Tampere University

Yan Feng

MULTIPLE OBJECT TRACKING

Supervisor: Prof. Esa Rahtu
Examiner: Prof. Joni Kämäräinen
Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
Mar 2020

ABSTRACT

Yan Feng: Multiple Object Tracking
M.Sc. Thesis
Tampere University
Master's Degree Programme in Data Engineering and Machine Learning
Mar 2020

Multiple object tracking, a middle-level task, is a critical foundation to support advanced research activities, like pose analysis or motion recognition. In this thesis, the relationship between object detection, single-object tracking, and multiple object tracking was explored and discussed.

On this basis, the Single Shot MultiBox Detector (SSD) [27], SiamMask [26] network, and Detect and Track (D&T) model [22] have been utilized, modified and evaluated.

D&T model is the offline Detection Based Tracking (DBT) network. We had observed the benefits of this correlation loss application via researching on D&T model, and we trained the D&T network on the dataset combination containing the person objects so as to make it useful in reality. In addition, SSD had been applied as the detector with the same tracking methods, during which process the effect of the diverse detectors on the MOT experiments could be figured out. The last experiment was executed on the Single Object Tracking (SOT) model-SiamMask. The original SiamMask network is an online Detection Free Tracking (DFT) network. In order to adapt to the situation of multi-target tracking, it had been modified to initialize multiple target objects with an SSD detector at every specific interval.

Having prepared all the multiple object tracking models, we carried out the evaluation with the MOT17DET dataset. The evaluation metrics for multiple object tracking provided us with a standard view of their performance. In this procedure, we also obtained some helpful knowledge and experience for future MOT improvement.

Keywords: multiple object tracking, single shot multibox detector, SiamMask, detect and track.
The originality of this thesis has been checked using the Turnitin Originality Check service.

Contents

1. Introduction 1

1.2. Motivation 3

1.3. Summary 3

2. Detection and Tracking 5

2.1. Object Detection 5

2.1.1. One-stage algorithm 7

2.1.2. Two-stage algorithm 8

2.2. Object tracking 12

3. Multiple-Object Tracking Methodologies 19

3.1. MOT Categorization 19

3.2. Detect to Track and Track to Detect 21

3.3. PyTorch-SSD -to-Object-Detection 24

3.4. Fast Online Object Tracking and Segmentation: A Unifying Approach 27

4. Experiments 31

4.1. Dataset 31

4.1.1. Standard dataset 31

4.1.2. Dataset Combination 35

4.2. Implementation 38

4.2.1. Detect to Track and Track to Detect 38

4.2.2. SSD to Object Detection 39

4.2.3. Fast Online Object Tracking and Segmentation 40

4.3. Evaluation 41

4.3.1. Evaluation dataset 41

4.3.2. Evaluation metric 42

4.3.3. Testing results 42

4.3.3.1. The first evaluation 42

4.3.3.2. The second evaluation 43

4.3.3.3. Further analysis **48**

4.4. Discussion **52**

5. Conclusion and future work **54**

5.1. Conclusion **54**

5.2. Future directions **54**

References **56**

1. Introduction

1.1. Overview

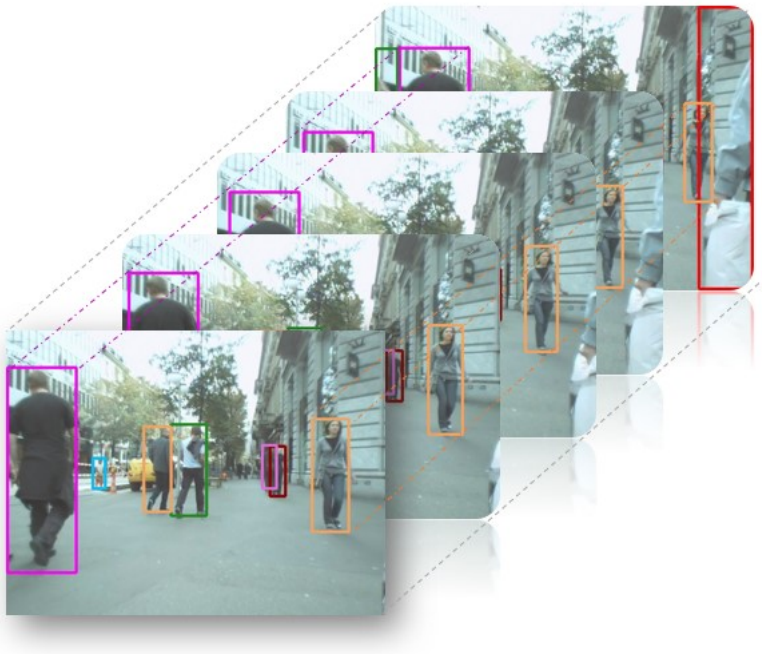

Figure 1. The ground-truth MOT result from MOT17DET [21]. They are selected as the five subsequent frames and organized following the time stamp.

Multiple Object Tracking (MOT) is also called Multiple Target Tracking (MTT). In general, MOT can be considered as the estimation of various variables. From its name, we can infer that its main task is to locate and identify moving objects in a frame sequence and then output trajectories for diverse purposes. In [12], the author mentioned that multiple targets could be explained as objects belonging to distinct categories. In other cases, numerous objects can state the different parts of a single object. We tended to interpret multiple objects as divergent class objects in this thesis.

As the medium level of computer vision, MOT supports advanced tasks such as pose recognition, motion analysis, etc. Referring to the application in real life, the main directions of MOT could cover our usual activities and ensure our security, comfortability, health, and convenience. For example, video monitoring could adopt MOT to detect strange actions. Hence, it contributed to saving a large amount of labor and property. In addition, MOT could be implemented to identify and deal with object interaction in complex scenes. Furthermore, AR and VR would be equipped with MOT to add more details,

such as the in-game character action settings. Most importantly, MOT could enhance the medical image.

Distinct from Single Object Tracking (SOT), MOT is facing more and more hardships. Besides the SOT issues, such as illumination, deformation, occlusion, and so on, MOT has to cope with the data association for the target objects. And MOT might deal with diverse problems, like the frequent object occlusion, the unknown beginning and ending time for trajectories, the different motions, the similar object appearance, the interaction among objects, etc.

For an MOT model, the general input is the detection response generated by the detector. Detection responses are referred to as detection hypothesis or detection observation describing the position and confidence of the target object in each frame. Most MOT models adopt the data association methods to match the target detection response or tracklets and then output the trajectories. Several tracklets belonging to the same object contributed to one trajectory, which corresponded to the movement sequence of this object during one time period.

Having got the basic concept of MOT tracking, we would dive into more specific categories in this field. In Chapter 3, partitioning standards are described: initialization, processing format, and output type. Detection-Based Tracking (DBT) and Detection-Free Tracking (DFT) are identified through the initializing approaches like Figure 2. As one DFT model, the SiamMask network [26] states the presentative sample as [40]. After manually initializing the bounding box of the target object in the first frame, the DFT model would detect and locate in the subsequent frames following the plotted boxes. In this thesis, we mainly researched and developed DBT models, presenting the corresponding video [41]. This type of network at first detected the targeted and then linked these detection responses into trajectories according to either on object features or probabilistic movement characteristics. We selected these models since they could deal with the new existence of the target objects in relatively long-term videos.

Processing mode would also decide the MOT categories. The difference between online tracking and offline tracking lies in whether the target observation for the next few frames is adopted when processing the current frame. Considering the processing format, the Detect and Track (D&T) model [22] and modified Single Shot MultiBox Detector (SSD) [27] method are the offline networks while the fast-online object tracking and segmentation (SiamMask) [26] executes the frames online.

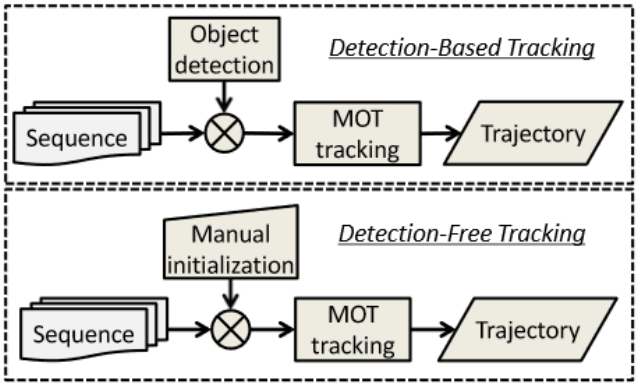

Figure 2. Procedure flow of two prominent tracking methods [12]. Top: Detection-Based Tracking (DBT), bottom: Detection-Free Tracking (DFT).

1.2. Motivation

From the previous part, it is easy to infer that the MOT model relies on the detector since it needs the detection response as the input. Therefore, the target objects used in the detector would limit the selection of tracking objects.

Considering the target object in MOT, the pedestrian is the significant object that shall be involved in our experiments. Firstly, it is typically a non-rigid object and an ideal example for MOT compared with other items. Secondly, the number of videos involved pedestrians exceeds other videos. It means the relevant applications would be more accessible and profitable than others. Finally, at least seventy percent of MOT research aims at pedestrians [12]. In short, tracing object classes and even detecting object classes should be made up of the person class.

Since the detector applied in MOT would affect the MOT output, changing the detection observation by replacing the detector could help us to compare and analyze the factors determining the performance of the MOT network.

In the overview, we discussed the application, requirements, and categories of MOT. Despite the high speed, some typical SOT models can achieve good results. It inspires us to modify the SOT model to accept multiple objects. Such that we can avoid designing the MOT network from the beginning and adopt the advantages of the SOT model.

Based on these three purposes, the corresponding experiments had been organized and conducted in chapter 4.

1.3. Summary

As has been described in the last section, we aim at building the MOT models which detect and track multiple common objects. To this end, our contribution is threefold.

Firstly, we modified the MOT model - Detect and Track (D&T) [22]. One of the drawbacks of D&T is that its tracking object classes do not involve pedestrians. Therefore, we reorganized the training dataset to add new converted videos and images about the person. After the training process, the D&T model could be utilized to detect and track the diverse class objects included people. Prior to retraining the D&T model on the MOT17DET dataset, it scores the mAP of 0.5765, with the baseline result reaching 0.687. This gap led by the hardware limitation and the large imbalanced training dataset combination.

Secondly, it is found that tracking outputs fail to satisfy our requirements. According to the analysis from [12], the detection response would be the core factor when we refer to DBT networks. At this prompt, the detector in the D&T network was changed from the R-FCN model [13] to the SSD model [27]. The tracking part is kept the same for comparison. Through experiments and ablation studies, we found that the SSD tracking model performed better in observing large-scale test results.

Lastly, the SiamMask network [26] had been modified for tracking multiple objects without plotting manually in the first frame. Its initialization part is responsible for the SSD model. Therefore, the performance of the SSD detector would influence the modified SiamMask model.

According to this sequence, the next section mainly explains object detection and tracking. The third chapter focuses on the description of the MOT categories, D&T network, SSD model, and SiamMask architecture. The experiments section details the processing steps as well as the evaluation. Specifically, the processes introduce how we prepared the new dataset, applied the SSD model with the tracking method, and modified the SiamMask. The same MOT dataset is evaluated to give a standard result for comparison. The last parts are the future work and conclusion as the expectation for high-quality MOT networks with their advanced functions to end this thesis.

2. Detection and Tracking

This section illustrates the literature review around object objection and tracking. The object detection chapter covers the one- and two-stage object detection algorithms. And the object-tracking section first introduces visual object tracking categories and then steps into the details of correlation filters and deep learning models.

2.1. Object Detection

Definition

As a cornerstone for multiple object tracking, we shall first explain object detection, which is the fundamental branch in computer vision. [1] defines object detection as "seeking to locate object instances from a large number of predefined categories in natural images.". Typically, almost object detection networks pursue high accuracy and efficiency. Detection accuracy concerns the location and recognition accuracies, while efficiency cares about the time, memory, and storage efficiencies.

Milestones in computer vision

In recent years, there are some significant breakthroughs about feature representations, detection frameworks, and datasets happening in the computer vision, containing object detection, as Figure 3 below describes.

As the critical feature presentation, Scale Invariant Feature Transform (SIFT) feature [31] starts the era of converting the global features to the local features, which corresponds to the variances in translation, scale, rotation, illumination, viewpoint, and occlusion. After SIFT was released, some innovative features were followingly proposed, such as the Histogram of Gradients (HOG) [61], Local Binary Patterns (LBP) [60], region covariances [59], Speeded-Up Robust Features (SURF) [62], etc. To combine these local features, some light concatenation and feature pooling encoders do a favor. The typical cases contain the Bag of Visual Words [63], Spatial Pyramid Matching of BoW models (SPM) [64], and Fisher Vector [65].

For object detection methods, the spirit of Deformable Part based Models (DPMs) [66] and the Cascades [67] brought great influence in the generic object detection before deep learning. DPM outperforms other conventional detectors for representing by components deployed in a deformable structure. The highlight of Cascades is to compute Haar-like [68] features from 'Integral Image' and to merge the classifiers yielded by AdaBoost algorithms in a cascade.

Besides the feature presentation and detectors, the picture below also illustrated some common datasets, such as PASCAL VOC [42], ILSVRC [69], and MS COCO [19].

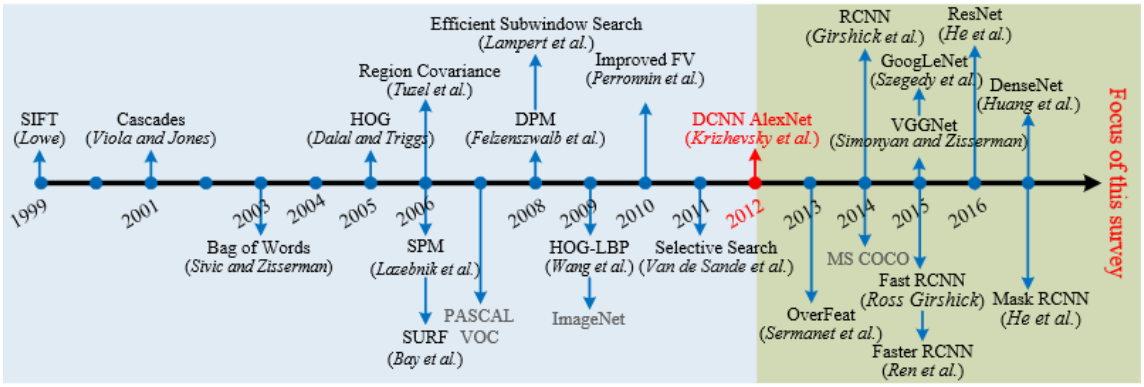

Figure 3. Milestones of object detection and recognition, including feature representations, detection framework, and datasets [1]. The listed milestones before 2012 are almost handcrafted features, while the approaches after 2012 are relevant to deep networks.

Milestones in object detection

This section mainly describes the object detection framework following the timeline in Figure 4. From 2012, DCNN [73] successfully revealed the starting time for deep detectors. After its birth, more advanced object detection models had been developed and published in Figure 4. As the CNN features are introduced into object detection, it is worth mentioning that DetectorNet [70], OverFeat [71], MultiBox [72], and RCNN [7] used CNN for their improvement at the same time. Furthermore, the convolutional frameworks having a great depth, such as AlexNet [73], VGGNet [74], GoogLeNet [75], ResNet [76], and DenseNet [77], are proposed. Considering their representative power, these models always perform as the backbone.

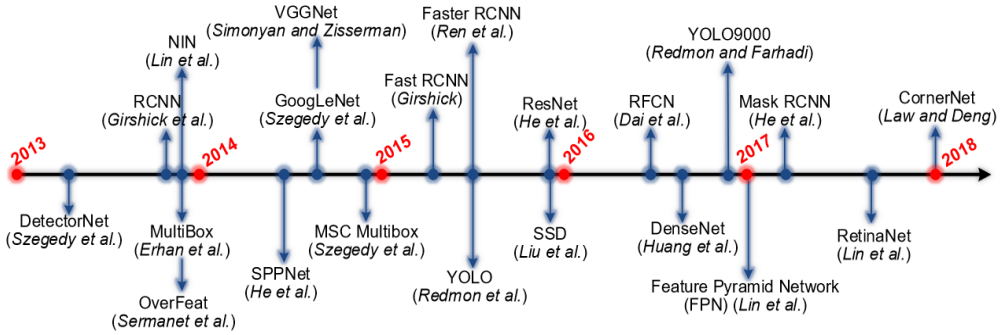

Figure 4. Milestones in generic object detection [1]. Landmark methods have emerged since deep learning entered the field, and they can be divided into two main categories.

These milestone approaches listed in Figure 4 can be divided into two typical categories: two-stage and one-stage detection frameworks. The two-stage network performs object detection through the proposed regions. In detail, the two-stage model firstly proposes

a set of regions of interest to avoid the infinite potential bounding box candidates. According to these region candidates, the two-stage model extracts the CNN features and then classifies them to computes results. In contrast, the one-stage algorithm establishes a unified framework to conduct object detection in a simple and efficient way.

R-CNN algorithms emerge as the popular two-stage networks for object detection, such as R-CNN [7], Fast R-CNN [43], Faster R-CNN [6], etc. They first adopt heuristic approaches, like selective search or CNN network, like RPN, to generate region proposals. Then, classification and regression can be performed based on region proposal. The typical one-stage algorithms, such as Yolo [2] and SSD [27], use only one CNN network to estimate the class and location for different target objects directly.

After two algorithm architectures, we next discuss their differences. When comparing the one-stage and two-stage algorithms, we concentrate on the result accuracy and computational efficiency. The two-stage algorithm produces more precise outputs, while these algorithms sometimes cannot affordable for real-time applications. On the opposite side, the one-stage algorithm results in significantly faster detections while deployable on lighter hardware. However, their output accuracy does not satisfy us as that of the two-stage methods.

2.1.1. One-stage algorithm

There are some typical one-stage networks, assembling both the detection and classification, like DetectorNet [70], OverFeat [71], CornerNet [78], SSD [27], and so on. As one of the popular one-stage algorithms, the Yolo network uses only one CNN to provide an end-to-end prediction in real-time. To dive into basic knowledge, we paid more attention to the initial Yolo architecture rather than the Yolo9000 model.

Yolo architecture uses a convolutional network to extract features and then employs a fully connected layer to derive predictions. It adopts the GooLeNet model as the backbone. As shown in Figure 5 below, Yolo contains 24 convolutional layers and 2 fully connected layers, For the convolutional layer, the $(1 * 1)$ convolution is mainly used for channel reduction, followed by the $(3 * 3)$ convolution. The convolutional layer and the fully connected layer use the Leaky ReLU activation function, and the last layer adopts a linear activation function.

To achieve the high speed, Yolo splits the resized images to (7*7) cells after it receives the raw images as input. According to these cells, the output layer directly produces the bounding box location and the corresponding category. To be specific, the author feeds the network with the entire graph and transforms object detection into a regression mission for reducing time. For the training, the loss function is defined to achieve a perfect balance among three aspects: coordinates in the form of (x,y,w,h), confidence, and

classification. Before evaluating Yolo, the outputs require being filtered through approaches like dropping the boxes with a low class-specific confidence score, non-maximum suppression (NMS), etc.

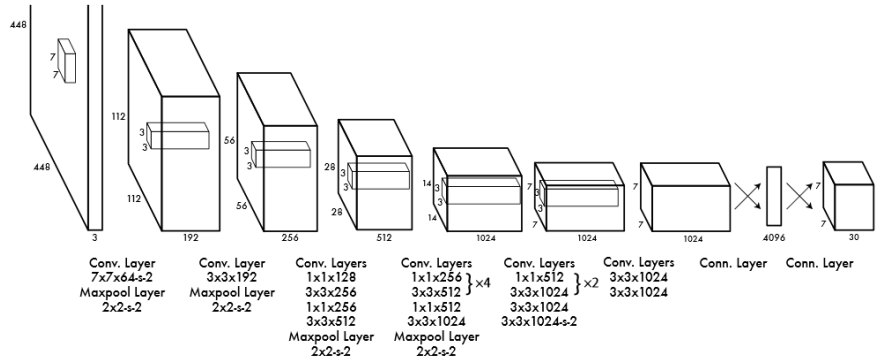

Figure 5. Yolo architecture, whose detection network contains 24 convolutional layers followed by 2 fully connected layers [2].

2.1.2. Two-stage algorithm

Among the two-stage algorithms, almost attention has been paid to the R-CNN series, including R-CNN [7], Fast RC-NN [43], Faster R-CNN [6], R-FCN [13], Mask R-CNN [79], Cascade R-CNN [80], and Light Head R-CNN [81]. Since RCNN, Fast RCNN, and Faster RCNN are widely used, we would discuss their differences and distinct features in this section.

For two-stage algorithms, the region-proposal design applied will decide their performance. So, we first present the diverse solutions about region-proposal extraction.

Selective search

[3] offers the introduction and implementation of the selective search. Compared with exhaustive search, segmentation, and other sampling strategies, selective search outstands since it can capture all scales, quickly compute and provide diversification directly. Specifically, it calculates the division and clusters similar regions based on features, like color, texture, size, and shape compatibility. Notedly, the hierarchical grouping algorithm and the diversified strategy contribute to the selective search algorithm

The hierarchical grouping algorithm was proposed since the region-based features of the image are more representative than image pixels. The author implemented the Efficient Graph-Based Image Segmentation [4] method invented by Felzenszwalb and Huttenlocher to generate the first region and iteratively groups the areas using the greedy algorithm: In the beginning, calculate the similarity between all adjacent regions. Next, combine the two most similar regions. Then calculate the similarity between the merged region and the adjacent region. The rest work is to repeat the second and third steps until

the entire image becomes a single region. In each iteration, the more significant regions are formed and added to the region proposal list. In short, they create a regional proposal from a smaller segment to a broader segment following the bottom-up principle.

The author released the diversification strategies using a variety of color spaces with diverse invariant attributes, different similarity measures, and distinct starting regions. The color spaces which had been tested with multiple alternatives, including RGB, the intensity of the grey-scale image, Lab, the rg channels of normalized RGB with intensity called rgI, HSV, RGB which is normalized RGB, the opponent color space exclude intensity denoted as C and the Hue channel H from HSV.

As a weighted function, the final similarity measure combines color similarity, texture similarity, size similarity, and appropriate similarity measures. In addition, the approach mentioned in [4] is the best option for proposing starting regions considering computational efficiency.

RPN

As the main component in Faster RCNN [6], Region Proposal Network (RPN) accelerates and optimizes the generation of proposals, which takes an image as input and outputs a set of rectangular object proposals with a score.

We will illustrate the PRN process in the following content. The first step is to generate anchor boxes of diverse sizes and length-to-width ratios with a sliding window. The multi-scale anchors support the detection network to compute as many scale features as possible efficiently. The typical anchor implementation has three shapes and three ratios. Hence, a total of nine anchors for one feature can initialize the bounding boxes. Then intersection-over-Union (IoU), which presents the overlap between the anchor and the ground-truth, is used to separate the positive anchor boxes from the negative anchor boxes. Softmax and reshape operations aid the positive anchor candidate extraction.

After the procedures above, the input data had been converted into anchor boxes coordinates and the probability of object existence in each anchor box. In detail, the RPN network maps each sample into four coordinate values and one probability value. The four coordinate values are applied in regression to locate the object with the help of the feature map. The probability value reflects the probability that the specified anchor box includes objects. For training, the RPN network adopts the losses of the binary classification and coordinates regression.

The region proposals are generated by RPN and filtered according to the probability value. Furthermore, they are fed into the R-CNN subnet for multi-classification and coordinate regression. Thereby, the multi-task loss is contributed by these two branch performances.

R-CNN network family

After discussing the region proposal strategies, we will introduce the R-CNN network family. It worths us to research three key roles: R-CNN [7], Fast R-CNN [43], and Faster R-CNN [6]. Region-based Convolution Neural Networks (R-CNN) [7] is the ancestor of the other two networks. Namely, its creative framework provides Fast R-CNN [43] and Faster R-CNN [6] with cornerstone. It made a breakthrough by improving the mAP on Pascal VOC [42] while DPMs [66] were at the bottleneck.

In Figure 6, the regions exacted by the selective search are warped into a square and then fed to a subsequent convolutional neural network for classification and regression. In the classification, the support vector machine (SVM) exams whether the objects presented in the region proposal. Besides, the regression network assists R-CNN architecture in predicting four offset values to achieve high-precision results.

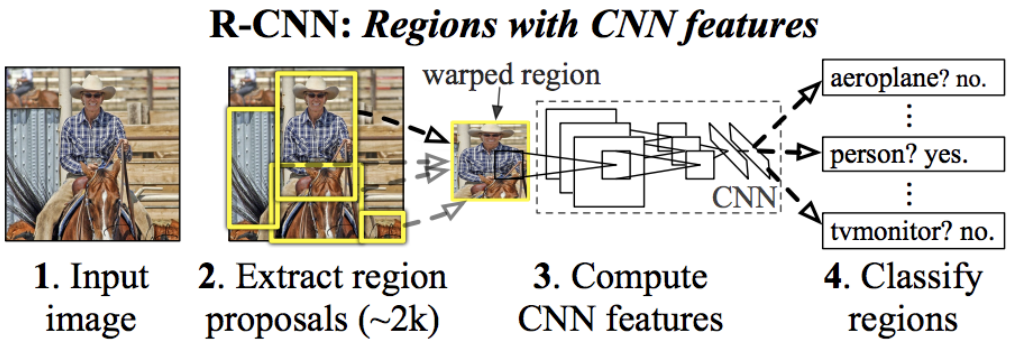

Figure 6. R-CNN architecture [7]. Firstly, the R-CNN model feeds the image as input. Secondly, approximately two thousand region proposals would be extracted. A powerful convolutional network converts the proposals to features. Finally, the category specific linear SVMs classifies the regions.

Through previous content, we have observed several downsides of R-CNN. Most obviously, the training steps are cumbersome, including CNN fine-tuning, SVM detector training, and bounding box regressor training. These complex steps and plenty of proposals constraint R-CNN to perform in real-time. Moreover, the selective search algorithm limits the self-learning of the region proposal generator, and therefore cannot be iteratively improved to reject erroneous candidate region proposals.

The shortcomings of R-CNN motivated a lot of creation. As one of them, Fast RCNN inspired by SSPnet architecture [8] benefits in high-quality results and high efficiency. Shown in Figure 7, it first normalizes the raw images and feeds them with corresponding object proposals to the CNN network. The convolutional layer does not perform feature extraction on the region proposals from selective search directly. The highlight point is that the author uses the ROI pooling layers to exact and form feature vectors from Conv features map according to object proposals. These designs avoid the extra calculation for

generating features on a large number of candidate proposals. As a result, it reduces training and testing time. Besides these advantages, the space requirement of Fast R-CNN [43] is not as crucial as that of R-CNN [7]. Since object classification and bounding box regression in R-CNN [7] work independently, these operations shall be supported by adequate features as training samples. On the contrary, Fast R-CNN [43] unifies these sibling layers with the full-connected layer in the CNN network to reduce the storage for redundant features. However, the selective search in R-CNN slows down the process, preventing users from performing real-time detection.

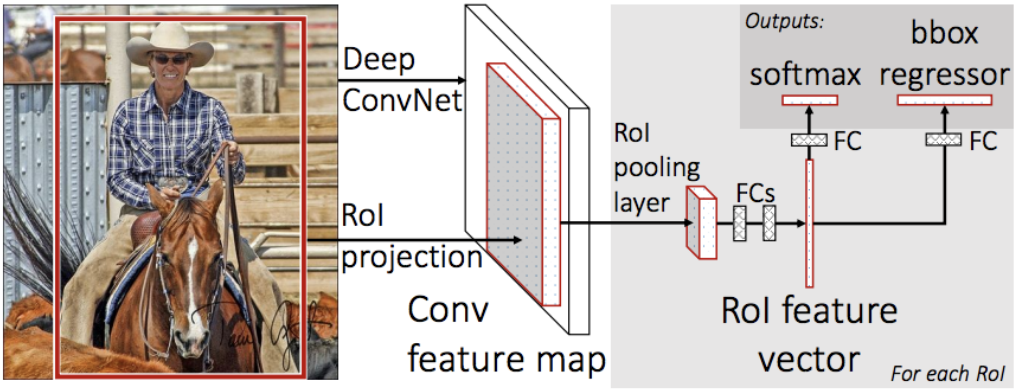

Figure 7. Fast R-CNN architecture [43]. Firstly, the convolutional network computes the feature map for the whole image. Then the corresponding part is extracted from the feature map for each region proposal. Regions of interests (RoI) pooling layer resizes the RoIs. And the fully connected layers (FCs) layers flatten the fixed-size region proposal feature maps into feature vectors. Finally, the softmax layer classifies objects. The bounding box regressor outputs the bounding box coordinates for each object class.

As one of the most popular object detection models, Faster R-CNN [6] outperforms Fast R-CNN [43] on speed and space. Compared with R-CNN [7] and Fast R-CNN [43], it implements the end-to-end object detection framework and compresses the time of region proposal generation. The innovation which Fast R-CNN [43] creates is to train a Region Proposal Network (RPN) on a convolutional feature map rather than adopting a certain region proposal method for region proposal generation. The afore-mentioned RPN computes the feature maps to output region proposals. Then, the predicted region proposals are reshaped by an RoI pooling layer to classify the images within the proposed region and to predict the offset values for the bounding boxes.

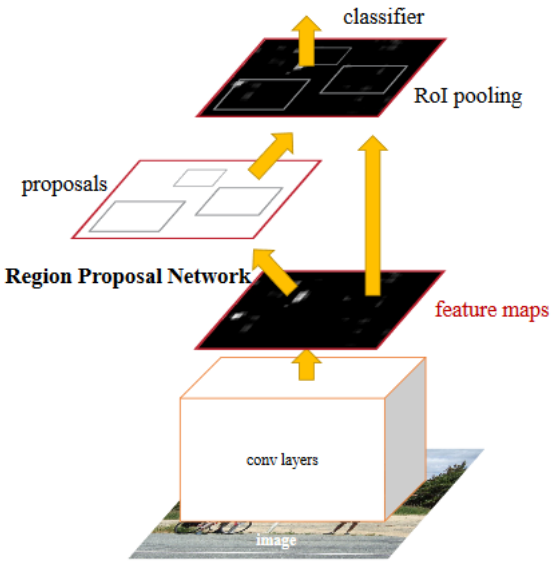

Figure 8. Faster R-CNN architecture [6]. Initially, an image is convolved into feature maps. Then RPN network generates the candidate regions considering their feature maps in the corresponding anchors. Finally, RoI pooling layers take the feature map for each region proposal, flattens it, and passes it through the classifiers to calculate the category and location for the proposal regions.

2.2. Object tracking

Object tracking shall be explained before we start introducing multiple object tracking. [9] defined object tracking as "to estimate the states of the target in the subsequent frames, given the initial state of a target object in the first image."

In general, object tracking faces several obstacles, like deformation, illumination variation, blur or fast motion, background clutter, out-of-plane rotation, in-plane rotation, scale variation, occlusion, and out-of-view. Due to these, it is not easy to invent and optimize object tracking algorithms. OBT and VOT datasets, introduced in the experiment section, are the most common data source for object tracking training and evaluation.

Visual object tracking categories

Developing generative and discriminative models are two main trends for visual object tracking. Generative methods model the target region in the current frame and search the most similar region in the next frame. This type of algorithm focuses on the features of the target objects without the context information. Thereby, the generation method works well under normal circumstances but always fails when the target object changes drastically or is occluded. The most popular generative approaches are Kalman filter [50], particle filter [48], and mean-shift [49]. Robust Scale-Adaptive Mean-Shift (ASMS) for tracking [5], based on the mean-shift algorithm, is recommended as the real-time tracking model by VOT2015. Since the mean-shift is applied to track the object by minimizing a

distance between two probability density functions, which are represented by a reference and candidate histograms. However, we could not cope well with the variant scale objects using the fixed search window. ASMS method solves this problem by adopting a robust scale estimation.

Referring to discriminative methods, they usually distinguish the target and context region as the positive and negative samples in the first frame. And then, train the classifier on features extracted from samples. Once the classifier fits the training dataset, it is time to discriminate against the next frame. Compared with generative methods, most of them perform better because they can classify the foreground and background region with the context features.[51]

Before the correlation filter and deep learning method appear, Structured Output Tracking with Kernels (Struck) [10] and Tracking-Learning-Detection (TLD) [44] are competitive among the visual object tracking methods. TLD shown in Figure 9 is a long-term object tracking method that consists of a tracking module, a detection module, and a learning module. In detail, the tracking module observes the target object movements between frames. The detection module locates target objects for each picture independently. The learning module evaluates the detecting results by checking the tracking module outputs. Then, it supplies training samples to the detection module for iterative optimization.

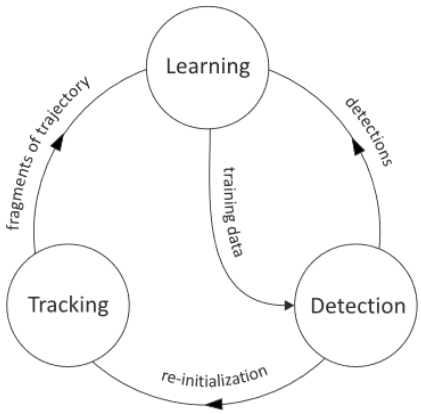

Figure 9. The block diagram of the TLD framework [44]. Under the proposal that motion is restricted among frames, the tracker predicts the object movements between the continuous frames. Detector processes every frame independently and determines the appearance location through fully scanning the image. Finally, observing the performance of the tracker and detector, learning estimates the error caused by the detector. In addition, it produces training examples to prevent failing.

Generally, traditional methods (such as adaptive tracking detection) train binary classifiers online to identify target objects from the background. Unlike them, Struck adopts a distinct method to generate structured output predictions. Its implementation operates

with the subsequent frames and then predicts the variances of object configuration between frames. Figure 10 presents the difference between Struck and other traditional models. It shows that Struck skipped the following steps: generating a set of samples and creating training labels with the help of estimated object positions and the learner.

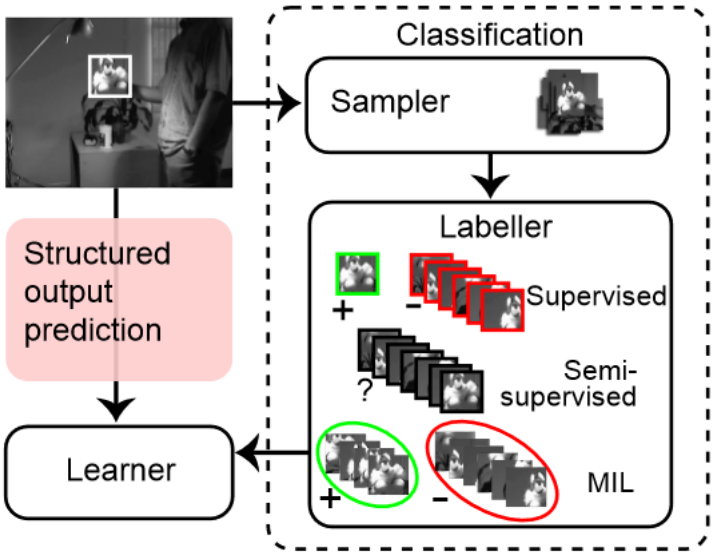

Figure 10. Different adaptive tracking-by-detection paradigms [10]. The struck approach on the left-hand side avoids the steps of traditional methods shown on the right-hand side.

Correlation filters

Correlation filter (CF), also named discriminative correlation filter (DCF), is a standard and straightforward classification algorithm. It is proposed from the signal processing and contributed by the Fourier principles. CF has drawn public attention due to its computational efficiency.

The essential procedure likes the following steps: In the case of a given template, the higher the response generated by the cross-correlation between the search area and the target object, the higher the correlation between them. Therefore, the user could carry out the object tracking by observing the response values.

The Minimum Output Sum of Squared Error (MOSSE) [56] is the first model that introduces the correlation filtering into object tracking. As a correlation filter of single-channel gray features, it lacks the characteristic feature and acts insensitive to scale variant objects.

The advanced CF has been improved by being plug-in with multi-channel formulations, spatial constraints, and in-depth features. On the basis of MOSSE, the circulant structure of Tracking-by-Detection with Kernels (CSK) [54] expands with dense sampling, padding, and kernel-trick. And then, the Kernel Correlation Filter (KCF) [55]

adopts the HOG [61] feature of multi-channel gradients after the CSK implementation. Color visual tracking [53] has been optimized with multi-channel color names (CN) based on CSK architecture. HOG [61] is a gradient feature, while CN is a color feature. Interestingly, they are complementary to each other. For the past two years, these two features have been adopted as the standard handcraft tracking features. Although the correlation filter has plenty of alternative features to choose from, it was at the bottleneck when dealing with the scale variance. The discriminative scale-space tracker (DSST) [52] gave a solution using the scale filter based on MOSSE. Motivated by its high calculation cost, [57] designed a Fast-Discriminative Scale Space Tracker (fDSST) to accelerate the computation by simplifying the dimensions. As a fusion of Distractor-Aware Tracker (DAT) [58]and DSST models, Staple [34] is achieved by combining HOG [61] and color histogram features.

Deep learning models

Most deep learning models are recommended because they have powerful learning capabilities. In other words, they can effectively grasp valuable clues from a large amount of labeled training data. However, the object tracking task only provides the bounding box of the first frame as a labeled sample. It restricts the training of deep learning models for object tracking.

When we lack enough training samples, one solution is to use auxiliary non-tracking training data for pre-training. It allows us to obtain a general representation of object features. During real-time tracking, we use the limited labeled data of the target object to fine-tune the pre-trained model. The fine-tuning enables the model specific to the labeled data of target objects. This transfer learning strategy requires less labeled data, but it could improve the performance of tracking models dramatically. It advances the object tracking and benefits the networks, such as DLT [32] and SO-DLT [33].

With the popularity of the classification competition, there is a trend to extract features using pre-trained CNN classification networks of existing large-scale classification datasets. Some object tracking models employ this feature extraction approach, and then they perform classification and compute tracking outputs with the extracted features. These networks not only avoid the dilemma where the shortage of tracking samples leaves but also make full use of the dominant representation of Conv features. As one of the representative models, a Fully Convolutional Network-based Tracker (FCNT) [45] constructs feature extraction networks and two complementary heat-map prediction networks according to the different characteristics of distinct CNN layers. In this way, it prevents the tracker from drifting by effectively suppressing the distractor. At the same time, the network operates robustly to the deformation of target objects.

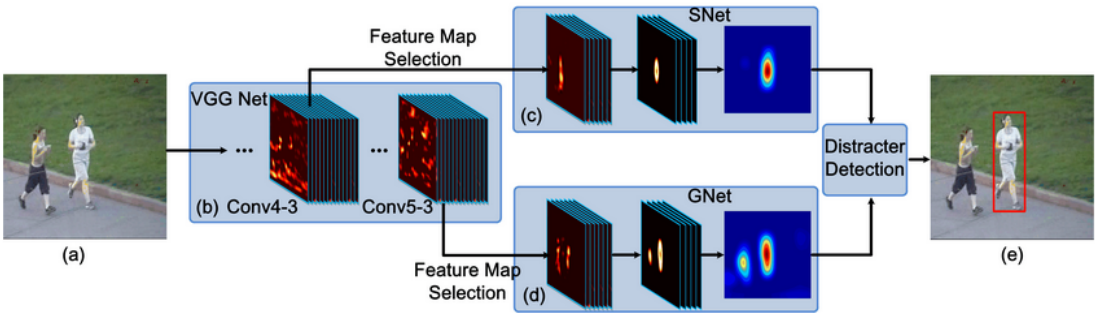

Figure 11. The pipeline of the FCNT tracking method [45]. The most relevant feature maps with the given target are chosen on the VGG layers (b). A general network (GNet) (d) and a specific network (SNet) (c) capture the category information and discriminate the target from the neighboring on the selected conv5-3 and con4-3 feature maps, respectively. Using GNet and SNet, the heatmap regression is computed in the first frame. For each of the remaining frames, the region of interest (RoI) (a) centered on the target position is fed into the network as input. Finally, a distracter detection scheme figures out the final target (e) based on the heatmap generated from (c) or (d).

While the image classification identifies which objects are in the same class, object tracking searches the same object between the sibling frames. Inspired by this difference, a Multi-Domain Convolutional Neural Network (MDNet) [35] proposes to pre-train CNNs with the frames from tracking videos to get general representation for target objects. As Figure 12 displays, the multi-domain network is divided into the sharing layer and the domain-specific layer. Each training sequence is treated as a separate domain, and each domain would be executed through a binary classification layer (fc6), which is used to distinguish the foreground from the background in the specified frame sequence. In the other networks, the shared layer can learn the generic feature presentation of the target objects in the tracking sequence. In addition, the domain-specific layer can solve the problem of inconsistent classification objects in different training sequences.

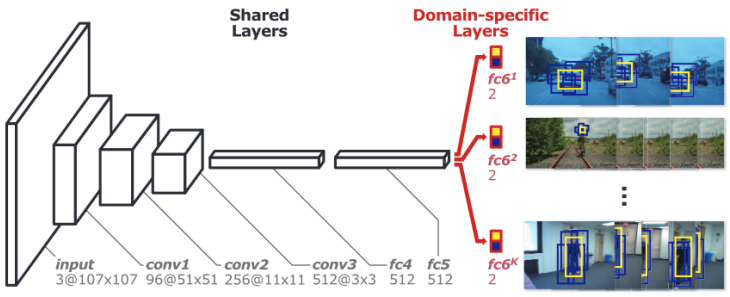

Figure 12. The architecture of Multi-Domain Network consists of shared layers and K branch of domain-specific layers [35].

In recent years, the RNN network, such as LSTM and GRU, accomplish quite a few excellent applications related to time-series. Recurrently Target-Attending Tracking (RTT) [36] is the first tracking algorithm that adopts RNN to model complex, large-scale associations for part-based tracking tasks. As Figure 13 presents, the RTT architecture uses a multi-directional recurrent neural network to simulate and to mine the reliable part that determines the overall tracking. And this network models on a two-dimensional plane, solving the tracking drift caused by prediction error accumulation and propagation.

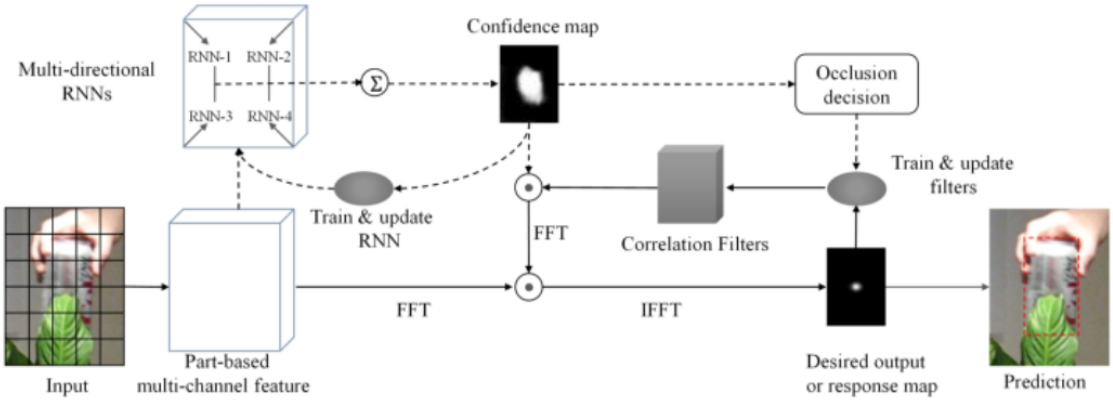

Figure 13. The architecture of the RTT Network [36]. When training and updating model, RTT learns discriminative correlation filters through adaptively regularizing the filters with confidence maps, which are estimated by multi-directional RNNs.

Deep learning supports building an end-to-end output tracking framework to meet real-time requirements. GOTURN [37] and SiameFC [11] are two typical object tracking models with high processing speed. As shown in the figure below, GOTURN has established a new tracking framework that learns the relationship between appearance and motion offline.

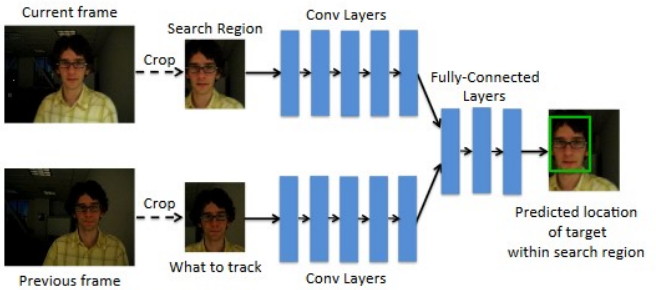

Figure 14. GOTURN network architecture for tracking [37]. A search region and target are cropped from the current frame and the previous frames and fed into convolu-

tional layers. Then, the target location within the search region is estimated by comparing the features from the target object to the features in the search region through the fully connected layers.

SiameFC [11] is proposed as a creative, fully convolutional twin network. As Figure 15, the author trained the SiameFC network end-to-end on ILSVRC15 [69]. Although SiameFC operates in a simple way, it achieves the best performance on multiple benchmarks.

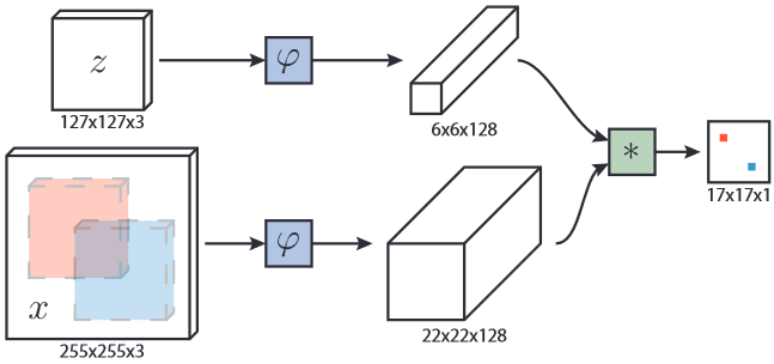

Figure 15. Fully convolutional Siamese architecture [11]. As the input to SiameFC, an exemplar image z and a candidate image x is fed to a deep convolutional network. With a convolutional embedding function φ and a cross-correlation layer, feature maps are converted and combined for similarity computation. The red and blue color points in the final score map hint similar degrees in sub-windows within the search image.

3. Multiple-Object Tracking Methodologies

Besides the knowledge about detection and tracking, key points of Multiple Object Tracking (MOT) also need to be well-understood and mastered, with its categorization being introduced in the first section. As described in the experimental section, we had applied the D&T [22] model on the dataset combination to obtain an acceptable MOT network. SSD [27] would be taken as the detector for the MOT model. SiamMask model [26] would be modified from an online SOT model into MOT architecture. Thus, the interpretation of D&T, SSD, and SiamMask model would be provided for the implementation of MOT in chapter 4.

3.1. MOT Categorization

According to [12], there are three standards for selecting and classifying multiple object tracking methods. They are initialization, processing format, and output type, all of which are playing the key role during the workflow of an MOT task. Differences in initialization allow us to infer whether the specified method is Detection-Based Tracking (DBT) or Detection-Free Tracking (DFT).

The DBT model first detects the target objects and then links them with tubes. This strategy can also be called "tracking-by-detection." The DBT models perform the particular type of object detection or motion detection on each frame of the given video sequence. After getting the object hypothesis, these methods would track objects according to their frame order or bench. Finally, this tracking hypothesizes would be plunged into trajectories.

DFT performs against DBT since it needs manual initialization. In other words, the researchers shall plot the target objects in the first frame of the given video. The DFT network then locates the tracking objects in the subsequent frames after initialization. As shown in Figure 16, the DBT model is more popular than the DFT model because of its capacity to discover new objects and to automatically cancel the disappearing objects. However, in the practical application of the DBT model, the detector needs to be trained in advance. In addition, it is necessary to consider that the DBT network depends on its detector.

Item	DBT	DFT
Initialization	automatic, imperfect	manual, perfect
# of objects	varying	fixed
Applications	specific type of objects (in most cases)	any type of objects
Advantages	ability to handle varying number of objects	free of object detector
Drawbacks	performance depends on object detection	manual initialization

Figure 16. Comparison between DBT and DFT [12]. The differences are depicted from their initialization, the number of objects, applications, advantages, and drawbacks.

With the purpose of implementing the MOT, we focus on data associations based on detected or drawn boxes. These related approaches are divided into online ones and offline ones. In the "Detect to track and track to detect," the offline method is adopted, which means that the object detections should be collected in advance and calculated according to the results. In contrast, online tracking is a frame-by-frame analysis of observations. Therefore, it can be called sequential tracking.

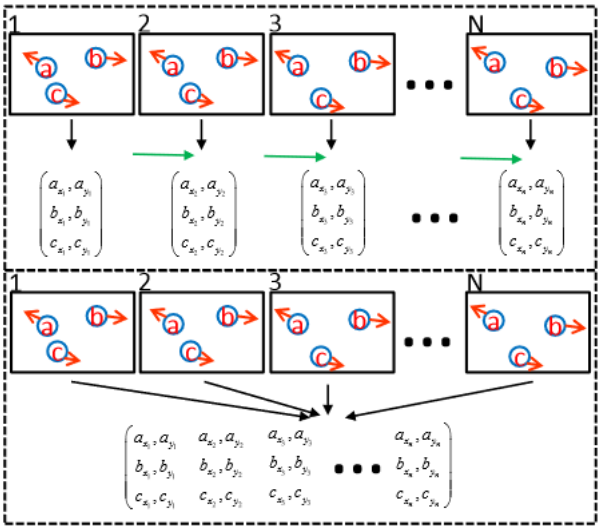

Figure 17. Description of online and offline tracking [12]. Top: online tracking method; bottom: offline tracking principle.

The last criterion is set up regarding the type of output. The randomness of output decided the MOT strategies belong to the probabilistic group or deterministic group.

3.2. Detect to Track and Track to Detect

Objective

Detect and Track (D&T) model [22] is proposed to solve the high-precision detection and tracking with a simple and effective method. The multi-task objection is used for frame-based object detection and cross-frame tracking regression, and the convolutional network structure is established, as shown in Figure 18. The following describes its detection and tracking principles.

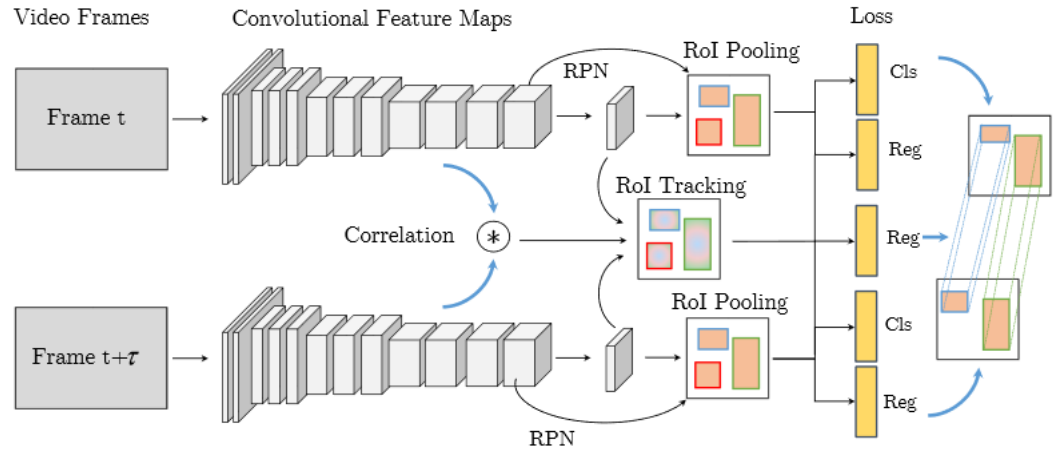

Figure 18. The architecture of the Detect and Track (D&T) model [22]. Two frames are input into convolutional layers. Based on their feature maps, RPN and RoI Pooling corporate to get the position-sensitive scores and regression maps. An RoI tracking operates position-sensitive regression maps from both frames with correlation maps and outputs the box transformation between two frames. When training the D&T model, its multi-task loss is applied with a combination of classification loss, regression loss, and tracking loss.

Detection Architecture

This D&T model provides region classification and regression with the region-based fully convolutional network (R-FCN) [13]. According to [13], R-FCN processes much faster than Faster R-CNN [6], while R-FCN still maintains competitive accuracy. Faster R-CNN [6] detector inserts the pooling layer into convolutions to keep translation invariance, which requires a tradeoff between accuracy and speed. To optimize this architecture, R-FCN was invented with its creative components: position-sensitive score maps and position-sensitive RoI pooling. In the position-sensitive score map, each ROI is divided into bins to encode position information. If C category objects are applied to classify, we will get $(C + 1) * k^2$ ROIs. For each category, the

ROI is voted on separately using the fractional mapping calculated with the pooled response from the raw image. Finally, the classification scores can be obtained through the SoftMax response.

In addition to classification, bounding box regression will be processed in a similar manner. The regression score graph with $4k^2$ dimension is calculated as a position-sensitive score graph. In this way, each ROI will get 4-d vector parameters as a translation of its position coordinates, width, and length after the position-sensitive ROI pool.

This detector takes ResNet-101 as the backbone network and filters the input samples through online hard cases [14] to generate hard cases. These hard examples will be fed for training, with a profound impact on classification and regression quality. At the same time, the author also modifies the backbone network, using the dilated convolution in conv5 to prevent the pooling from shrinking the heat map.

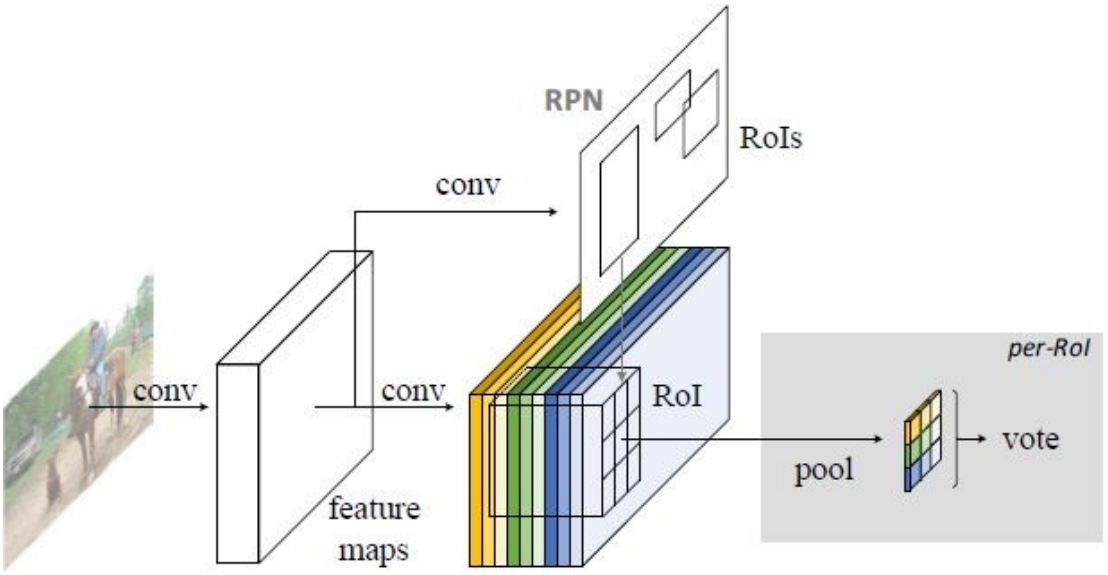

Figure 19. The overall architecture of R-FCN [13]. In it, RPN generates candidate ROIs which are used on the score maps.

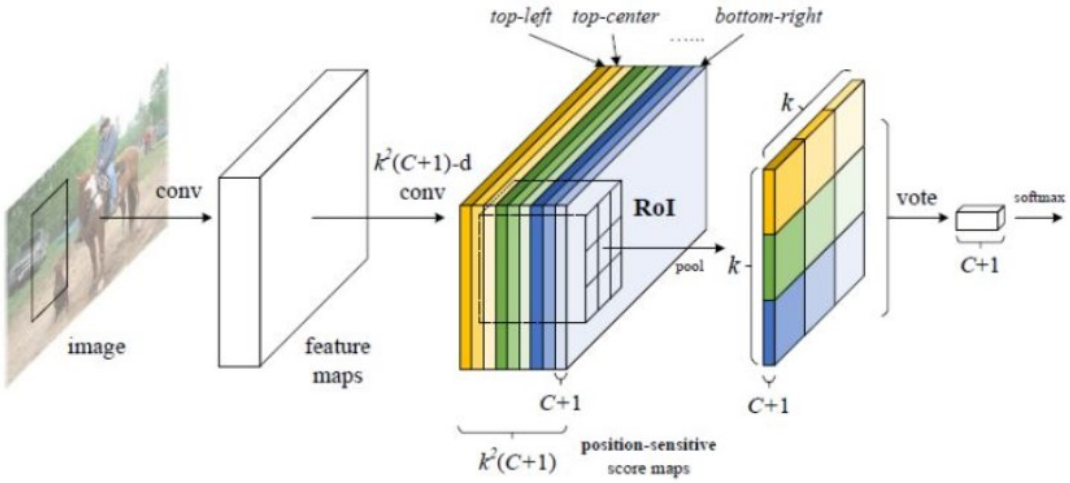

Figure 20. The core solution of R-FCN for object detection [13]. A fully convolutional network computes k × k position-sensitive score maps for the input image. The author applies the pooling one of the k^2 maps for each of the k × k bins within an RoI.

Tracking Architecture

The Detect and Track (D&T) model [22] is designed to improve the tracking quality according to the adjacent frames. Two points attract our attention: Correlation features representing object co-occurrence across time were calculated to aid the ConvNet during tracking; On the other hand, the frame-level detections based on the cross-frame tracklets had been linked to high-precision detection at the video level.

During the single object tracking, the correlation tracker focuses on the single target templates and the search image. Since the objective of the D&T model is to track multiple objects of videos simultaneously, the correlation feature map should cover all the positions. As all possible circular shifts in a feature map will yield the large output dimensions, correlation operates within a local neighborhood for output dimensionality reduction. In more detail, the restricted window would be utilized to perform a correlation between the sibling frames to reduce the computation. Moreover, the correlation features come from the con3, conv4, and conv5 layers of the detection architecture-R-FCN network [13]. The heatmap through the correlation process exacts the box regression transformation [△x, △y, △w, △h] between the sibling frames in order to correct the predictive results. The end-to-end training process had been achieved through the additional tracking loss over these transformations.

One advantage of this model is that the target object bounding boxes in the output video move smoothly. [15] adopted a unique solution for trajectory generation in order to overcome the challenge of filtering related data and maintaining detection quality. The detailed solution is described as following: For the establishment of a scoring function combining detection and tracking time, the best matching path will be computed through

the Viterbi algorithm meaning dynamic programming. Its principle is relevant to the solution of multi-step and multi-selective scenes. During the generation of tracklets, the detection of each frame can be considered as a step, and multiple bounding boxes in the detection are used as multiple selections. After building optimal paths, smooth path labeling and temporal trimming are required to complete the tracking. During iterating each frame for the obtained paths, the max detection score would be assigned and stored for each class. Furthermore, this score is subject to a penalty for discontinuous marking. It is then determined as the final path by examining the labels of the class-specific path, and the score on that path is calculated to illustrate the average accuracy. As a result, filtering paths with poor performance become increasingly convenient.

Future implementation

Although the original D&T model can perform multiple object tracking smoothly, its tracking targets are unfamiliar in standard life. In chapter 4, how to organize a dataset combination and train the D&T model on this plug-in dataset is described. In the end, we will implement a model that can detect and track a set of common objects, such as people, vehicles, and so on.

3.3. PyTorch-SSD -to-Object-Detection

Objective

To build a model capable of detecting and locating specific objects in images, [27] invented the Single Shot MultiBox Detector (SSD) as a one-stage object-detection algorithm. This approach is based on a feed-forward convolutional network, which produces a fixed-size collection of bounding boxes and scores for the presence of object class instances, followed by a non-maximum suppression (NMS) step to obtain the final filtered detections.

SSD emerges to make up for the downsides of the other one-stage detectors, like Yolo [2]. Yolo is neither sensitive to the sizes of the target objects nor robust to objects with scale changes and fractures. In contrast, the SSD network [27] incorporates the anchors' idea in Faster R-CNN [6], performing feature layer extraction, calculating the frame regression and classification operations in turn, which can adapt to the training and detection tasks of multiple scale targets. Once SSD was created, everyone has seen the feasibility of real-time high-precision target detection.

Architecture

The main design of the SSD network is feature layer extraction, which starts the SSD network. Then the frame regression and classification are performed in turn.

Since different levels of feature maps can represent different levels of semantic information, the SSD model operating on multi-layer features is suitable for target detection at different scales. For example, the low-level feature maps can represent more detailed semantic information so that they can improve the quality of semantic segmentation and are suitable for small-scale target learning. On the other hand, high-level feature maps can represent senior-stage semantic information of the high-level stage, thus producing smooth segmentation results. Furthermore, they are suitable for in-depth learning of large-scale goals.

SSD architecture contains six stages. Each stage can learn a feature map and then carries out bounding box regression and classification. First, a five-block convolutional network of VGG16 is used as the first stage in the SDD model, and then the fc6 and fc7 in VGG16 were converted into two convolutional layers, with conv6 and conv7 being as the second and third stages. On this basis, the SSD network continues to add conv8, conv9, conv10, and conv11 four-block networks to extract higher-level semantic information, with its structure of the SSD being shown in Figure 21 below. In each stage operation, the network contains multiple convolutional layer operations, each of which is essentially a small convolution.

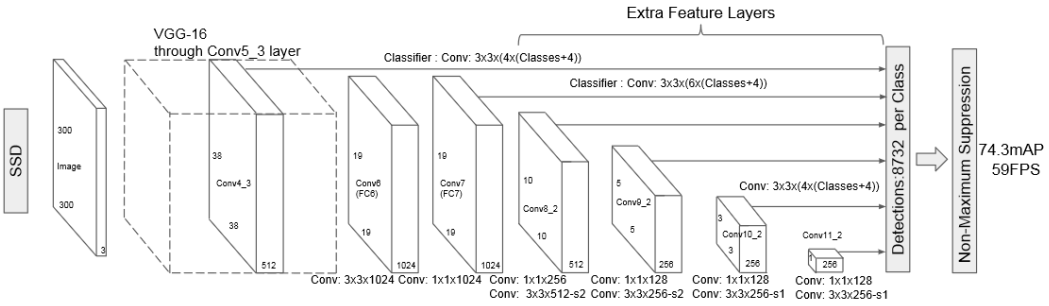

Figure 21. The architecture of the SSD detection model [27]. Multiple feature layers are attached to the end of a base network. Using them, SSD estimates the transformation to default boxes with different scales and aspect ratios and their corresponding confidences. In addition, the width and length of rectangular present the size of the feature map, and thickness presents the channel of the feature map.

In the design of the target detection network, the default box generation should be assigned with priority, which directly determines the type of target task and detection performance. In SSD, the author learned the Anchors mechanism from Faster R-CNN [6]. In each stage, the distinct number of default boxes with specific aspect ratios and scales are generated according to the size of the feature map. These pre-calculated, fixed boxes

that collectively represent this universe of probable and approximate box predictions can also be called priors.

Figure 22 describes the details of these priors applied to various low-level and high-level feature maps from conv4_3, conv7, conv8_2, conv9_2, conv10_2, and conv11_2. If the prior scale is s, then its area is going to be equal to the area of the square of length s. A larger feature graph has a smaller scale prior and is therefore ideal for detecting smaller objects. All feature maps will have priors with ratios 1:1, 2:1, 1:2. The intermediate feature maps of conv7, conv8_2, and conv9_2 will also have priors with ratios 3:1, 1:3. Moreover, all feature maps will have one additional prior with an aspect ratio of 1:1, and its scale is the geometric mean of the current and subsequent feature map scales.

Feature Map From	Feature Map Dimensions	Prior Scale	Aspect Ratios	Number of Priors per Position	Total Number of Priors on this Feature Map
conv4_3	38, 38	0.1	1:1, 2:1, 1:2 + an extra prior	4	5776
conv7	19, 19	0.2	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	2166
conv8_2	10, 10	0.375	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	600
conv9_2	5, 5	0.55	1:1, 2:1, 1:2, 3:1, 1:3 + an extra prior	6	150
conv10_2	3, 3	0.725	1:1, 2:1, 1:2 + an extra prior	4	36
conv11_2	1, 1	0.9	1:1, 2:1, 1:2 + an extra prior	4	4
Grand Total	–	–	–	–	**8732 priors**

Figure 22. The details of these priors applied to various low-level and high-level feature maps of SSD from conv4_3, conv7, conv8_2, conv9_2, conv10_2, and conv11_2 [46].

Once producing the default boxes on each feature map, SSD could generate the feature vectors corresponding to the bounding box regression and classification. For bounding box regression, only a 4-dimensional vector is needed, which represents the scale of the border (two directions of the coordinate axis) and the translation vector (two directions of the coordinate axis). For classification, the SSD network adopts a strategy for scoring each category. Assuming the number of categories in the dataset as C, and the total number of categories is $(C + 1)$ class, which involves background categories. So that SSD network uses $(C + 1 + 4)$-dimensional vectors to represent the final scores that combines the bounding box regression and classification. For instance, given a VOC dataset (including 20 categories) as an experimental dataset, each default box generates a $(20 + 1 + 4 = 25)$ dimensional eigenvector. Therefore, there will be 8732 predicted boxes in the encoded-offset pattern and 8732 sets of class scores.

After the prior preparation is completed, the loss function is applied to train and correct the SSD model on the specified dataset. In this paper [27], the weighted sum of position loss and confidence loss constitutes the objective loss function. Specifically, the localization loss is the averaged smooth L1 loss between the encoded offsets of positively

matched localization boxes and their ground truths. Furthermore, the confidence loss is simply the sum of the cross-entropy losses among the positive and hard negative matches. Positive and negative matches are identified by checking whether the overlap of Jaccard with prior and ground truth values exceeds 0.5. Moreover, the hard-negative mining principle, which keeps the number of hard negative overwhelm the positive ones, is adopted to accelerate the training of classification.

In order to obtain the final labelled boundary boxes that can be interpreted by humans, their respective prior offset results are decoded to obtain the boundary coordinates. For each non-background class, candidate boxes for the particular class of objects would be obtained through being filtered with a score. On the basis of these candidate tests, non-maximal suppression (NMS) is quite essential to obtain quality tests.

Future implementation

Among diverse multi-target tracking methods, we prefer detection-based tracking, which relies on high-quality detectors to implement. Therefore, in chapter 4.2.2, how we replace the detector in the D&T detector with the SSD model to observe its varying performance is expounded in detail.

3.4. Fast Online Object Tracking and Segmentation: A Unifying Approach

Objective

For real-time visual object tracking and semi-supervised video object segmentation, a simple method was developed: SiamMask [26]. It is designed to eliminate the gap between arbitrary object tracking and VOS for the purposes of retaining the offline trainability, online speed of these methods, and refining their representation of the target object. In the VOS field, SiamMask wins for its high speed. Compared with other VOT approached, the upper limit of target positioning accuracy is reached by SiamMask.

In general, the DFT model starts at the target position in the first frame and then predicts the target position in the next frame. Hence, how to define and locate the target object in the subsequent frames directly affects the performance of tracking approaches. Initially, the object tracking algorithms mark the targets using the axial rectangles. The accuracy of the tracking algorithms required to be improved, with the difficulty of object tracking competition. VOT2015 annotates the target object with a rotated rectangular. After that, VOT2016 proposes to create a mask through a bounding box. Actually, this rotated rectangle can be regarded as an approximation of the mask. More specifically, using masks is one way to get an upper limit on accuracy.

It is expensive to prepare the masks and to perform the training process for conventional tracking algorithms. Almost these algorithms use the semantic segmentation network to perform a two-category training online and then perform subsequent frame prediction, which takes a few minutes during the training process. Recently, more and more algorithms that do not require online finetune have been proposed. However, their speeds are still not satisfactory. For example, FAVOS [38] and OSMN [39] require 1s and 120ms every frame, respectively, which is still a specific difference from the real real-time operation. On the other hand, the first frame of the VOS algorithm requires a given target's mask, which is hard in human-computer interaction scenarios.

Methods

Visual object tracking can be divided into diverse categories according to the calculating methods of target object locations. The first category contains approaches of predicting scores, mainly including correlation filtering and SiamFC models. SiamFC, a fully convoluted Siamese network of offline training that compares an exemplar image against a (larger) search image x to obtain a dense response map. The object position and size are obtained by predicting the score map of the candidate region and computing the image pyramid. However, we cannot get the aspect ratio of the object through SiamFC.

The second category contains the bounding box regression methods represented by GOTURN and SiamRPN. SiamRPN, being the network that improves the performance of SiamFC by RPN, allows the use of a boundary box with a variable aspect ratio to estimate the target position. That is the reason why the output of SiamRPN, without the guarantee of prediction stability, can make the more accurate bounding boxes generated from the positive SiamRPN prediction and satisfies the researchers. The decisive point revealed by the SiamRPN is that the predicted aspect ratio from the network can adjust the bounding boxes.

Architecture

Based on the objectives and background of the Siamese network mentioned above, the architectures of SiamFC and SiamRPN are augmented with the seg-mentation branch and the loss mask, obtaining the two-branch and three-branch variants of SiamMask. With these variances, the corresponding losses can be optimized to obtain the improved model, as shown in Figure 23.

However, mask predictions are much more complex than score and box predictions. The representation method in [26] is to use a vector to encode a RoW mask. The RoW is the response map that presents the similarity between the template and the search area. The vector transformation utilizes a neural network that convolves two layers: one with 256 and the other with $(63 * 63)$ channels. This transformation enables each prediction

position to be equipped with a fairly high output dimension, which can encode a large amount of information. Furthermore, the depth-wise convolution is employed to concatenate the convolution to achieve efficient operation. The loss for two-branch variants of SiamMask is a weighted function which consists of the losses of mask prediction and SiamFC score map. While the loss for three-branch variants of SiamMask is a weighted function composed of the losses of mask prediction, SiamFC score map, and SiamRPN prediction. In all, these components constitute the main model framework of SiamMask.

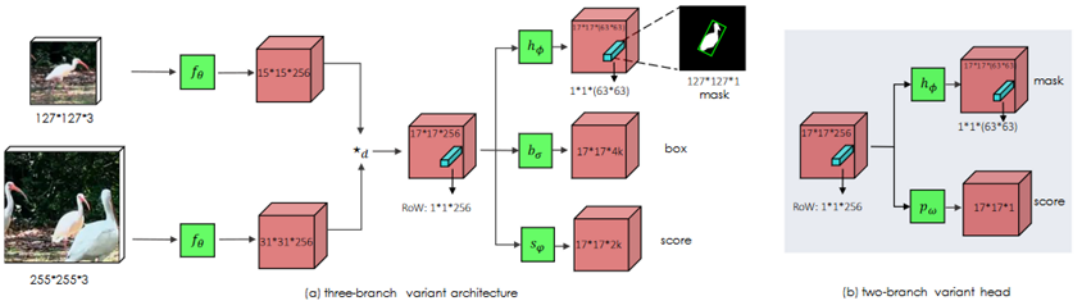

Figure 23. The architecture of SiamMask [26]. The left-side section presents the three-branch architecture, while the right-side section states two-branch architecture. Convolutional layers of ResNet-50 f_θ computes the features with the input search image (bottom) and examplar (top). Using depth-wise correlation $*_d$, responses of a candidate window (RoWs) are generated to feed several branch models. Two branch architecture, like SiamFC, contains the h_ϕ and p_w networks for processing masks and scores. Three branch architecture, similar to SiamMask, includes h_ϕ , b_σ and s_φ networks for final masks, boxes, and scores. Their training losses correspond to two and three variants, respectively.

Although the Mask branch had been established, the accuracy of the prediction output generated directly from the Mask branch has not reached the author's expectation. Therefore, the refine module [16] shown in the figure below is used to improve the accuracy of the segmentation by combining the low- and high-resolution features. In addition, the refine module adopts a top-down structure, which decreases the spatial resolution. At the same time, the number of channels increases with rising depth. In this way, rich information can be grasped without exceeding the amount of computation. In order to maintain sufficient inverse probability and high efficiency, convolution, ReLU, bilinear up-sampling, and concatenation are utilized in the refine module. This part draws on the idea of sharpmask, together with which deepmask is the objection segmentation proposal framework proposed by Facebook in 2015-2016.

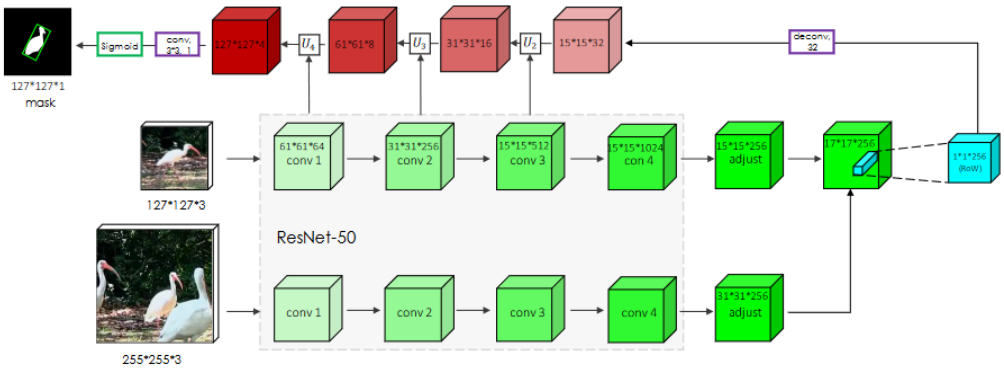

Figure 24. Schematic illustration of the stacked refinement modules in SiamMask [26]. Multiple refinement modules are contributed by upsampling layers and skip connections. They merge low- and high-resolution features for outputs more accurate object masks.

Compared with VOS, the VOT always requires a bounding box rather than a binary mask. There are several options (Min-max, MBR, and Opt) to convert the output mask to a boundary box. Min-max represents the axial bounding rectangle generated based on the mask; MBR represents the minimum bounding rectangle that is rotated according to the mask; Opt is an optimization strategy for automatic boundary box generation proposed in VOT-2016. And the author of [26] uses the MBR, which is convenient to generate. The box generated by the VOT optimization method is of good quality, while the box generated by the optimization algorithm is slow.

Future implementation

As has been mentioned, SiamMask is the Single Object Tracking (SOT) approach, as well as the approach of detection-free tracking. Through the implementation described in the next chapter, we modify it as a detection-based tracking way for multiple objects.

4. Experiments

Experiments were conducted around the following four topics: dataset, implementation, evaluation, and discussion. The dataset section begins with standard datasets, and then it explains how we organized a new dataset combination. The implementation section describes the processes of training and modifying three existing MOT models. In order to compare our networks fairly, we employed multiple evaluation metrics. Finally, the discussion section summarizes the disadvantages of these experiments and suggests possible future improvements.

4.1. Dataset

With the popularity of multi-object tracking increasing, it has inspired us to explore and develop MOT applications. Considering the target objects of MOT, both researchers and investors are interested in people, animals, and vehicles. Unfortunately, common datasets cannot meet our requirements. For example, the ImageNet dataset used in the D&T model does not include person videos, although it has a wide variety of categories. Besides that, some unfamiliar objects in ImageNet (such as turtles or foxes) are unnecessary because they rarely appear in our daily lives. To this end, we prepared the dataset combination based on the standard datasets.

In detail, we organized the next section around two subjects: a standard dataset and a new dataset combination. The standard dataset contains training and evaluation datasets used in D & T models, SSD detectors, and SiamMask networks. The dataset combination is used to fine-tune D & T and SSD models. It was contributed by the new videos and images of the person.

4.1.1. Standard dataset

ILSVRC2015

LSVRC is the ImageNet large scale visual recognition challenge. ILSVRC2015 includes not only the regular object detection competition for labeled images but also the taster object detection competition for annotated videos. The paper [22] mentioned that ILSVRC2015 contains 30 classes in 3862 training videos and 555 validation videos. Referring to the image dataset, at most 2k images per class for 30 categories are publicly available.

Since Detect and Track (D&T) model [22] adopts the ILSVRC2015 dataset for training and evaluation, we organized the newly collected dataset into a tree structure of ILSVRC2015. As shown in Figure 25 below, its architecture has three main directories

under the ImageNet dataset: "Data," "Annotations," and "ImageSets," which are responsible for storing, annotating, and listing the image and video data respectively. Their subfolders "VID" and "DET" store the materials related to the videos and images separately. Finally, the “train” and “val” directories on the leaves of this tree structure split the whole images, videos, and their corresponding annotation files into training and validation sets.

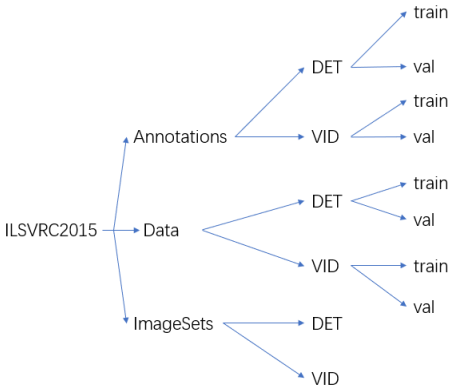

Figure 25. The tree structure of the ImageNet directory. ‘DET’ and ’VID’ present the images and videos, respectively. ‘train’ and ‘val’ state the training and validation sets.

Pascal VOC

Pascal VOC [42] is a series of bench datasets used for object classification and location. It starts introducing an annual competition to perform the standard evaluation on the object detection algorithms. In 2005, Pascal VOC had only four-category objects. After this year, the number of standard dataset classes has increased to 20. Since 2009, the dataset collection not only retains the previous dataset but also adds a newly collected dataset.

The dataset used in the SSD experiment consists of Pascal VOC data from 2007 and 2012. The following two figures describe the objects and images of VOCs in 2007 and 2012.

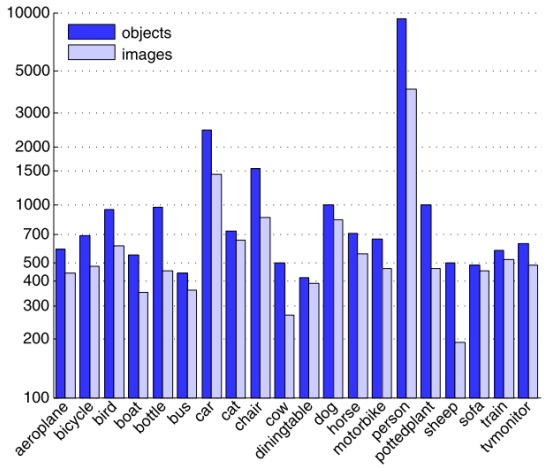

Figure 26. Summary of the entire VOC2007 dataset [42]. This histogram describes the class distribution of objects and images in the entire VOC2007 dataset. The vertical axis adopts the log scale.

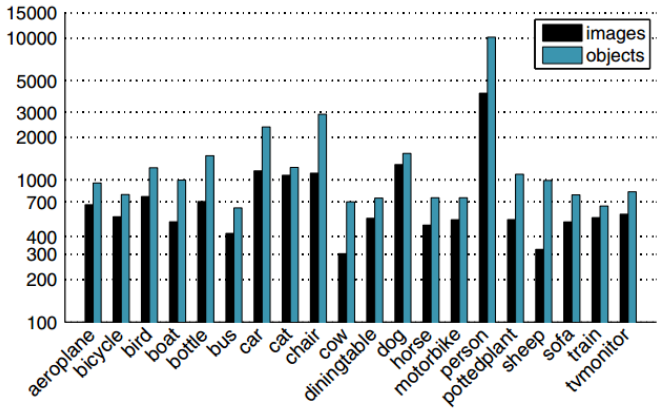

Figure 27. Summary of the main VOC2012 dataset [47]. This histogram describes the class distribution of objects and images in the training and validation VOC2012 dataset. The vertical axis adopts the log scale.

VOT

[17] Starting in 2013, the Visual Object Tracking (VOT) Challenge has been held to compare tracker performances. The dataset VOT is a color video sequence that benefits the performances of the color feature algorithms. Compared to the Visual Tracker Benchmark (OTB) dataset similar to the VOT dataset, its sequence is accurately labeled, and the evaluation index is easier to check. Notedly, almost the VOT video datasets are short-term.

Youtube-VOS

[18] Youtube-VOS is a large-scale video object segmentation dataset containing 4453 YouTube video clips and 94 object categories. It collects high-resolution videos containing the specified-class objects from YouTube-8M, which is a large-scale video classification dataset that includes millions of YouTube videos.

MS COCO

[19] MS COCO contains large-scale object detection, segmentation, and subtitle datasets. The creation of MS COCO makes up for the shortcomings of ImageNet. Objects in ImageNet are mostly large-scale and well-centered. However, the actual situation always opposite to it. Therefore, the MS COCO dataset includes complex scenes containing ordinary objects. This means that the MS COCO dataset contains occlusive objects and a large number of small objects that are difficult to label.

MOT

MOT is the dataset used in pedestrian detection challenges. In its subset- MOT17DET, there are seven training videos filmed in different situations. Since the resolutions of MOT17DET, excluding MOT17-05, are the same as 1920 × 1080. They range from 14 to 30 frames per second and range from 600 to 1050 in length. The total time and target frame of these videos are 215 seconds and 112297, respectively. The average density (the average number of pedestrians per frame) has reached 21.1. For more details, see Figure 28.

	Sample	Name	FPS	Resolution	Length	Boxes	Density	Description
1		MOT17-13	25	1920x1080	750 (00:30)	11642	15.5	Filmed from a bus on a busy intersection
2		MOT17-11	30	1920x1080	900 (00:30)	9436	10.5	Forward moving camera in a busy shopping mall
3		MOT17-10	30	1920x1080	654 (00:22)	12839	19.6	A pedestrian scene filmed at night by a moving camera
4		MOT17-09	30	1920x1080	525 (00:18)	5325	10.1	A pedestrian street scene filmed from a low angle.
5		MOT17-05	14	640x480	837 (01:00)	6917	8.3	Street scene from a moving platform
6		MOT17-04	30	1920x1080	1050 (00:35)	47557	45.3	Pedestrian street at night, elevated viewpoint
7		MOT17-02	30	1920x1080	600 (00:20)	18581	31.0	People walking around a large square.

Figure 28. Details of the entire training dataset from MOT17 DET [21]. ‘Name,’’ frames by second,’’ resolution,’’ length,’’ boxes,’ ‘density,’ and ‘description’ of seven training MOT17DET videos are listed.

For each video, there are two folders: "gt" and "img1" in this directory. The "im1" includes all the frames of this video. Their names mean themselves order, and their formats are "jpg." The "gt" folder stores a file recording the ground truth information. Ground truth information consists of frame id, tracking id, coordinates of the top-left corner, the width and height of the bounding box, flag for evaluation, the type of object, and the visibility ratio of each bounding box.

To compare the experimental results with the ground truths, we formed our result records similar to the ground truth files. Notedly, the tracking id, the type of object, and the visibility ratio of each bounding box shall be set -1 in the result files. In addition, detecting confidence replaces the evaluation flag.

4.1.2. Dataset Combination

Overview

As we mentioned before, our dataset should contain common objects, especially the person. Thereby we established a new dataset combination based on the standard datasets, such as ILSVRC2015, Pascal VOC, and MOT. And the new video and image dataset of the person introduced later joined our dataset combination. Since building a new dataset combination is to train the D&T and SSD detector, we converted the format of all datasets to the format of ILSVRC2015.

After collection, transformation, and calculation, we obtained 114057 training frames and images containing a total of 270097 objects. There are 107046 validation frames and images, including a total of 242639 objects. These objects belong to eight classes: bird, bus, car, dog, domestic cat, bicycle, motorcycle, and person. Figure 29 below shows the class distribution over the entire new dataset combination.

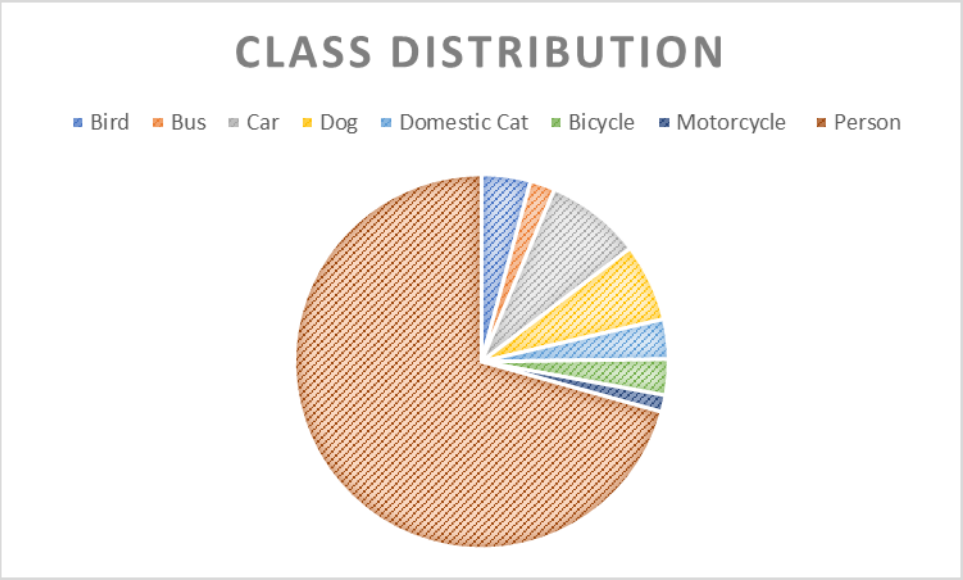

Figure 29. Class distribution of a new dataset. This pie chart states the class distribution with colorful slices. The proportion of each slice shows as the class percentage over the entire dataset.

For more details, table 1 and table 4 display class distributions of training and validation datasets. And table 2 and table 3 present the video and the image class distributions in the training dataset. Table 5 and table 6 state the video and the image class distributions in the validation dataset.

Table 1. The class distribution of the training dataset.

Class	Bird	Bus	Car	Dog	Domestic Cat	Bicycle	Motorcycle	Person
Number	7987	4214	12891	9330	5469	6310	5662	218234

Table 2. The video class distribution in training dataset

Class	Bird	Bus	Car	Dog	Domestic Cat	Bicycle	Motorcycle	Person
Number	3852	749	5719	5036	1701	3153	1805	94070

Table 3. The image class distribution in training dataset

Class	Bird	Bus	Car	Dog	Domestic Cat	Bicycle	Motorcycle	Person
Number	4135	3465	7172	4294	3768	3157	3857	124164

Table 4. The class distribution in the validation dataset

Class	Bird	Bus	Car	Dog	Domestic Cat	Bicycle	Motorcycle	Person
Number	13748	6785	29443	25055	12185	9526	1775	144122

Table 5. The video class distribution in the validation dataset

Class	Bird	Bus	Car	Dog	Domestic Cat	Bicycle	Motorcycle	Person
Number	9862	6313	26690	20237	11629	8854	1139	111733

Table 6. The image class distribution in the validation dataset

Class	Bird	Bus	Car	Dog	Domestic Cat	Bicycle	Motorcycle	Person
Number	3886	472	2753	4818	556	672	636	32389

New video dataset

Since the AVA and ActEV / VIRAT datasets have target objects of different sizes, they were selected as the people video sources in the new dataset combination.

AVA

The AVA project [82] provides audiovisual annotations of videos to enhance our understanding of human activities. All annotated videos are 15-minute movie clips. Human annotators have exhaustively labeled each of the clips, and the use of movie clips in the dataset is expected to enable a wider variety of recording conditions and representations of human activity.

We adopted the AVA Action dataset as the new video source. It is sourced from the 15th to 30th minute time intervals of 430 different movies, given a sampling frequency of 1 Hz. It gives us nearly 900 keyframes for each movie. In each keyframe, each person is marked with (possibly multiple) bounding boxes and tracking identification.

By processing the AVA data source in CSV files, we could transform their format into the ImageNet form. In the AVA files, each row contains an annotation for one person performing an action in an interval, where that annotation is associated with a certain frame. Different persons and multiple action labels are described in separate rows. The format of a row is the following: “video_id,” “middle_frame_timestamp,” “person_box,” “action_id,” “person_id.”

- video_id: YouTube identifier

- middle_frame_timestamp: from the start of YouTube in seconds.
- person_box: top-left (x1, y1) and bottom-right (x2, y2) normalized with respect to frame size, where (0.0, 0.0) corresponds to the top left, while (1.0, 1.0) corresponds to bottom right.
- person_id: a distinct integer allowing this box to be connected to other boxes corresponding to the same person in adjacent frames of this video.

According to the YouTube identifier, we downloaded 40 and 10 high-quality videos and saved them for further transformation. Then, by converting the downloaded videos into frames, we could store the frames under the “Data” folder. If there is not enough storage space, we shall retain only the frame corresponding to "middle_frame_timestamp”. The "person box" records the normalized coordinates of the person bounding boxes. When transforming is ongoing, the annotation file records the ground truth objects with the product of the "person box" and the frame size and "person id" in XML. Since we utilized the "train.txt" and "val.txt" under the "ImageSets" folder to list the frame file paths for training and evaluation, the frame file and XML file should be renamed as the frame order for the convenience.

The averaged time length of ImageNet videos is around 10 seconds. The author of [22] subsampled ten frames per video. Thus, the real fps can be regarded as 1. In this way, we sampled every annotated frame of the AVA dataset.

ActEV/VIRAT

From [24], we know that ActEV/VIRAT [23] establishes for activity detection research in streaming video [20]. This dataset is an improved version of VIRAT, with more videos and annotations. It includes 455 videos at 30 fps from 12 scenes, more than 12 hours of recordings. Most videos own high resolution as $(1920 * 1080)$.

32 and 4 videos from ActEV/VIRAT are utilized as our training and validation datasets, respectively. “Data” folder stores these videos and processed frames. Moreover, coordinates and tracking identifications from the processed dataset in [24] should be organized and written in an XML file under the “Annotation” folder. Since the frequency of the ActEV/VIRAT dataset differs from that of the AVA dataset, we subsampled 100 frames per ActEV/VIRAT video when recording the frame file paths for training and evaluation

New image dataset

Besides the video data source, the Figure Eight dataset [25] has been converted as a part of our new dataset combination.

Figure Eight

As a data-annotated platform, the Figure Eight dataset contains 9 million images that have been annotated with image-level labels and bounding boxes spanning thousands of classes. We adopted its person images and put the chosen images under the "DET" of the "DATA" folder. Moreover, the bounding box coordinates are transformed from CSV format into XML format and saved under the "Annotation" folder.

4.2. Implementation

4.2.1. Detect to Track and Track to Detect

[22] notes that the original dataset for D&T training and validation comes from the ImageNet. The author decided to adopt it since this data source contains not only the image but also the video source.

As we mentioned earlier, we cannot employ the pretrained D&T model since it cannot detect and track the common object, especially the person. Due to this, we trained the D&T model on the new dataset combination described in 4.1.2.

Training and Testing of the original version

For paper [22], training D&T architecture starts with the R-FCN model [13] and fine-tunes it on the ImageNet VID training set with randomly sampling a set of two adjacent frames from a different video in each iteration.

In the Pytorch implementation, the author employed the common heuristic of passing alternating samples from VID and DET of ImageNet for training. Additionally, ten-frames are sampled per video snippet as the training input. It avoids biasing the training towards longer snippets.

The following tables report the training and evaluation details of the baseline single-frame R-FCN [13] and D & T models. The trained single-frame RFCN initialized the weights of the D & T network. However, since the author evaluated the model on each frame instead of multiple adjacent frames, we cannot treat the verification results as fair.

Table 7: Baseline single-frame R-FCN

model	#GPUs	batch size	lr	lr_decay	max_epoch	time/epoch	mem/GPU	mAP
Res-101	2	2	1e-3	5	11	--	8021MiB	70.3

Table 8: D&T network

model	#GPUs	batch size	lr	lr_decay	max_epoch	time/epoch	mem/GPU	mAP
Res-101	2	2	1e-4	5	7	--	8021MiB	68.7

Training and Testing of our models

We trained the D&T model one the new dataset combination. However, the experimental results present not as well as the previous work. We infer there are three reasons.

Firstly, we cannot implement the pretrained model, like D&T and R-FCN models. Author [22] trained them on ImageNet rather than our dataset combination, so they contain different checkpoints

Secondly, we are not able to train the D&T model with the great backbone network, such as resnet101. Due to the memory limitations in GPU hardware, our D&T model used resnet50 or resnet18 as a backbone. For training, we applied a learning rate of 1e-4, learning decay of 5, and max epoch of 11. The table below shows that our D&T architecture is evaluated on the single frame of the validation sequence. And it scores the mAP of 0.5765.

Last but not least, the imbalanced dataset combination slowed down the training process. To be specific, we cannot use a learning rate of 1e-3 to accelerate and optimize our training.

Table 9: Comparison of different models

model	batch size	lr	lr_decay	max_epoch	dataset	Map
Res-50	1	1.00E-04	3	6	imagenet_vid+imagenet_det(1)	0.4064 (1)
Res-50	1	1.00E-03	5	11	imagenet_vid+imagenet_det(1)	0.6015 (2)
Res-50	1	1.00E-04	5	11	imagenet_vid+imagenet_det(2)	0.5765 (3)
Res-18	1	1.00E-04	3	10	imagenet_vid+imagenet_det(1)	0.2783(4)

Notedly, we would evaluate the D&T network on the MOT dataset. After this comparison (table 9), our model is fine-tuned with images, videos, and their corresponding annotations from 2DMOT2015 and Pascal VOC datasets.

4.2.2. SSD to Object Detection

Inspired by detection-based tracking, we released the second experiment by replacing the R-FCN detector with the SSD model on the D&T architecture. Since the high-precision detector, like SSD, can get the precise bounding boxes for each frame of the testing video. And the tracking approach, which links the frame-level detection [22], can produce high-precision detections at the video level.

The tracking approach of the D&T model could cope with the detection responses simply and effectively. It is essential to explain the tracking procedure. The author in [22] defined a class link score that combines detection and tracking. It determines whether the tracklets should be linked to the object tube. With the Viterbi algorithm, we can figure out the best path to maximize the score for a given video duration. In order to make the tracker robust, the tracking approach adopts the highest-scoring re-weighting with a non-maximum suppression effect.

In short, the second implementation focuses on improving detection accuracy.

Training and Testing of the original version

In the paper [27] and its Pytorch implement, we found an SSD detector using VGG16 as the backbone. In addition, the author fine-tuned the model by using a batch size of 32, a learning rate of 1e-3, a momentum of 0.9, and a weight decay of 5e-4 in stochastic gradient descent (SGD) training.

For a fair comparison, the author filtered the output boxes by setting the minimum score to 0.01, NMS overlap to 0.45, and top k to 200. Parsed predictions are evaluated against the ground truth objects with the evaluation metric as the mean average precision (mAP). The best checkpoint of the SSD model scores the mAPs across 20 classes as 0.771. It came from epoch 186 with a validation loss of 2.515.

Training and Testing of our models

In this experiment, we trained the SSD detector on our dataset combination instead of on the Pascal VOC. For training parameters, we adopted a batch size of 8 images, a learning rate of 1e-3, a momentum of 0.9, and a weight decay of 5e-4. Then we could get the best model with a loss of 3.040 at the 55^{th} epoch. With the same testing method, our SSD network achieves the mean average precision (mAP) spanning nine classes as 0.551.

After changing the batch size of pictures from 8 to 32 and retaining other parameters, we retrained the SSD model for 95 epochs. When evaluating the best model with a loss of 2.870, the mAP across the nine categories rose to 0.614.

Since our mAPs are less than the original experimental results. We decided to fine-tune the pretrained SSD detector on our dataset combination. The pretrained model had been generated through the 186-epoch training on Pascal VOC. After fine-tuning this model on the dataset combination for 328 epochs, we got the best checkpoint with a validation loss of 2.730. For a fair comparison, the fine-tuned SSD model would be assessed with MOT17DET in evaluation.

4.2.3. Fast Online Object Tracking and Segmentation

In [26], the author trained the SiamMask model on the pre-processed datasets of Youtube-VOS, COCO, ImageNet-DET, and ImageNet-VID. And, he evaluated the SiamMask on VOT, DAVIS, and Youtube-VOS.

Our third implementation aims at achieving multiple-object detection and tracking jointly and automatically with the single-object-tracking model, SiamMask network. The content below details the experimental steps.

The first step is to convert the testing video into individual frames. Since the trained SiamMask operates online, producing class-agnostic object segmentation masks and rotated bounding boxes at 55 frames per second. We converted the input videos into frames at the same frequency as 55 frames per second.

Secondly, we modified the relevant codes, such as" demo.py," "test.py" (in "tools" folder) and "custom.py" (in "experiments/siammask_sharp" folder) for the tolerance of multiple-object tracking,

Thirdly, the SSD detector is imported for classification and initialization automatically. Specifically, the SSD model initializes objects on the input frame every 30 frames to keep detection accurate. We stored the bounding boxes and labels of the target objects to reduce the execution time. During this initialization, our programs would compute the IOU that the tracking bounding boxes overlap with adjacent SSD outputs. According to IOU, the "target" list, which stores the tracking object information, would delete or add the SSD detections every certain interval. In detail, if the IOU is below the specified threshold, our program will remove the corresponding tracking boxes. And SSD model detections would be appended if their IOU with the previous tracking boxes exceeds the threshold. For correcting the results, the "target" list would be cleared and reinitialized with the SSD detections every 60 frames.

By converting the plotted subsequent frames, the final video presents to end this experiment

4.3. Evaluation

The evaluation of multiple object tracking is hard to be defined and carried out with the right mothed. Like original PyTorch testing on the D&T model, it measures the detection performance rather than the MOT performance. When we evaluated the D&T model on each snippet of validation sets, the evaluation script operates each frame instead of multiple adjacent frames.

Thence, it is desirable to test three models on the same dataset and to compare the results processed by the same multiple object tracking metrics.

4.3.1. Evaluation dataset

Due to the heavy workload of tagging the tracking ID, the evaluation video was chosen as the MOT17DET described in Section 4.1.1. In this way, we do not need to copy with the tracking ID. And annotations are only prepared for pedestrians in MOT17DET [28].

On the official website [21], the ground truths of the testing set are not publicly available. Consequently, the experimental results are obtained through computing the training set and the corresponding ground truth. Although our models can detect and track multi-class objects, few datasets contain multiple category objects for multi-object tracking

evaluation. Thereby, we blocked our models from dealing with other class objects other than pedestrians when we evaluated networks on MOT17DET.

4.3.2. Evaluation metric

Evaluation metric of multiple object tracking is so significant that it offers criteria for the fair comparison between MOT models. Almost MOT implementations adopt the 'tracking-by-detection' strategy. Therefore, we can divide MOT evaluation metrics into two groups: detection and tracking evaluation metrics.

In MOT17DET, its evaluation scripts employ the MATLAB to produce the detection measures, such as recall, precision, FAR, GT, TP, FP, FN, MODA, and MODP. True positives and false positives present a total number of true positives and the number of false positives separately. In other words, the true positives count the outputs which match annotated targets, and the number of the rest outputs is a false positive. The target objects which are not identified as the positive would be collected as missed targets or false negatives. This distinguishing process is executed considering the dissimilarity calculation. In general, we value dissimilarity through measuring the IOU between the ground truth bounding and the rectangle outputs.

Recall and precision indicate the percentages of detected targets over ground-truth detections and correctly detect targets over total result detections respectively. FAR states the number of false alarms per frame. And GT describes the total number of ground truth boxes in the testing dataset.

MOTA and MOTP belong to the tracking evaluation metrics. MOTA to evaluate a tracker's performance with false negatives, false positives, and mismatch rates while MOTP states the average dissimilarity between all true positives and their corresponding ground truth targets.

Since our evaluation does not use tracking identifiers, we have skipped the interpretation of ID-related evaluation indicators such as ID F1 score, ID precision, ID call, and ID switching. Moreover, without regard to identity, MODA / MODP is almost the same as MOTA / MOTP.

4.3.3. Testing results

4.3.3.1. The first evaluation

Before fine-tuning the models like the second evaluation, we evaluated D&T, the unifying SSD tracking network, and the modified SiamMask architecture directly on MOT17DET. The evaluation resulted in the tables below.

Adjusting the parameters could decide the performance of MOT models. The jump gap indicates the threshold for the stale tube. We set the longest gap to 100 or 25 during

the evaluation of the D & T model. The gap in SSD detection and tracking implement was limited as 15. Moreover, in the SiamMask experiments, the minimal score of SSD detection was adjusted as 0.45 to get relatively better results.

Recently, the mean average precision emerges as one of the most popular evaluation metrics. When we compared the corresponding mAPs, the D&T model outperformed the other two models. Since the current results have not reached our requirement, the models were retrained before the second evaluation.

Table 10: The testing result of D&T network

	Average precision	Rcll	Prcn	FAR	GT	TP	FP	FN	MOTA	MOTP	num_frames
MOT17-02	0.3315	33.00%	58.50%	2.84	7273	2398	1703	4875	9.60%	74.30%	600
MOT17-04	0.1639	15.50%	75.30%	1.4	28909	4486	1473	24423	10.40%	70.60%	1050
MOT17-05	0.3486	57.00%	40.90%	3.6	3659	2086	3011	1573	-25.30%	75.30%	837
MOT17-09	0.5159	53.70%	70.40%	1.35	3148	1691	710	1457	31.20%	77.30%	525
MOT17-10	0.1571	15.60%	46.10%	2.55	9159	1425	1669	7734	-2.70%	68.80%	654
MOT17-11	0.2658	25.90%	79.80%	0.44	6122	1585	400	4537	19.40%	79.60%	900
MOT17-13	0.0909	2.80%	33.50%	0.6	8033	225	447	7808	-2.80%	69.40%	750
Overall	0.2258	20.90%	59.60%	1.77	66387	13896	9413	52491	6.80%	73.60%	5316

Table 11: The testing result of SSD detection and tracking implement

	Average Precision	Rcll	Prcn	FAR	GT	TP	FP	FN	MOTA	MOTP	num_frames
MOT17-02	0.1351	17.20%	20.30%	8.21	7273	1251	4924	6022	-50.05%	69.40%	600
MOT17-04	0.0909	6.70%	84.90%	0.29	25451	1696	301	23755	5.50%	69.30%	1050
MOT17-05	0.2697	42.40%	57.00%	0.73	1904	807	610	1097	10.30%	71.90%	837
MOT17-09	0.4391	47.90%	69.10%	1.29	3148	1507	675	1641	26.40%	72.00%	525
MOT17-10	0.0909	9.40%	59.00%	0.53	5280	498	346	4782	2.90%	68.90%	654
MOT17-11	0.1781	29.80%	72.80%	0.33	2644	787	294	1857	18.60%	76.10%	900
MOT17-13	0.0909	8.00%	44.30%	0.38	2835	227	285	2608	-2.00%	60.20%	750
Overall	0.1408	11.10%	47.70%	1.4	61189	6773	7435	54416	-1.10%	70.70%	5316

Table 12: The testing result of SiamMask

	Average Precision	Rcll	Prcn	FAR	GT	TP	FP	FN	MOTA	MOTP	num_frames
MOT17-02	0.2124	20.70%	62.50%	1.51	7288	1506	904	5782	8.30%	75.80%	600
MOT17-04	0.1736	11.50%	87.40%	0.46	28936	3336	480	25600	9.90%	78%	1050
MOT17-05	0.4801	54.70%	65.70%	1.25	3667	2006	1047	1661	26.20%	72%	837
MOT17-09	0.5097	51.40%	83.60%	0.61	3154	1622	319	1532	41.30%	72%	525
MOT17-10	0.0741	16.30%	66.80%	1.14	9172	1498	744	7674	8.20%	75%	654
MOT17-11	0.389	47.30%	67.10%	1.58	6137	2900	1424	3237	24.10%	75%	900
MOT17-13	0.0909	4.20%	70.80%	0.15	6698	279	4	0	2.40%	73%	750
Overall	0.1675	20.20%	72.30%	0.95	65052	13147	5033	51905	12.50%	75%	5316

4.3.3.2. The second evaluation

After retraining the D&T model on the dataset combination of Pascal VOC and MOT2D2015, we obtained the fine-tuned D&T network. In addition, the SSD detector had been retrained across the entire new data set. With these fine-tuned models, we executed the evaluation of MOT17DET videos again. The following tables report the second evaluation results.

Considering the parameter settings, they had been adjusted and optimized for different models. In the evaluations of D&T architecture and SSD tracking implement, the jump gaps for filtering tubes were set as 200. Furthermore, in the SiamMask experiments, the minimal score of SSD detection was set as 0.3 to get more detection.

After being retrained, the D&T model contained more checkpoints. Due to the memory limitations in GPU hardware, it cannot perform the multiple objects tracking with the jump gap of 200 on MOT17-04 and MOT17-13 videos. Moreover, the video platforms of these two videos are not commonly available. MOT17-04 is filmed from an elevated place, while MOT17-13 is recorded in a car. Based on these factors, we skipped the experiments on the MOT17-04 and MOT17-13 videos. Using the rest five videos, we completed the second experiment.

Analysis of improvement

Through the second evaluation, we observe the fine-tuned models achieved better results. To identify on which video the improvement was relatively significant, we made a comparison between the six tables. From tables 10 and 13, it can be found that the D&T model outputted higher quality results on MOT17-10 and MOT17-11 videos. These two videos were all taken from the moving platforms. The recall result of MOT17-10 grew from 15.6% to 33.40%. While the recall result of MOT17-11 grew from 25.90% to 54.60%. The different growths caught our attention. We deduced that this might be released by illumination. In detail, MOT17-11 is filmed indoor while MOT17-10 is logged outdoor at night.

From the table 11 and 14, we can see that the fine-tuned SSD detector benefits all the MOT17 videos, especially the MOT17-11. The average precision on MOT17-11 had increased and reached the same stage as the average precision on MOT17-05 and MOT17-09. Unfortunately, the average precisions on MOT17-02 and MOT17-10 were kept under thirty percent. The unsatisfying results might be caused by the dim light and the crowd density in the MOT17-02 and MOT17-10 videos.

Since our SiamMask model employed the SSD detector to initialize the target object location, its output was influenced by the performance of the detector. Through the comparison between tables 12 and 15, we noticed the recall on MOT17-10 had been raised by using the fine-tuned detector.

Table 13: The testing result of D&T network

	Average Precision	Rcll	Prcn	FAR	GT	TP	FP	FN	MOTA	MOTP	num_frames
MOT17-02	0.3458	35.10%	61.00%	2.72	7273	2551	1632	4722	12.60%	75.80%	600
MOT17-05	0.4861	59.60%	46.80%	2.95	3659	2179	2473	1480	-8.00%	75.30%	837
MOT17-09	0.5119	50.20%	71.50%	1.2	3148	1579	628	1569	30.20%	78.50%	525
MOT17-10	0.3306	33.40%	71.00%	1.92	9159	3062	1253	6097	19.80%	72.70%	654
MOT17-11	0.4735	54.60%	58.80%	2.61	6122	3344	2347	2778	16.30%	80.70%	900
Overall	0.4091	43.20%	60.40%	2.37	29403	12715	8333	16688	14.90%	76.60%	5316

Table 14: The testing result of SSD detection and tracking implement

	Average Precision	Rcll	Prcn	FAR	GT	TP	FP	FN	MOTA	MOTP	num_frames
MOT17-02	0.23	24.10%	41.00%	4.21	7273	1752	2517	5521	-10.50%	69.80%	600
MOT17-05	0.5703	60.50%	39.00%	4.14	3659	2213	3467	1446	-34.30%	72.20%	837
MOT17-09	0.5195	53.60%	55.80%	2.55	3148	1688	1338	1460	11.10%	72.60%	525
MOT17-10	0.1613	19.60%	46.90%	3.1	9159	1791	2026	7368	-2.60%	68.90%	654
MOT17-11	0.5096	55.80%	54.50%	3.17	6122	3418	2851	2704	9.30%	74.40%	900
Overall	0.3294	36.90%	47.10%	3.47	29403	10862	12199	18541	-4.50%	72.00%	3516

Table 15: The testing result of SiamMask

	Average Precision	Rcll	Prcn	FAR	GT	TP	FP	FN	MOTA	MOTP	num_frames
MOT17-02	0.2503	20.80%	83.30%	0.51	7288	1516	305	5772	16.60%	75.80%	600
MOT17-05	0.4731	53.40%	68.80%	1.06	3667	1957	886	1710	29.20%	72.80%	837
MOT17-09	0.394	46.50%	79.40%	0.72	3154	1466	380	1688	34.40%	76.60%	525
MOT17-10	0.2476	20.90%	80.20%	0.73	9172	1921	475	7251	15.80%	75.70%	654
MOT17-11	0.4248	40.40%	82.30%	0.59	6137	2480	532	3657	31.70%	75.60%	900
Overall	0.329	31.70%	78.40%	0.73	29418	9340	2578	20078	23.00%	75.20%	3516

Comparison among models

According to the average precisions in tables 13, 14, and 15, the winner is the D&T network. When we concentrate on recall and MOTP, the D&T network also outperforms among models. These competitive metrics results state that the predictions from the D&T model are most similar to the ground truth target objects. However, the SiamMask model has its advantage when we compare their precision, false alarm, and MOTA.

Although the annotated video cannot be displayed, we can display some frame detection. The D & T model operates on the MOT17-02, MOT17-09, and MOT17-05 datasets and outputs the following pictures. The retrained network produces more accurate results than the model originally evaluated. As a consequence, more bounding boxes with class labels and prediction confidence appear in the generated image.

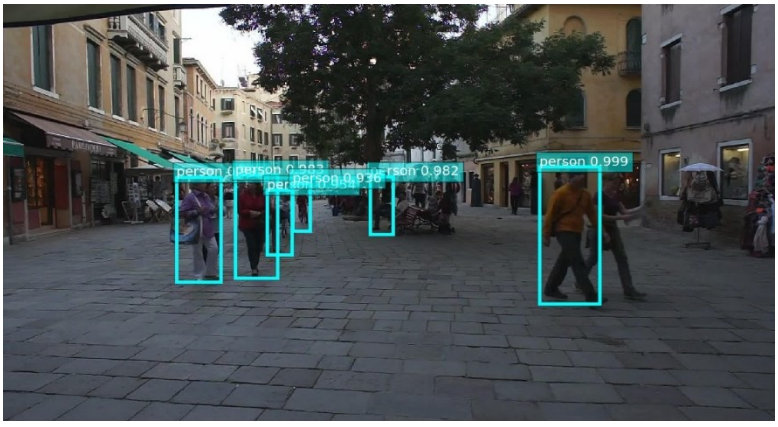

Figure 30. D&T model predicted frame from MOT17-02 video

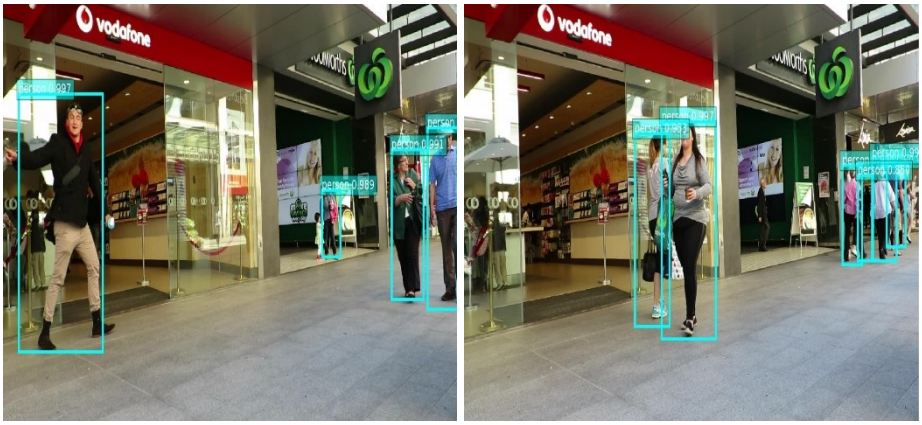

Figure 31. D&T model predicted frame from MOT17-09 video

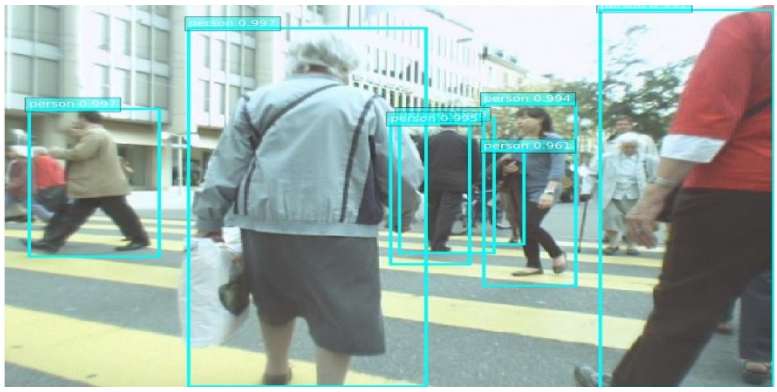

Figure 32. D&T model predicted frame from MOT17-05 video

SSD tracking model processes the MOT17-11 and MOT17-09 datasets and outputs the pictures below. The bounding boxes with predicted labels and confidences are plotted for target objects in these figures.

Figure 33. SSD tracking model predicted frame from MOT17-11 video

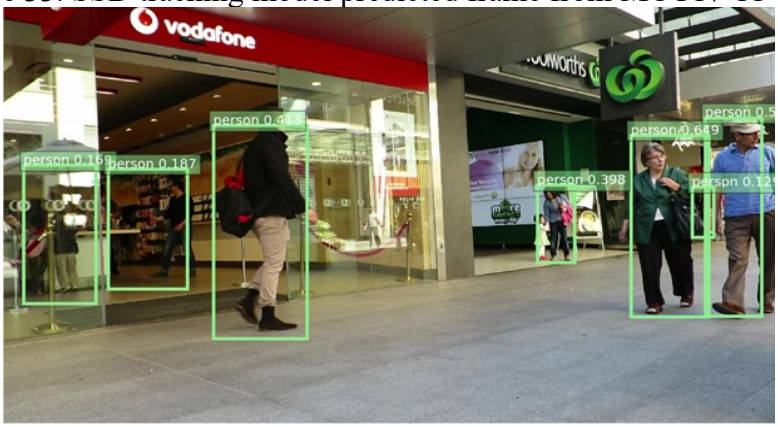

Figure 34. SSD tracking model predicted frame from MOT17-09 video

Modified SiamMask model copes with the MOT17-05, MOT17-11and MOT17-10 datasets and produces the following images. For each detected object, a bounding box with the class label is plotted based on its corresponding masks as figures show.

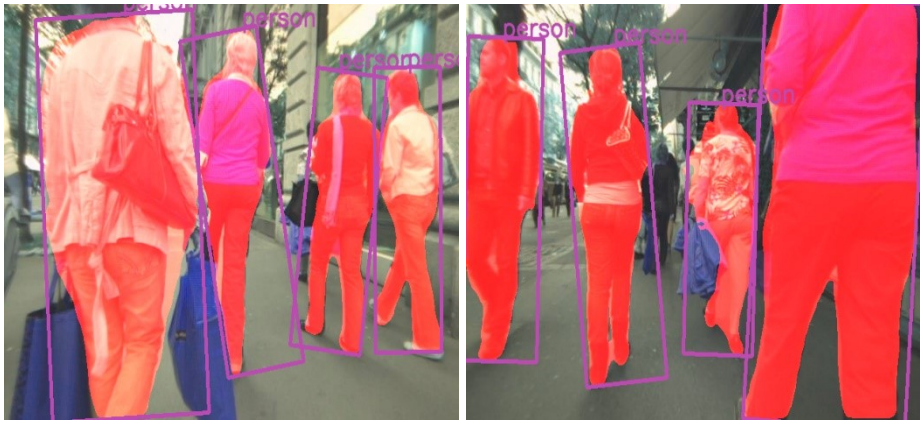

Figure 35. Modified SiamMask model predicted frame from MOT17-05 video

Figure 36. Modified SiamMask model predicted frame from MOT17-11 video

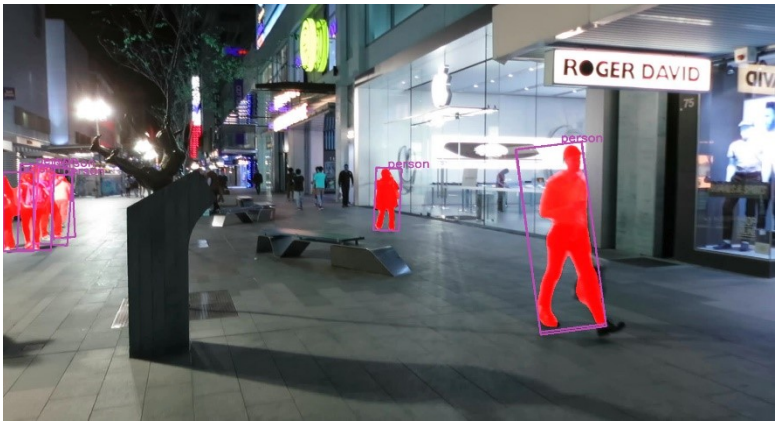

Figure 37. Modified SiamMask model predicted frame from MOT17-10 video

According to these predicted frames, we found the D&T and SSD models detect more target objects than the SiamMask network. Moreover, the D&T model supplies more

smooth tracking trajectories when compared with SSD tracking models. For the detected object, SiamMask produces segments and then builds the rotated bounding boxes from the binary masks. If we concentrate on the detected objects, SiamMask can achieve better results through outputting more accurate bounding boxes.

4.3.3.3. Further analysis

Regarding the influence of the width and height for the bounding boxes, the ground truth and the corresponding predicted bounding boxes have been divided into small- and large-scale groups. Therefore, we can conduct an evaluation and produce the results, such as the precision, recall, and average precision for the small- and large-scale groups, respectively.

When we discuss grouping details, it is necessary to discuss the scale unit, the clustering strategy, and the number of clusters. Since the resolutions of MOT17DET, excluding 'MOT17-05, are the same as 1920×1080. As the figure below shows, K-means clustering approach partitions ground truth bounding boxes of ‘MOT17-02', 'MOT17-05', 'MOT17-09', 'MOT17-10', and 'MOT17-11' datasets into small- and large-scale groups according to their resolutions. The purple and yellow points belong to the small- and large-scale groups separately. Dividing the datasets into three groups:' small," medium,' and' large' could lead to more precise results. However, if we split the dataset into three groups, some groups will be empty due to the few datasets.

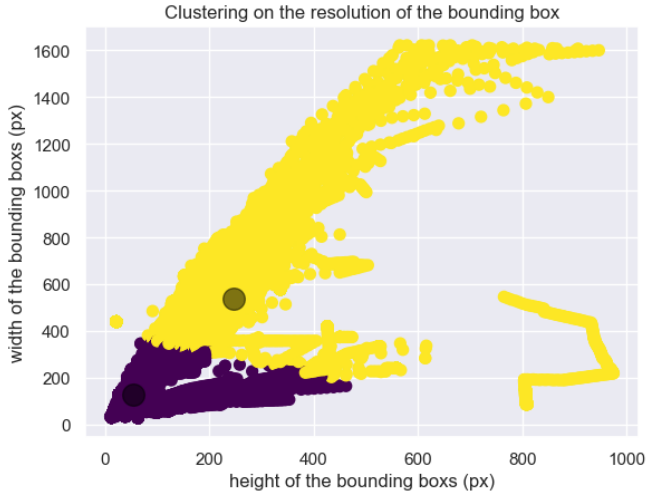

Figure 38. Kmeans cluster presentation distributed on the datasets of 'MOT17-02', 'MOT17-05', 'MOT17-09', 'MOT17-10' and 'MOT17-11'. Each point stands for a ground truth bounding box. By clustering based on the box scale, the points are grouped as small- and large-scale sets, correspondingly colored as purple and yellow. Two black points show the centers of large- and small- scale groups.

After dividing the bounding boxes from the ground-truth datasets as the small- and large-scale groups, we plotted the bounding boxes belonging to two groups on the same frame to state their differences. The plotted images are presented as the figures below. In each pair of images, the left-side image presents the specified frame with the large-scale bounding boxes, while the right-side picture shows the same frame with the small-scale bounding boxes.

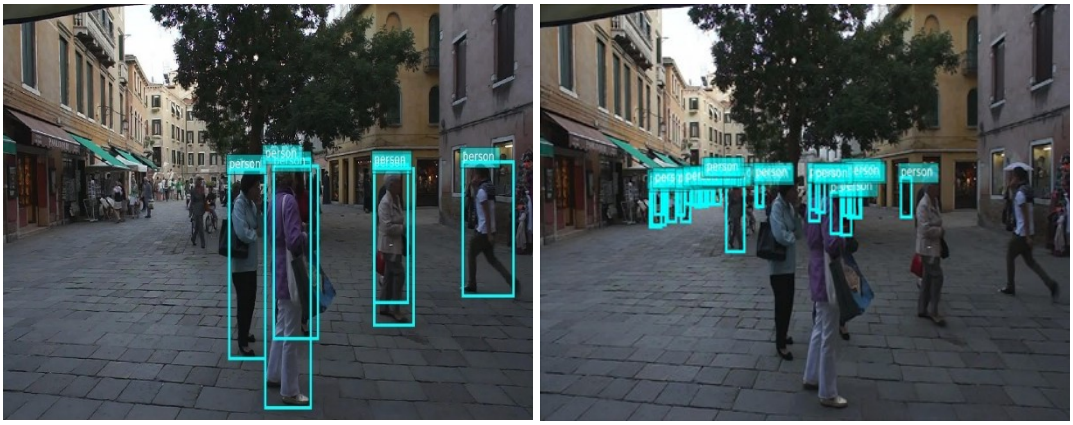

Figure 39. The large-scale and small-scale bounding boxes plotted on the 267th frame from 'MOT17-02' dataset

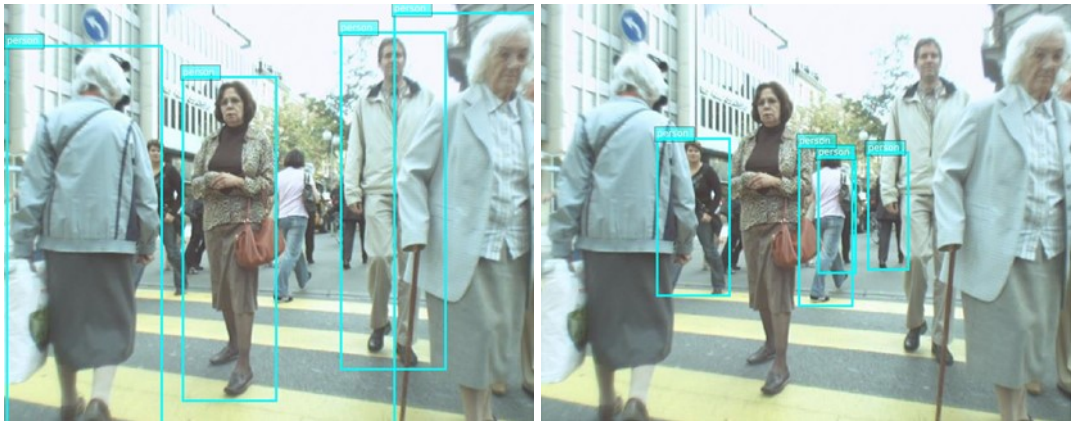

Figure 40. The large-scale and small-scale bounding boxes plotted on the 292nd frame from 'MOT17-05' dataset

Figure 41. The large-scale and small-scale bounding boxes plotted on the 453rd frame from 'MOT17-09' dataset

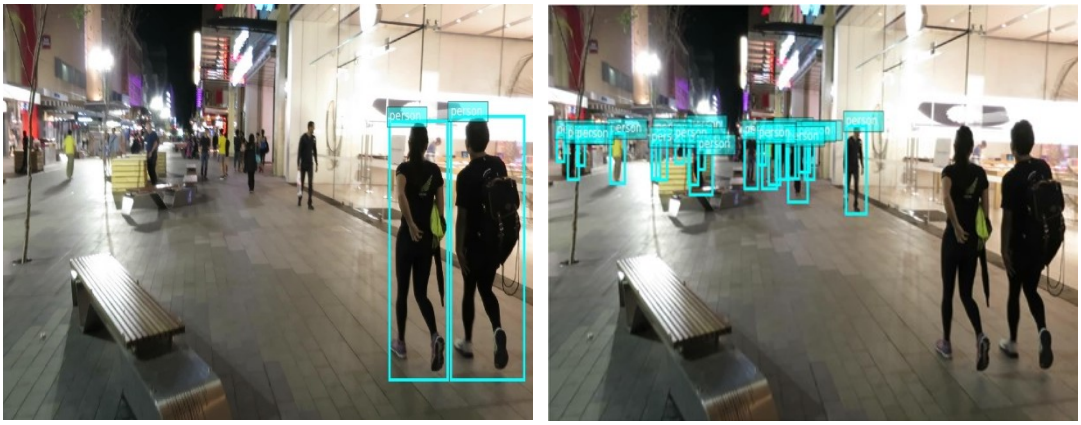

Figure 42. The large-scale and small-scale bounding boxes plotted on the 196th frame from 'MOT17-10' dataset

Figure 43. The large-scale and small-scale bounding boxes plotted on the 255th frame from 'MOT17-11' dataset

After clustering all bounding boxes from ground truth files, we applied a fitted K-means model to predict the labels for the detection boxes and then to group boxes. Depending on the labels, we matched and compared the boxes in the same clusters. To be specific, a small-scale detection box is evaluated with the small-scale ground truth bounding box. At the same time, the large-scale detection results are compared with corresponding ground truth datasets.

For the small-scale group, the results do not reflect well as we expect. The D&T model achieves the best average precision as 0.3093. For more details, see Tables 16, 17, and 18.

Table 16: Small-scale testing result of D&T network

	Average Precision	Rcll	Prcn
MOT17-02	0.237	22.00%	49.80%
MOT17-05	0.4249	51.50%	39.50%
MOT17-09	0.308	33.90%	53.30%
MOT17-10	0.3341	34.70%	71.60%
MOT17-11	0.2576	27.50%	36.60%
Overall	0.3093	32.10%	52.70%

Table 17: Small-scale testing result of SSD detection and tracking implement

	Average Precision	Rcll	Prcn
MOT17-02	0.1413	13.50%	29.90%
MOT17-05	0.3959	44.50%	27.10%
MOT17-09	0.3865	40.70%	43.90%
MOT17-10	0.149	15.00%	40.90%
MOT17-11	0.0515	16.30%	36.10%
Overall	0.1976	20.30%	34.10%

Table 18: Small-scale testing result of SiamMask

	Average Precision	Rcll	Prcn
MOT17-02	0.0673	10.30%	70.30%
MOT17-05	0.295	37.60%	59.60%
MOT17-09	0.2153	23.40%	70.40%
MOT17-10	0.1692	17.40%	78.30%
MOT17-11	0.0909	0.20%	21.40%
Overall	0.0712	16.70%	69.90%

For large groups, the results of Tables 19, 20, and 21 meet our requirements. The SSD detection with tracking function can realize the best average accuracy of 0.6725. Simultaneously, the average accuracy obtained by D & T network is 0.5798, while the average accuracy of SiamMask is 0.5831.

Table 19: the large-scale testing result of D&T network

	Average Precision	Rcll	Prcn
MOT17-02	0.7792	87.20%	73.90%
MOT17-05	0.655	70.70%	57.90%
MOT17-09	0.6273	68.80%	86.20%
MOT17-10	0.1681	17.20%	50.80%
MOT17-11	0.5564	69.30%	65.60%
Overall	0.5798	67.40%	67.90%

Table 20: Large-scale testing result of SSD detection and tracking implement

	Average Precision	Rcll	Prcn
MOT17-02	0.5698	64.60%	53.80%
MOT17-05	0.7647	87.10%	62.10%
MOT17-09	0.6198	68.50%	66.80%
MOT17-10	0.6257	65.00%	67.80%
MOT17-11	0.6848	77.70%	56.40%
Overall	0.6725	74.40%	59.10%

Table 21: Large-scale testing result of SiamMask

	Average Precision	Rcll	Prcn
MOT17-02	0.6128	63.70%	88.80%
MOT17-05	0.657	79.70%	74.40%
MOT17-09	0.6406	71.70%	77.30%
MOT17-10	0.506	55.30%	82.20%
MOT17-11	0.5932	64.10%	81.40%
Overall	0.5831	66.70%	80.40%

From experiments, we could see that our models work better on the large-scale dataset than on the small-scale dataset. This phenomenon hints for us to improve the MOT performance on small-scale objects to optimize our network.

4.4. Discussion

The previous content reflects the advantages and disadvantages of our network. It worths us to explore and to analyze these experiments. Furthermore, we received a memorable lesson from the weakness of our models.

Three shortcomings related to the training process of the D & T model should be addressed. First, the pre-trained model R-FCN [13] provided by the author is not suitable for the desired target object. As such, either training or evaluation did not involve the pretrained model. And we used ResNet50 as the backbone to train the D&T network. In future work, the new D&T model could start with the pretrained R-FCN model, including suitable objects. The second weakness is the unbalanced class datasets. Though we optimized the batch scheme, there was no obvious improvement in results. It left us to adjust the loss function and to regulate the batch plan. The last but least, the gap parameter needs

careful selection when performing an evaluation. This parameter determines the length of trajectories. If we set the gap too long, some vanishing object bounding boxes will remain in the tracking tube. In contrast, short-gap tubes cannot help us to grasp many target objects like long-gap tubes. Therefore, short-gap tubes will result in lower recalls. For long-term video detection, its algorithm needs to be explored and developed.

In the second experiment, the SSD detector took the place of the R-FCN detection model. Although its average accuracy in MOT evaluation of some videos, such as MOT17-05, MOT17-09, and MOT17-11, satisfies the users. However, the target objects don not move smoothly on the trajectories as on the D&T trajectories. In my view, it is because its training process lacks correlation loss. Adding a correlation loss and training the SSD detector with the weighted loss is an innovative solution.

To modify SiamMask to accept multiple objects, we initialized the bounding box for each target object with SSD. In this way, the model will reinitialize these objects when a certain interval comes. Specifically, at certain moments, bounding boxes with high confidences in the detection of specific frames should be retained or added. Otherwise, the tracklets will cancel them. As the tracker is trained on a short-term video dataset and lacks a correction mechanism for a long-term tracking tube, it is difficult to determine the length of the interval. Therefore, improving the video adaptability of existing SiamMask models requires us to explore.

Furthermore, the section- ‘Further analysis’ reminds us to improve the MOT accuracy on the small-scale objects to improve the overview performances for our networks.

5. Conclusion and future work

We split this section into two parts: conclusion and future work. The conclusion presents our proposal, approaches, process, and evaluation results. Future work discusses the possible MOT directions.

5.1. Conclusion

It is easy to reveal that MOT relies on two basic bench works: object detection and single object tracking. Therefore, the typical object detection model-SSD, single-object tracking network-SiamMask, and the multiple-object tracking architecture D&T model had been researched and used.

To apply MOT on the required target objects, we firstly retrained the D&T model on the appropriate dataset combination. In the second experiment, we considered the influences on MOT due to the varying detector. So, the SSD detector takes the place of the R-FCN detector. And it cooperates with the Viterbi tracking approach to achieve multiple object tracking. Although the modified SSD method produces accurate results with short-term videos, the generated tubes are not as smooth as the previous work. In the third experiment, we transformed the SiamMask into a multiple-object tracking method. Furthermore, it is developed from detection-free tracking to detection-based tracking with the support of an SSD detector. Because it was originally designed for short-term video, there is a lack of proper data association methods for video adaptation.

In the experiment, we performed three models on MOT17 videos. Then evaluation metrics are calculated for the standard results. Moreover, the plotted frames produced from each model present together to show the specific performance.

5.2. Future directions

In the future, we could optimize the MOT model in terms of video adaptation, crowd density, and completeness of objection representation, as well as implementation under multiple cameras.

In previous experiments, we have discussed the problem that short-term tracking models do not operate well with long-term videos. Therefore, designing algorithms to extend the trajectory is a crucial issue.

Detection is always affected by small-scale objects in the crowd. For traditional object detection methods, a trade-off needs to be made between crowd density and completeness of object representation. Although some creative methods have been invented, such as "finding small faces" [29], how to combine them with MOT models is still our concern.

The evaluation dataset (MOT17DET) was taken from multiple viewpoints. Correlating multi-view data of the same video can restore stereo vision, which makes us feel real.

Referring to forward-looking projects, MOT cooperates with the 3D scene understanding, other computer vision tasks, such as pose analysis and the next action prediction.

References

[1] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (1809). Deep learning for generic object detection: A survey. *International Journal of Computer Vision, 1-58.*

[2] 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).*

[3] 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.*

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

[5] Vojir, T., Noskova, J., & Matas, J. (2014). Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters, 49, 250-258.*

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

[7] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).*

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916.*

[9] Wu, Y., Lim, J., & Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9), 1834-1848.*

[10] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M. M., Hicks, S. L., & Torr, P. H. (2015). Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence, 38(10), 2096-2109.*

[11] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., & Torr, P. H. (2016, October). Fully-convolutional siamese networks for object tracking. *In European conference on computer vision (pp. 850-865). Springer, Cham.*

[12] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Zhao, X., & Kim, T. K. (2014). Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618.*

[13] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. *In Advances in neural information processing systems (pp. 379-387).*

[14] Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training region-based object detectors with online hard example mining. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 761-769).*

[15] Saha, S., Singh, G., Sapienza, M., Torr, P. H., & Cuzzolin, F. (2016). Deep learning for detecting multiple space-time action tubes in videos. *arXiv preprint arXiv:1608.01529.*

[16] Pinheiro, P. O., Lin, T. Y., Collobert, R., & Dollár, P. (2016, October). Learning to refine object segments. *In European Conference on Computer Vision (pp. 75-91). Springer, Cham.*

[17] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, A. Khajenezhad, A. Salahledin, A. Soltani-Farani, A. Zarezade, A. Petrosino, A. Milton, B. Bozorgtabar, B. Li, C. S. Chan, C. Heng, D. Ward, D. Kearney, D. Monekosso, H. C. Karaimer, H. R. Rabiee, J. Zhu, J. Gao, J. Xiao, J. Zhang, J. Xing, K. Huang, K. Lebeda, L. Cao, M. E. Maresca, M. K. Lim, M. E. Helw, M. Felsberg, P. Remagnino, R. Bowden, R. Goecke, R. Stolkin, S. Y. Lim, S. Maher, S. Poullot, S. Wong, S. Satoh, W. Chen, W. Hu, X. Zhang, Y. Li, Z. Niu, "The Visual Object Tracking VOT2013 challenge results", *Proc. IEEE Int. Conf. Comput. Vis. Workshops, pp. 98-111, 2013.*

[18] Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., & Huang, T. (2018). Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327.*

[19]: Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. *In European conference on computer vision (pp. 740-755). Springer, Cham.*

[20]: ActEV: Activities in Extended Video: *https://actev.nist.gov/*

[21]: MOT17Det dataset: *https://motchallenge.net/data/MOT17Det/*

[22]: Feichtenhofer, C., Pinz, A., & Zisserman, A. (2017). Detect to track and track to detect. *In Proceedings of the IEEE International Conference on Computer Vision (pp. 3038-3046).*

[23]: Awad, G., Butt, A., Curtis, K., Lee, Y., Fiscus, J., Godil, A., ... & Kraaij, W. (2018, November). Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search.

[24]: Liang, J., Jiang, L., Niebles, J. C., Hauptmann, A. G., & Fei-Fei, L. (2019). Peeking into the future: Predicting future person activities and locations in videos. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5725-5734).*

[25]: Figure Eight Datasets: *https://www.figure-eight.com/dataset/*

[26]: Wang, Q., Zhang, L., Bertinetto, L., Hu, W., & Torr, P. H. (2019). Fast online object tracking and segmentation: A unifying approach. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1328-1338).*

[27]: Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. *In European conference on computer vision (pp. 21-37). Springer, Cham.*

[28]: Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831.*

[29]: Hu, P., & Ramanan, D. (2017). Finding tiny faces. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 951-959).*

[30]: Xu, L., Ren, J. S., Liu, C., & Jia, J. (2014). Deep convolutional neural network for image deconvolution. *In Advances in neural information processing systems (pp. 1790-1798).*

[31]: Lindeberg, T. (2012). Scale invariant feature transform.

[32]: N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. *In Advances in Neural Information Processing Systems, pages 809–817, 2013.*

[33]: Wang, N., Li, S., Gupta, A., & Yeung, D. Y. (2015). Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587.*

[34]: Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., & Torr, P. H. (2016). Staple: Complementary learners for real-time tracking. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1401-1409).*

[35]: Nam, H., & Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4293-4302).*

[36]: Cui, Z., Xiao, S., Feng, J., & Yan, S. (2016). Recurrently target-attending tracking. *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1449-1458).*

[37]: Held, D., Thrun, S., & Savarese, S. (2016, October). Learning to track at 100 fps with deep regression networks. *In European Conference on Computer Vision (pp. 749-765). Springer, Cham.*

[38]: Cheng, J., Tsai, Y. H., Hung, W. C., Wang, S., & Yang, M. H. (2018). Fast and accurate online video object segmentation via tracking parts. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7415-7424).*

[39]: Yang, L., Wang, Y., Xiong, X., Yang, J., & Katsaggelos, A. K. (2018). Efficient video object segmentation via network modulation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6499-6507).*

[40]: DFT video demo: *https://www.youtube.com/watch?v=I_iOVrcpEBw&feature=youtu.be*

[41]: DBT video demo: *https://www.robots.ox.ac.uk/~vgg/research/detect-track/videos/12.mp4*

[42]: Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision, 88(2), 303-338.*

[43]: Girshick, Ross. Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.*

[44]: Kalal, Z., Mikolajczyk, K. and Matas, J., 2011. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence, 34(7), pp.1409-1422.*

[45]: Wang, Lijun, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. *In Proceedings of the IEEE international conference on computer vision, pp. 3119-3127. 2015.*

[46]: Pytorch implementation of SSD model: *https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection*

[47]: Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision, 111(1), 98-136.*

[48]: Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing, 50(2), 174-188.*

[49]: Comaniciu, D., Ramesh, V., & Meer, P. (2000, June). Real-time tracking of non-rigid objects using mean shift. *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662) (Vol. 2, pp. 142-149). IEEE.*

[50]: Funk, N. (2003). A study of the Kalman filter applied to visual tracking. *University of Alberta, Project for CMPUT, 652(6).*

[51]: Yu, Q., Dinh, T. B., & Medioni, G. (2008, October). Online tracking and reacquisition using co-trained generative and discriminative trackers. *In European conference on computer vision (pp. 678-691). Springer, Berlin, Heidelberg.*

[52]: Danelljan, M., Häger, G., Khan, F. S., & Felsberg, M. (2016). Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence, 39(8), 1561-1575.*

[53]: Danelljan, M., Shahbaz Khan, F., Felsberg, M., & Van de Weijer, J. (2014). Adaptive color attributes for real-time visual tracking. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1090-1097).*

[54]: Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2012, October). Exploiting the circulant structure of tracking-by-detection with kernels. *In European conference on computer vision (pp. 702-715). Springer, Berlin, Heidelberg.*

[55]: Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence, 37(3), 583-596.*

[56]: Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010, June). Visual object tracking using adaptive correlation filters. *In 2010 IEEE computer society conference on computer vision and pattern recognition (pp. 2544-2550). IEEE.*

[57]: Danelljan, M., Häger, G., Khan, F. S., & Felsberg, M. (2016). Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence, 39(8), 1561-1575.*

[58]: Possegger, H., Mauthner, T., & Bischof, H. (2015). In defense of color-based model-free tracking. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2113-2120).*

[59]: Tuzel, O., Porikli, F., & Meer P. (2006). Region covariance: A fast descriptor for detection and classification. *In ECCV (pp. 589–600).*

[60]: Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence, 24(7), 971-987.*

[61]: Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. *In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.*

[62]: Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. *In European conference on computer vision (pp. 404-417). Springer, Berlin, Heidelberg.*

[63]: Sivic, J., & Zisserman, A. (2003, October). Video Google: A text retrieval approach to object matching in videos. *In null (p. 1470). IEEE.*

[64]: Lazebnik, S., Schmid, C., & Ponce, J. (2006, June). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (Vol. 2, pp. 2169-2178). IEEE.*

[65]: Perronnin, F., Sánchez, J., & Mensink, T. (2010). Improving the fisher kernel for large scale image classification. *In ECCV (pp. 143–156).*

[66]: Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008, June). A discriminatively trained, multiscale, deformable part model. *In 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8). IEEE.*

[67]: Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR, 1, 1–8.*

[68]: Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR, 1, 1–8.*

[69]: ILSVRC detection challenge results. (2018). *http://www.image-net.org/challenges/LSVRC/.*

[70]: Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. *In NIPS (pp. 2553–2561).*

[71]: Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). OverFeat: Integrated recognition, localization and detection using convolutional networks. *In ICLR.*

[72]: Erhan, D., Szegedy, C., Toshev, A., & Anguelov, D. (2014). Scalable object detection using deep neural networks. *In CVPR (pp. 2147–2154).*

[73]: Krizhevsky, A., Sutskever, I., & Hinton, G. (2012a). ImageNet classification with deep convolutional neural networks. *In NIPS (pp. 1097–1105).*

[74]: Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large scale image recognition. *In ICLR.*

[75]: Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *In CVPR (pp. 1–9).*

[76]: He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In CVPR (pp. 770–778).*

[77]: Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017a). Densely connected convolutional networks. *In CVPR.*

[78]: Law, H., & Deng, J. (2018). CornerNet: Detecting objects as paired keypoints. *In ECCV.*

[79]: He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask RCNN. *In ICCV*.

[80]: Cai, Z., & Vasconcelos, N. (2018). Cascade RCNN: Delving into high quality object detection. *In CVPR*.

[81]: Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018c). Light head RCNN: In defense of two stage object detector. *In CVPR*.

[82]: AVA dataset: *https://research.google.com/ava/*

[83]: Pytorch implementation of SSD: *https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Object-Detection*