection speed of the model, and the vertical axis represents the detection accuracy of the model. These results are based on VOC2007 dataset. VOC2007 is an open dataset with approximately 10,000 labeled images of RGB channel. It is one of the commonly used datasets in the field of computer vison. Since traditional object detection algorithms are outdated, we have listed the performance comparison results of the two-stage models and single-shot models. As can be seen from the results, Yolov2 shows better performance in both speed and accuracy. We don't have the results of Yolov3 on the VOC2007 dataset, but we can predict that Yolov3 will perform better than Yolov2. Since these results is based on the RGB color image dataset, it can only be used as a reference. There is currently no test result on X-ray dataset, so this time we conduct our experiments on X-ray dataset to see the results. We chose Yolov2 as the object detection model for our experiments. This is because

Yolov2 is the most stable network by now, and our hardware resources are limited. Experiments on Yolov3 need more time, this will be our future work.

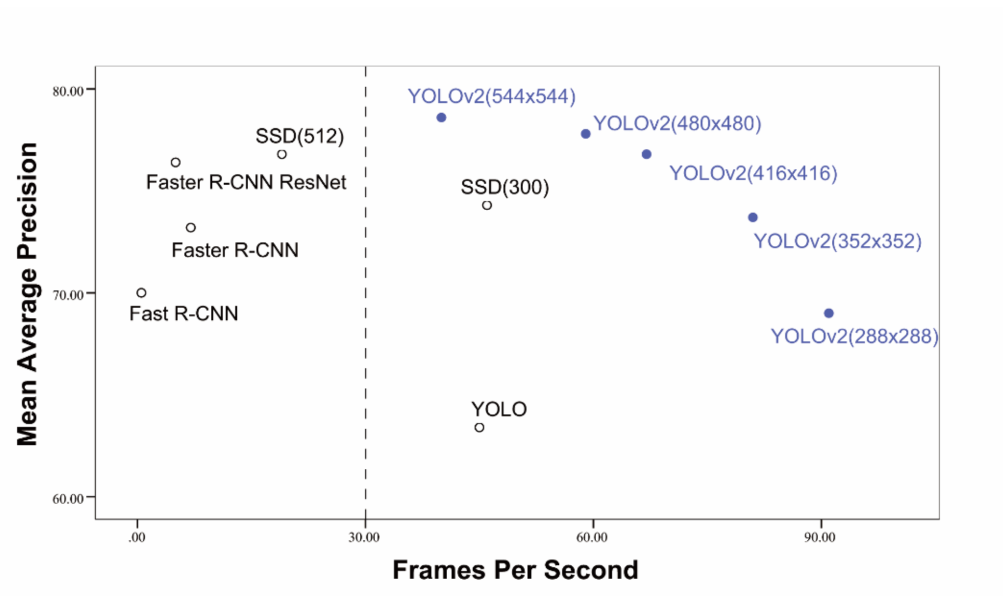


Fig. 24: Performance comparison of object detection CNN models based on VOC2007 dataset

Chapter 3
Proposed Method

3.1 Overview

In this chapter, we introduce our research method for dangerous object detection on X-ray images. In the first part, we provide some basic knowledge of CNNs to help readers who are not familiar with these concepts. In the second part, we introduce how to label the images and create annotation files for training data. In the third part, we describe our raw X-ray data collection method. In the fourth part, we explain how to synthesize X-ray images. In the last part, we propose the whole framework of our security system.

3.2 Preliminary

3.2.1 Definition of mAP and Recall

In the object detection task, there are two metric measures which are often used to evaluate the performance. One is mean average precision (mAP) and the other is Recall. Before explaining these two concepts, we explain the follow four mathematical definitions first. When preparing a training dataset, we will label the objects that need to be detected. We usually label them with rectangular boxes, which represent the real position of the objects in the image. These rectangular boxes are called ground truth. Assuming that the rectangular box with solid line in **Fig. 25** is ground truth, the real detection result is usually deviated from the ground truth. We use the dotted rectangle in **Fig. 25** to represent the real detection result. We define the intersection of the real detection result and ground truth as true positive (TP), which means this detected part is related to the object. Instead, we define the rest of the dotted rectangular to be false positive (FP), which means this area cannot be classified as a part of the object. The area outside the two rectangular boxes is defined as true negative (TN), which means that the model correctly classifies this area as a part not belonging to the object. The rest of solid rectangular is defined as false negative (FN), which means this area belongs to the object, but the model fails to detect this area. Equation (3) shows the definition of Recall. Recall shows the ability of the object detection model to reject nor-related information.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{3}$$

To explain mAP, we first introduce the definition of Precision. The mathematical formula of Precision is shown in Equation (4), which is used to measure the ability of the object detection model to detect relevant information. Precision and Recall have different mathematical implications. In the case that we want to detect more objects and make as few missing objects as possible. We should try to make Recall as large as possible. In the case that we want to detect a higher proportion of relative objects. The lower the proportion of unrelative objects, the better. We should try to make Precision as large as possible. Although there is no relationship between both values, we rarely encounter the situation that both Precision and Recall are large enough in real experiments.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{4}$$

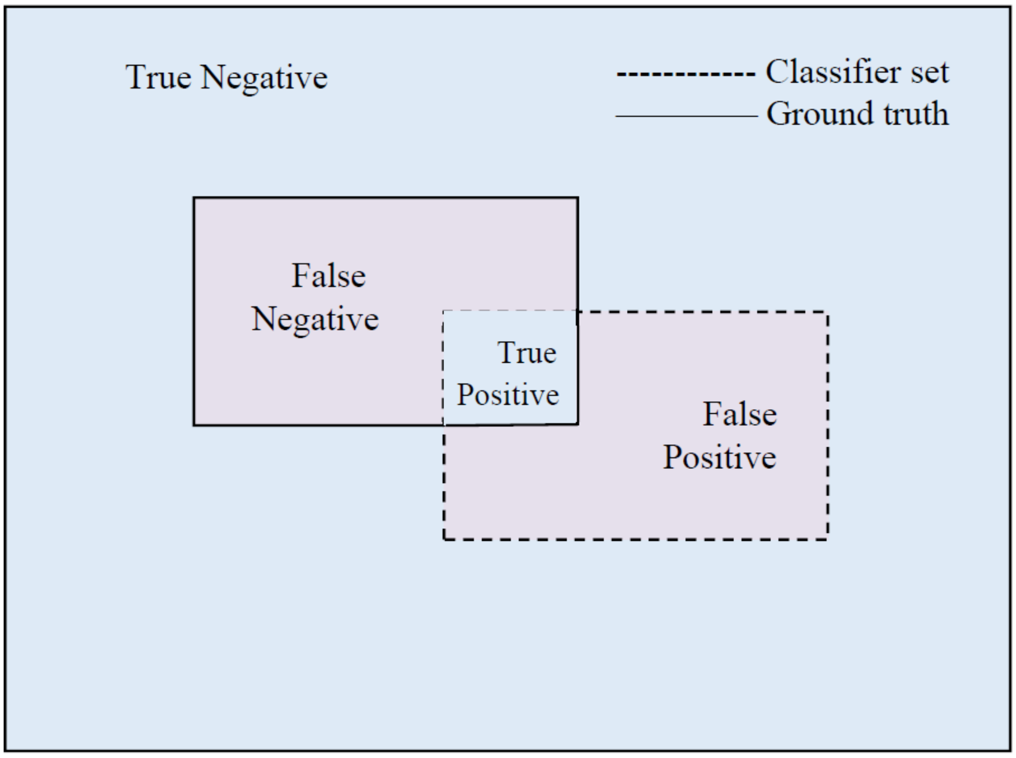


Fig. 25: Definition of TP, TN, FP and FN

Therefore, we believe that these two values are mutually constrained. That is to say, when Recall becomes larger, Precision will become smaller. Conversely, when Precision becomes larger, Recall will become smaller. In the experiments, we need to find a balance point according to actual needs and choose the appropriate Precision and Recall values. This balance point is called threshold, and different Recall values and Precision values can be obtained by adjusting the threshold value.

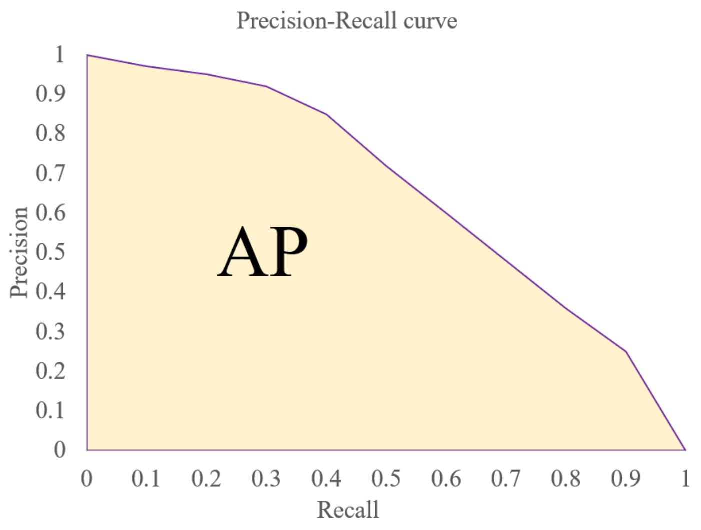


Fig. 26: Example of Precision-Recall curve

By varying the threshold, Precision and Recall are changing and we can get a Precision-Recall curve (see **Fig. 26**). We define the area which is enclosed by the curve and the X axis as average precision (AP). In the case that the curve is continuous, it is obviously that this area can be obtained by calculating the integral (see Equation (5)).

$$\mathrm{AP} = \int_0^1 PR\, dr \tag{5}$$

For discrete Precision-Recall curve, we can calculate AP by summing (see Equation (6))

$$\mathrm{AP} = \sum_{i=1}^{n} P(i)\Delta r(i) \tag{6}$$

In the experiment, we can get multiple AP values from each classification. The mAP value is the average of APs. Both mAP and Recall are important metric measures for evaluate the performance of the algorithm.

3.2.2 Positive and negative examples

In deep learning, the training dataset is usually divided into two parts. One part is positive examples and the other part is negative examples. Proper preparation of positive and negative examples is a prerequisite for training a good classifier. Positive and negative examples are not arbitrarily chosen. A good dataset can simulate complex and varied real environments, enabling the classifier to learn various situations. For example, if we want to train a classifier to detect cars, there is no doubt that the positive examples are different kinds of cars. The question is that what is the negative example. Specifically, the negative examples can be any object that is not a car. However, in order to make the training dataset more like the real environment, we usually choose objects related to the driving environment as negative examples, such as trees, mountains, rivers, roads, pedestrians and buildings. In this case, it makes no sense to choose objects in the room such as tables, stairs and beds as negative examples. Let's take another example, if we want to detect faces of patients lying in the hospital bed. The negative examples should be doctors, nurses, windows and medicines. Taking images of blackboards and clouds may make no sense because these objects do not match the real environment of the hospital. In the case of dangerous object detection of baggage security, which is our experiment. The positive examples are three kinds of dangerous objects: scissor, knife and bottle. We take some safe objects that often appear in security checks as negative samples such as bags, clothes, tape and umbrellas.

In large datasets, the number of negative examples is much larger than the number of positive examples. Even in small datasets, the number of negative samples is one to three times more than the number of positive examples. So how can we generate so many examples? If our dataset is very limited, we can cut a high pixel resolution image into multiple low pixel resolution images to generate negative examples. Suppose that we have an image with pixel resolution of 10000*10000 and the detector can work on the resolution of 50*50. Now, we cut this 10000*10000 resolution image into pieces of 50*50. We can get 40,000 negative examples. Then we repeat this process and cut this large resolution image into pieces of 20*20. We can get 250,000 negative examples. The process can be repeated multiple times. Finally, we can enlarge these negative examples and force them into the same size. This method has a limitation that the generated negative examples are not sufficient diverse. However, if we can create a dataset with thousands of high pixel resolution images and cut all theses images into pieces. We can get millions of negative examples with high variance. After than we can train a good detector.

When we train the labeled dataset with Yolo, we do not need to deliberately prepare positive and negative examples. This is because Yolo can automatically prepare positive and negative examples through anchor prediction. Let us introduce the concept of Intersection over Union (IOU). **Fig. 27** shows the definition of IOU, the upper left rectangle is the ground truth, and the lower right rectangle is the detection result. Since the detection results is always different from the ground truth, IOU is defined by dividing the intersection of the detection result and the ground truth by the union part of them. In Yolo, the threshold of IOU is always set to be 0.5. That is to say, when IOU value of the detection result is greater than 0.5, it will be treated as a positive example. When IOU value of the detection result is less than 0.5, it will be regarded as a negative example. By adjusting the threshold of IOU, we can control the proportion of positive and negative examples. This method is also applicable in Yolov2 and Yolov3.

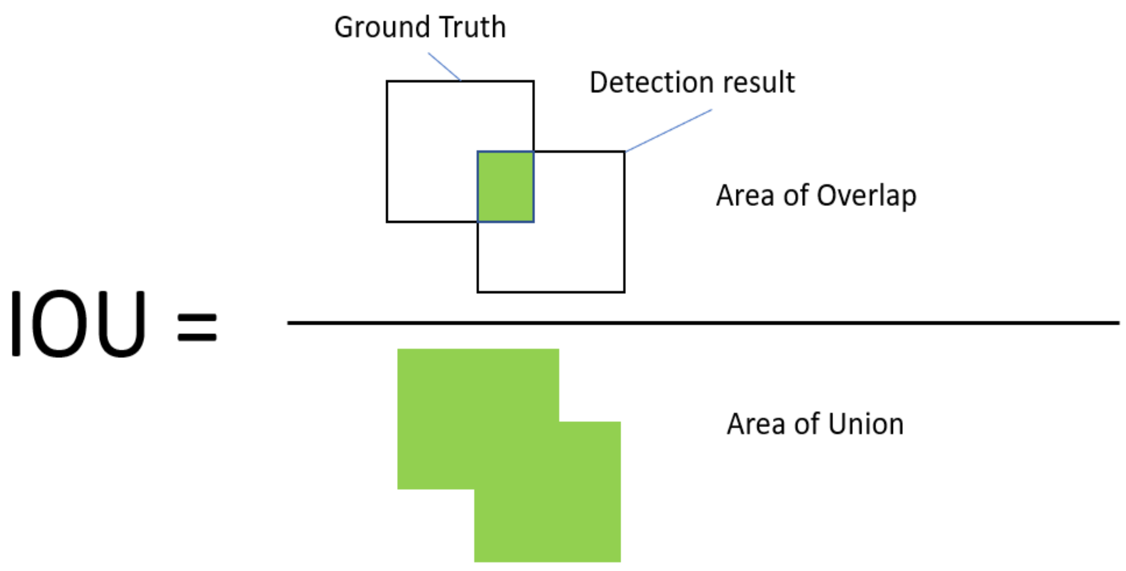


Fig. 27: Definition of IOU

3.3 Preparing training dataset

We use a software called labelImg to label our X-ray images (see **Fig. 28**). This is a software for labeling images, and the software menu is on the left side. Click open button or open dir button to open an image or a folder. Click Change Save Dir to modify the path to save the label file. Click Prev Image or Next Image to switch between the previous or next image. Click Create RectBox to draw a rectangle and add a label to the image.

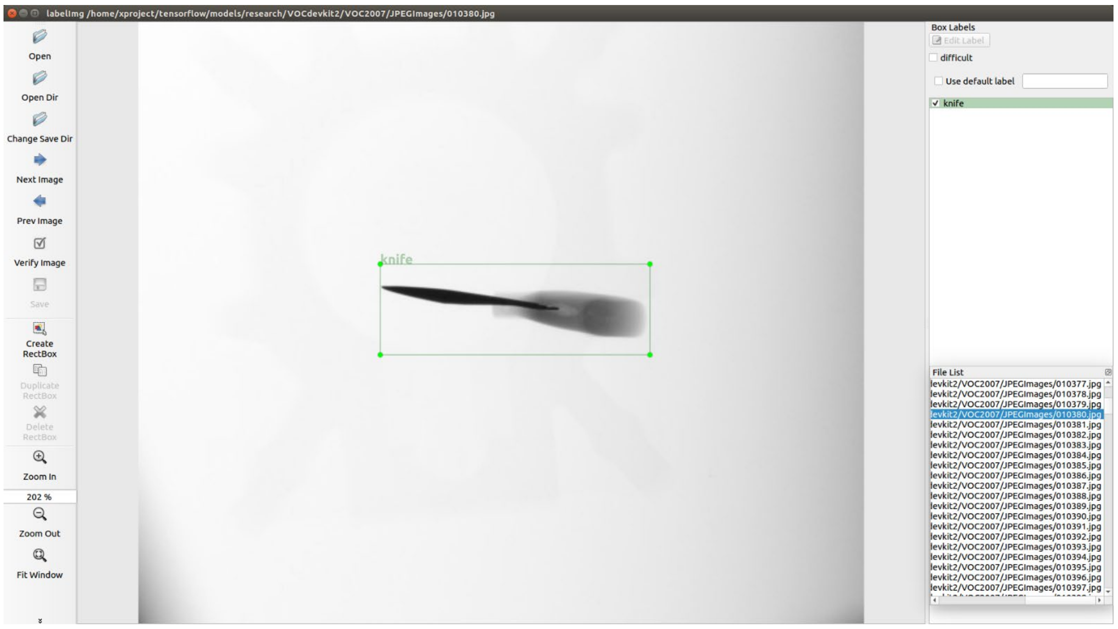
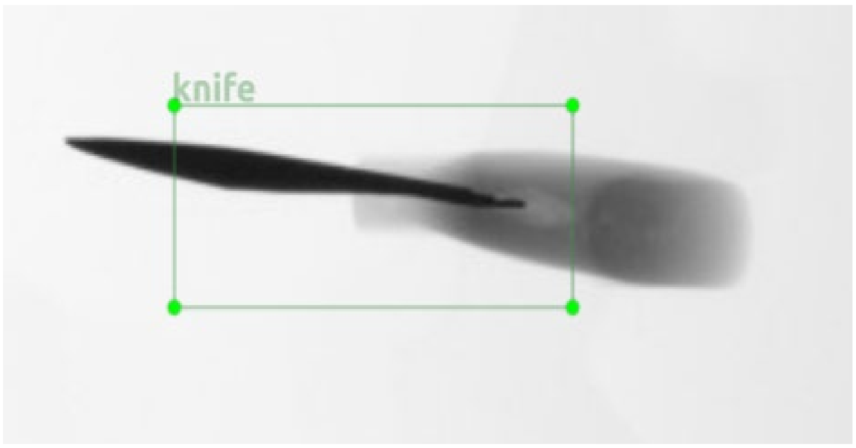
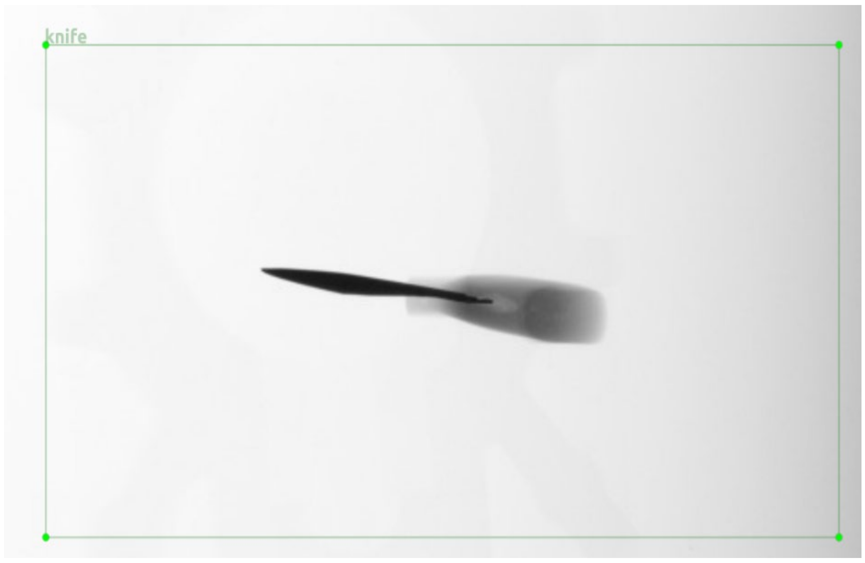

labelImg /home/xproject/tensorflow/models/research/VOCdevkit2/VOC2007/JPEGImages/010380.jpg
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
Create RectBox
Duplicate RectBox
Delete RectBox
Zoom In
202 %
Zoom Out
Fit Window
knife
Box Labels
Edit Label
difficult
Use default label
knife
File List
levkit2/VOC2007/JPEGImages/010377.jpg
levkit2/VOC2007/JPEGImages/010378.jpg
levkit2/VOC2007/JPEGImages/010379.jpg
levkit2/VOC2007/JPEGImages/010380.jpg
levkit2/VOC2007/JPEGImages/010381.jpg
levkit2/VOC2007/JPEGImages/010382.jpg
levkit2/VOC2007/JPEGImages/010383.jpg
levkit2/VOC2007/JPEGImages/010384.jpg
levkit2/VOC2007/JPEGImages/010385.jpg
levkit2/VOC2007/JPEGImages/010386.jpg
levkit2/VOC2007/JPEGImages/010387.jpg
levkit2/VOC2007/JPEGImages/010388.jpg
levkit2/VOC2007/JPEGImages/010389.jpg
levkit2/VOC2007/JPEGImages/010390.jpg
levkit2/VOC2007/JPEGImages/010391.jpg
levkit2/VOC2007/JPEGImages/010392.jpg
levkit2/VOC2007/JPEGImages/010393.jpg
levkit2/VOC2007/JPEGImages/010394.jpg
levkit2/VOC2007/JPEGImages/010395.jpg
levkit2/VOC2007/JPEGImages/010396.jpg
levkit2/VOC2007/JPEGImages/010397.jpg

Fig. 28: Software interface of labelImg

Fig. 28 shows how to label an X-ray image with only one knife. Note that the size of the rectangle should cover the object exactly, too large and too small is no good. **Fig. 29** shows two error labels. The rectangle in **Fig. 29** (a) is too small, which will cause the network cannot learn the entire features of the object. The rectangle in **Fig. 29** (b) is too large, which will cause the network to learn the extra features of the background other than the object. Both of these labeling methods will lead to erroneous learning of the network and reduce the detection accuracy.



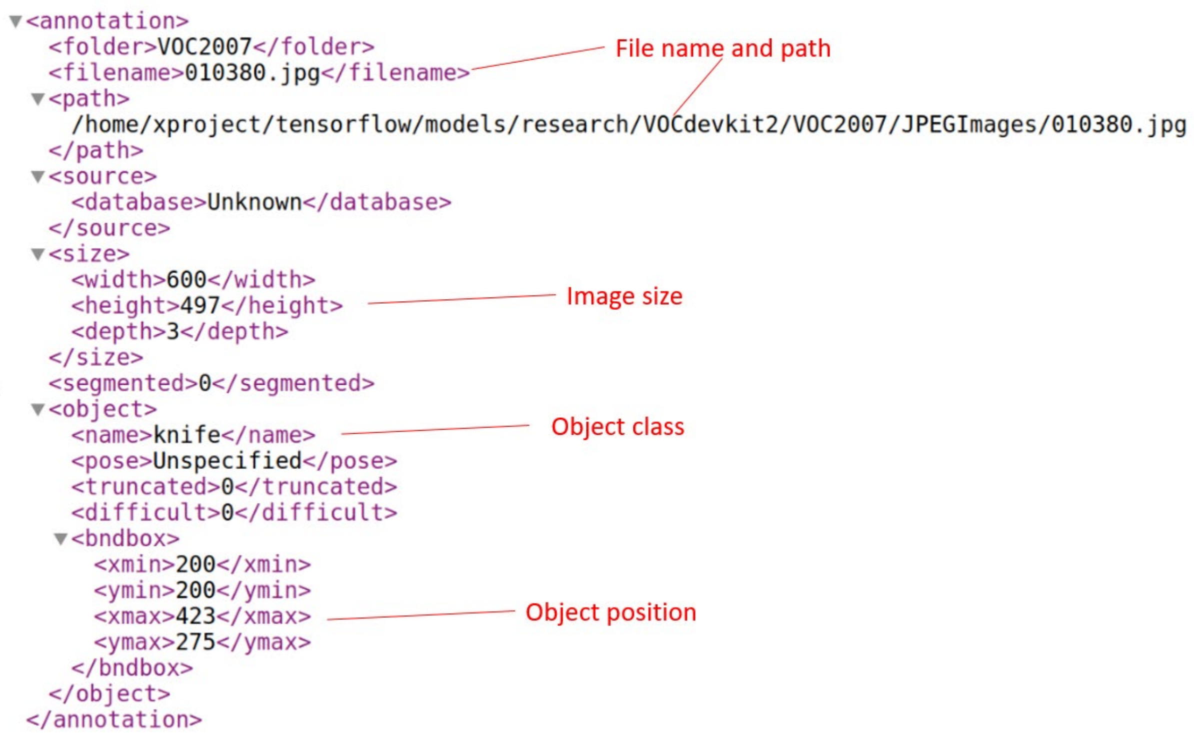
(a) Label is too small



(b) Label is too large

Fig. 29: Examples that labels are not suitable

Each labeled image will be mapped with an annotation file. The annotation is based on Pascal VOC2007 format standard, which contains all the information about the image such as image name, path, size, objects in the image and their position (see **Fig. 30**). This tree-like structure can be used for training with any object detection model. Yolo can convert these annotation files into the format required for Yolo training through the built-in conversion code.



```
▼<annotation>
   <folder>VOC2007</folder>
   <filename>010380.jpg</filename>
  ▼<path>
     /home/xproject/tensorflow/models/research/VOCdevkit2/VOC2007/JPEGImages/010380.jpg
   </path>
  ▼<source>
     <database>Unknown</database>
   </source>
  ▼<size>
     <width>600</width>
     <height>497</height>
     <depth>3</depth>
   </size>
   <segmented>0</segmented>
  ▼<object>
     <name>knife</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <difficult>0</difficult>
    ▼<bndbox>
       <xmin>200</xmin>
       <ymin>200</ymin>
       <xmax>423</xmax>
       <ymax>275</ymax>
     </bndbox>
   </object>
</annotation>
```

Fig. 30: Annotation files of labeled images

3.4 Raw X-ray image data collection

Our dataset contains two parts. The first part is raw X-ray dataset. Our raw X-ray dataset contains 1104 X-ray images, which are collected by us one by one. These images are used directly as part of the training dataset. We will also select some raw X-ray images as seeds for synthesizing images, and we will explain our synthesizing method later. The device we use to collect data is a medical machine (see **Fig. 31**). This device is an experimental device for developing portable X-ray device. By adjusting the illumination distance and illumination intensity of this device, we can simulate the functions of our portable X-ray device to achieve the same effect. We set the illumination distance to be 150cm and input voltage to be 70kV. The size of X-ray images we collected is 3000*2488 pixels resolution. Considering that this size is not suitable for directly training because of our limited hardware sources. We resized this size into 600*497 by factor 0.2. We mainly focus on three kinds of dangerous objects detection: scissor, knife, bottle. These three are the most common objects in the security check. We have prepared more than a dozen of these three kinds of dangerous objects with different shapes, sizes and some common safe things.

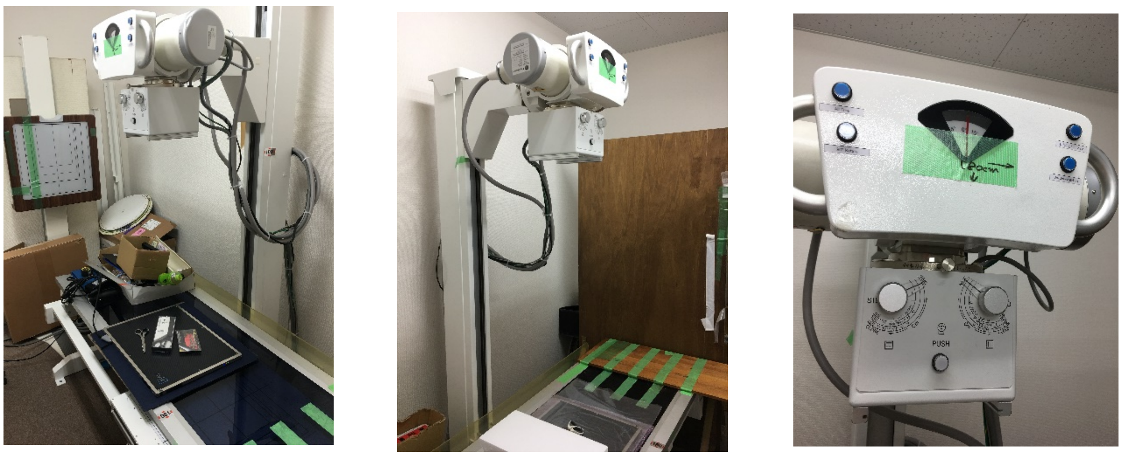
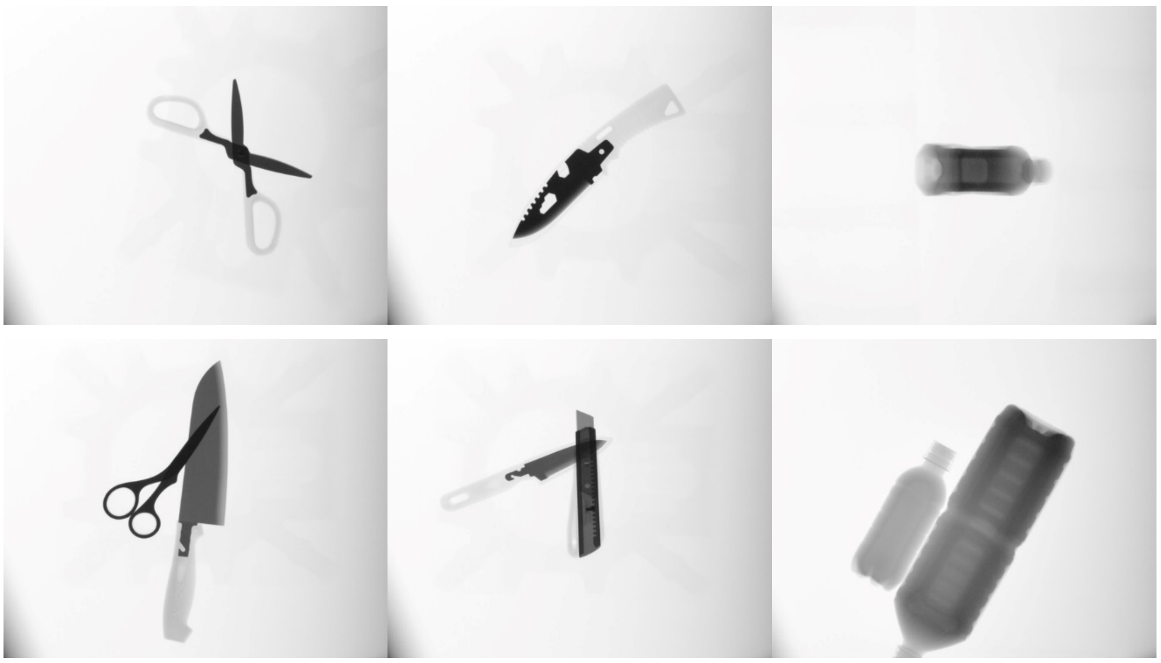


Fig. 31: Medical X-ray device

In the real security checks, the number of dangerous objects in the baggage is usually not much. In most cases, there are only one to three dangerous items, and the rest are safe items. This means that collecting images by stacking dangerous objects in large quantities is not realistic. Considering the extremely complicated way of placing objects in the baggage, we used some auxiliary materials to rotate the objects and randomly placed them at various positions We also considered the situation that objects overlap with each other. By combing dangerous objects and safety objects, the contents of the real baggage are restored to the utmost extent.

There are two kinds of raw X-ray images. One is an image that contains only one or two dangerous objects, which is called simple image (see **Fig. 32** (a)). This kind of image is easy to train, allowing the network to better learn the features of a single dangerous object. The other kind of image contains many objects and they overlap with each other, we call this is a complex image (see **Fig. 32** (b)). This kind of image is close to the real baggage content, allowing the network to learn the features in complex baggage. Our complex images include images that are extremely difficult to identify, which allows ours network to handle more complex detections. In these images, scissors are the most complex and varied. We collected images of scissors that were opened at different angle. At the same time, some types of knives are retractable. We have also taken images of knives with different telescopic ranges. Since the lighting environment of each data collection is different, the brightness of the images we collected is slightly different, but this does not affect our experiments.


(a) Examples of simple image

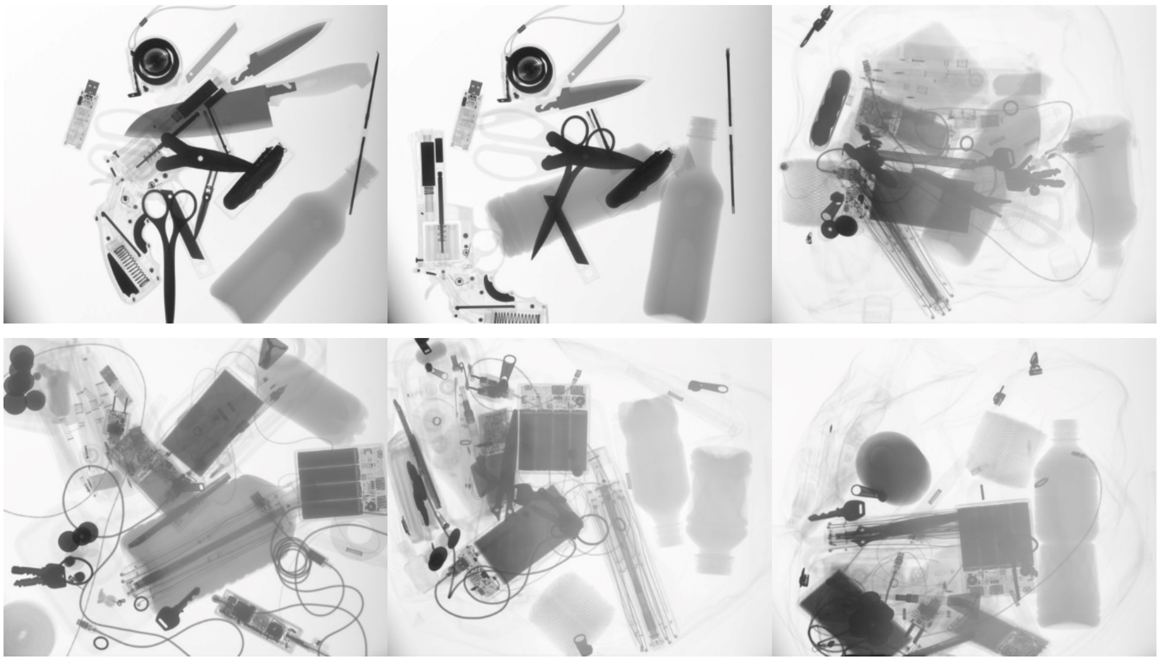

(b) Examples of complex image

Fig. 32: Examples of raw X-ray data

3.5 Synthesizing X-ray data

3.5.1 Beer–Lambert law

We use Tanaka's method to synthesize X-ray images [51]. His method is an improvement of the method proposed by Jaccard, N., Rogers et.al [24]. We introduce Jaccard's method first, his method was using for cargo security, which is very similar to our baggage security.

Jaccard's method suppose that all X-ray images obey the following Beer-Lambert law:

$$I_{xy} = I_0 \exp(-\int u_{xy}(z)\, dz), \tag{7}$$

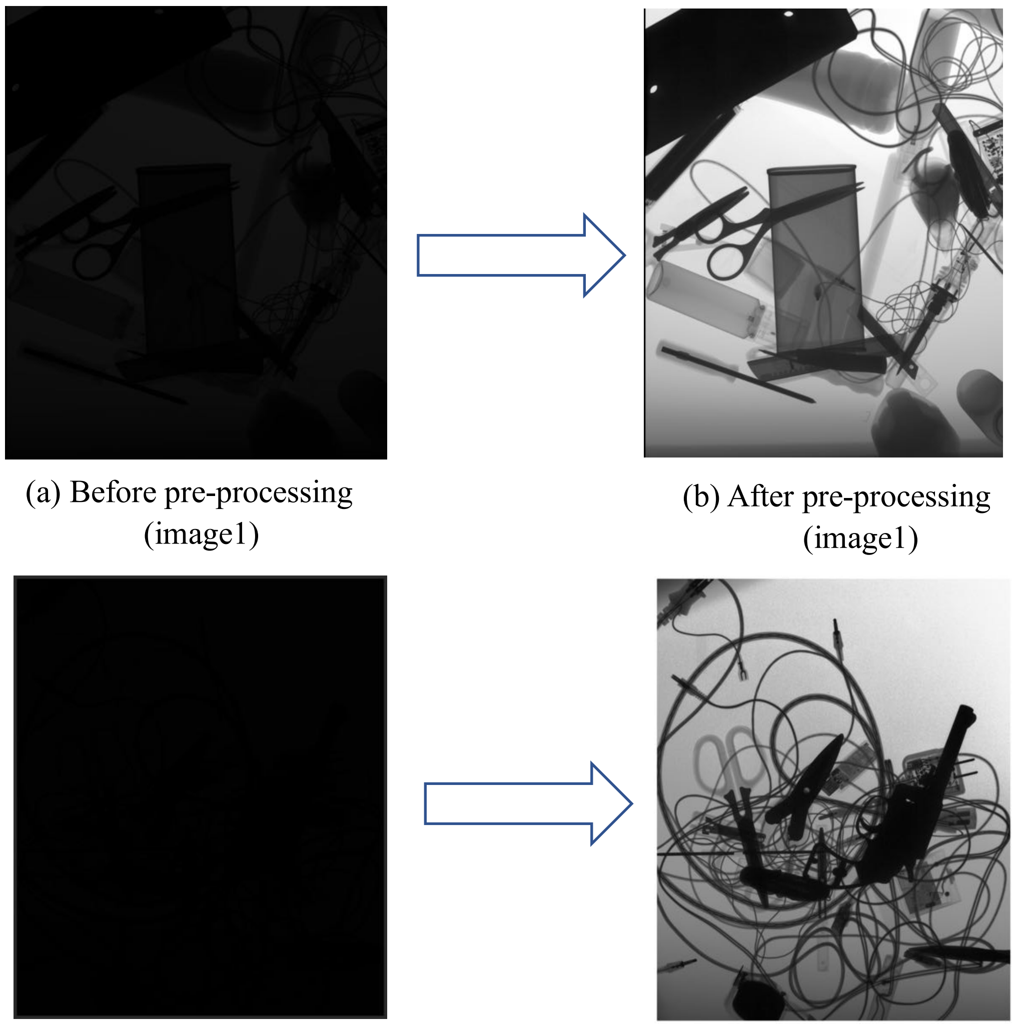
where I_{xy} is the illumination intensity at the position represented by coordinate (x, y), I_0 is the irradiation intensity of X-ray source beam, and $u_{xy}(z)$ is photon reduction coefficient in the Z-axis direction. If the objects to be illuminated can be treated as a whole, this formula indicates that the X-ray intensity of the objects satisfies an integral attenuation relationship with the X-ray intensity of the illumination source. In the case of cargo, the attenuation contributions are from two parts. One is from the object (O) and the other is from container (C). Then we have the following formula:

$$\begin{aligned} I_{xy} &= I_0 \exp(-\int_O u_{xy}(z)\, dz) \exp(-\int_C u_{xy}(z)\, dz) \\ &= I_0 O_{xy} C_{xy} \end{aligned} \tag{8}$$

Baggage security X-ray images also obey the Equation (8). We can obtain the I_{xy} by measuring the luminance value of the X-ray image. That is to say, we need to measure C_{xy} and I_0, then calculating the value of O_{xy}. If we obtain the value of O_{xy}, the corresponding X-ray image can be used as a background image. We also can extract the objects from other X-ray images. By taking using of both background and extracted objects, we can synthesize new X-ray images by multiplication.

3.5.2 Image pre-processing

To synthesize a large amount of X-ray images, we take some of raw X-ray images as seeds. These images cannot be directly used for synthesizing because their brightness is not inconsistent. Therefore, we pre-process these images to ensure they are consistent in brightness. The image pre-processing stage contains two steps. The first step is histogram equalization. As can be seen from our example of raw X-ray data, the appearance of the objects in the image is clear, but the background is rather vague. This makes the details of the image difficult to identify. The purpose of histogram equalization is to evenly distribute the brightness of objects and backgrounds so that the details of the image can be fully represented. Through histogram equalization, we can effectively eliminate the influence of environmental factors on the images. The second step is gaussian filtering. Gaussian filtering can remove impurities form the details of the image to give a cleaner image.



(a) Before pre-processing (image1)

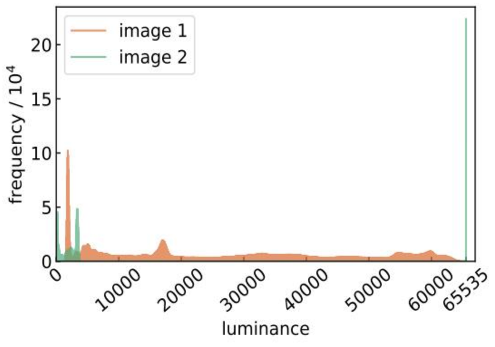
(b) After pre-processing (image1)

(c) Before pre-processing (image2)

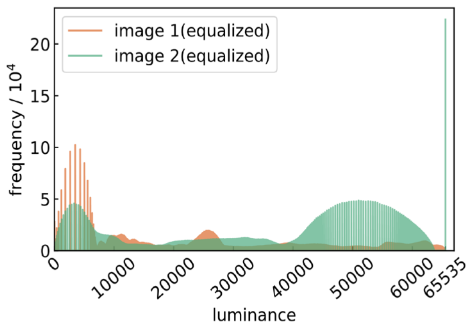
(d) After pre-processing (image2)

Fig. 33: Examples of image pre-processing[51]

Fig. 33[51] gives two examples of image pre-processing. **Fig. 33** (a) and **Fig. 33** (c) are the two raw X-ray images. In order to better explain the role of pre-processing, we used two example images, the real raw X-ray images are not so dark. Obviously, these two images are difficult to read because of the darkness and even human operators can hardly observe the contents inside. After pre-processing these two images, both of them become bright with uniformly distributed brightness and clearly visible. We show the histogram of the two example images in **Fig. 34**[51]. It can be found that the luminance distribution of the two images before histogram equalization is concentrated on the edge portion (see **Fig. 34** (a)), and it becomes smoother after histogram equalization (see **Fig. 34** (b)).



(a) Before histogram equalization

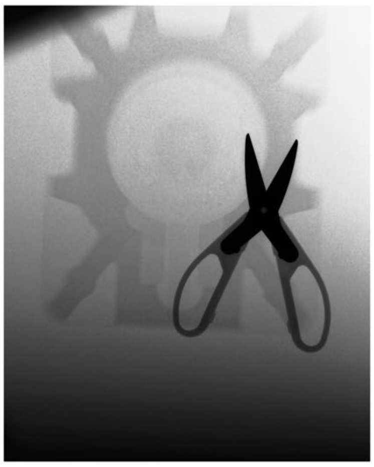


(b) After histogram equalization

Fig. 34: Histogram of the two example images[51]

3.5.3 Dangerous object extraction

In this stage, we extract dangerous objects from pre-processed images. We need to extract the contour of the object we want to extract. In order to extract a complete contour, we first binarize the image so that we can better observe the contour to be extracted. We use a threshold for binarizing image. For each pixel on the image, if the pixel value is higher than the threshold, this pixel value will be set to 0. Otherwise, it will be set to 255. By this way, we can binarize the image. **Fig. 35**[51] gives an example of binarizing image with the threshold value of 55. It can be found that at the threshold value of 55, the contour of the scissor after the binarization of this image is not complete. In this case, we need to adjust the threshold value and binarize the image repeatedly until we can see the complete contour.


(a) Preprocessed original image

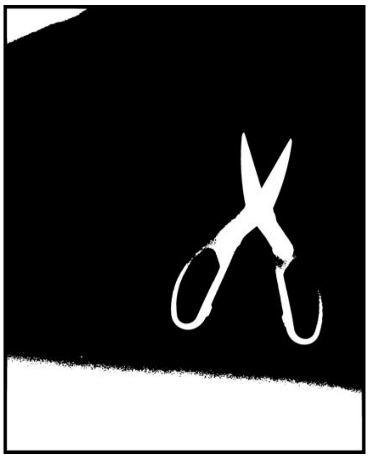

(b) Binarized image (threshold = 55)

Fig. 35: Example of binarizing the image[51]

The setting of the threshold value needs to be based on experience, and the threshold value of each image is also different. Whether the contour is successfully extracted or not is also judged by eyes. This is a shortcoming of this method, and we will improve it in the future. **Fig. 36**[51] shows different extracted contours with different threshold values. The red circle parts indicate the parts we failed to extract. We can find that in this scissor example, the threshold value of both 55 and 65 cannot extract the complete contour and the contour extracted with threshold value of 75 is perfect. After successfully extracting the contour, it is necessary to cut the scissor's contour along the circumscribed rectangle, and the set the luminance value of the background part of the rectangle that does not belong to the scissor to 1. This is because we only need the contour part of the scissor for synthesizing, and we need to eliminate the interference of the background part on the synthesis. Then, we take O_{xy} value as 1 and perform synthesizing process with this dangerous object contour and the prepared background images.

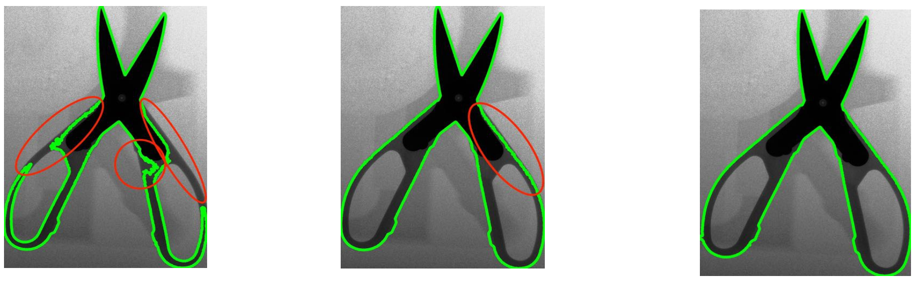

(a) Threshold = 55 (b) Threshold = 65 (c) Threshold = 75

Fig. 36: Sampling contours extracted with different thresholds[51]

3.5.4 Synthesizing data

Now that we have the background image and the contours, in order to generate robust dataset, we perform the follow operations on the contours:

- Randomly enlarging or reducing each contour by α times ($0.7 \leq \alpha \leq 1.3$).
- Randomly rotating each contour by $\beta°$ ($0 \leq \beta \leq 359$).
- Randomly selecting position of each contour on the background image.

Multiple objects can be synthesized on each background image, and each object can be synthesized with any number of background images. The position and size of the object are changed randomly. Through the random combination of the background images and the objects, we can synthesize a large number of X-ray images. **Fig. 37** gives some examples of synthesized images. We have marked synthesized dangerous objects using red rectangular boxes. These red rectangular boxes are not included in the real synthesized X-ray images. Here we resized the synthesized image into 512*614 and successfully synthesized 5017 images for experiments. At the time of synthesis, the algorithm recorded the positions of the dangerous objects and automatically labeled each image. This saves a lot of time for manual labeling. In the next chapter, we will conduct five kinds of comparison experiments and prove that our synthesized images improve the detection performance.

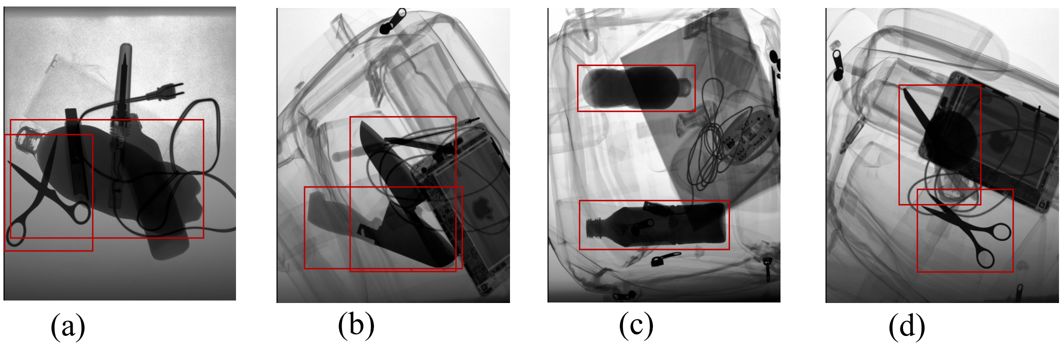


Fig. 37: Examples of synthesized X-ray images

3.6 Framework of our dangerous objects detection method

Now, let me explains how our X-ray security system works. First, it is necessary to prepare some labeled raw X-ray images. These images can be obtained from common X-ray devices, but we strongly recommend using our portable X-ray devices. At the same time, these X-ray images should be single-view X-ray images. These raw X-ray images will be used directly as part of the training dataset for training the CNN. Some of these images will also be taken as seeds for synthesizing X-ray images. The synthesizing process contains three modules. The first is preprocessing module, in this module, the system performs histogram equalization and gaussian filtering on the seeds. Then these preprocessed images will be extracted by dangerous object extraction module. In the last module, the system will synthesize X-ray images. As long as the seeds are sufficient, the system can synthesize any number of X-ray images and automatically generate annotations. These synthesized images are also taken as a part of training dataset. Finally, both raw X-ray images and synthesized images will be trained for an object detection CNN model. Training from scratch is not recommended, the common way for training CNN model is taking use of a pre-trained model for fine-tuning. There is no published pre-trained model based on X-ray image datasets. We use a pre-trained model based on RGB image datasets instead. We show our system framework in **Fig. 38**.

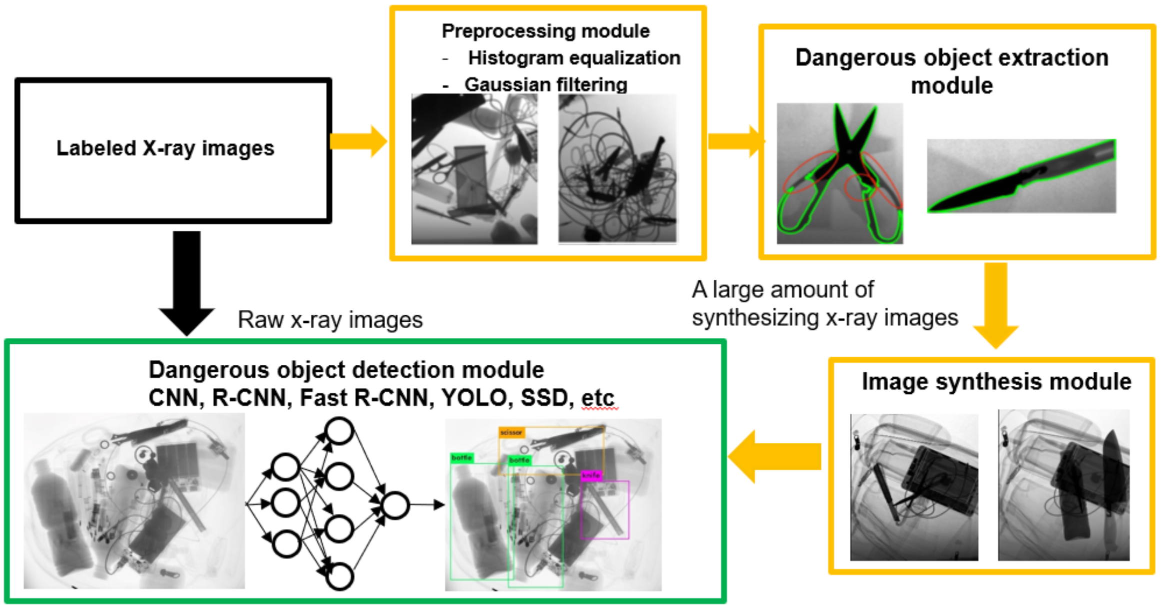

Labeled X-ray images
Preprocessing module
- Histogram equalization
- Gaussian filtering
Dangerous object extraction module
Raw x-ray images
A large amount of synthesizing x-ray images
Dangerous object detection module
CNN, R-CNN, Fast R-CNN, YOLO, SSD, etc
Image synthesis module

Fig. 38: Framework of our dangerous objects detection method

Chapter 4
Experiments and Results

4.1 Overview

In this chapter, we show detection performance of our X-ray security system. We show the design of our experiments and conduct five kinds of experiments to prove that our synthesized images contribute the detection performance. We also show the significance of t-test of our experiment results.

4.2 Design of five kinds of experiments

We have collected 1104 raw X-ray images with size of 600*497 and synthesized 5017 X-ray images with the size of 512*614. These images will be divided into two parts in a certain proportion: training dataset and testing dataset. Both of two datasets contains raw X-ray images and synthesized images. Our experiment results will be counted from three parts. The first part is detection performance based on raw X-ray images. The second part is detection performance based on synthesized X-ray images. The third part is total detection performance based on two kinds of X-ray images. This statistical method can better compare the detection results of raw X-ray images and synthesized X-ray images and reflect the role of our synthesized images. For fair comparison, we conduct the following five kinds of experiments:

- Training only raw X-ray images.
- Training only synthesized X-ray image.
- Training both raw X-ray images and synthesized X-ray images with the same proportion.
- Training raw X-ray images and a small amount of synthesized X-ray images.
- Training raw X-ray images and a large amount of synthesized X-ray images.

We run each kind of experiments 20 times to obtain stable results. In each running time, we randomly divide our datasets with constant proportion. We also use control variable method to make these experiments comparable. We set learning rate to be 0.001 and batch size to be 64 in all experiments. The amount of training and testing dataset of each kind of experiment is listed in **Table 2**. In the previous two experiments, we train 662 images with only raw X-ray images and only synthesized X-ray images respectively. These two kinds of experiments' results are the based line of dangerous object detection task. Exp.3 cut the training dataset into two parts with the same proportion and keep the total training images the same as previous two experiments. So that we can compare the two kinds of X-ray images from Exp.3. The amount of raw X-ray images for training in Exp.4 and Exp.5 is set to be the same as the total of training data in previous three kinds of experiments and the amount of synthesized X-ray images is different. This is to observe the effect of adding different amount of synthesized X-ray images on performance.

Table 2: Five kinds of experiments

Exp	**Training dataset**			**Testing dataset**		
	Raw images	**Synthesized images**	**Total**	**Raw images**	**Synthesized images**	**Total**
Exp.1	662	0	662	442	442	884
Exp.2	0	662	662	442	442	884
Exp.3	331	331	662	773	773	1546
Exp.4	662	331	993	442	442	884
Exp.5	662	3010	3672	442	2007	2449

4.3 Experiment results

We show the details of 20 times running results of five experiments. Our experiments are mainly focus on three kinds of dangerous objects detection: bottle, knife and scissor. The six rows of the table represent the running time, testing AP of bottle, testing AP of knife, testing AP of scissor, mAP and Recall. Here AP and mAP are reserved two decimal places, and Recall is expressed as an integer in percentage. All results are obtained with the threshold of 0.25.

Table 3: Total testing results in each running time of Exp.1

(training 662 raw X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	35.84	45.11	35.13	32.47	41.87	40.33	42.59	39.88	43.54	41.80	41.45	44.81	37.22	37.81	39.33	40.98	44.32	43.22	48.93	44.53
knife	46.74	52.03	34.85	42.04	45.90	50.84	53.32	48.12	51.56	50.44	49.91	48.31	49.63	52.47	47.21	52.20	52.32	49.01	51.03	54.53
scissor	75.83	76.55	56.99	67.36	74.99	73.60	77.63	74.60	72.91	74.22	73.39	78.33	74.87	77.04	75.43	75.51	75.43	70.79	76.92	77.04
mAP	52.80	57.89	42.33	47.29	54.26	54.93	57.85	54.20	56.01	55.49	54.91	57.15	53.91	55.77	53.99	56.23	57.36	54.34	58.96	58.70
Recall	57	60	43	49	56	59	58	57	58	58	58	57	60	56	57	58	60	57	59	58

Table 4: Raw images testing results in each running time of Exp.1

(training 662 raw X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	89.86	89.97	86.46	89.25	87.41	86.64	89.99	89.67	89.45	88.17	89.59	88.50	88.62	86.50	87.81	88.09	90.35	89.37	87.72	89.27
knife	75.55	83.04	73.57	74.76	77.46	80.22	78.25	77.05	77.79	79.48	77.14	82.90	83.74	77.58	78.66	82.45	77.18	82.42	77.29	79.01
scissor	86.11	86.54	75.82	81.85	84.64	85.45	87.10	85.17	84.76	86.22	84.60	89.49	85.19	85.13	86.51	86.11	83.65	86.03	86.10	87.61
mAP	83.84	86.52	78.62	81.96	83.17	84.10	85.12	83.96	84.00	84.63	83.78	86.96	85.85	83.07	84.33	85.55	83.73	85.94	83.70	85.30
Recall	86	85	75	79	82	84	85	85	84	85	85	87	87	84	86	86	85	85	84	85

Table 5: Synthesized images testing results in each running time of Exp.1

(training 662 raw X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	11.46	17.35	6.88	10.71	18.57	14.43	16.21	13.86	16.21	9.41	13.10	18.58	8.56	8.32	12.80	16.20	17.25	11.70	27.78	18.00
knife	17.53	20.95	11.31	13.87	12.74	19.61	23.19	15.42	19.85	21.76	19.64	18.01	20.06	22.40	18.70	21.75	24.65	17.22	19.20	24.65
scissor	53.01	59.62	18.67	24.58	52.99	48.34	59.45	55.90	48.57	54.11	50.56	48.79	51.36	58.53	52.60	53.35	56.03	42.05	60.52	59.41
mAP	27.33	32.64	12.29	16.39	28.10	27.46	32.95	28.39	28.21	28.43	27.77	28.46	26.66	29.75	28.03	30.43	32.64	23.66	35.83	34.02
Recall	30	35	12	19	30	34	32	30	33	31	31	29	34	28	30	32	35	30	35	31

Table 6: Total testing results in each running time of Exp.2

(training 662 synthesized X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	76.43	79.65	69.75	79.30	80.32	74.80	79.38	76.91	79.16	79.90	73.68	76.24	80.08	75.42	70.85	77.00	78.25	80.02	79.18	79.18
knife	63.44	66.81	62.05	68.19	68.27	66.37	63.96	65.87	67.23	63.73	70.12	64.77	65.6	65.58	64.33	67.91	66.78	69.78	67.44	63.36
scissor	66.64	65.37	58.46	61.63	68.73	67.24	67.09	64.72	68.14	62.17	65.72	66.36	63.45	68.04	60.45	67.62	67.43	68.83	69.80	59.73
mAP	68.84	70.61	63.42	69.71	72.44	69.47	70.14	69.16	71.51	68.60	69.84	69.12	69.71	69.68	65.21	70.85	70.82	72.88	72.14	67.42
Recall	66	67	62	69	70	67	69	66	69	65	67	64	69	65	65	67	68	71	71	64

Table 7: Raw images testing results in each running time of Exp.2

(training 662 synthesized X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	48.09	63.86	38.54	57.76	64.39	45.62	59.96	47.50	54.40	58.50	45.90	51.60	65.12	50.18	39.09	55.51	57.86	57.51	60.10	57.87
knife	43.03	52.60	42.64	53.20	55.26	48.14	48.73	48.17	53.29	44.93	53.25	45.35	50.06	42.58	43.85	48.96	51.73	54.24	57.71	53.61
scissor	55.77	54.11	45.75	51.50	60.29	56.22	56.64	54.01	58.35	52.92	56.36	55.00	55.24	57.02	49.61	59.24	57.44	57.12	63.81	49.85
mAP	48.97	56.85	42.31	54.15	59.98	49.99	55.11	49.89	55.35	52.12	51.83	50.65	56.81	49.93	44.18	54.57	55.68	56.29	60.54	53.78
Recall	45	51	39	51	54	47	52	46	52	46	48	44	52	44	42	50	51	54	55	48

Table 8: Synthesized images testing results in each running time of Exp.2

(training 662 synthesized X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	89.61	89.77	87.59	89.35	89.72	89.92	90.66	90.00	90.36	90.20	88.26	89.70	90.07	90.29	89.55	88.65	88.71	90.24	89.79	89.93
knife	78.64	81.08	77.85	84.31	83.30	84.58	78.39	79.74	80.86	78.78	85.81	80.35	78.35	84.24	82.43	78.95	81.94	82.36	77.99	71.49
scissor	89.85	87.05	89.45	86.66	87.34	89.16	88.96	88.69	88.79	80.24	89.97	88.62	81.13	90.12	85.65	81.15	88.26	90.41	89.90	79.22
mAP	86.03	85.97	84.96	86.77	86.78	87.89	86.00	86.14	86.67	83.08	88.01	86.22	83.18	88.22	85.88	82.92	86.30	87.67	85.89	80.21
Recall	86	84	84	87	85	87	87	86	85	84	86	84	85	86	87	84	85	87	86	80

Table 9: Total testing results in each running time of Exp.3
(training 331 raw X-ray images + 331 synthesized X-ray images, testing 773 raw X-ray images + 773 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	80.18	87.56	86.44	87.29	85.59	87.90	86.44	88.89	87.57	87.50	86.84	87.06	87.18	87.73	86.14	86.57	84.33	86.61	87.13	86.10
knife	70.83	76.61	73.01	73.52	74.64	72.87	73.69	76.12	70.18	74.91	75.02	75.36	73.91	75.55	72.31	74.24	75.84	74.74	75.16	72.71
scissor	79.39	80.00	82.87	79.26	79.78	79.04	80.11	79.32	82.18	79.97	84.45	85.04	79.74	83.93	79.86	84.02	79.80	79.94	82.65	78.71
mAP	76.80	81.39	80.78	80.02	80.00	79.94	80.08	81.44	79.98	80.80	82.11	82.49	80.28	82.41	79.44	81.61	79.99	80.43	81.65	79.18
Recall	77	82	79	79	80	81	80	80	79	81	81	82	80	82	79	82	80	81	82	76

Table 10: Raw images testing results in each running time of Exp.3
(training 331 raw X-ray images + 331 synthesized X-ray images, testing 773 raw X-ray images + 773 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	76.93	86.13	86.46	86.34	86.19	86.23	86.10	87.48	87.56	86.13	86.64	85.04	87.87	87.84	86.57	84.17	86.11	87.52	87.82	83.08
knife	68.42	76.05	71.59	73.15	74.10	72.01	73.84	73.72	69.79	73.52	76.34	76.37	75.40	76.43	71.88	74.48	75.44	74.07	74.04	71.20
scissor	78.88	79.93	79.95	79.02	83.04	78.69	79.98	78.96	81.27	79.59	84.19	84.93	79.69	80.55	79.52	84.52	78.81	78.79	79.73	77.23
mAP	74.75	80.71	79.33	79.51	81.11	78.98	79.97	80.05	79.54	79.74	82.39	82.12	80.99	81.61	79.32	81.06	80.12	80.12	80.53	77.17
Recall	76	81	77	77	81	80	80	78	78	81	80	81	80	81	79	82	80	81	82	73

Table 11: Synthesized images testing results in each running time of Exp.3
(training 331 raw X-ray images + 331 synthesized X-ray images, testing 773 raw X-ray images + 773 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	84.58	88.93	86.73	87.85	86.46	89.01	87.79	89.59	87.88	88.19	86.78	87.76	86.69	87.29	87.37	88.17	85.78	86.47	87.71	87.66
knife	71.89	77.18	75.28	74.22	75.55	73.28	73.43	77.89	70.93	76.07	74.54	74.69	72.44	75.25	73.90	74.28	76.33	75.62	75.66	74.40
scissor	82.85	84.80	86.59	80.31	79.46	82.16	83.72	80.17	85.55	85.80	85.29	85.74	80.40	85.70	84.69	82.81	85.31	87.97	83.45	80.87
mAP	79.78	83.64	82.87	80.79	80.49	81.48	81.65	82.55	81.45	83.35	82.21	82.73	79.84	82.75	81.99	81.75	82.47	83.35	82.27	80.98
Recall	79	84	80	81	80	82	80	83	80	82	81	82	79	82	80	81	81	81	83	79

Table 12: Total testing results in each running time of Exp.4

(training 662 raw X-ray images + 331 synthesized X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	87.55	87.04	87.96	88.46	88.81	87.68	88.53	88.16	88.97	87.64	89.26	87.75	87.71	88.72	86.90	88.34	89.43	88.97	87.16	87.89
knife	80.05	75.07	78.30	77.41	76.87	78.22	78.69	75.47	74.85	79.79	78.78	75.49	81.41	78.17	75.51	75.40	76.41	78.16	77.02	76.31
scissor	88.22	83.30	86.47	85.21	87.62	83.23	80.94	86.76	85.50	87.68	85.04	79.95	87.58	88.24	87.53	86.23	80.43	85.59	84.80	85.94
mAP	85.27	81.80	84.24	83.69	84.43	83.04	82.72	83.46	83.11	85.04	84.36	81.06	85.57	85.04	83.32	83.32	82.09	84.24	82.99	83.38
Recall	83	81	84	83	84	81	83	82	84	86	85	81	86	86	82	82	81	84	81	83

Table 13: Raw images testing results in each running time of Exp.4

(training 662 raw X-ray images + 331 synthesized X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	87.66	89.97	88.76	89.98	90.03	88.29	89.82	88.78	89.99	88.67	89.62	88.59	88.47	89.69	89.10	89.27	90.26	89.40	88.46	89.60
knife	82.55	76.85	81.74	78.44	82.80	79.19	77.58	78.07	77.58	81.47	79.33	79.51	83.43	83.42	78.44	80.90	77.53	78.61	78.05	82.31
scissor	88.14	83.54	87.79	86.52	87.56	79.92	80.76	86.33	83.67	88.07	86.18	79.42	87.94	88.11	88.80	88.02	80.90	86.80	85.02	88.96
mAP	86.12	83.45	86.10	84.98	86.80	82.47	82.72	84.39	83.75	86.07	85.05	82.51	86.61	87.07	85.45	86.06	82.90	84.94	83.85	86.96
Recall	84	84	87	85	86	83	83	83	85	88	86	83	87	89	87	86	83	84	81	87

Table 14: Synthesized images testing results in each running time of Exp.4

(training 662 raw X-ray images + 331 synthesized X-ray images, testing 442 raw X-ray images + 442 synthesized X-ray images, 10,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	87.90	84.13	86.25	86.77	87.54	87.65	87.18	88.00	88.30	87.74	88.60	88.35	87.39	88.01	85.11	87.86	87.94	88.63	86.76	87.13
knife	76.71	73.64	75.17	75.33	75.17	77.86	79.14	73.93	73.00	77.41	80.31	73.31	80.48	75.94	72.11	70.69	75.99	78.06	76.33	74.03
scissor	87.91	82.23	84.85	83.28	88.25	84.64	80.87	87.92	87.41	87.10	80.04	84.43	86.87	88.65	83.32	83.54	83.04	81.20	85.47	82.94
mAP	84.18	80.00	82.09	81.79	83.65	83.38	82.40	83.28	82.90	84.09	82.98	82.03	84.91	84.20	80.18	80.70	82.32	82.63	82.85	81.37
Recall	83	79	80	80	82	79	82	81	83	84	85	80	84	84	78	79	80	83	80	79

Table 15: Total testing results in each running time of Exp.5
(training 662 raw X-ray images + 3010 synthesized X-ray images, testing 442 raw X-ray images + 2007 synthesized X-ray images, 25,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	90.77	90.54	90.59	90.65	90.55	90.66	90.53	90.67	90.75	90.75	90.67	90.41	90.70	90.75	90.74	90.73	90.44	90.68	90.71	90.62
knife	89.41	89.40	88.59	87.10	88.45	89.12	88.63	89.75	88.60	89.26	88.82	89.35	89.90	88.83	88.56	89.32	88.40	89.26	88.42	89.56
scissor	90.38	90.47	90.00	89.90	90.30	90.06	90.26	90.49	90.31	90.39	90.46	90.60	90.43	90.04	90.10	90.47	90.28	90.51	90.47	90.47
mAP	90.19	90.14	89.73	89.22	89.77	89.95	89.81	90.30	89.89	90.13	89.98	90.12	90.35	89.88	89.80	90.17	89.71	90.15	89.87	90.22
Recall	92	92	91	91	91	92	91	93	92	92	92	93	92	93	91	93	91	93	92	93

Table 16: Raw images testing results in each running time of Exp.5
(training 662 raw X-ray images + 3010 synthesized X-ray images, testing 442 raw X-ray images + 2007 synthesized X-ray images, 25,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	90.24	89.19	89.58	88.92	88.60	89.34	88.71	90.34	90.15	89.59	89.29	88.55	89.77	89.32	89.27	89.68	85.52	89.99	90.31	89.45
knife	86.04	83.58	79.74	78.01	82.39	81.84	77.89	80.87	84.57	78.72	81.15	80.72	83.13	79.63	82.29	82.91	82.68	85.16	84.06	84.05
scissor	88.85	89.95	85.85	86.29	86.55	87.56	85.71	88.04	89.60	88.16	89.13	89.40	86.62	88.19	86.11	89.12	88.78	88.23	88.93	88.38
mAP	88.38	87.57	85.06	84.41	85.85	86.25	84.10	86.41	88.11	85.49	86.53	86.22	86.51	85.71	85.89	87.24	85.66	87.79	87.77	87.29
Recall	89	88	88	87	86	88	85	88	90	88	89	88	85	88	86	88	86	89	89	89

Table 17: Synthesized images testing results in each running time of Exp.5
(training 662 raw X-ray images + 3010 synthesized X-ray images, testing 442 raw X-ray images + 2007 synthesized X-ray images, 25,000 iterations)

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
bottle	90.81	90.63	90.72	90.81	90.79	90.79	90.74	90.71	90.82	90.82	90.79	90.58	90.74	90.85	90.83	90.80	90.62	90.74	90.75	90.71
knife	89.71	90.05	89.30	88.60	89.31	89.81	89.64	90.37	89.37	89.96	89.59	89.92	90.28	89.66	89.21	90.17	89.32	89.62	89.12	89.98
scissor	90.67	90.56	90.72	90.25	90.74	90.44	90.79	90.75	90.60	90.80	90.75	90.79	90.78	90.71	90.60	90.83	90.51	90.78	90.69	90.72
mAP	90.40	90.41	90.25	89.89	90.28	90.35	90.39	90.61	90.26	90.52	90.38	90.43	90.60	90.40	90.21	90.60	90.15	90.38	90.19	90.47
Recall	93	93	92	92	92	93	92	95	92	94	93	94	94	93	92	93	92	93	92	94

We show all running results of Exp.1 in **Table 3**, **Table 4** and **Table 5**. Similarly, **Table 6**, **Table 7** and **Table 8** give all running results of Exp.2. **Table 9**, **Table 10** and **Table 11** give all running results of Exp.3. **Table 12**, **Table 13** and **Table 14** give all running results of Exp.4. **Table 15**, **Table 16** and **Table 17** give all running results of Exp.5. We compare these results using the mean and standard deviation values of 20 times runs. We show all MEAN ± STDEV values of each experiment in **Table 18**, **Table 19**, **Table 20**, **Table 21** and **Table 22** respectively.

Table 18: Mean ± STDEV values of the first experiment (Exp. 1) running 20 times.

YOLOv2 results	Recall (%)	mAP	Bottle (AP)	Knife (AP)	Scissor (AP)
Total test (884)	54.05 ± 3.99	54.72 ± 3.88	41.06 ± 3.95	49.12 ± 4.45	73.97 ± 4.72
Raw data test (442)	84.20 ± 2.78	84.21 ± 1.81	88.63 ± 1.23	78.78 ± 2.88	85.20 ± 2.70
Synthesized data test (442)	30.05 ± 5.51	27.97 ± 5.51	14.37 ± 4.81	19.13 ± 3.69	50.42 ± 10.94

Table 19: Mean ± STDEV values of the second experiment (Exp. 2) running 20 times.

YOLOv2 results	Recall (%)	mAP	Bottle (AP)	Knife (AP)	Scissor (AP)
Total test (884)	67.05 ± 2.46	69.58 ± 2.27	77.28 ± 3.08	66.08 ± 2.21	65.38 ± 3.31
Raw data test (442)	48.55 ± 4.36	52.95 ± 4.65	53.97 ± 7.86	49.57 ± 4.64	55.31 ± 4.06
Synthesized data test (442)	85.25 ± 1.68	85.74 ± 2.00	89.62 ± 0.77	80.57 ± 3.28	87.03 ± 3.62

Table 20: Mean ± STDEV values of the third experiment (Exp. 3) running 20 times.

YOLOv2 results	Recall (%)	mAP	Bottle (AP)	Knife (AP)	Scissor (AP)
Total test (1546)	80.15 ± 1.66	80.54 ± 1.31	86.55 ± 1.78	74.06 ± 1.70	81.00 ± 2.06
Raw data test (773)	79.40 ± 2.28	79.96 ± 1.70	85.91 ± 2.44	73.59 ± 2.23	80.36 ± 2.14
Synthesized data test (773)	81.00 ± 1.41	81.92 ± 1.12	87.43 ± 1.16	74.64 ± 1.72	83.68 ± 2.46

Table 21: Mean ± STDEV values of the fourth experiment (Exp. 4) running 20 times.

YOLOv2 results	Recall (%)	mAP	Bottle (AP)	Knife (AP)	Scissor (AP)
Total test (884)	83.10 ± 1.74	83.61 ± 1.19	88.15 ± 0.74	77.37 ± 1.84	85.31 ± 2.55
Raw data test (442)	85.05 ± 2.09	84.91 ± 1.57	89.22 ± 0.72	79.89 ± 2.20	85.62 ± 3.14
Synthesized data test (442)	81.25 ± 2.12	82.60 ± 1.34	87.36 ± 1.14	75.73 ± 2.64	84.70 ± 2.65

Table 22: Mean ± STDEV values of the fifth experiment (Exp. 5) running 20 times.

YOLOv2 results	Recall (%)	mAP	Bottle (AP)	Knife (AP)	Scissor (AP)
Total test (2449)	92.00 ± 0.79	89.97 ± 0.26	90.65 ± 0.10	88.94 ± 0.63	90.32 ± 0.20
Raw data test (442)	87.70 ± 0.01	86.41 ± 1.19	89.29 ± 1.04	81.97 ± 2.35	87.97 ± 1.33
Synthesized data test (2007)	92.90 ± 0.91	90.36 ± 0.17	90.75 ± 0.07	89.65 ± 0.44	90.67 ± 0.14

Table 23: Independent samples test (95% interval confidence) of raw images test results.

Comparison of Raw data test results			Levene's Test for Equality of Variances		t-test for Equality of Means
			F-value	Sig	Sig (p-value)
Exp. 1 vs Exp. 2	mAP	Equal variances assumed	12.915	0.001	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	7.653	0.009	0.000
		Equal variances not assumed			0.000
Exp. 1 vs Exp. 3	mAP	Equal variances assumed	0.066	0.798	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	0.005	0.944	0.000
		Equal variances not assumed			0.000
Exp. 1 vs Exp. 4	mAP	Equal variances assumed	0.074	0.787	0.0975
		Equal variances not assumed			0.0975
	Recall	Equal variances assumed	0.001	0.978	0.141
		Equal variances not assumed			0.141
Exp. 1 vs Exp. 5	mAP	Equal variances assumed	0.761	0.388	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	1.464	0.234	0.000
		Equal variances not assumed			0.000

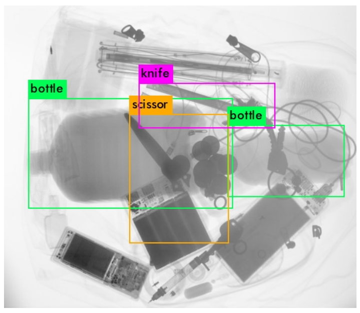
To prove that our experiment results are statistically significant, we show the significance of t-test based on our experiment results. In order to confirm whether our synthesized data can improve the detection performance, we perform significance of t-test using raw images test results of Exp.1 and raw images test results of other experiments (see **Table 23**). We also performed a significance of t-test on the raw images test results and synthesized images test results in each experiment to analysis the difference between raw images and synthesized images (see **Table 24**).

Table 24: Independent samples test (95% interval confidence) of raw images test results and synthesized images test results for each experiment.

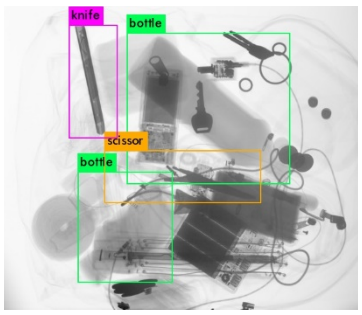
Comparison of raw data test results and synthesized data test results for each experiment			Levene's Test for Equality of Variances		t-test for Equality of Means
			F-value	Sig	Sig (p-value)
Exp. 1	mAP	Equal variances assumed	4.862	0.034	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	1.900	0.176	0.000
		Equal variances not assumed			0.000
Exp. 2	mAP	Equal variances assumed	10.661	0.002	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	18.708	0.000	0.000
		Equal variances not assumed			0.000
Exp. 3	mAP	Equal variances assumed	0.520	0.475	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	3.569	0.067	0.0055
		Equal variances not assumed			0.006
Exp. 4	mAP	Equal variances assumed	1.235	0.273	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	0.149	0.702	0.000
		Equal variances not assumed			0.000
Exp. 5	mAP	Equal variances assumed	27.374	0.000	0.000
		Equal variances not assumed			0.000
	Recall	Equal variances assumed	3.279	0.078	0.000
		Equal variances not assumed			0.000

4.4 Detection examples of our security system

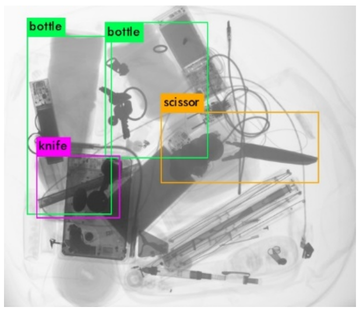
We select a trained Yolov2 model which has the best performance in Exp.5 to show some detection examples for convincing. Our testing dataset contains both simple images and complex images. Considering that simple images are easy to be detected, we select some of complex images as examples to show. **Fig. 39** gives six detection examples and we show all dangerous objects detection score from left to right under each example images. Because raw images testing results are the most valuable reference, all example images we selected are raw X-ray images. We can find that the objects in some example images are difficult to distinguish even for experienced human operators, but our trained model can accurately detect these dangerous objects.



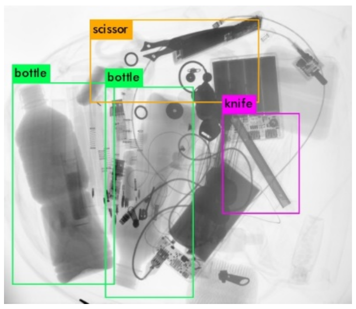
(a) bottle1: 88%, scissor: 68%, knife: 87%, bottle2: 84%.



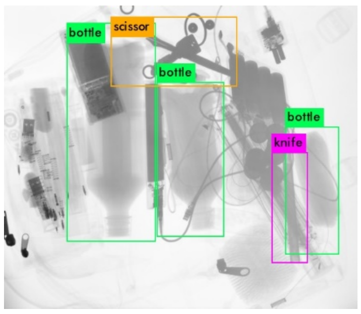
(b) knife: 77%, bottle1: 84%, scissor: 82%, bottle2: 91%.



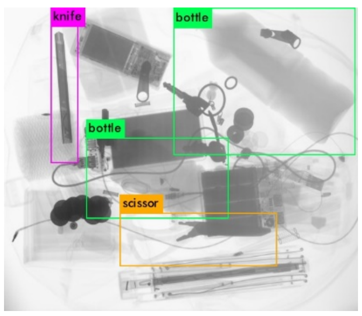
(c) bottle1: 80%, knife: 71%, bottle2: 90%, scissor: 85%.



(d) bottle1: 89%, scissor: 90%, bottle2: 87%, knife: 79%



(e) bottle1: 89%, scissor: 86%, bottle2: 82%, knife: 55%, bottle3: 54%



(f) knife: 77%, bottle: 85%, scissor: 86%, bottle2: 94%

Fig. 39: Detection examples of the best performance model in all experiments

Chapter 5 Discussions

5.1 Discussions

In the real security checks, the raw images detection performance is the most important. This is because synthesized images are virtual data after all, and its test results can only be used as a reference. Therefore, in our experimental results, we should pay attention to whether the detection result of raw images has been improved. In Exp.1, we found that when we only train raw X-ray images, the raw images testing results are quite well (mAP: 84.21, Recall: 84.20). At the same time, the AP value of bottle has reached 88.63, which is the highest one in these three kinds of dangerous objects. The AP value of knife is 78.78, which is the lowest one in these three kinds dangerous objects. This is because bottles are relatively large and easy to detect. Yolov2 is not good at detecting small objects, resulting in the detection accurate of knife is lower. We can see that in Exp.1, the synthesized images testing results (mAP: 27.97, Recall: 30.05) are in sharp contrast with the raw images testing results. This is because we only trained raw X-ray images in Exp.1 and raw X-ray images are easier to be detected than synthesized X-ray images. The results in Exp.2 verified this phenomenon. When we only train synthesized X-ray images with the same amount of Exp.1, we found that the synthesized images testing results in Exp.2 (mAP: 85.74, Recall: 85.25) were as good as raw images testing results (mAP: 84.21, Recall: 84.20) in Exp.1, and might even be better. This is because when we only train synthesized images, the trained model has better adaptability the synthesized images. There is an interesting phenomenon here, the testing results of Exp.2 did not show symmetry with Exp.1. In other words, the raw images testing results in Exp.2 (mAP: 52.95, Recall: 48.55) were not as low as expected, which are both close to 50 in mAP testing and Recall testing. At the same time, the total testing results in Exp.2 (mAP: 69.58, Recall: 67.05) are also higher than the total testing results in Exp.1 (mAP: 54.72, Recall: 54.05). This initially indicates the performance improvement of our synthesized X-ray images dataset. In Exp.3, we got the raw images results of (mAP: 79.96, Recall: 79.40) and the synthesized images results of (mAP: 81.92, Recall: 81.00). We found that when we train both raw X-ray images and synthesized X-ray images with the same amount, both mAP and Recall of synthesized images testing results are a little higher than raw images testing results. The significance of t-test (p-value of mAP: 0.000, p-value of Recall: 0.006) in **Table 24** indicates that this is not due to random factors. This is because the brightness of synthesized images is darker than raw images, which means that the features of synthesized images extracted by CNN are hard to disappear in deep convolutional layers, allowing the network learns these features better. Since we trained both raw X-ray images datasets and synthesized images datasets, the trained model showed good performance on two kinds of testing datasets. We note that the synthesized images testing results in Exp.3 (mAP: 81.92, Recall: 81.00) are worse than the synthesized images testing results in Exp.2 (mAP: 85.74, Recall: 85.25) and the raw images testing results in Exp.3 (mAP: 79.96, Recall: 79.40) are also worse than the raw images testing results in Exp.1 (mAP: 84.21, Recall: 84.20). This is because the total number of training images in the three experiments is equal, and the number of raw images and synthesized images in Exp.3 is smaller than that in Exp.1 and Exp.2 respectively, resulting in insufficient training of both two kinds of images. As we mentioned before, the raw images testing result is the most important and our purpose is to improve the raw images detection performance. In Exp.4, we restored the amount of raw X-ray images for training and added a small amount of synthesized X-ray images to see if these synthesized images can affect results. We got the raw images testing results in Exp 4 of (mAP: 84.91, Recall: 85.05) and both of mAP and Recall of the results are a little higher than raw images testing results in Exp.1 (mAP: 84.21, Recall: 84.20). However, the significance of t-test shows that (p-value of mAP: 0.0975 p-value of Recall: 0.141) shows their difference is not significant. This indicates that a small number of synthesized images did not make a significant contribution to our raw images testing results. The synthesized images testing results in Exp.4 (mAP: 82.60, Recall: 81.25) are also close to the synthesized images testing results in Exp.3 (mAP: 81.92, Recall: 81.00). This because we used the same amounts of synthesized images for training in both experiments, which also indicates that our experiment results are stable. When we trained a large number of synthesized images in Exp.5, we got raw images testing results of (mAP: 86.41, Recall: 87.70) and synthesized images testing results of (mAP: 90.36, Recall: 92.90).

We found that the raw images testing results in Exp.5 (mAP: 86.41, Recall: 87.70) is also higher than raw images testing results in Exp.1 (mAP: 84.21, Recall: 84.20). This time, the significance of t-test (p-value of mAP: 0.000 p-value of Recall: 0.000) shows that this testing results are significant. This means that although a small number of synthesized images do not contribute to the raw images testing results, a large number of synthesized images work on the accuracy. the significance of t-test based on Exp.5 (see **Table 24**) also tells us the synthesized images testing results (mAP: 90.36, Recall: 92.90) which both mAP and Recall are the highest values in all experiments are significant. This means that the raw images testing results still have space for improvement.

5.2 Limitations

There several limitations in our system that need to be noted.

- The brightness of our synthesized X-ray images is different from raw X-ray images. The synthesized X-ray images are darker, which are easy to be detected. The reason why synthesized X-ray images are darker is that the histogram equalization process average the brightness of objects and the background of the image. Since the objects are darker than the background, the whole image will become darker after histogram equalization. This difference in brightness causes our synthesized images not to be completely the same style as raw X-ray images, limiting the features learning of raw X-ray images by the network.
- Although our trained model can detect these dangerous objects in real time, the detection accuracy is still cannot meet the needs of practical applications. In other word, our model is not yet fully automated, and manual assistance is still required in practical applications. In the best case, we got the detection mAP of 86.41 and Recall of 87.70. In a real practical security system, the accuracy requirement is 95% or even higher. Our model is still some distance away form this precision and needs to be improved.

- The dangerous our model can detect is limited. Our model can only detect three kinds of dangerous objects. In real life, the kinds of dangerous objects are extremely complicated. In addition to scissors, knives and bottles, there are dangerous items such as guns, drugs, and explosives. Due to the limited dataset, our model is not able to detect these mentioned dangerous items. We need to augment the dataset and reinforce the model so that the model can detect more dangerous items.
- The pre-trained model we used for fine-tuning is based on RGB color image datasets. Because our dataset are all single-view X-ray images, fine-tuning a model pretrained with RGB color image datasets is not a good idea. There is still no open pre-trained model based on X-ray datasets.

5.3 Future work

- We will fix the image synthesizing algorithm to synthesize X-ray images which are the same style as raw X-ray images. At the same time, we will make the features of dangerous objects in the synthesized images clearer and without distortion.
- We will try other object detection model to conduct our experiments. We will train Yolov3 using our X-ray dataset, we think that Yolov3 can show better performance than Yolov2. On the other hand, using Faster R-CNN is also one of our plans. We believe that the X-ray dataset can get higher detection accuracy on Faster R-CNN.
- We will collect more raw X-ray images which contain other dangerous objects to expand our X-ray dataset. We will study how to detect guns, drugs and explosives to reinforce our model.
- We will try to find other open X-ray datasets for pre-training the model and start fine-tuning from the pre-trained model which is based on other X-ray datasets.

Chapter 6
Conclusion

In this thesis, we focus on the topic of CNN-based X-ray dangerous objects detection. We have discussed the shortcomings of commonly used multi-view security systems. In order to enhance the detection efficiency, we proposed a portable X-ray device for collecting single-view X-ray images. We have elaborated on the improvement process of the object detection models and finally selected Yolov2 to conduct our experiments. We have collected 1104 raw X-ray images for experiments. Considering that raw X-ray dataset is limited, we used Tanaka's method for synthesizing X-ray images. In the synthesizing process, the algorithm can generate annotations automatically, greatly reduced the workload of labeling the images. We synthesized 5017 X-ray images for expanding our training dataset.

We applied both raw X-ray images and synthesized X-ray images for training the model. In order to prove that our synthesized images contribute to raw images detection performance, we design five kinds of experiments. To obtain stable results, we run each kind of experiments 20 times. Our experiments mainly focus on three kinds of dangerous objects detection: scissor, knife, bottle. Finally, we achieved raw images testing mAP of 86.41 and Recall of 87.70. We also showed the significance of t-test on our results. The t-test showed that training a small amount of synthesized images cannot improve the raw images detection accuracy, but training a large amount of raw images can improve the detection performance.

Acknowledgements

We thank Yokoshima Shin and Nikaido Yoji in T & S Co., Ltd., for providing us with portable X-ray device information and data collection environments.

References

[1] Hubel, David H., and Torsten N. Wiesel. "Receptive fields and functional architecture of monkey striate cortex." The Journal of physiology 195.1 (1968): 215-243.

[2] Wang, Haohan, Bhiksha Raj, and Eric P. Xing. "On the Origin of Deep Learning." arXiv preprint arXiv:1702.07800 (2017).

[3] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. No. ICS-8506. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[4] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th international conference on machine learning (ICML-10). 2010.

[5] Caruana, Rich, Steve Lawrence, and C. Lee Giles. "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping." Advances in neural information processing systems. 2001.

[6] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

[7] Xu, Bing, et al. "Empirical evaluation of rectified activations in convolutional network." arXiv preprint arXiv:1505.00853 (2015).

[8] https://en.wikipedia.org/wiki/Backpropagation

[9] https://github.com/FrankCAN/Tensorflow-Tutorial-master-morvan

[10] Murphy, Kevin P. "Machine Learning: A Probabilistic Perspective. Adaptive Computation and Machine Learning." (2012).

[11] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

[12] Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. Proc. 8th Annual Conf. Cognitive Science Society.

[13] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural Networks: The Official Journal of the International Neural Network Society, 12(1), 145–151.

[14] Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543– 547.

[15] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121–2159.

[16] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[17] H. Robinds and S. Monro, "A stochastic approximation method," Annals of Mathematical Statistics, vol. 22, pp. 400–407, 1951.

[18] Singh, S., Singh, M.: Explosives detection systems (EDS) for aviation security. Signal Processing. 83, 31-55 (2003).

[19] Chen, Z., Zheng, Y., Abidi, B. R., Page, D. L., Abidi, M. A.: A combinational approach to the fusion, de-noising and enhancement of dual-energy x-ray luggage images. In Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference, IEEE, pp. 2-2 (2005).

[20] He, X. P., Han, P., Lu, X. G., Wu, R. B.: A new enhancement technique of x-ray carry-on luggage images based on dwt and fuzzy theory. In Computer Science and Information Technology, 2008. ICCSIT'08. International Conference, IEEE, pp. 855-858 (2008).

[21] Baştan, Muhammet, Mohammad Reza Yousefi, and Thomas M. Breuel. "Visual words on baggage X-ray images." Computer analysis of images and patterns. Springer, Berlin, Heidelberg, 2011.

[22] Turcsany, D, Mouton, A., Breckon, T. P.: Improving feature-based object recognition for X-ray baggage security screening using primed visualwords. In Industrial Technology (ICIT), 2013 IEEE International Conference, IEEE, pp.1140-1145 (2013).

[23] Mery, Domingo, et al. "Automated X-ray object recognition using an efficient search algorithm in multiple views." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2013.

[24] Jaccard, N., Rogers, T. W., Griffin, L. D.: Automated detection of cars in transmission Xray images of freight containers. In 2014 International Conference on Advanced Video and Signal Based Surveillance (AVSS), IEEE, pp. 387-392 (2014).

[25] Rogers, T. W., Jaccard, N., Protonotarios, E. D., Ollier, J., Morton, E. J., Griffin, L. D.: Threat image projection (tip) into x-ray images of cargo containers for training humans and machines. In 2016 IEEE International Carnahan Conference on Security Technology (ICCST), IEEE, pp. 1-7 (2016).

[26] Baştan, M.: Multi-view object detection in dual-energy X-ray images. Machine Vision and Applications. 26, pp. 1045-1060 (2015).

[27] Kundegorski, M. E., Akçay, S., Devereux, M., Mouton, A., & Breckon, T. P.: On using feature descriptors as visual words for object detection within x-ray baggage security screening (2016).

[28] Akçay, S., Kundegorski, M. E., Devereux, M., Breckon, T. P.: Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery, IEEE (2016).

[29] Riffo, V., Mery, D.: Automated Detection of Threat Objects Using Adapted Implicit Shape Model. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 46, 472-482 (2016).

[30] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[31] Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.

[32] Lienhart, Rainer, and Jochen Maydt. "An extended set of haar-like features for rapid object detection." Image Processing. 2002. Proceedings. 2002 International Conference on. Vol. 1. IEEE, 2002.

[33] Valiant, Leslie G. "A theory of the learnable." Communications of the ACM 27.11 (1984): 1134-1142.

[34] Freund, Yoav, and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting." Journal of computer and system sciences 55.1 (1997): 119-139.

[35] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[36] Felzenszwalb, Pedro F., et al. "Object detection with discriminatively trained part-based models." IEEE transactions on pattern analysis and machine intelligence 32.9 (2010): 1627-1645.

[37] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.

[38] Van de Sande, Koen EA, et al. "Segmentation as selective search for object recognition." Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011.

[39] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." european conference on computer vision. Springer, Cham, 2014.

[40] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[41] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[42] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[43] Neubeck, Alexander, and Luc Van Gool. "Efficient non-maximum suppression." Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. Vol. 3. IEEE, 2006.

[44] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[45] Lin, Tsung-Yi, et al. "Feature Pyramid Networks for Object Detection." CVPR. Vol. 1. No. 2. 2017.

[46] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." arXiv preprint (2017).

[47] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[48] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[49] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[50] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[51] 田中　優亮,“X線危険物認識システム構築のためのデータ合成とその評価”, 卒業論文,東京大学,2018.

List of Publications

1. Lekang Zou, Tanaka Yusuke, and Iba Hitoshi, "Dangerous objects detection of X-ray images using convolution neural network", to be published in the 2018 second International Conference on Security with Intelligent Computing and Big-data Services (SICBS-2018).