Moving Object Detection Based On Weakly Supervised Deep Learning Method

Candidate Number: 1051010

St Anne's College

University of Oxford

A dissertation presented for the degree of

Master of Computer Science

Trinity 2021

Abstract

Moving object detection is of great importance to some tasks in the industry. Based on the fast development of deep learning, deep learning based models have achieved promising results. Numerous deep learning models for object detection need object-level supervision which contains the coordinates of ground-truth boxes. However, the object-level annotations are expensive and difficult to obtain. Therefore, in this project, an adapted weakly supervised deep learning based framework for moving object detection is proposed. The framework adopted a few effective modules to improve the performance such as Instance Refinement, Guided Attention, Box Regression and Knowledge Distillation Module.

The project consists of four main chapters. The motivation and objectives are elaborated in the introduction, and several main challenges are listed. Related Work chapter focuses on the recent deep learning methods for object detection and some conventional models for moving object detection. Besides, the structures and principles of the baseline framework (adapted WSDDN) are described in detail.

After the introduction of the architecture of the framework. The experiments demonstrate the effectiveness of each module through an ablation study, and the proposed framework can finally achieve 14.3% mAP on the Calipsa data set. The visualizations of prediction boxes prove the ability to identify the class and detect moving objects of the framework. Besides, class activation maps (CAM) visualizations show the strong attention of the GAM module on the moving targets.

Chapter five discusses some apparent defects of the proposed framework such as the local minimum problem and multiple objects of the same class that may exist in the images. In response to these problems, several promising solutions that may be explored in future work are given.

***Keywords*—** Moving Object Detection - Weak Supervision - Deep Learning

Table of Contents

1 | Introduction

1.1 Motivation

Over the last 10 years, there have been major improvements in the field of deep learning and Convolutional Neural Network (CNN) in particular. CNNs are widely used in computer vision and achieved great improvement in plenty of open problems. So far, deep learning model has now become a standard and most effective approach to address the problems of several important areas in computer vision such as image classification, segmentation, objection detection, etc.

Object detection is a task aiming to find all objects of a given set such as cat, dog, and person. The object detection model is required to generate the bounding boxes coordinates and class annotations of each object within a given image. A bounding box consists of the coordinates of top-left and bottom-right corner points of a minimal axis-aligned rectangle.

Fully supervised deep learning methods have been applied in the area of object detection widely and studied very well. It means that object-level annotations (bounding boxes coordinates and class annotations) are given in the training set. However, accurate object-level annotations are very time-consuming and expensive to obtain in industrial data sets due to the huge number of images (usually more than 100,000 images in total), therefore making the fully supervised methods sometimes impractical in industry.

Class Activation Map (CAM) (Zhou, Khosla, Lapedriza, Oliva, & Torralba, 2016), proposed by Zhou et al., is a technique of generating a heat map of activation which focuses on the object area. This work discloses the localization ability of CNN and inspires many researchers to train object detection models with only

image-level annotations (each image is annotated with a list of all classes which are to be found in it, but not their locations) in order to save the heavy cost of preparing bounding boxes for each image. This kind of task is called weakly supervised object detection (WSOD). Despite the advantage of saving the significant cost of preparation, the WSOD task is still very challenging. Usually, the performance of WSOD models can only attain 30% to 60% of the average precision (AP) generated by fully supervised object detection (FSOD) models, so there is still a large gap between the performance of WSOD and FSOD.

In real-life tasks, it is generally not necessary to find all the known objects in an image. For instance, CCTV monitoring systems are only required to detect and track moving objects in a series of frames. To tackle this problem, almost all existing research has utilized FSOD models which cost a huge amount of time in preparing bounding boxes and their annotations. Therefore, in this project, WSOD deep learning models are expected to solve the problem of moving object detection.

1.2 Problem Definition

Given two frames in a video (almost consecutive), the model is required to output the details of all moving objects of known classes. The details include the coordinates of bounding boxes, confidence values and class labels of all detected moving objects. (See Figure 1.1)

1.3 Objectives

The objective is to build a general end-to-end deep learning WSOD framework that combines the phases of object detection and object filtering to solve the mov-

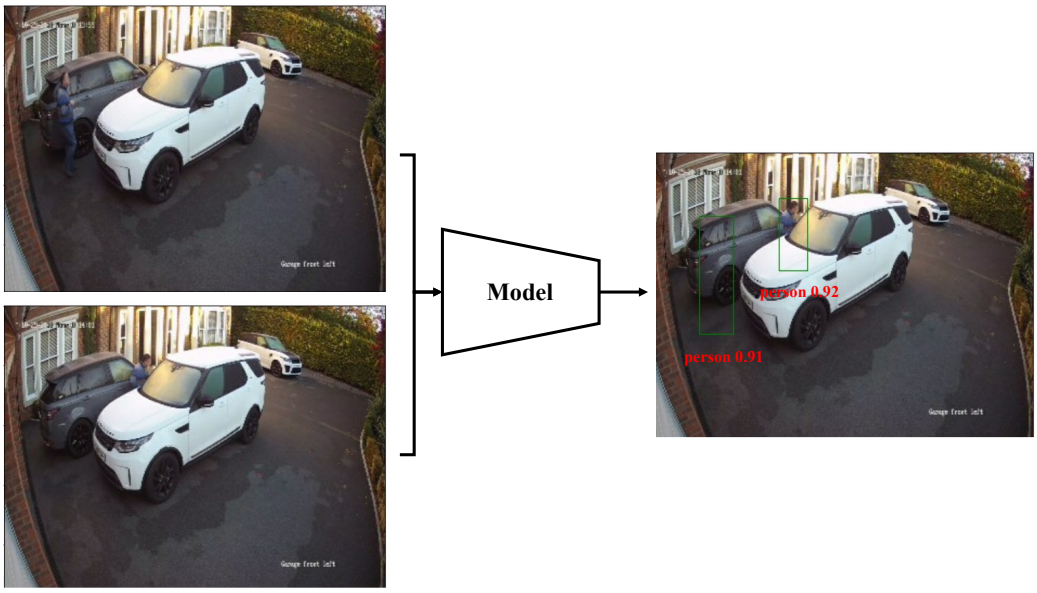

Figure 1.1: **Problem definition**. Left: the input to the model is two concurrent frames from a security video. Right: output from the model is an annotated image. Green bounding boxes show the location of a person in the two frames. Red annotations show the label (person) together with a confidence measure.

ing object detection problem.

The framework is expected to achieve reasonable performance on a real data set from industry, and generalize to other data sets (with different resolutions and from different environments).

In this project, the framework will be trained and tested on the Calipsa data set. Calipsa is a company aiming to address the problem of false CCTV alarms via AI technology. The Calipsa data set is produced by CCTV monitoring systems and contains 277887 pairs of images in the training set and 30714 pairs in the test set. The image pairs are temporally sparse and the time difference between consecutive frames is variable. In addition, the image quality is highly variable.

1.4 Challenges

There are plenty of challenges confronting the weakly supervised moving object detection. Some of the most significant challenges are listed as follows.

1. **Quality of images**: The low resolution (640×480) and different brightness will confuse the model.
2. **Small object**: The small moving objects may be mistakenly regarded as background noise.
3. **Shadow and occlusion**: Shadows should not be detected as objects. Objects which are occluded from the light source (in shadow) are hard to identify.
4. **Local minimum problem**: The WSOD framework tends to output the bounding box of the most characteristic part of the moving object instead of the whole body (e.g. face of a person or headlight of a truck).

2 | Related Work

2.1 Object localization in CNNs

Convolutional Neural Networks (CNNs) excel at recognizing and extracting patterns in the input image, such as lines, gradients and circles. It is the property that enable CNNs to become a powerful feature extractor for a variety of computer vision tasks. However, very few researchers attempted to study the ability of localization of CNN until (Zhou, Jagadeesh, & Piramuthu, 2015) pointed out it.

These authors concluded CNNs are capable of storing the localization information without the supervision of bounding boxes. Although having this remarkable ability to localize objects in CNNs, such ability will be lost if the downstream fully connected layers are used for classification task.

In order to preserve the localization ability of CNNs, Global Average Pooling (GAP), instead of multiple fully connected layers, can be applied after all convolutional units. GAP is commonly deployed as a structural regularizer to prevent over-fitting during training, but in (Zhou et al., 2016), the authors operate an inner product on the feature vector generated by GAP and the weight of the last convolutional layer to create a class activation map (CAM) which represents the activation areas of each class effectively.

2.2 Weakly Supervised Object Detection

Recently, weakly supervised models for object detection can be split into two main kinds of approaches: segmentation-based method and Multiple Instance Learning

(MIL) (Dietterich, Lathrop, & Lozano-Pérez, 1997). So far, due to the outstanding performance of MIL methods, the MIL-based model actually becomes the most popular approach in WSOD task.

2.2.1 Multiple Instance Learning

In the original definition of MIL (Dietterich et al., 1997), the basic training unit is called a bag $\mathcal{B}$ and each bag consists of a collection of instances $\{\mathbf{I}_j\}$. There are two constraints in MIL: (1) If all instances of a bag are labeled as negative, the bag is treated as negative. (2) If at least one instance of a bag is labeled as positive, the bag is positive.

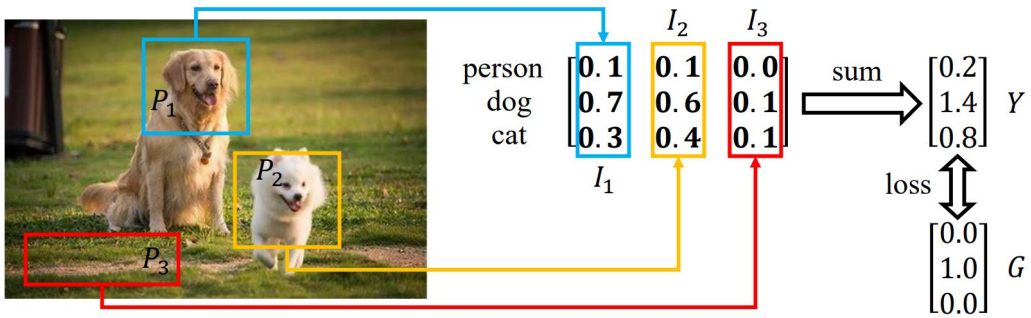

Figure 2.1: **MIL computation example in WSOD task**. Several proposals exist in the left image and the MIL module output a proposal score vector (column of the matrix) for each proposal. The image score $\mathbf{Y}$ is the sum of all score vectors which is expected to be close to the ground-truth label $\mathbf{G}$.

Based on the definition of MIL, WSOD task can be transformed into a MIL problem naturally. Consider a data set of N images. Every image in the data set is viewed as a bag $\mathcal{B}_i$ ($i \in [1, N]$) in MIL, and all proposal bounding boxes with this image (proposals for short) $\{\mathbf{P}_j^i\}$ are the instances (elements) of this bag $\mathcal{B}_i$. Typically, for each instance $\mathbf{P}_j^i$, a model will generate a feature vector $\mathbf{I}_j^i \in \mathcal{R}^C$, where C is usually denoted as the number of classes in the data set.

Note that in the WSOD task, each image is only associated with an image-level annotation. The annotation $\mathbf{G}_i \in \mathcal{R}^C$ is a vector that only contains 1 or 0, representing whether or not the class appears in the image. The output $\mathbf{Y}_i$ of a WSOD model is the same size of $\mathbf{G}_i$ for the convenience of backward propagation. A simple method to obtain the output $\mathbf{Y}_i$ is to aggregate $\{\mathbf{P}_j^i\}$ via aggregation function (such as sum) which satisfies the requirement of MIL.

$$\mathbf{Y}_i = \mathrm{AGG}(\mathbf{I}_j^i) \quad \Longleftrightarrow \quad \mathbf{G}_i \tag{2.1}$$

where AGG is an aggregation function and $\Longleftrightarrow$ represents loss computation.

2.2.2 Basic MIL-based WSOD Framework

WSDDN (Bilen & Vedaldi, 2016) is a milestone work in the WSOD task. Bilen et al. first introduced the MIL module into WSOD task. In addition, WSDDN utilized two branches to perform the localization and classification tasks respectively. Despite the low performance and simplicity, almost all WSOD frameworks are developed based on WSDDN. In this section, we describe and explain some fundamental modules and processes in this framework.

2.2.2.1 Proposal Generation

Current FSOD frameworks are almost all developed based on two classical FSOD frameworks: Fast-RCNN (Girshick, 2015) and YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016). The YOLO framework directly produces the prediction boxes by the last layer while Fast-RCNN generates the translation and scaling offsets of a collection of proposals.

In the FSOD task, the annotation contains the coordinates of ground-truth boxes so that the YOLO framework can perform the regression task directly on the co-

ordinates. However, in the WSOD task, object-level annotations do not exist so the prediction boxes are part of proposals.

There are two types of proposal generation methods: sliding window (SW) and searching algorithms.

The SW proposals are identical to the sliding windows during the convolutional operation, except for varied sizes of windows applied in SW operations. Despite the rapid production of proposals in SW operations, more than ten thousand of them will largely reduce the efficiency of Region of Interest (RoI) pooling layers and fully connected layers in the forward process.

Conversely, searching algorithms (which detect complete objects based on edges, colors, etc.) produce far fewer proposals, normally less than 2000. Therefore, leveraging searching algorithms to produce proposals offline prevails in WSOD frameworks. The most common algorithms are Selective Search (SS) (Uijlings, Van De Sande, Gevers, & Smeulders, 2013) and Edge Box (EB) (Zitnick & Dollár, 2014).

SS (Uijlings et al., 2013) algorithm first produces small proposal boxes according to the segmentation results by the method of (Felzenszwalb & Huttenlocher, 2004). The next step is to constantly merge small boxes according to the metrics of similarity including color, size, shape and texture. The EB (Zitnick & Dollár, 2014) algorithm produces proposals based on edge detection and clustering.

In WSDDN (Bilen & Vedaldi, 2016), the authors try both algorithms and find that the model trained on EB proposals achieves slightly better performance (1% average precision) than on SS proposals. Furthermore, the model attains more than 3% improvement when the proposal scores of EB are given as an attention map in the framework.

2.2.2.2 Feature Extractor

The feature extractor, often called backbone in the object detection task, aims to transform a 3-channel RGB image into a high-dimensional (hundreds of channels) feature tensor. This feature tensor encodes plenty of information (such as the location of each foreground object) stored in an image, and they are always decoded by fully connected layers to tackle different kinds of problems such as class predictions and offset predictions. The prevailing feature extractors in object detection tasks are AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and VGG16 (Simonyan & Zisserman, 2014).

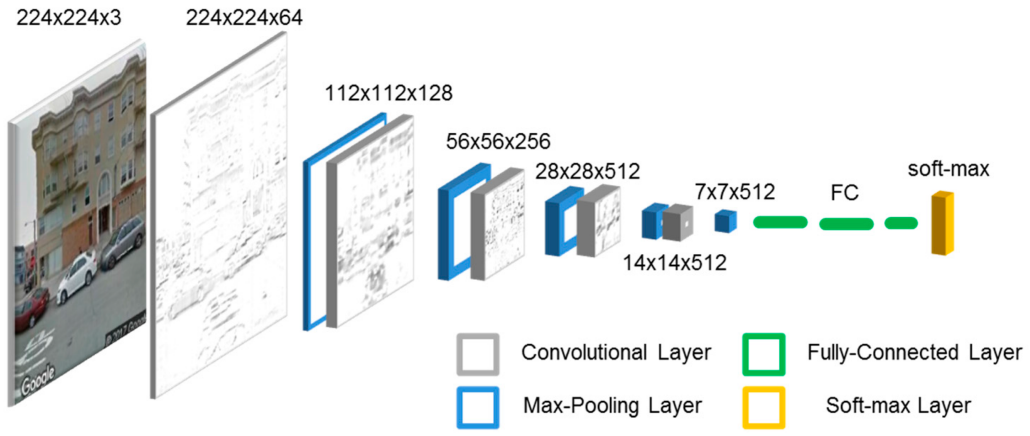

Figure 2.2: **VGG16 Network Architecture** (Simonyan & Zisserman, 2014). The feature extractor represents the VGG16 network without the fully connected layers and the last soft-max layer.

There are also other influential feature extractors including ResNet (Simonyan & Zisserman, 2014), GoogleNet (Szegedy et al., 2015), etc. However, for the object detection task, the extractor gains significant superiority when the kernel sizes of convolutional layers are small because it can reduce the loss of object localization precision in images shown in the experiment (Shen et al., 2020).

Following WSDDN (Bilen & Vedaldi, 2016) and for impartial comparisons, re-

cent WSOD frameworks commonly utilize the VGG16 backbone pre-trained on the Imagenet (Deng et al., 2009) data set.

2.2.2.3 Detection Head

The detection head contains the RoI pooling layer and the MIL module. It is designed to transform a feature tensor into a 2D proposal score tensor. Each row of it represents the class scores for a proposal.

RoI Pooling was first introduced in Fast-RCNN (Girshick, 2015) and has been widely used in object detection frameworks since then. It functions as a pooling layer in which the pooling areas are not sliding windows. The actual polling areas are the boxes according to the locations of the proposals. In addition, all pooling areas will be transformed into small tensors of a unified size such as 7×7 in common.

The MIL module in WSDDN (Bilen & Vedaldi, 2016) encompasses two streams, one for class scores predictions and the other for providing confidence of each proposal.

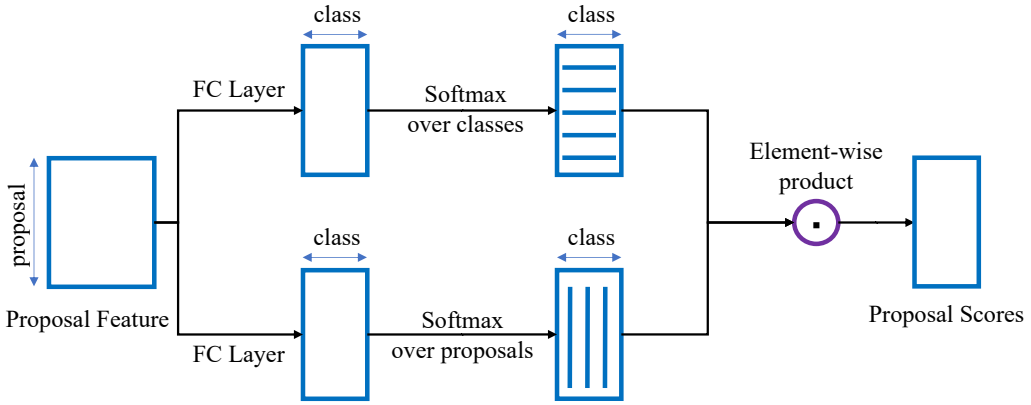

Figure 2.3: **MIL Module**.

As represented in Figure 2.3, the detection head performs element-wise product

on the middle features generated by two streams in the MIL module, so that we can obtain proposal scores.

2.2.3 Effective Modifications

Based on WSDDN, various effective WSOD frameworks emerge. In this section, we briefly introduce some of the most powerful and classical frameworks that achieve great improvement over WSDDN and have a vast impact on future works.

In (Diba, Sharma, Pazandeh, Pirsiavash, & Van Gool, 2017), Diba et al. proposed a cascaded CNN which combines the segmentation and WSOD in cascaded style. The segmentation is guided by a class activation map (CAM) and feature tensor produced by the backbone. Segmentation results help weigh the importance of each proposal and make the MIL module focus on object areas. This cascaded method achieves 3.5% mAP improvement on the VOC2007 test data set comparing to WSDDN (mAP is an evaluation metric for object detection, details can be referred to (Everingham & Winn, 2011)).

OICR (Tang, Wang, Bai, & Liu, 2017) was the state-of-the-art WSOD framework in 2017. Tang et al. developed an online instance classifier refinement method to refine proposal scores, by which the local optimum problem (see 1.4) can be partially addressed. Instance (proposal) labels inferred from the MIL module are propagated to their overlapped instances, and the next-level classifier will be trained by these new labels. The author applied three refinement modules and attained 42.0% mAP on the VOC2007 test data set. The authors proposed to train an FSOD network Fast-RCNN (Girshick, 2015) with the object-level annotations produced by the OICR framework. The final performance arrives at 47.0% mAP.

Besides, Tang et al. developed a new WSOD framework based on OICR in 2018, called PCL (Tang, Wang, Bai, et al., 2018). PCL also applied a multi-level in-

stance learning manner, but it replaced the spatial label propagation with proposal cluster learning. Concretely, the instance label inference is based on a cluster of instances, and the center of the cluster is determined by a few instances with high scores other than one instance with the top score. The performance of PCL achieves approximately 3% mAP improvement on the VOC2007 test data set.

With the inspiration of OICR and PCL, Yang et al. proposed that the WSOD framework (Yang, Li, & Dou, 2019) can apply the box regression using the pseudo ground truth. Within the multi-level refinement method of OICR, the ground truth of next-level refinement is the instance of the top score at the current level. Therefore, box regression's ground truth could be the predicted instance produced by the last-level refinement. Furthermore, Yang et al. proposed another GAM module that adapted CAM to generate an attention mask. The attention mask will be directly operated on the feature map produced by the backbone. Their framework finally attains 48.6% mAP on the VOC2007 test data set without training an FSOD framework.

Zeng et al. considered bottom-up information while measuring the proposal score (Zeng, Liu, Fu, Chao, & Zhang, 2019) to improve the location precision. Previous WSOD frameworks only focus on the features extracted from the whole image by convolutional and pooling layers. This kind of feature is called top-down information. Zeng et al. combined several objectness measurements proposed (Alexe, Deselaers, & Ferrari, 2010) with regular proposal scores to evaluate each proposal. Based on OICR, WSOD2 (Zeng et al., 2019) achieves 53.6% mAP on the VOC2007 test data set, but new complicated measurements reduce the speed of the forward inference to some extent.

Apart from applying box regression to improve location precision, WSRPN (Tang, Wang, Wang, et al., 2018) adopted Region Proposal Network (Ren, He, Girshick,

& Sun, 2015) in the WSOD framework to improve the quality of proposals. Their experiments show the performance reaches 45.3% mAP on the VOC2007 test data set without training an FSOD framework.

2.3 Moving Objection Detection

The methods to solve moving object detection tasks can be roughly divided into three kinds of approaches: Background Subtraction, Optical Flow and deep learning models. In this section, we will briefly introduce these types of methods and compare their strength and weakness.

2.3.1 Background Subtraction

Temporal Differencing (Liu, Ai, & Xu, 2001), also known as frame differencing, is the most simple technique among background subtraction methods. It subtracts two nearby frames from each other on a pixel-level basis. Both of them are transformed from RGB into gray-scale figures at first. The output is the absolute value of their subtraction. In most research works, this technique is combined with deblurring and threshold masking, so as to distinguish authentic movements from noise such as the changing of light conditions or minute offsets of the camera.

Consider a series of frames $F^{(i)} \in \mathcal{R}^{3\times W_F \times H_F}$ where $i \in [1, N_F]$, N_F is the number of frames, W_F is the width of frame and H_F is the height of frame. We denote $F_R^{(i)}$, $F_G^{(i)}$ and $F_B^{(i)}$ as three matrices according to difference RGB channels.

Here we utilize the Luminosity Method (Kanan & Cottrell, 2012) to transform the RGB figures into the gray-scale images.

$$G^{(i)} = 0.281F_R^{(i)} + 0.562F_G^{(i)} + 0.093F_B^{(i)} \tag{2.2}$$

Figure 2.4: **Temporal Differencing Output**. Right-hand images show the difference between the relevant two image frames. Note that changes in tree position (top) and shadows (bottom) have contributed to the result.

where $G^{(i)}$ is the gray-scale matrix of the i th frame.

The temporal difference $TD^{(i)}$ is computed as

$$TD^{(i)} = |G^{(i)} - G^{(i-1)}| \tag{2.3}$$

Normally, there is a threshold γ for preventing background noise so that the result is a binary image.

$$TD^{(i)}(x,y) = \begin{cases} 0 & TD^{(i)}(x,y) < \gamma \\ 255 & TD^{(i)}(x,y) >= \gamma \end{cases} \tag{2.4}$$

(x,y) is a pixel of moving objects if $TD^{(i)}(x,y) > 0$.

Temporal differencing is easy to implement and requires few computational resources. However, the performance is highly reliant on the velocity of target ob-

jects and the threshold parameter.

There are also considerable methods related to background subtraction such as Mean Filter (Benezeth, Jodoin, Emile, Laurent, & Rosenberger, 2008), Running Gaussian average (Wren, Azarbayejani, Darrell, & Pentland, 1997), Background Mixture models (Stauffer & Grimson, 1999), etc. These approaches apply different probability models and kernels to detect the foreground.

2.3.2 Optical Flow

Optical Flow (Beauchemin & Barron, 1995) is also a conventional type of approach able to tackle the moving object detection task. When an object moves, it forms a series of continuously alternating frames on the human retina, and the information of these changes flows through the retina at different times, as if it were a flow of optics.

In practice, the velocity vector of pixels in the image is regarded as optical flow. the optical flow method utilized in moving object detection determines the continuity of the optical flow field. If there are no moving objects in the frames then the field is continuous.

The basic assumptions of the Optical Flow method are listed as follows:

1. Brightness constancy: Even if a pixel in an image moves to other positions, the brightness keeps the same.

2. Small Movement: The moving objects in two frames are very close.

Consider the brightness intensity field $I(x, y, t)$, where x, y are the physical coordinates and t denotes time. We can obtain the following equation by Taylor series

based on small movement assumption:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t)+\frac{\partial I}{\partial x}\Delta x+\frac{\partial I}{\partial y}\Delta y+\frac{\partial I}{\partial t}\Delta t+O(\Delta x, \Delta y, \Delta t) \tag{2.5}$$

With the assumption of brightness constancy $I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t)$ and ignoring the high-order term $O(\Delta x, \Delta y, \Delta t)$, we have

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{2.6}$$

Dividing by Δt,

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{2.7}$$

which is equal to

$$\nabla I \cdot V = -I_t \tag{2.8}$$

where V is the optical flow field.

In order to solve Equation 2.8, we need another constraint equation (Lucas, Kanade, et al., 1981; Horn & Schunck, 1981) to estimate the actual flow.

Despite the high accuracy of the Optical Flow method, the assumptions of this approach are strict, especially the assumption that the movements must be small. Therefore, this method can be not applied to some data sets in the industry.

2.3.3 Deep Learning Method

With the fast development of the deep learning technique, many recent papers (Zhu et al., 2020; Zhang, Li, Zhang, Wu, & Zhao, 2015; Patil & Murala, 2018;

Ye, Li, Chen, Wachs, & Bouman, 2018) have attempted to apply deep learning frameworks in moving object detection.

Zhu et al. (Zhu et al., 2020) adopted a two-stage deep learning framework. The first stage was designed to extract the feature of all moving areas; The second stage is another deep CNN that classifies and accurately localizes the moving objects. During preprocessing, temporal differencing is applied to provide more direct information of moving areas.

Mandal et al. (Mandal, Kumar, Saran, et al., 2020) proposed a multi-level feature extraction backbone. This design can effectively detect small objects thus obtaining great performance in detecting moving objects.

Chen et al. (Chen, Wang, Zhu, Tang, & Lu, 2017) adapted the LSTM (Hochreiter & Schmidhuber, 1997) module and encoder-decoder (Badrinarayanan, Kendall, & Cipolla, 2017) structure in a deep learning framework that is able to receive multiple frames at the same time and remember long-term features or information. Also, they applied an attention layer to enhance the activation of moving parts.

3 | Methodology

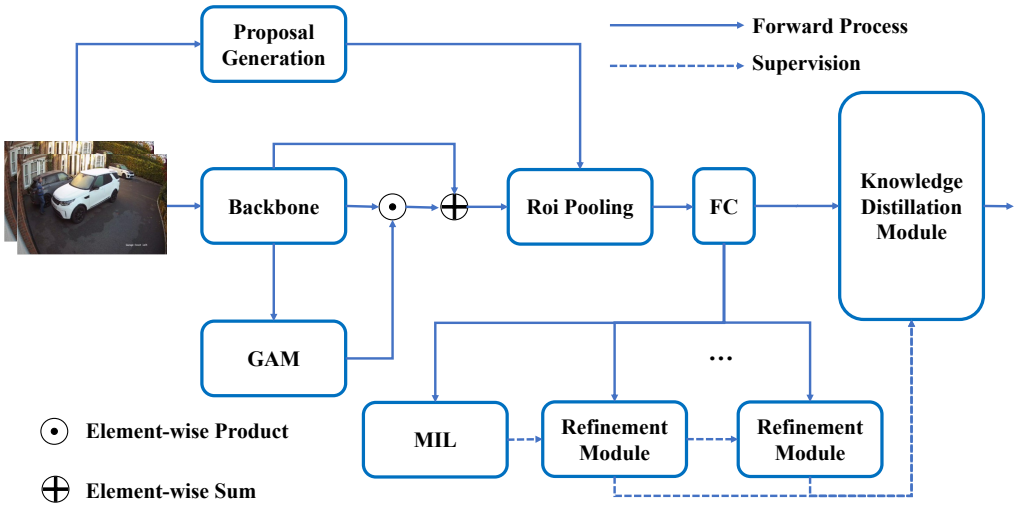

Figure 3.1: **WSOD Framework Architecture.** The framework used in this project receives an image pair and outputs a proposal score $\mathbf{X}^P \in \mathbb{R}^{N_p \times C}$, where N_p represents the number of proposals and C is the number of classes.

Building on the successful features of neural networks explored in Chapter 2, we designed a framework for weakly supervised object detection. The architecture of our framework is represented in Figure 3.1. The input of the framework is an image pair which are two consecutive frames. The backbone consumes two images and produces a feature map. The Guided Attention Module (GAM, see Section 3.5) functions as an attention module to enhance the feature map. The RoI pooling layer receives the proposals and the enhanced feature map. The output of RoI pooling (see Section 3.2) and several fully connected layers is called the proposal feature. MIL, refinement and knowledge distillation modules all receive this proposal feature (the output tensor of the FC layer in Figure 3.1) as an input and generate a proposal score. All refinement modules aim to refine the classifiers inside by the pseudo ground truth. The concrete refinement process will

be elaborated in the Section 3.4. In order to maintain the information, the knowledge distillation module is adopted. It aggregates all proposal scores produced by multiple refinement modules and outputs a final proposal score.

In this chapter, all modules will be described in the following sections.

3.1 Backbone

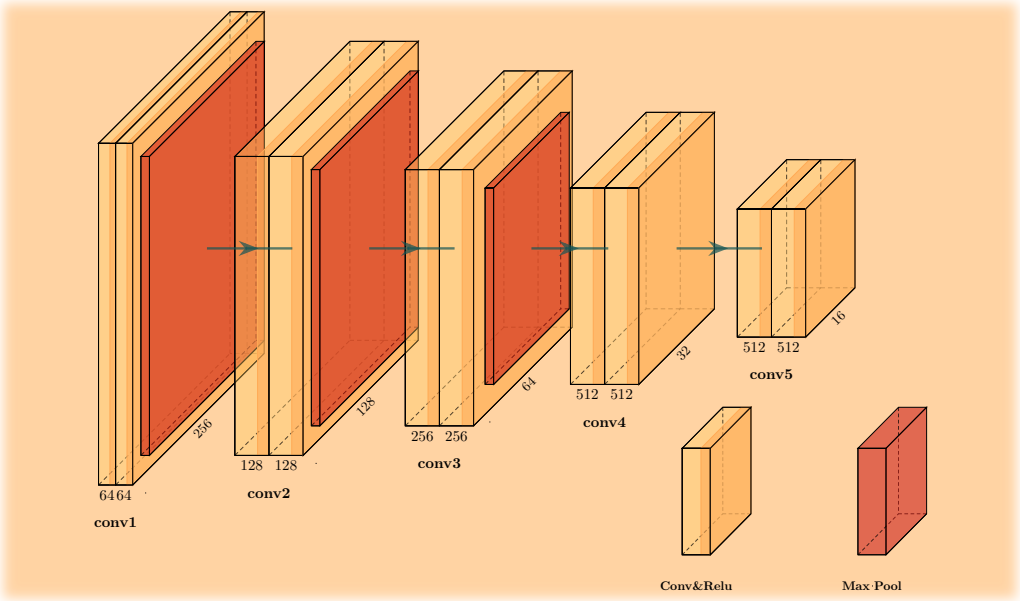

Figure 3.2: **Backbone of the framework.** This module is based on VGG-16. Adaptations from the original VGG-16 are explained in the text.

As mentioned in Section 2.2.2.2, the backbone plays an important role in the object detection task. Considering the concrete problem defined in Section 1.2, an adapted VGG16 (Simonyan & Zisserman, 2014) architecture was applied as the backbone of the framework (See Figure 3.2). The details of layers in the backbone are listed in Table 3.1.

There are a few differences between this adapted VGG16 backbone and the original architecture.

Layer Name	Layers
Conv1	Conv2d(3, 64) + Relu
	Conv2d(64, 64) + Relu
	MaxPool2d
Conv2	Conv2d(64, 128) + Relu
	Conv2d(128, 128) + Relu
	MaxPool2d
Conv3	Conv2d(128, 256) + Relu
	Conv2d(256, 256) + Relu
	Conv2d(256, 256) + Relu
	MaxPool2d
Conv4	Conv2d(256, 512) + Relu
	Conv2d(512, 512) + Relu
	Conv2d(512, 512) + Relu
	MaxPool2d
Conv5	Conv2d(512, 512) + Relu
	Conv2d(512, 512) + Relu
	Conv2d(512, 512) + Relu
	MaxPool2d

Table 3.1: **The layers of the adapted VGG-16.** The kernel size and stride of all convolutional layers are 3 and 1. The layer $conv5$ applies dilated convolutions where the dilation parameter equals 2.

1. In order to prevent losing numerous informative features caused by down-sampling, the last two max-pooling layers are removed in this backbone. Therefore, only three max-pooling layers remain, and the ratio of down-sampling is eight which means one individual pixel of the output corresponds to 8×8 pixels in the input image.

2. The number of channels in the first convolutional layer is doubled because the input comes as a pair of images.

3. Batch Normalization layers do not appear in the adapted VGG16 backbone. Many recent works do not apply Batch Normalization or Group Normalization in the backbone due to poor performance. These normalization layers

are normally utilized as regularization to prevent over-fitting. However, the loss propagated backward from the MIL module is usually not accurate so that more regularization layers will make the framework more difficult to train.

4. According to (Shen et al., 2020), their WSOD backbone adopts dilated convolutions (Yu & Koltun, 2015) in order to enlarge the reception field and obtain high-resolution feature maps without down-sampling. Therefore, our backbone applies dilated convolutional layers in $conv5$.

3.2 RoI Pooling

The RoI Pooling (Girshick, 2015) utilizes multiple max pooling operations to convert the feature in regions of interest into the feature maps of fixed size, normally 7×7.

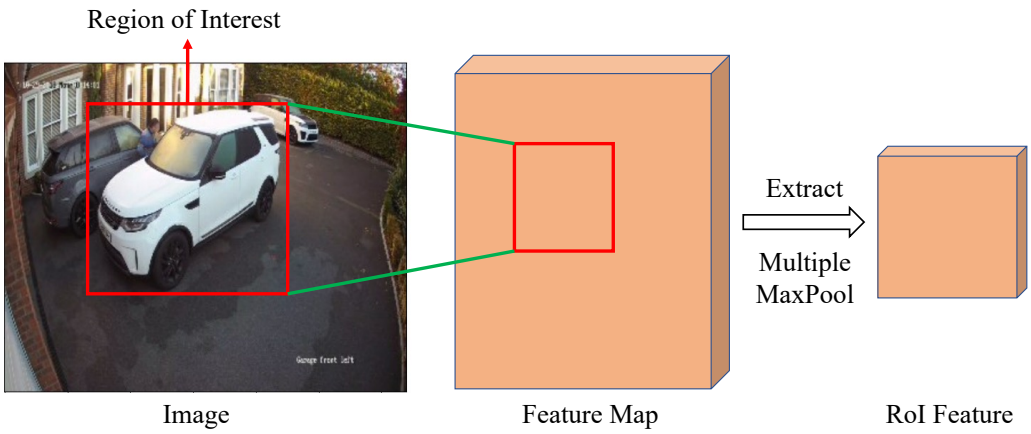

Figure 3.3: **RoI Pooling.** Extract the area in the feature map corresponding to the region of interest in the image, and perform the multiple max pooling operations.

Suppose the size of the input image is (W_I, H_I) and the coordinate of the region of interest is (r, c, w, h) where (r, c) is the coordinate of the top-left point and (w, h) is the size of this region. In addition, the size of the feature map is (W_F, H_F). The

final output RoI Feature map (W_R, H_R) is usually defined as $(7, 7)$.

The 4-vector coordinates of the corresponding area (red box of the feature map in Figure 3.3) in the feature map is calculated by

$$[r_a, c_a, w_a, h_a] = [r, c, w, h] \cdot W_F / W_I \tag{3.1}$$

This rectangle area will be cropped from the feature map and down-sampled by multiple max pooling operations.

Assume the expected size of RoI feature map is (W_R, H_R), then there will be roughly $(w_a/W_R) \cdot (h_a/H_R)$ max pooling operations to obtain the RoI feature map. Here for simplicity, $w_a > W_R$ and $h_a > H_R$ can be assumed.

Each region of interest (proposal) leads to an RoI feature map, and each RoI feature map will be transformed into an RoI feature vector by a fully connected layer. Therefore, the proposal feature matrix (the input in Figure 2.3) is produced by stacking multiple RoI feature vectors.

3.3 MIL Detector

As shown in Figure 2.3, the proposal feature $\mathbf{X}^R \in \mathbb{R}^{N_p \times N_r}$ is the input of the MIL module. N_p and N_r denote the number of proposals and the length of the RoI feature vector.

Both streams have their own fully-connected layers to extract useful information and reduce N_r to C, where C is the number of classes. The upper stream is responsible for predicting the confidence value of every class in each proposal, whereas the lower stream aims to provide the probability of existing moving objects in each proposal. Therefore, the soft-max operations in both streams differ.

The concrete calculations are

$$\hat{\mathbf{X}}^R = \hat{\text{FC}}(\mathbf{X}^R) \in \mathbb{R}^{N_p \times C} \tag{3.2}$$

$$\mathbf{X}_{ij}^{R_{upper}} = \frac{\exp(\hat{\mathbf{X}}_{ij}^R)}{\sum_j \exp(\hat{\mathbf{X}}_{ij}^R)} \tag{3.3}$$

$$\tilde{\mathbf{X}}^R = \tilde{\text{FC}}(\mathbf{X}^R) \in \mathbb{R}^{N_p \times C} \tag{3.4}$$

$$\mathbf{X}_{ij}^{R_{lower}} = \frac{\exp(\tilde{\mathbf{X}}_{ij}^R)}{\sum_i \exp(\tilde{\mathbf{X}}_{ij}^R)} \tag{3.5}$$

where $\mathbf{X}^{R_{upper}}$ and $\mathbf{X}^{R_{lower}}$ are the output of both streams, and C denotes the number of classes. Note that the parameters of FC layers are not shared (see above, denoted $\hat{\text{FC}}$ and $\tilde{\text{FC}}$).

The proposal score $\mathbf{X}^P$ can be obtained by performing element-wise product on the output of both streams.

$$\mathbf{X}_{ij}^P = \mathbf{X}_{ij}^{R_{upper}} \cdot \mathbf{X}_{ij}^{R_{lower}} \tag{3.6}$$

Because the annotation $\mathbf{Y} \in \{0, 1\}^C$ represents the absence of each one of the classes using 0 or 1, the proposal score $\mathbf{X}^P \in \mathbb{R}^{N_p \times C}$ should be aggregated as described in Equation 2.1. The output, called classification score $\mathbf{\Phi} \in \mathbb{R}^C$, is calculated by

$$\mathbf{\Phi}_j = \sum_i \mathbf{X}_{ij}^P \tag{3.7}$$

The MIL module is trained by standard binary cross entropy loss:

$$L_{\text{MIL}} = -\sum_{c=1}^{C} [\mathbf{Y}_c \log \mathbf{\Phi}_c + (1 - \mathbf{Y}_c) \log(1 - \mathbf{\Phi}_c)] \tag{3.8}$$

3.4 Online Instance Classifier Refinement

As the figures shown in (Bilen & Vedaldi, 2016), the MIL detector tends to focus on the most characteristic part of target objects which is a common problem in weakly supervised works. This phenomenon is caused by the local minimum problem. Normally, after several iterations of training, the proposals, which contains the most characteristic part such as heads of people, tyres of vehicles, will be given higher confidence by the detector.

However, for the FSOD detector, there will be a box regression module to modify the coordinates of the proposal which have the highest confidence value so that the prediction box will more accurately wrap the whole body of target objects.

Therefore, without a box regression module, the WSOD detector will directly output the original proposal which can only locate the most distinctive part.

In order to solve this problem, Tang et al. designed an Online Instance Classifier Refinement (OICR) (Tang et al., 2017) module to refine multi-level classifiers. The motivation is that the confidence value of the proposals, which include the whole object, should be increased and higher than the ones that only contain the characteristic part.

Consider the proposals that have high spatial overlaps with the top-scoring one. It is probable that these proposals contain parts of the whole object. Therefore, they can be labeled as the same class as the top-scoring proposal, and regarded as pseudo ground truth to train the next-level classifier. After plenty of training iterations, the next-level classifier is capable of identifying the whole object including the parts which are not distinctive.

As shown in Figure 3.4, the top-scoring proposal (brown box) only contains the

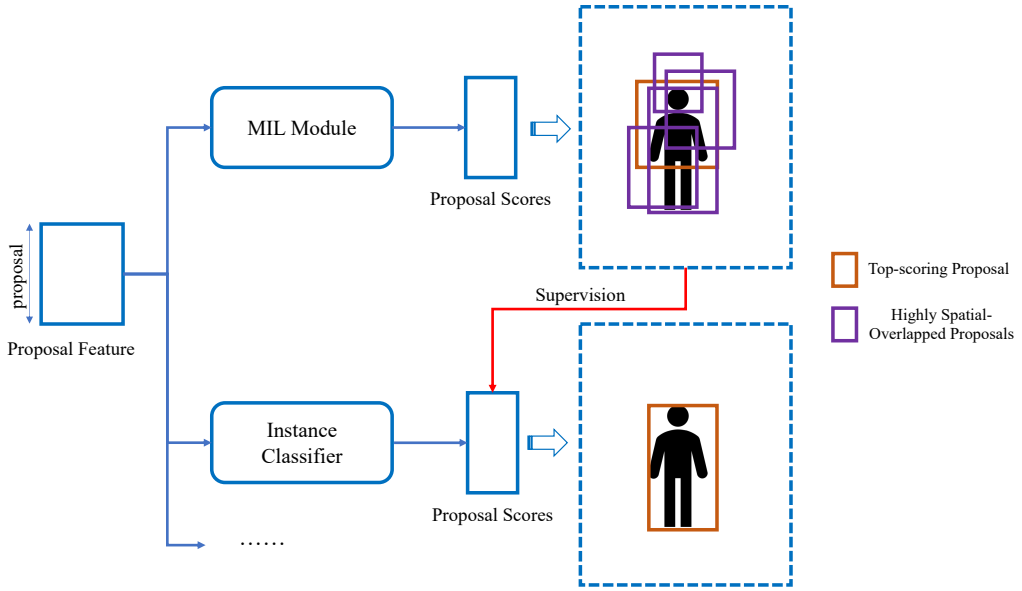

Figure 3.4: **Online Instance Classifier Refinement.** The proposal score generated by the MIL module is responsible for supervising the first instance classifier. And the proposal score produced by the k-th classifier will supervise the (k+1)-th classifier.

upper part of a person object, but there are several proposals that have high overlaps (purple boxes) with the top-scoring one. The best threshold (determined by the experiments) of this overlap measured by Intersection over Union (IoU) is 0.5 (Tang et al., 2017) (IoU metric computes the proportion of the intersection area over the union area of two boxes, see Figure 4.1). All the proposals that have overlaps higher than the threshold will be treated as foreground proposals with persons inside. These labels (now regarded as ground truth) will be utilized to train the next-level instance classifier. The next-level classifier will have knowledge of every part of a person object so that the proposal containing the complete person object may be given the highest score by the next-level classifier.

The detailed process of OICR is described in Algorithm 1. For each class that existed in the image pair, the top-scoring proposal j_c^k is picked out. For all proposals, there are two conditions. If the overlap between it and any top-scoring proposal

Algorithm 1: Online Instance Classifier Refinement

Input: The proposal score $\mathbf{X}^{Pk}$ of the k-th classifier; The label vector of an image pair $\mathbf{Y} \in \{0,1\}^C$

Output: The proposal label $\mathbf{V}_r^k = [v_{r0}^k, \ldots, v_{rC}^k] \in 0, 1^{C+1}$; The loss weights w_r^k. $r \in [1, N_p]$. N_p is the number of proposals. C denotes the number of classes.

Initialize $\mathbf{I} = [I_1, \ldots, I_{N_p}] = -\inf$;
Initialize $V_{rc}^{k+1} = 0,\ c \in [1, C]$;
Initialize $V_{r0}^{k+1} = 1$;
for c *in* $[1, C]$ **do**
  if $\mathbf{Y}_c = 1$ **then**
    Get the index of top-scoring proposal j_c^k;
    for r *in* $[1, N_p]$ **do**
      $I_r^{'} = \text{IOU}(r, j_c^k)$;
      if $I_r > I_r^{'}$ **then**
        $I_r = I_r^{'}$;
        $w_r^{k+1} = \mathbf{X}_{j_c^k, c}^{Pk}$;
        // I_t is the threshold of overlap that determines whether a nearby proposal is a foreground object
        if $I_r > I_t$ **then**
          $V_{r,c^{'}}^{k+1} = 0, c^{'} \neq c$;
          $V_{rc}^{k+1} = 1$;

is higher than a threshold I_t, then this proposal is labeled as the same class of this top-scoring proposal, and the loss weight is set to the corresponding confidence $\mathbf{X}_{j_c^k, c}^{Pk}$. If all overlaps are less than the threshold I_t, then this proposal is labeled as background class ($c = 0$).

Note that the index of the top-scoring proposal j_c^k for each class c is computed by

$$j_c^k = \arg\max_r \mathbf{X}_{rc}^{Pk} \tag{3.9}$$

where $\mathbf{X}^{Pk}$ is the k-th proposal score.

Considering the noisy ground truth generated by the OICR process, the loss weights w_r^k can be inserted into the loss function of multi-class cross-entropy. The loss function of each supervision is defined as

$$L_{\text{refine}}^k = -\frac{1}{N_p}\sum_{r=1}^{N_p}\sum_{c=0}^{C} w_r^k \mathbf{V}_{rc}^k \log \mathbf{X}_{rc}^{Pk} \tag{3.10}$$

where k is the time of refinement.

3.5 GAM Module

Guided Attention Module (GAM) (Yang et al., 2019) is applied to weigh the feature map and enhance the score of the target object area. The conventional way of deploying an attention module is just getting the product of the feature map and the attention map, and adding it to the original feature map. However, there are two main differences between GAM and the common attention module:

1. The GAM contains both spatial and channel attention which means the attention map generated by GAM has the shape $\mathbb{R}^{D\times W\times H}$ instead of only $\mathbb{R}^{W\times H}$ while the feature map has the shape $\mathbb{R}^{D\times W\times H}$.

2. The GAM module owns an isolated loss calculation whereas other attention modules are only trained with the total framework. The additional classification loss can accelerate the learning of attention weights.

The calculation of the attention map $\mathbf{A} \in \mathbb{R}^{D\times W\times H}$ is

$$\mathbf{A}_{cij} = \text{ReLU}(\mathbf{w^T}_c \mathbf{X}_{ij} + \mathbf{b}_c), \quad c \in [1, D] \tag{3.11}$$

where $\mathbf{w^T}$ and $\mathbf{b}$ are the parameter and bias of the convolutional layer, and $c \in [1, D]$ represents the channel. The ReLU activation function (Agarap, 2018) is

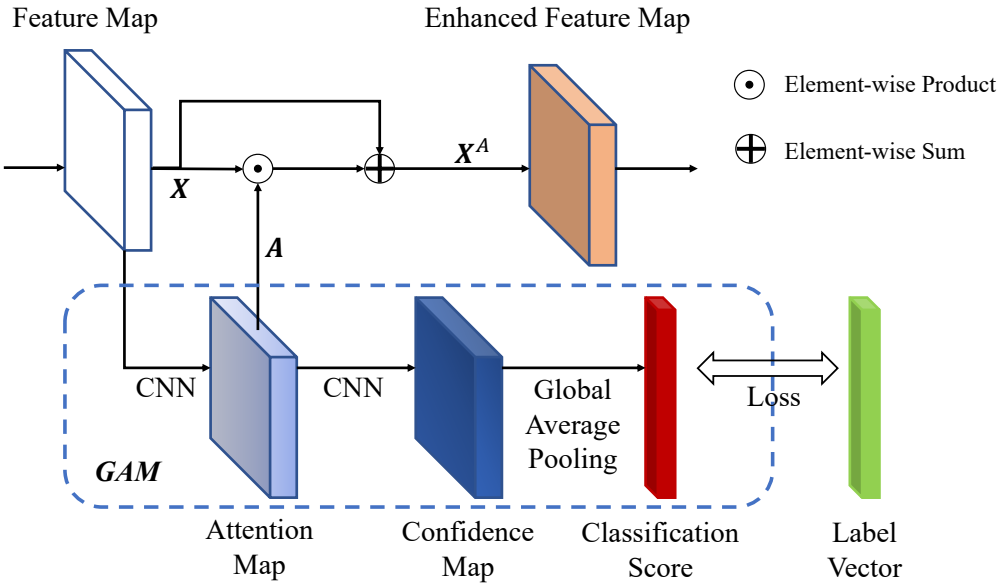

Figure 3.5: **Guided Attention Module (GAM).** The GAM module outputs an attention map to weigh the feature map and encourage the framework to focus on the moving area. A classification score is also generated in the GAM module to perform the backward propagation, and speed up the parameter learning.

computed as

$$\text{ReLU}(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.12}$$

The enhanced feature map $\mathbf{X}^A$ is obtained by

$$\mathbf{X}^A_{cij} = (1 + \mathbf{A}_{cij})\mathbf{X}_{cij} \tag{3.13}$$

In order to accelerate the learning of GAM, an additional backward propagation is introduced in GAM. Inspired by CAM (Zhou et al., 2016), global average pooling (GAP) is adopted to produce a classification score $\mathbf{\Phi}^{\text{GAM}} \in \mathbb{R}^C$, where C is the number of classes.

The concrete process to obtain the classification score $\mathbf{\Phi}^{\text{GAM}}$ is

$$\hat{\mathbf{A}}_{cij} = \text{ReLU}(\mathbf{w}^{\mathbf{T}}{}_c\mathbf{A}_{ij} + \mathbf{b}_c), \quad c \in [1, C] \tag{3.14}$$

$$\mathbf{\Phi}^{\text{GAM}} = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}\hat{\mathbf{A}}_{\cdot ij} \in \mathbb{R}^C \tag{3.15}$$

where $\hat{\mathbf{A}}$ is the confidence map (see Figure 3.5). W, H are the width and height of the feature map $\mathbf{X}$ and the attention map $\mathbf{A}$.

Similar to the MIL loss function L_{MIL}, the GAM loss function L_{GAM} is also a binary cross entropy loss which is

$$L_{\text{GAM}} = -\sum_{c=1}^{C}[\mathbf{Y}_c \log \mathbf{\Phi}_c^{\text{GAM}} + (1 - \mathbf{Y}_c)\log(1 - \mathbf{\Phi}_c^{\text{GAM}})] \tag{3.16}$$

where $\mathbf{Y}$ is the label vector (ground truth).

3.6 Box Regression

Due to the lack of object-level annotations, normal WSOD frameworks are not able to perform regression tasks on the coordinates of prediction boxes so that almost all of them would directly output the top-scoring proposal.

However, inspired by OICR (Tang et al., 2017) that top-scoring proposals are regarded as the pseudo ground truth of next-level classifier, the top-scoring proposals produced by the last classifier can also be treated as the pseudo-ground-truth boxes to train a regression module.

Following the convention in (Girshick, 2015), the regression module only outputs the coordinate offsets $\mathbf{O}^P \in (-1, 1)^{N_p \times 4}$ of the original top-scoring proposal. The four elements of each row of $\mathbf{O}^P$ is $\mathbf{O}^P_{i,\cdot} = [dx, dy, dw, dh] \in (-1, 1)^{N_p \times 4}$ where

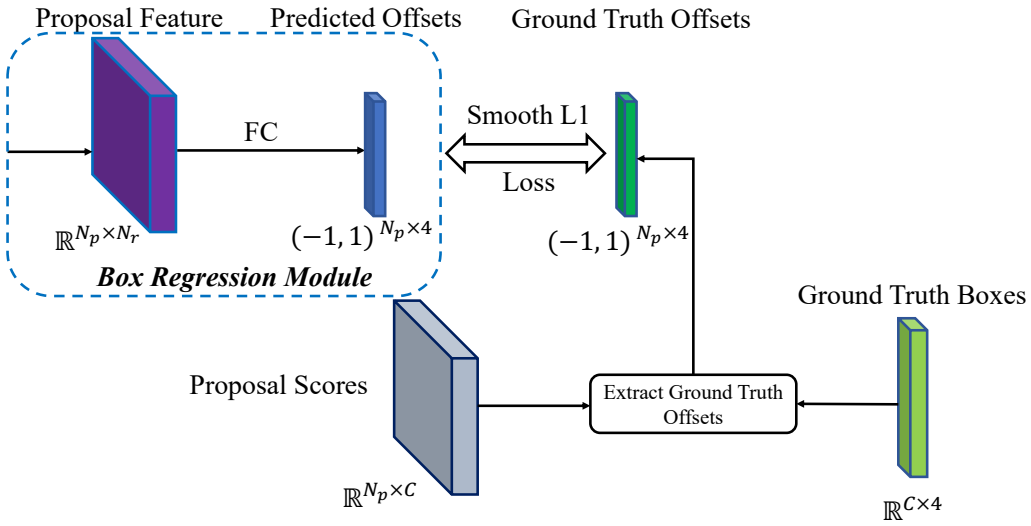

Figure 3.6: **Box Regression Module.** Pseudo-ground-truth offsets are selected from some top-scoring proposals. The smooth L_1 loss is calculated between the pseudo-ground-truth offsets and the predicted offsets produced by the box regression module.

$i \in [1, N_p]$.

As represented in Figure 3.6, the box regression module utilizes a fully connected layer and an activation function **tanh** to transform the proposal feature $\mathbf{X}^P$ into predicted offsets $\mathbf{O}^P$. The calculation can be described as

$$\mathbf{O}^P = \tanh(\mathbf{X}^P \cdot \mathbf{w}) \ \in \mathbb{R}^{N_p \times 4} \tag{3.17}$$

where $\mathbf{w} \in \mathbb{R}^{N_r \times 4}$ is the parameter of the fully-connected layer. For convenience, the bias is ignored here. The tanh function is defined as $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$.

The rectified proposals $\hat{\mathbf{P}}$ are

$$\hat{\mathbf{P}}_r = \begin{bmatrix} \mathbf{P}_{rw}\mathbf{O}^P_{rx} + \mathbf{P}_{rx} \\ \mathbf{P}_{rh}\mathbf{O}^P_{ry} + \mathbf{P}_{ry} \\ \mathbf{P}_{rw}\exp(\mathbf{O}^P_{rw}) \\ \mathbf{P}_{rh}\exp(\mathbf{O}^P_{rh}) \end{bmatrix}, \quad r \in [1, N_p] \tag{3.18}$$

where $\mathbf{P} \in \mathbb{R}^{N_p \times 4}$ denotes the original proposals, and x, y, w, h represents the indexes $1, 2, 3, 4$.

In order to perform learning in the box regression module, ground truth offsets need to be extracted from the proposal score and ground truth boxes. Based on OICR (Algorithm 1), the proposal label $\mathbf{V} \in 0, 1^{N_p \times (C+1)}$ is obtained and then transformed into a vector $\tilde{\mathbf{V}} \in [0, C]^{N_p}$ denoting the index of ground truth boxes. $\tilde{\mathbf{V}}_r = 0$ represents that proposal r does not overlap any ground truth boxes over the given threshold I_t. In addition, only the proposals that satisfy $\tilde{\mathbf{V}}_r \neq 0, \quad r \in [1, N_p]$ ought to be considered in backward propagation. The boolean mask of foreground proposal $\mathbf{M} \in [true, false]^{N_p}$ is computed by

$$\mathbf{M}_r = (\tilde{\mathbf{V}}_r \neq 0) \tag{3.19}$$

The ground truth offsets $\mathbf{O}^G \in (-1, 1)^{N_p \times 4}$ are defined as

$$\mathbf{O}^G_r = \begin{cases} \mathbf{G}_{\tilde{\mathbf{V}}_r} \ominus \mathbf{P}_r, & \tilde{\mathbf{V}}_r \neq 0 \\ \mathbf{O}^P_r, & \tilde{\mathbf{V}}_r = 0 \end{cases}, \quad r \in [1, N_p] \tag{3.20}$$

where $\mathbf{P} \in \mathbb{R}^{N_p \times 4}$ represents the coordinates of all original proposals, and opera-

tion $\ominus$ is defined as follows:

$$\mathbf{G}_{\tilde{\mathbf{V}}_r} \ominus \mathbf{P}_r = \begin{bmatrix} (\mathbf{G}_{\tilde{\mathbf{V}}_r,x} - \mathbf{P}_{rx})/\mathbf{P}_{rw} \\ (\mathbf{G}_{\tilde{\mathbf{V}}_r,y} - \mathbf{P}_{ry})/\mathbf{P}_{rh} \\ \ln(\mathbf{G}_{\tilde{\mathbf{V}}_r,w}/\mathbf{P}_{rw}) \\ \ln(\mathbf{G}_{\tilde{\mathbf{V}}_r,h}/\mathbf{P}_{rh}) \end{bmatrix} \tag{3.21}$$

For the loss function between $\mathbf{O}^G$ and $\mathbf{O}^P$, smooth L_1 function is used as a convention (Ren et al., 2015):

$$L_{reg} = \frac{1}{N_p} \sum_{r=1}^{N_p} \text{smooth}_{L_1}(||\mathbf{O}_r^G - \mathbf{O}_r^P||_1) \tag{3.22}$$

$||\cdot||_1$ is the 1-norm function.

The definition of smooth L_1 function is

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & |x| \leq 1 \\ |x| - 0.5, & |x| > 1 \end{cases} \tag{3.23}$$

3.7 Knowledge Distillation Module

In the multiple refinement modules, the supervision is only produced by the last classifier. For instance, the (k+1)-th refinement directly receives the output of the classifier of the k-th refinement module. However, numerous information could be lost in the process of multiple refinements. The (k+2)-th refinement module does not have the complete knowledge of the k-th one so that the supervisions may be noisy.

To prevent the loss of information during multiple refinements, (Zeni & Jung,

2020) proposed a distillation knowledge module to maintain more information till the last refinement process.

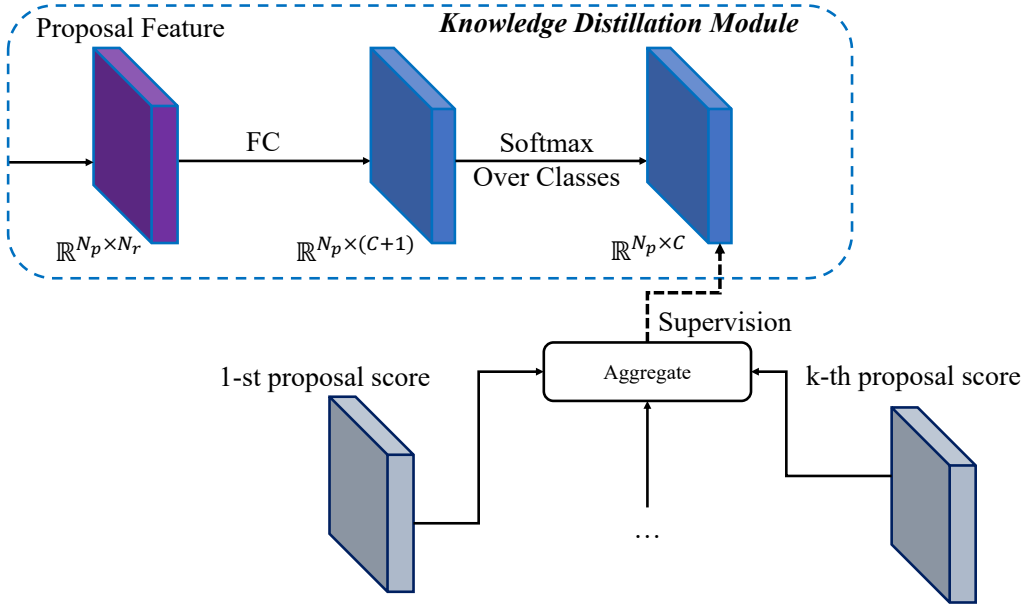

Figure 3.7: **Knowledge Distillation Module.** The knowledge distillation module receives the aggregation of the proposal scores generated by all refinement modules, and outputs a final proposal score.

As shown in Figure 3.7, the knowledge distillation module is very similar to the refinement module except for the supervision process. In OICR, the pseudo ground truth for (k+1)-th refinement is only extracted from the proposal score generated by the k-th classifier. In the knowledge distillation module, the pseudo ground truth comes from the aggregation of the proposal scores produced by all classifiers. The aggregation function AGG$(\dots)$ can be various, and in this project, the aggregated proposal score is the mean value of all proposal scores.

$$\text{AGG}(\mathbf{X}^{P1}, ..., \mathbf{X}^{PK}) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{X}^{Pk} \tag{3.24}$$

where K represents the number of refinements, and $\mathbf{X}^{Pk}$ denotes the proposal

score produced by the k-th refinement module.

The loss calculation of knowledge distillation module L_{distill} is the same as OICR loss computation (see Equation 3.10).

3.8 Loss function

Taking the framework architecture as a whole from end to end, the final loss function combines all individual loss functions including MIL, refinement, GAM and distillation modules.

$$L = L_{\text{MIL}} + L_{\text{GAM}} + \sum_{k=1}^{K} L_{\text{refine}}^{k} + L_{\text{distill}} \tag{3.25}$$

4 | Experiments

4.1 Datasets and Evaluation Metrics

The data set evaluated in this project is from Calipsa, and is referred to as "the Calipsa data set". Calipsa is a company aiming to address the problem of false CCTV alarms via AI technology. The Calipsa data set is produced by CCTV monitoring systems and contains 277887 pairs of images in the training set and 30714 pairs in the test set. The image pairs are temporally sparse and the time difference between consecutive frames is variable. Limited by the resources of computation, only 27000 pairs of images in the training set and 3000 pairs in the test set are used in this project. For the training set, only the image pairs and image-level annotations are included. The object-level annotations are neglected during the whole training. There are four classes that exist in the Calipsa data set: cyclist, person, car and truck. Due to the imbalance of these classes, especially the lack of cyclist objects, the moving cyclists are very difficult for the framework to detect. Therefore, the evaluation metric on cyclists is extremely low (close to 0).

For testing, Average Precision (AP, (Everingham & Winn, 2011)) is the main metric for evaluation on the test set. The AP metric strictly follows the standard PASCAL VOC protocol, and the threshold of IoU (the metric of overlap between two boxes, see Figure 4.1) between the predicted and ground-truth boxes is 0.5 which obeys the convention in object detection tasks. mAP represents the mean value of the AP of all classes.

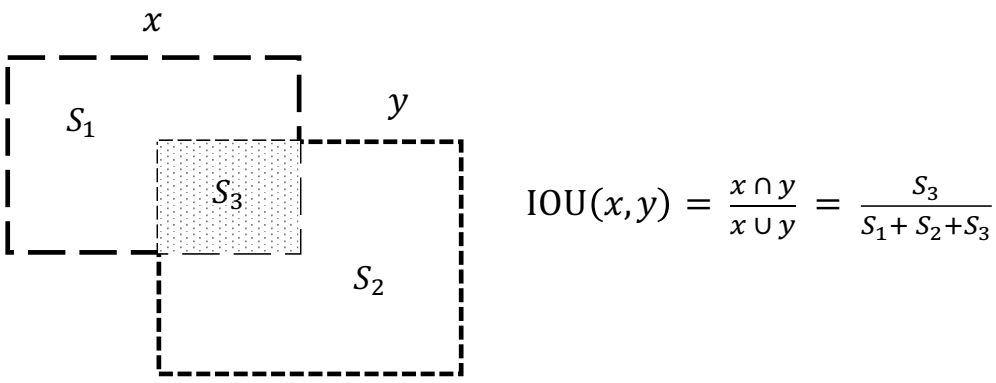

Figure 4.1: **Intersection over Union (IoU)** The IoU is a metric of the overlap between two boxes. The calculation process is shown above.

4.2 Implementation Details

The framework is built on the adapted VGG-16 (Figure 2.2) and the last two fully connected layers are moved after the RoI pooling layer (see Figure 3.1). The parameters of the backbone and the fully connected layers are initialized using the VGG-16 parameters pre-trained on ImageNet. Note that the number of channels is doubled in the backbone, so the parameters of the original VGG-16 are also copied and stacked along the frame channel dimension.

There are also plenty of newly added modules and layers in addition to the backbone in this framework. These newly added modules are initialized using Gaussian distributions with 0-mean and 0.01-deviations. All of these biases are initialized to 0. The learning rate is set to 5×10^{-5} for the mini-batch size 1 during the first 20,000 iterations, and then the learning rate is decreased to 1×10^{-5} for the remaining 20,000 iterations. The momentum and weight decay of the optimizer SGD (Bottou, 2010) are set to 0.9 and 0.0005 respectively.

For the proposal generation, there are two algorithms used in this project: Selective Search (SS) and Edge Box (EB). For both algorithms, 400 proposals are

produced for each image pair. In a single iteration of training, one pair of images will be randomly chosen from the training set and randomly resized to a scale in the set $\{480, 576, 688, 864, 1200\}$ (The short side is resized to one of the scales). In addition, the probability of horizontal flip of a pair is 0.5 during training.

During the test, for each pair of images, it will be resized to all five scales. And for each scaled pair, an additional flipped pair will also be generated. Therefore, 10 pairs of images will be produced for one single pair in an iteration of the test, so there will be 10 proposal scores in total. The mean of these proposal scores will be evaluated in the test.

The framework and all experiments are implemented based on the Pytorch deep learning framework (Paszke et al., 2019). All of the experiments are run on an NVIDIA Tesla T4 GPU.

4.3 Ablation Study

In this chapter, the experiments are conducted to validate the effectiveness of each module including refinement, GAM, box regression, knowledge distillation modules. Besides, hyper-parameters are tuned in order to find the best model.

The baseline framework in this project only contains the backbone and the MIL module, which is an adapted WSDDN (Bilen & Vedaldi, 2016).

Table 4.1 exhibits the results of frameworks combining different modules when the IoU threshold is set to 0.5 (conventional value). The baseline model can only achieve 2.3% mAP. With the help of three refinement modules, the performance attains 6.0% which proves the effectiveness of refinement. Based on the baseline and three refinement modules, three different techniques are being compared. GAM, which enables the framework to focus on the moving areas, helps to im-

Methods	car	truck	person	mAP
Baseline	9.1	0.0	0.0	2.3
Baseline + Refine	2.7	18.7	2.6	6.0
Baseline + Refine + GAM	11.7	18.8	12.6	10.7
Baseline + Refine + Dist	2.5	20.4	14.4	9.4
Baseline + Refine + Regression	10.2	27.1	8.0	11.4
Baseline + Refine + GAM + Reg	3.6	29.5	14.8	12.1
Baseline + Refine + GAM + Reg + Dist	14.0	23.7	19.2	**14.3**

Table 4.1: **Comparison of AP performance (%) on Calipsa test set (IoU threshold: 0.5).** The cyclist class is ignored because the cyclist APs of all frameworks are in the range $[0.0\%, 0.1\%]$. **Refine** denotes three refinements.

Methods	car	truck	person	mAP
Baseline	25.2	22.9	32.5	20.2
Baseline + Refine	27.7	37.3	50.5	**28.8**
Baseline + Refine + GAM	23.2	44.5	32.7	25.2
Baseline + Refine + Dist	26.0	39.9	36.9	25.8
Baseline + Refine + Regression	26.0	41.7	35.7	26.1
Baseline + Refine + GAM + Reg	24.8	45.3	33.1	26.0
Baseline + Refine + GAM + Reg + Dist	28.6	40.0	38.8	27.0

Table 4.2: **Comparison of AP performance (%) on Calipsa test set (IoU threshold: 0.01).** The cyclist class is ignored because the cyclist APs of all frameworks are in the range $[0.0\%, 0.6\%]$.**Refine** denotes three refinements.

prove mAP by 4.7% to 10.7%. The box regression module seems to be most effective that improves mAP by 5.4%. The knowledge distillation module helps to improve the performance as well, but the improvement of mAP by 3.4% is less than the other two combinations. This is reasonable that both the GAM and box regression module both bring new knowledge to the framework, whereas the knowledge distillation only maintains and distills the original information.

Considering the great effectiveness of GAM and box regression module, they are combined to evaluate the performance. However, the improvement of this combination is only 0.7% mAP (over GAM alone) which possibly means that GAM and box regression both provide similar information of moving areas in a redundant

fashion. After combining all modules, the final framework can attain 14.3% mAP which is a 12.0% improvement over the baseline.

Table 4.2 shows the performances when the IoU threshold is set to 0.01. Under this environment, as long as the prediction boxes overlap the ground-truth boxes a little, they will be considered as correct boxes. In such conditions, localization precision is almost ignored so that the performance represents the capability of detecting a moving object. From the results of Table 4.2, the framework that only has baseline and refinement modules can achieve the best mAP. All other added modules harm the performance. Besides, the performance gap between these frameworks is small except for the baseline.

Note that the cyclist class is ignored in Table 4.1 and 4.2 because of the extreme lack of cyclists in the training set of the Calipsa data set. Therefore, the AP for the cyclist is normally ranged from 0.0% to 0.1% when the IoU threshold equals 0.5.

Figure 4.2, 4.3 and 4.4 show some of the predictions of the framework combining all modules. The first two columns are the input of the framework. The third-column image contains the prediction boxes and ground truths. The green box represents the ground truth and the red box represents the prediction.

Despite the multiple refinements and box regression, the framework still tends to focus on the characteristic part especially when the object accounts for a large proportion in the figure such as the first two rows in Figure 4.3. The second problem is that the framework sometimes outputs the boxes only containing the moving objects in the second image (the second column). For example, in the first row in Figure 4.2, two men are moving and one moving man leaves in the second image so that the framework can only detect the man who is present in both images. Besides, the confidence value given by the framework is not high.

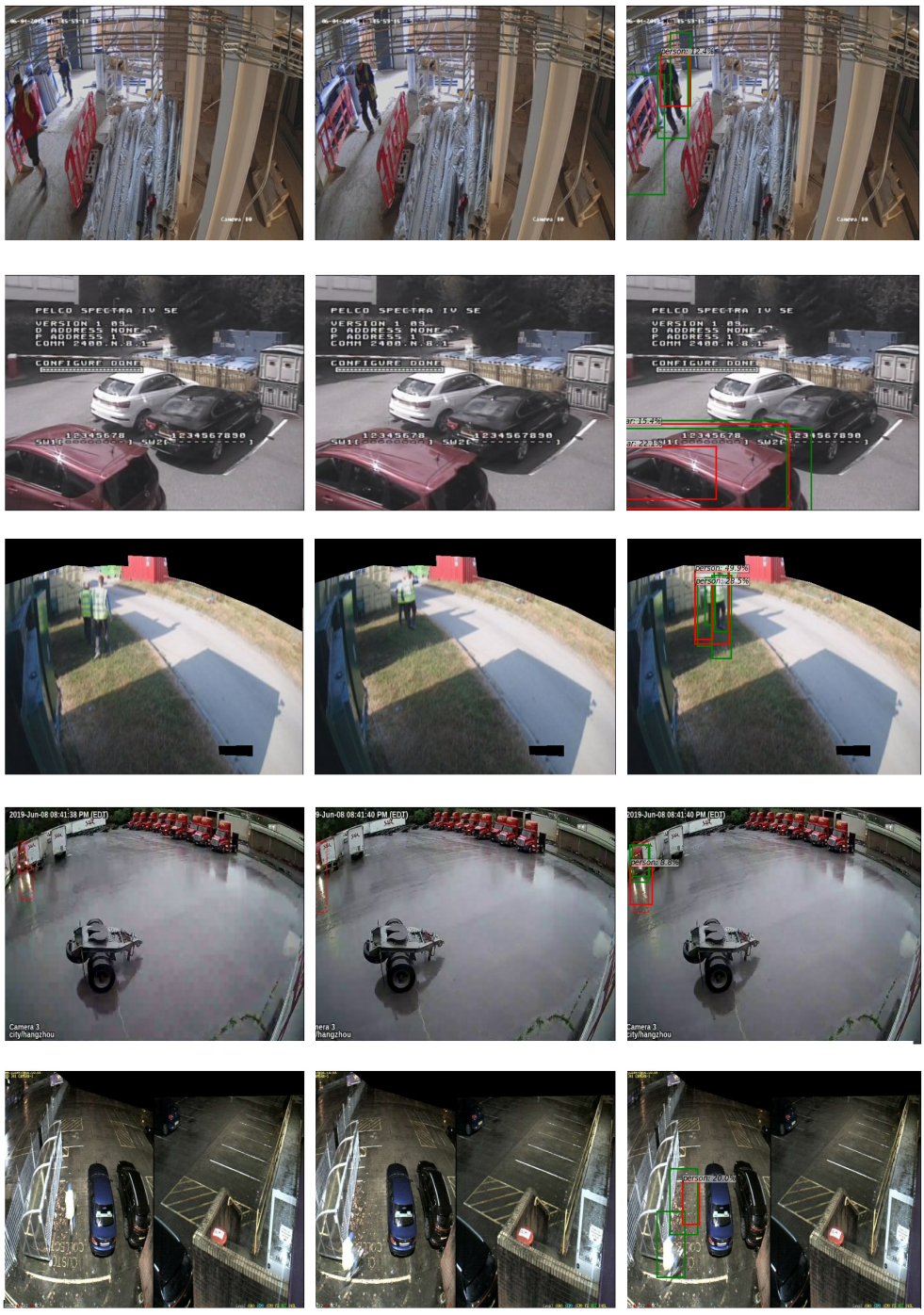

Figure 4.2: **Predictions of the framework. (confidence score >** 0.5%**)** Red boxes are predictions and Green ones are ground truths.

Figure 4.3: **Predictions of the framework. (confidence score >** 0.5%**)** Red boxes are predictions and Green ones are ground truths.

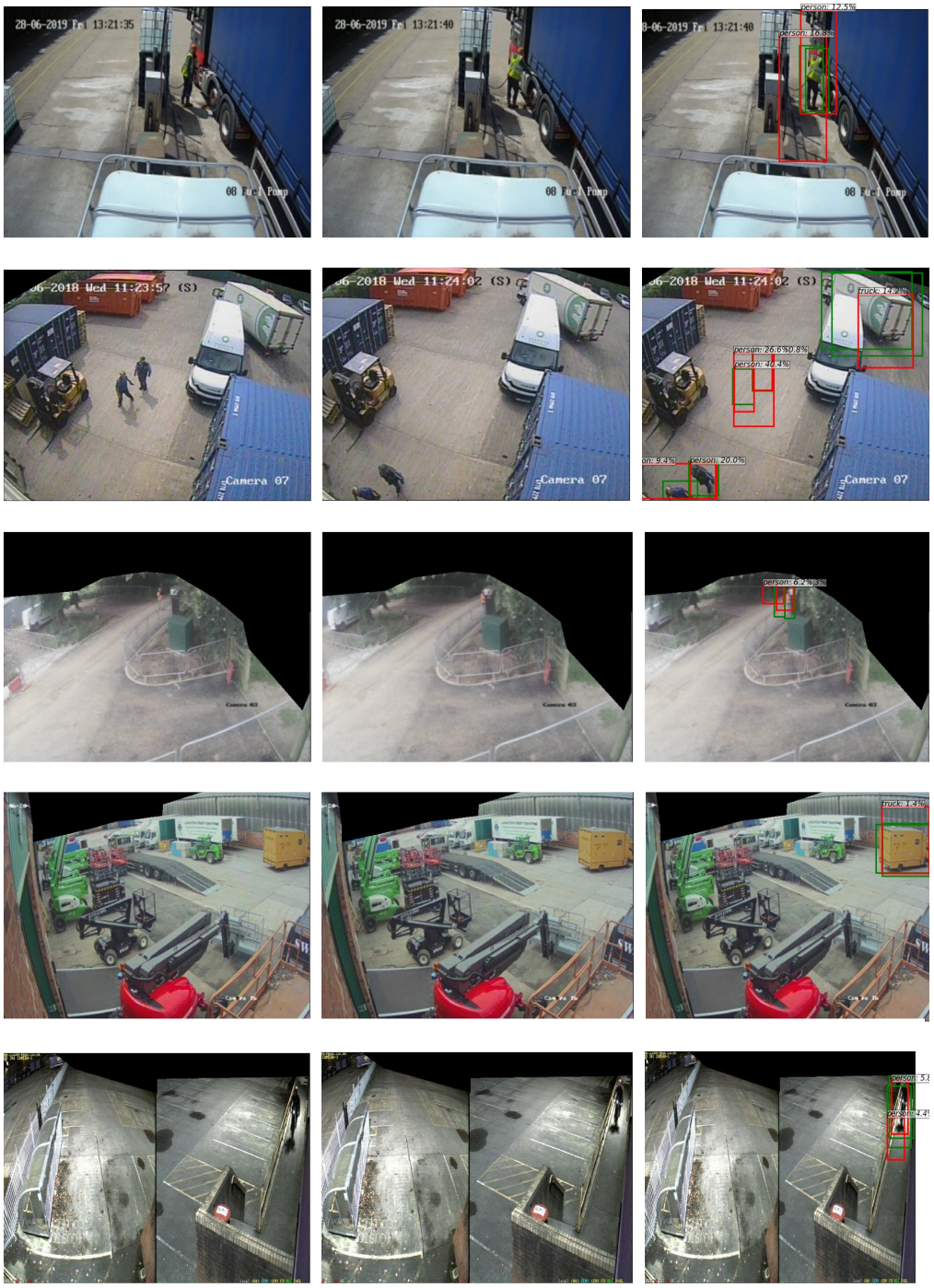

Figure 4.4: **Predictions of the framework. (confidence score >** 0.5%**)** Red boxes are predictions and Green ones are ground truths.

The framework has very little confidence in some good prediction boxes such as the last row of Figure 4.3 only showing 1.8% confidence.

4.3.1 Hyper-parameters of the Refinement Module

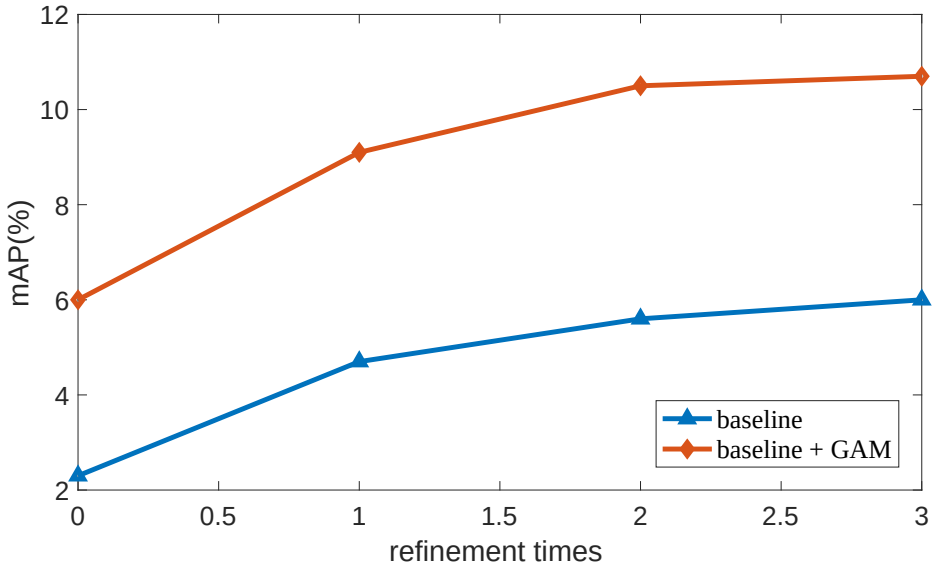

Figure 4.5: **Results on Calipsa data set for different number of refinements.** The blue line indicates the different number of refinements added on the baseline framework, and the orange one represents the refinements added on the baseline combining the GAM module.

In this subsection, several experiments are conducted to see the effectiveness of different hyper-parameters of refinements.

Consider the number of refinements (see Figure 4.5). The results show that only one refinement added can improve the mAP by about 3%. Multiple times of classifier refinement can achieve more improvements, but the performance gained is getting smaller gradually. Considering the cost of refinement, three refinements are enough for the framework. More than three refinements may only improve mAP by 0.1%. (extrapolating the plot in Figure 4.5).

While performing the refinement, the module will regard the proposals, which

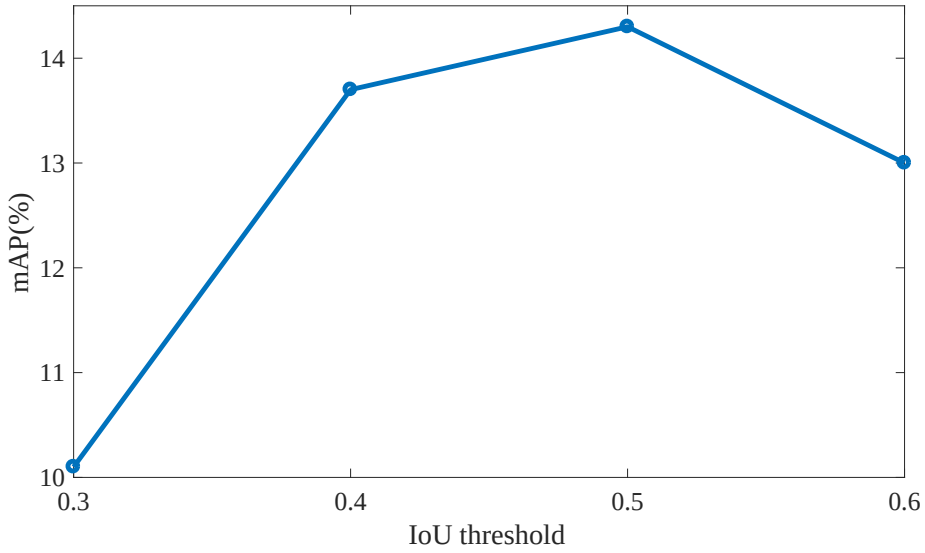

Figure 4.6: **Results on Calipsa data set for different IoU threshold of refinement modules.** The framework contains all modules and there are three refinement modules in the framework.

highly overlap the top-scoring proposal, as the foreground object of the same class. The metric **IoU** is utilized to measure the overlap between two proposals and the threshold of IoU is critical to the performance of a refinement module. Based on the results in Figure 4.6, 3-time refinements with 0.5 IoU threshold can help achieve the best performance 14.3%. When the IoU threshold is set to 0.4 or 0.6, the mAP only drops a little (mAP ranges from 13.7% to 14.3%).

4.3.2 Moving Attention Branch

As described in Section 3.5, the GAM module functions as an attention module and enables the framework to focus on the moving part in the images. However, the input of the GAM module is in the only output of the backbone which does not contain any prior knowledge. In addition, the GAM module needs an isolated backward process which is expensive in computation.

Therefore, in this section, three different structures of the Moving Attention Branch

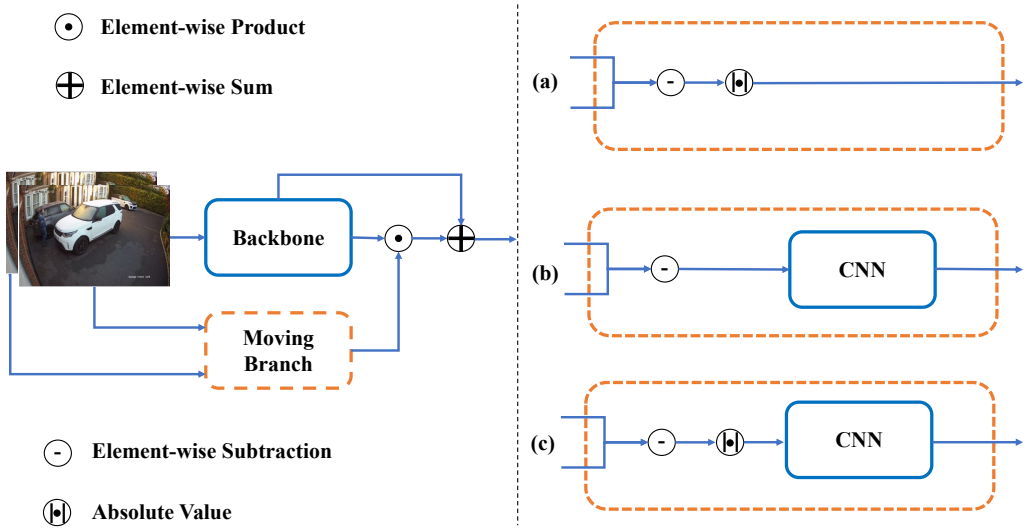

Figure 4.7: **Different Designs of Moving Attention Branch.** Each of the three options (a-c) is based on taking the pixel-level difference between the two images.

are proposed. The concrete structures are proposed as alternatives to the GAM module. The concrete structures of the three options are shown diagrammatically in Figure 4.7. Branch (a) only consists of an element-wise subtraction and an absolute operation. Branch (b) contains an element-wise subtraction and a following convolutional layer. Branch (c) is composed of an element-wise subtraction, an absolute operation and a convolutional layer. The designs are inspired by the Temporal Differencing 2.3.1 method used in moving object detection.

Methods	car	truck	person	mAP
Baseline + Refine	2.7	18.7	2.6	6.0
Baseline + Refine + Branch a	4.8	14.2	15.1	8.6
Baseline + Refine + Branch b	0.6	3.6	1.1	1.4
Baseline + Refine + Branch c	3.5	16.3	16.5	9.1
Baseline + Refine + GAM	11.7	18.8	12.6	**10.7**

Table 4.3: **Comparison of AP performance (%) on Calipsa test set (IoU threshold: 0.5).** The cyclist class is ignored because the cyclist APs of all frameworks are in the range $[0.0\%, 0.6\%]$. **Refine** denotes three refinements.

In Table 4.3, all branches (a), (b) and (c) can bring mAP improvement to Base-

Methods	car	truck	person	mAP
Baseline + Refine	27.7	37.3	50.5	28.8
Baseline + Refine + Branch a	27.5	30.6	36.6	23.7
Baseline + Refine + Branch b	33.9	38.0	34.1	**33.4**
Baseline + Refine + Branch c	21.7	33.0	34.8	22.4
Baseline + Refine + GAM	23.2	44.5	32.7	25.2

Table 4.4: **Comparison of AP performance (%) on Calipsa test set (IoU threshold: 0.01).** The cyclist class is ignored because the cyclist APs of all frameworks are in the range $[0.0\%, 0.6\%]$ except for the framework with Branch (b) (27.5%). **Refine** denotes three refinements.

line+Refine framework, but the improvements are all smaller than the GAM module. For the framework (Baseline + Refine + Branch b), due to the absence of an absolute value operation, the attention map may contain negative values which have a large impact on the precision of localization. Therefore the mAP performance of this framework is very low (1.4%). Branch (c) applies an additional convolutional layer to Branch (a) thus obtaining greater performance. Considering no additional backward propagation and loss computation is added, the improvement 3.1% demonstrates the effectiveness of Branch (c).

Table 4.4 shows the performance of different frameworks with IoU threshold 0.01, Branch (b) brings the greatest improvement among the three branches and achieves 33.4% which also exceeds the original framework (Baseline + Refine). However, the high performance does not necessarily exhibit the capability to detect moving objects of Branch (b) because the framework may output the prediction boxes that almost cover a large proportion of the whole image.

Consider that the only difference between Branch b and c is the absolute operation. The reasons of such poor performance with IoU threshold 0.5 (1.4%) may be: (1) The absolute operation increases the difficulty of parameter learning in the moving branch because the absolute function is undifferentiable at 0. (2) The

attention map output by the Branch b may be activated around the target object (object contour) which makes almost all nearby proposals get a higher confidence score. It is difficult for the framework to find the most accurate proposal. Thus the performance with IoU threshold 0.5 is extremely low.

4.4 CAM visualization

In Figure 4.8 and 4.9, a few interesting class activation maps (CAM) are shown. The generation of these CAMs is slightly different from (Zhou et al., 2016). Considering the channel size of the confidence map (Figure 3.5) exactly equals the number of classes, each channel of the confidence map represents an activation map of a class. The order of classes is the same as the ground-truth label because of the way that L_{GAM} is defined. Dark areas are not activated. The lighter parts represent the more activated areas.

The first two columns (Figure 4.8 and 4.9) are the input of the framework. The third column is the activation map of the most activated class (the activation map which contains the maximally activated pixel). The class names are written on the top-left corner of the third-column images.

All these class activation maps clearly demonstrate that the GAM module is capable of identifying different classes of objects. Furthermore, it is able to detect moving objects from static ones. For example, in the second row of Figure 4.8, the moving car in the bottom is significantly more activated than the stationary cars. Also in the fourth row of Figure 4.8, there are ten trucks that exist in the image but only the moving truck on the left of the image is activated.

The GAM module tends to identify the class via color or texture instead of shape. In the last row of Figure 4.8, the man in the images is mistakenly identified as

a truck. The reasons may be: (1) The orange color of the coat is similar to the common color of trucks. (2) There are very few training images that contain a man from the top view.

Despite the high similarity between trucks and cars, the GAM module is able to differentiate both classes very well. For instance, in the fifth row of Figure 4.9, only the area containing the moving truck is activated whereas the areas of all other small cars are dark. Similar results can be seen in the other rows in Figure 4.9.

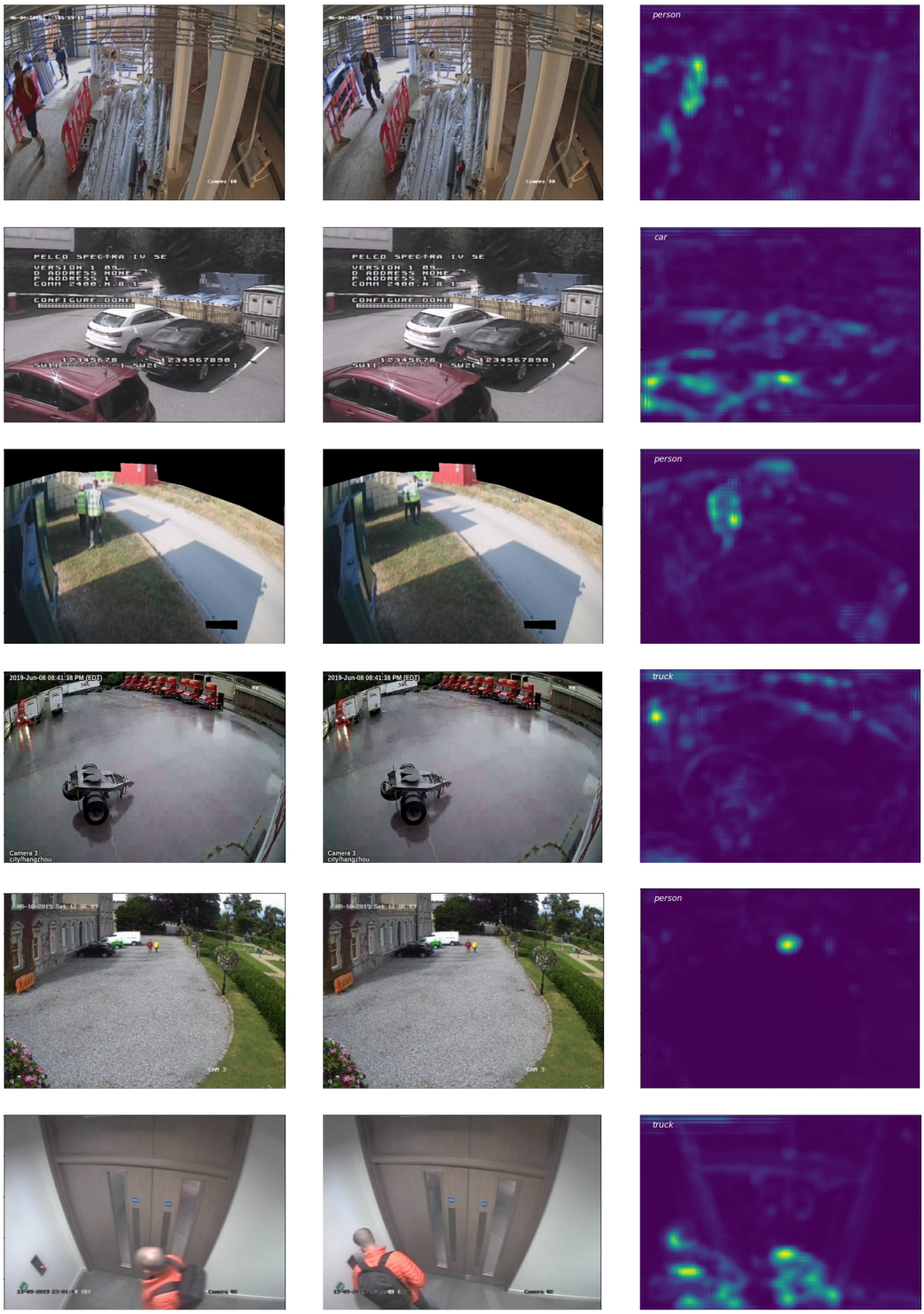

Figure 4.8: **Visualization of the class activation maps (CAM).** Note that each map is of the most activated class (see labels) and that lighter areas are more activated. The man in the last row is identified as a truck due to the orange coat.

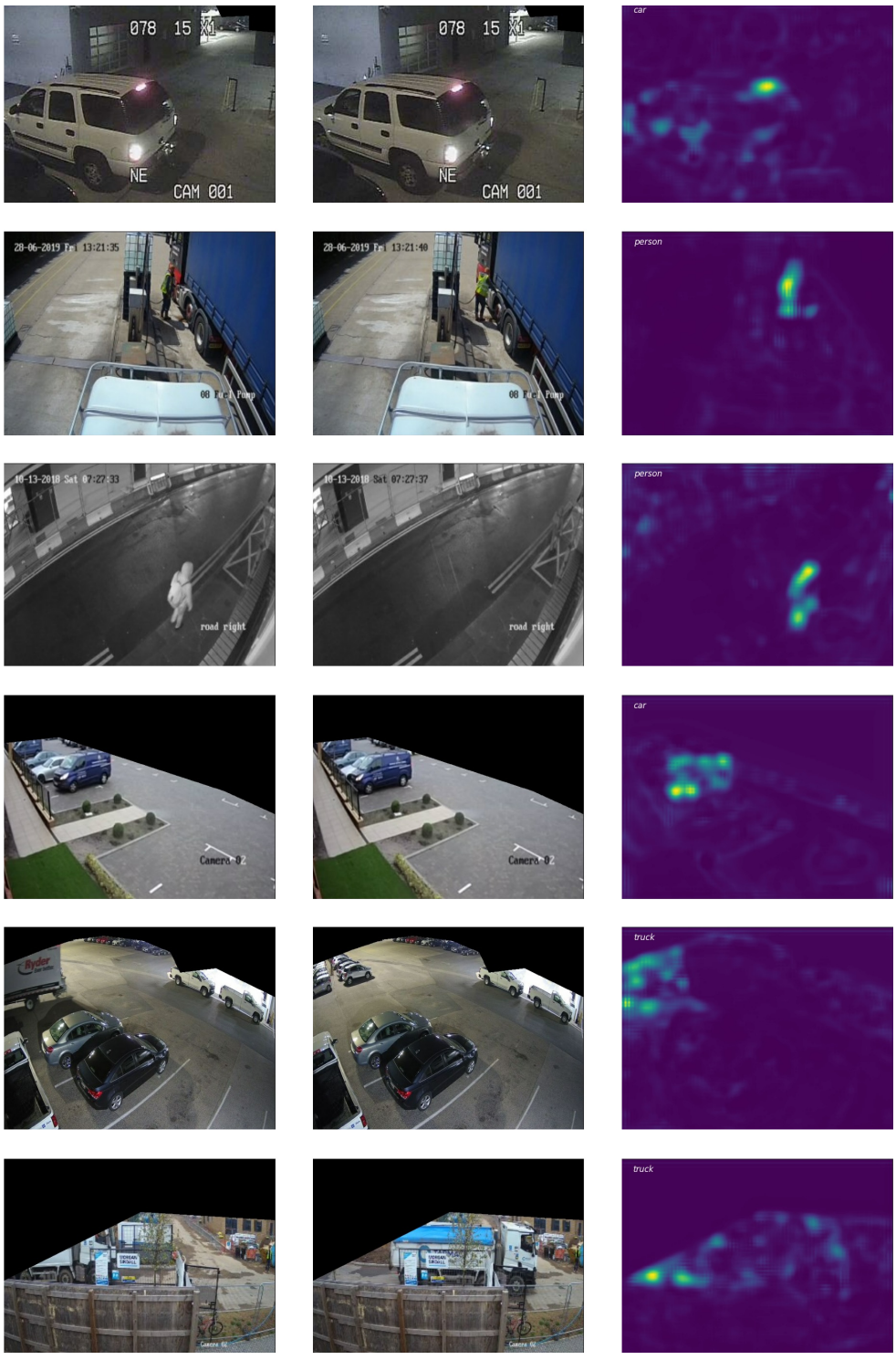

Figure 4.9: **Visualization of the class activation maps (CAM).** Note that each map is of the most activated class (see labels) and that lighter areas are more activated. Trucks and cars are successfully differentiated in these CAMS.

5 | Conclusion

In this dissertation, we go through the related literature in moving object detection and present a deep learning based framework for weakly supervised moving object detection. The framework is constructed based on OICR (Tang et al., 2017) with the addition of other effective modules to improve the mAP performance on the Calipsa data set. The framework contains several Refinement Modules, a Guided Attention Module (GAM), a Box Regression Module and a Knowledge Distillation Module.

The experiments in chapter four demonstrate that the proposed framework obtains substantial AP improvements compared with the baseline. Considering that the huge training set (27,000 pairs of images) is fed into the framework and the frozen backbone is pre-trained on the Imagenet data set, the proposed framework is anticipated to generalize well to other data sets (only need to fine-tune).

The whole project took me about five months to finish all the works including literature reading, coding, doing experiments and writing the dissertation. The source code of the whole project is constructed based on the GitHub repository (wsddn.pyrtorch (Dursun, 2019)). In my final project, there are 3682 lines of python source code in total except for 937 lines of comments and 380 lines of configuration. For the new modules or processes presented in this dissertation, more than 2000 lines of codes in the project are completely written by me. In addition to the brand new modules, I also adapted or changed more than 500 lines of codes such as the scripts for training, evaluation or information printing. During these five months, there are 130 commits recorded by the git tool. As introduced in Section 4.2, all experiments take 40000 iterations during training, and every 20 iterations (as a batch) take 45 seconds to finish on an individual GPU. Therefore, it

costs 2 days approximately to train a single framework (when the loss converges). Limited by the computation resources (one NVIDIA Tesla T4 GPU), I have only done 50 to 60 experiments. A great number of experimental results are not shown in this dissertation because of the poor performances or numerical issues.

From this project, I learnt that real-world deep learning tasks are more difficult that those I confronted in the machine learning course.

1. The sizes of data set in the industry are much greater than standard data sets (always more than ten times). For instance, the Calipsa data set is 30 times bigger than the VOC2007.

2. The problems in the industry are more complicated than the standard academic problems. The proposed framework should only output moving targets, whereas stationary objects should be ignored.

3. The quality of images and annotations may be worse than standard data sets. In the VOC2007 data set, the target objects always account for a large area in the images (more than 40% areas), whereas some of the target objects in the Calipsa data set only account for less than 5% of the area of the image. Most target objects in the VOC2007 data set are very clear while some of the target objects in the Calipsa data set are blurred.

4. Class imbalance is a serious problem in the Calipsa data set while the number of target objects in each class is roughly the same in the VOC2007 data set. Therefore, the AP performance of the proposed framework on the cyclist class is close to 0. If we ignore the cyclist class and only train the objects of the other three classes in this project, the final performance should be greater than or equal to 19.1% mAP.

5. In conference papers, authors do not apply dozens of techniques to aug-

ment the data set for two reasons: (1) Too many data augmentation methods may retard the convergence of a model which costs time and computation resources. (2) Researchers tend to focus on the effectiveness of a module instead of the performance of the complete framework, so they only utilize the normal augmentation methods that everyone uses. However, in real-world problems, data augmentation is of great importance to improve the performance and generalization ability of a framework (with numerous computation resources). In this project, limited by time and resources, the images are only resized and flipped as data augmentation methods.

The best performance of the proposed framework (14.3%) does not attain my initial expectation. For the pure object detection task on the VOC2007 data set, my OICR implementation achieves 35% mAP without fine-tuning (41% presented in the original paper (Tang et al., 2017)). Considering the challenging data set (low resolution and quality) and the more difficult task (moving object detection), the mAP performance is expected to attain 20% mAP without fine-tuning. Despite that the performance is lower than expected, the proposed framework is able to accurately detect some of the moving objects in the known classes from the visualizations of the chapter four (see Figures 4.2, 4.3 and 4.4). In the industry, this proposed framework may be utilized to generate coarse object-level annotations for moving object detection tasks, and the accurate annotations can be filtered out manually to train a fully supervised framework.

Meanwhile, this proposed framework has a lot of room for improvements which are elaborated in the next section (Section 5.1).

5.1 Future Work

Chapter four shows some visualizations of the prediction boxes and class activation maps output by the framework which implies a few potential defects of it. These defects indicate some areas which should be addressed in future work:

1. During training, the IoU threshold I_t in the refinement module is set to 0.5 consistently. However, in the initial phase of training, the top-scoring proposal is small due to the local minimum problem so that the overlaps between the top-scoring proposal and almost all other ones are smaller than 0.5. In other words, during the initial phase of training, the refinements are useless. A proposed solution to this is that the IoU threshold can be dynamically adjusted from a low to a high value as iterations progress during training.

2. In the refinement process, only one pseudo-ground-truth box (top-scoring proposal) is selected for each class, whereas multiple objects of this class may appear in a given image pair. More than one proposal should be chosen as pseudo-ground-truth boxes according to a given score threshold. For example, if the top four scores of a class are 0.4, 0.3, 0.2, 0.05 and the score threshold is set to 0.1, then there will be three proposals selected as pseudo-ground-truth boxes.

3. Considering the difference between the VOC2007 data set and the Calipsa data set, the backbone could be pre-trained on the Calipsa set for a classification task.

4. More data augmentation methods can be applied, such as random crop and random rotation, in order to improve the generalization ability of the framework.

Bibliography

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Alexe, B., Deselaers, T., & Ferrari, V. (2010). What is an object? In *2010 ieee computer society conference on computer vision and pattern recognition* (pp. 73–80).

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481–2495.

Beauchemin, S. S., & Barron, J. L. (1995). The computation of optical flow. *ACM computing surveys (CSUR)*, *27*(3), 433–466.

Benezeth, Y., Jodoin, P.-M., Emile, B., Laurent, H., & Rosenberger, C. (2008). Review and evaluation of commonly-implemented background subtraction algorithms. In *2008 19th international conference on pattern recognition* (pp. 1–4).

Bilen, H., & Vedaldi, A. (2016). Weakly supervised deep detection networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2846–2854).

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat'2010* (pp. 177–186). Springer.

Chen, Y., Wang, J., Zhu, B., Tang, M., & Lu, H. (2017). Pixelwise deep sequence learning for moving object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, *29*(9), 2567–2579.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Diba, A., Sharma, V., Pazandeh, A., Pirsiavash, H., & Van Gool, L. (2017). Weakly supervised cascaded convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 914–922).

Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, *89*(1-2), 31–71.

Dursun, A. (2019). *wsddn.pytorch.* https://github.com/adursun/wsddn.pytorch. GitHub.

Everingham, M., & Winn, J. (2011). The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, *8*, 5.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, *59*(2), 167–181.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 1440–1448).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, *17*(1-3), 185–203.

Kanan, C., & Cottrell, G. W. (2012). Color-to-grayscale: does the method matter in image recognition? *PloS one*, *7*(1), e29740.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Liu, Y., Ai, H., & Xu, G.-y. (2001). Moving object detection and tracking based on background subtraction. In *Object detection, classification, and tracking technologies* (Vol. 4554, pp. 62–66).

Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision..

Mandal, M., Kumar, L. K., Saran, M. S., et al. (2020). Motionrec: A unified deep framework for moving object recognition. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 2734–2743).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*, 8026–8037.

Patil, P. W., & Murala, S. (2018). Msfgnet: A novel compact end-to-end deep network for moving object detection. *IEEE Transactions on Intelligent Transportation Systems*, *20*(11), 4066–4077.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*, 91–99.

Shen, Y., Ji, R., Wang, Y., Chen, Z., Zheng, F., Huang, F., & Wu, Y. (2020). Enabling deep residual networks for weakly supervised object detection. In *European conference on computer vision* (pp. 118–136).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 ieee computer society conference on computer vision and pattern recognition (cat. no pr00149)* (Vol. 2, pp. 246–252).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich,

A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Tang, P., Wang, X., Bai, S., Shen, W., Bai, X., Liu, W., & Yuille, A. (2018). Pcl: Proposal cluster learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, *42*(1), 176–191.

Tang, P., Wang, X., Bai, X., & Liu, W. (2017). Multiple instance detection network with online instance classifier refinement. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2843–2851).

Tang, P., Wang, X., Wang, A., Yan, Y., Liu, W., Huang, J., & Yuille, A. (2018). Weakly supervised region proposal network and object detection. In *Proceedings of the european conference on computer vision (eccv)* (pp. 352–368).

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154–171.

Wren, C. R., Azarbayejani, A., Darrell, T., & Pentland, A. P. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on pattern analysis and machine intelligence*, *19*(7), 780–785.

Yang, K., Li, D., & Dou, Y. (2019). Towards precise end-to-end weakly supervised object detection network. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 8372–8381).

Ye, D. H., Li, J., Chen, Q., Wachs, J., & Bouman, C. (2018). Deep learning for moving object detection and tracking from a single camera in unmanned aerial vehicles (uavs). *Electronic Imaging*, *2018*(10), 466–1.

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zeng, Z., Liu, B., Fu, J., Chao, H., & Zhang, L. (2019). Wsod2: Learning

bottom-up and top-down objectness distillation for weakly-supervised object detection. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 8292–8300).

Zeni, L. F., & Jung, C. R. (2020). Distilling knowledge from refinement in multiple instance detection networks. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition workshops* (pp. 768–769).

Zhang, Y., Li, X., Zhang, Z., Wu, F., & Zhao, L. (2015). Deep learning driven blockwise moving object detection with binary scene modeling. *Neurocomputing*, *168*, 454–463.

Zhou, B., Jagadeesh, V., & Piramuthu, R. (2015). Conceptlearner: Discovering visual concepts from weakly labeled image collections. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1492–1500).

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2921–2929).

Zhu, H., Yan, X., Tang, H., Chang, Y., Li, B., & Yuan, X. (2020). Moving object detection with deep cnns. *Ieee Access*, *8*, 29729–29741.

Zitnick, C. L., & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *European conference on computer vision* (pp. 391–405).