

VISUAL OBJECT TRACKING USING SEMI SUPERVISED CONVOLUTIONAL  
FILTERS

A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES  
OF  
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMİR CAN SEVİNDİK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR  
THE DEGREE OF MASTER OF SCIENCE  
IN  
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2020



Approval of the thesis:

**VISUAL OBJECT TRACKING USING SEMI SUPERVISED  
CONVOLUTIONAL FILTERS**

submitted by **EMİR CAN SEVİNDİK** in partial fulfillment of the requirements for  
the degree of **Master of Science in Electrical and Electronics Engineering De-  
partment, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar  
Dean, Graduate School of **Natural and Applied Sciences** \_\_\_\_\_

Prof. Dr. İlkey Ulusoy  
Head of Department, **Electrical and Electronics Engineering** \_\_\_\_\_

Prof. Dr. A. Aydın Alatan  
Supervisor, **Electrical and Electronics Engineering, METU** \_\_\_\_\_

**Examining Committee Members:**

Prof. Dr. Afşar Saranlı  
Electrical and Electronics Engineering, METU \_\_\_\_\_

Prof. Dr. A. Aydın Alatan  
Electrical and Electronics Engineering, METU \_\_\_\_\_

Prof. Dr. Hakan Çevikalp  
Electrical and Electronics Engineering, E. Osmangazi University \_\_\_\_\_

Assoc. Prof. Dr. Emre Özkan  
Electrical and Electronics Engineering, METU \_\_\_\_\_

Assoc. Prof. Dr. Sinan Kalkan  
Computer Engineering, METU \_\_\_\_\_

Date: 23.09.2020



**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Surname: EMİR CAN SEVİNDİK

Signature :



## **ABSTRACT**

### **VISUAL OBJECT TRACKING USING SEMI SUPERVISED CONVOLUTIONAL FILTERS**

SEVİNDİK, EMİR CAN

M.S., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. A. Aydın Alatan

September 2020, 105 pages

Visual object tracking aims to find a single object position in a video frame, when a annotated bounding box is provided in the first frame. Correlation filters have always produced excellent results in terms of accuracy, while enjoying quite low computational complexity. The main property of correlation filter based trackers is to find a filter that can generate high values around the true target object location, whereas relatively low values for the locations away from the object. Recently, deep learning based methods have emerged to learn the optimal discriminative features to be utilized in correlation filters with promising results. Training of such deep feature extractors are usually performed by using both supervised and unsupervised learning techniques. In this thesis, the impact of semi supervised convolutional filters for the visual tracking problem is investigated in order to obtain robust features predicting the object location with high accuracy and being invariant to any kind of appearance change. Two different semi-supervision techniques are proposed and trained on ILSVRC2015 and TrackingNet dataset separately. They are also tested on widely used OTB50 and OTB100 tracking benchmark datasets. Semi-supervision

on ILSVRC2015 dataset leads 1.1% gain on success plot AUC value, 2.4% increase on precision plot AUC value and 2.2% increase on success rate in terms of OTB50 benchmark parameters. Similarly on OTB100 test set, 1.9% gain on success plot AUC value, 2.0% on precision plot AUC value and 2.8% increase on success rate is observed during semi-supervised experiments. In addition to the semi supervision methods, two joint supervision methodologies are also examined to observe the performance differences. The results show that both semi-supervision and joint supervision perform better than the fully supervised models, and such techniques still have superiorities on each other for different occasions.

Keywords: visual object tracking, semi-supervised, convolutional features, correlation filters

## ÖZ

### **YARI DENETİMLİ EĞİTİLMİŞ KATLAMALI FİLTRELER İLE GÖRSEL OBJE TAKİBİ**

SEVİNDİK, EMİR CAN

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

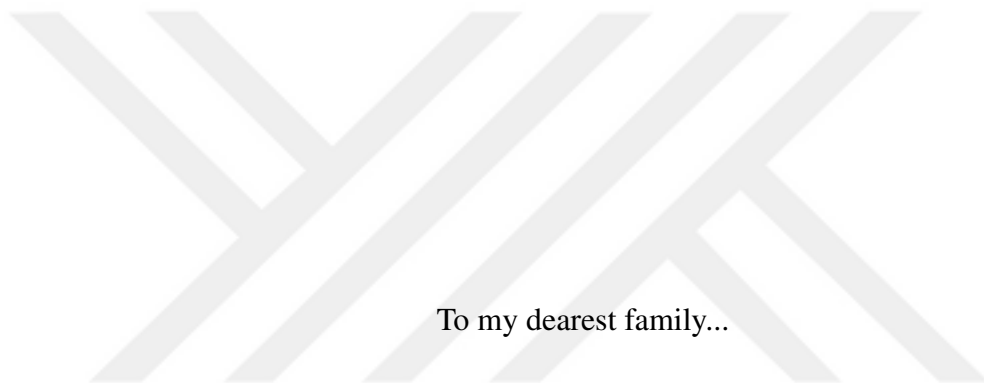
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Eylül 2020 , 105 sayfa

Görsel nesne takibinin amacı ilk video karesinde işaretlenen hedefin yerini diğer video karelerinde doğru bir şekilde bulabilmektir. Korelasyon filtreleri görsel takip problemi için her zaman hem doğruluk açısından mükemmel sonuçlar vermiş olup hem de düşük hesaplama karmaşıklığına sahip olmuştur. Korelasyon filtresi tabanlı takip algoritmaları hedefin gerçekte bulunduğu bölgelerde yüksek değerler verip hedeften uzaklaştıkça düşük değerler verecek bir filtre bulma hedefindedirler. Korelasyon filtrelerinin en uygun ayırt edici öznelikleri öğrenebilmesi için son zamanlarda gelecek vaad eden derin öğrenme tabanlı yöntemler ortaya çıkmıştır. Öznelik çıkarımı için kullanılan bu derin sinir ağları tam denetimli veya denetimsiz eğitim teknikleri kullanılarak eğitilmişlerdir. Bu tez çalışmasında nesnelerin farklı açılarda pozisyonlarını en doğru şekilde tahmin edecek görsel takip problemi için derin öğrenme tabanlı, yarı denetimli öğrenme kullanılarak eğitilmiş katlamalı niteliklerin etkileri incelenmiştir. Bu tezde iki farklı yarı denetimli eğitim yöntemi önerilmiştir. Bu yöntemler ILSVRC2015 ve TrackingNet veri kümeleri üzerinde ayrı ayrı eğitilip OTB50 ve OTB100 veri kümeleri üzerinde ayrı ayrı test edilmiştir. ILSVRC veri kümesinde

yarı denetimli eğitilen modeller tam denetimli eğitimlere karşı başarı grafiğinde eğri altı alan değeri olarak OTB50 ve OTB100 test kümelerinde sırasıyla %1.1 ve %1.9 artış, kesinlik grafiğinde eğri altı alan değeri üzerinde sırasıyla %2.4 ve %2.0 artış, genel başarı değerinde ise sırasıyla %2.2 ve %2.8 artış sağladığı gözlenmiştir. Bu yarı denetimli yöntemlere ek olarak karşılaştırma amacıyla ortak denetimli methodlarda denenmiştir. Her iki denetim yöntemlerinin de tam denetimli yöntemden daha iyi sonuçlar verdiği gözlenmiş olup bu iki denetim yönteminin birbirine karşı üstünlükleri farklı koşullarda ortaya çıkmıştır.

Anahtar Kelimeler: görsel nesne takibi, yarı denetimli, evrişimli süzgeçler, korelasyon süzgeçleri



To my dearest family...

## **ACKNOWLEDGMENTS**

First of all, I would like to express my gratitude and appreciation to my supervisor Prof. Dr. A. Aydın Alatan for being open-minded, supportive and understanding throughout my study. I would like to thank him for his guidance, positive attitude and constructive criticism for this research.

I am grateful of my family of four for their endless support, understanding and love. My father S. Sırrı Sevindik, mother Hamide Sevindik and my beautiful sister Selin Sevindik have stood by me all the time and encouraged me during the difficult times of my life.

I am also thankful to the employer and employees of the company Kuartis, especially to Ahmet Saraçoğlu and Osman Serdar Gedik for providing motivation, suggestions and guidance.

The author is financially supported by TUBITAK BİDEB 2210-A 2017/2 Scholarship during this study.

## TABLE OF CONTENTS

|                                                     |     |
|-----------------------------------------------------|-----|
| ABSTRACT . . . . .                                  | v   |
| ÖZ . . . . .                                        | vii |
| ACKNOWLEDGMENTS . . . . .                           | x   |
| TABLE OF CONTENTS . . . . .                         | xi  |
| LIST OF TABLES . . . . .                            | xiv |
| LIST OF FIGURES . . . . .                           | xv  |
| LIST OF ABBREVIATIONS . . . . .                     | xix |
| CHAPTERS                                            |     |
| 1 INTRODUCTION . . . . .                            | 1   |
| 1.1 Challenges of Visual Object Tracking . . . . .  | 2   |
| 1.2 Scope of the Thesis and Contributions . . . . . | 4   |
| 1.3 The Outline of the Thesis . . . . .             | 6   |
| 2 LITERATURE SURVEY AND RELATED WORK . . . . .      | 7   |
| 2.1 Correlation Filter Based Trackers . . . . .     | 7   |
| 2.1.1 Discriminative Correlation Filters . . . . .  | 9   |
| 2.1.2 Update Rule . . . . .                         | 10  |
| 2.1.3 Scale Estimation . . . . .                    | 13  |
| 2.1.4 Spatial and Temporal Regularization . . . . . | 13  |

|       |                                                               |    |
|-------|---------------------------------------------------------------|----|
| 2.1.5 | Feature Extractors . . . . .                                  | 14 |
| 2.2   | Regression Based Trackers . . . . .                           | 16 |
| 2.2.1 | Region Proposal Networks . . . . .                            | 18 |
| 2.2.2 | Intersection Over Union Net . . . . .                         | 21 |
| 2.3   | Semi Supervised Training . . . . .                            | 24 |
| 2.3.1 | Object Detection and Semantic Segmentation . . . . .          | 28 |
| 2.3.2 | Pose and Depth Estimation . . . . .                           | 30 |
| 3     | SUPERVISED AND UNSUPERVISED DCF BASED TRACKERS . . . . .      | 33 |
| 3.1   | DCFNet: DCF Network For Visual Tracking . . . . .             | 33 |
| 3.2   | UDT: Unsupervised Deep Tracking . . . . .                     | 36 |
| 4     | SEMI SUPERVISED TRAINING FOR VISUAL OBJECT TRACKING . . . . . | 45 |
| 4.1   | Proposed Methods . . . . .                                    | 45 |
| 4.2   | Network Configuration and Training Settings . . . . .         | 49 |
| 4.3   | Datasets . . . . .                                            | 49 |
| 4.3.1 | OTB50 and OTB100 . . . . .                                    | 49 |
| 4.3.2 | ILSVRC 2015 . . . . .                                         | 50 |
| 4.3.3 | TrackingNet . . . . .                                         | 51 |
| 4.4   | Evaluation Metrics for Tracking Experiments . . . . .         | 52 |
| 4.5   | Test Groups . . . . .                                         | 54 |
| 4.5.1 | Weakly Supervised Test Groups . . . . .                       | 55 |
| 4.5.2 | SWU: Supervised-Weakly-Unsupervised Test Groups . . . . .     | 57 |
| 4.5.3 | JSU: Joint Supervised-Unsupervised Test Groups . . . . .      | 58 |
| 4.6   | Experimental Results and Analysis . . . . .                   | 61 |



|         |                                                             |    |
|---------|-------------------------------------------------------------|----|
| 4.6.1   | Weakly Supervised Test Groups . . . . .                     | 61 |
| 4.6.1.1 | Experimental Results for Test1 and Test2 . . . . .          | 61 |
| 4.6.1.2 | Experimental Results for Test3 and Test4 . . . . .          | 66 |
| 4.6.2   | SWU Test Groups . . . . .                                   | 71 |
| 4.6.2.1 | Experimental Results for Test5 and Test6 . . . . .          | 71 |
| 4.6.2.2 | Experimental Results for Test7 and Test8 . . . . .          | 76 |
| 4.6.3   | JSU Test Groups . . . . .                                   | 80 |
| 4.6.3.1 | Experimental Results for Test9 and Test10 . . . . .         | 80 |
| 4.6.3.2 | Experimental Results for Test11 and Test12 . . . . .        | 84 |
| 4.6.4   | Overall Comparison of Test Groups Trained on ILSVRC . . . . | 88 |
| 4.6.5   | Overall Comparison of Test Groups Trained on TrackingNet .  | 89 |
| 4.6.6   | Comparison of SOTA Trackers and Proposed Models . . . . .   | 91 |
| 5       | CONCLUSIONS . . . . .                                       | 93 |
|         | REFERENCES . . . . .                                        | 97 |

## LIST OF TABLES

### TABLES

|            |                                                                                                                                            |    |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|----|
| Table 3.1  | Network configuration of UDT . . . . .                                                                                                     | 41 |
| Table 4.1  | Properties of models used in Test1 are shown. . . . .                                                                                      | 55 |
| Table 4.2  | Properties of models in Test2 are shown. . . . .                                                                                           | 56 |
| Table 4.3  | Properties of models in Test3 are shown. . . . .                                                                                           | 56 |
| Table 4.4  | Properties of models in Test4 are shown. . . . .                                                                                           | 56 |
| Table 4.5  | Properties of models used in Test5 are shown. . . . .                                                                                      | 57 |
| Table 4.6  | Properties of models used in Test6 are shown. . . . .                                                                                      | 57 |
| Table 4.7  | Properties of models used in Test7 are shown. . . . .                                                                                      | 58 |
| Table 4.8  | Properties of models used in Test8 are shown. . . . .                                                                                      | 58 |
| Table 4.9  | Properties of models used in Test9 are shown. . . . .                                                                                      | 59 |
| Table 4.10 | Properties of models used in Test10 are shown. . . . .                                                                                     | 59 |
| Table 4.11 | Properties of models used in Test11 are shown. . . . .                                                                                     | 60 |
| Table 4.12 | Properties of models used in Test12 are shown. . . . .                                                                                     | 60 |
| Table 4.13 | Properties of all test groups are shown. . . . .                                                                                           | 60 |
| Table 4.14 | OTB100 performance results of state-of-the-art trackers and top performing proposed models together with fully supervised models . . . . . | 91 |

## LIST OF FIGURES

### FIGURES

|             |                                                                      |    |
|-------------|----------------------------------------------------------------------|----|
| Figure 2.1  | Standard CFB tracking workflow . . . . .                             | 11 |
| Figure 2.2  | Visualization of Histogram of Oriented Gradient features. . . . .    | 15 |
| Figure 2.3  | Illustration of deep features . . . . .                              | 15 |
| Figure 2.4  | The overview of SiamFC architecture[2] . . . . .                     | 16 |
| Figure 2.5  | The overall framework of Siamese-RPN[3] . . . . .                    | 18 |
| Figure 2.6  | The general view of SiamRPN++ framework[4]. . . . .                  | 22 |
| Figure 2.7  | Overview of network architecture of ATOM[29]. . . . .                | 23 |
| Figure 2.8  | Illustration of three typical types of weak supervision[31]. . . . . | 25 |
| Figure 2.9  | Illustration of three incomplete supervision methods[31]. . . . .    | 27 |
| Figure 2.10 | The overview of Weakly Supervised Deep Detection Network[47].        | 29 |
| Figure 2.11 | The overview of learning schemes used in SUW-Learn[68] . . . . .     | 31 |
| Figure 3.1  | The general DCFNet architecture [8] . . . . .                        | 33 |
| Figure 3.2  | OTB2013 and OTB2015 success plots of DCFNet . . . . .                | 36 |
| Figure 3.3  | Supervised and unsupervised learning comparison . . . . .            | 37 |
| Figure 3.4  | General unsupervised deep tracking algorithm . . . . .               | 37 |
| Figure 3.5  | Single frame and multiple frames validation methods . . . . .        | 39 |

|             |                                                                                                                                 |    |
|-------------|---------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 3.6  | Randomly cropped images from the center of frames in ILSVRC2015 dataset for UDT training [7]. . . . .                           | 41 |
| Figure 3.7  | OTB2015 test results of UDT and UDT+ trackers . . . . .                                                                         | 42 |
| Figure 3.8  | TempleColor[10] test result of UDT and UDT+ trackers . . . . .                                                                  | 43 |
| Figure 4.1  | The probability value distribution for SWU_C and SWU_P semi supervised training methods. . . . .                                | 46 |
| Figure 4.2  | The overall flow of semi-supervised trainings. . . . .                                                                          | 47 |
| Figure 4.3  | The training workflow of joint supervised unsupervised trainings. . . . .                                                       | 48 |
| Figure 4.4  | OTB50: (a) Distribution of attributes on the whole dataset, and (b) the distribution of occlusion (OCC) attribute [72]. . . . . | 50 |
| Figure 4.5  | Comparison of datasets in terms of the length and number of videos [5]. . . . .                                                 | 52 |
| Figure 4.6  | Comparison of model results in Test1 in terms of success and precision plots . . . . .                                          | 62 |
| Figure 4.7  | Comparison of model results in Test1 in terms of success rate . . . . .                                                         | 62 |
| Figure 4.8  | Comparison of model results in Test2 in terms of success and precision plots . . . . .                                          | 64 |
| Figure 4.9  | Comparison of model results in Test2 in terms of success rate . . . . .                                                         | 64 |
| Figure 4.10 | Comparison of model results in Test1 and Test2 in terms of success and precision plots . . . . .                                | 66 |
| Figure 4.11 | Comparison of model results in Test3 in terms of success and precision plots . . . . .                                          | 67 |
| Figure 4.12 | Comparison of model results in Test3 in terms of success rate . . . . .                                                         | 67 |
| Figure 4.13 | Comparison of model results in Test4 in terms of success and precision plots . . . . .                                          | 69 |

|             |                                                                                                       |    |
|-------------|-------------------------------------------------------------------------------------------------------|----|
| Figure 4.14 | Comparison of model results in Test4 in terms of success rate . .                                     | 69 |
| Figure 4.15 | Comparison of model results of Test3 and Test4 in terms of suc-<br>cess and precision plots . . . . . | 71 |
| Figure 4.16 | Comparison of model results in Test5 in terms of success and<br>precision plots . . . . .             | 72 |
| Figure 4.17 | Comparison of model results on Test5 in terms of success rate . .                                     | 72 |
| Figure 4.18 | Comparison of model results in Test6 in terms of success and<br>precision plots . . . . .             | 74 |
| Figure 4.19 | Comparison of model results in Test6 in terms of success rate . .                                     | 74 |
| Figure 4.20 | Comparison of model results of Test5 and Test6 in terms of suc-<br>cess and precision plots . . . . . | 76 |
| Figure 4.21 | Comparison of model results in Test7 in terms of success and<br>precision plots . . . . .             | 77 |
| Figure 4.22 | Comparison of model results in Test7 in terms of success rate . .                                     | 77 |
| Figure 4.23 | Comparison of model results in Test8 in terms of success and<br>precision plots . . . . .             | 78 |
| Figure 4.24 | Comparison of model results in Test8 in terms of success rate . .                                     | 79 |
| Figure 4.25 | Comparison of model results of Test7 and Test8 in terms of suc-<br>cess and precision plots . . . . . | 80 |
| Figure 4.26 | Comparison of model results in Test9 in terms of success and<br>precision plots . . . . .             | 81 |
| Figure 4.27 | Comparison of model results in Test9 in terms of success rate . .                                     | 81 |
| Figure 4.28 | Comparison of model results in Test10 in terms of success and<br>precision plots . . . . .            | 83 |
| Figure 4.29 | Comparison of model results in Test10 in terms of success rate .                                      | 83 |

|             |                                                                                                     |    |
|-------------|-----------------------------------------------------------------------------------------------------|----|
| Figure 4.30 | Comparison of model results in Test11 in terms of success and precision plots . . . . .             | 85 |
| Figure 4.31 | Comparison of model results in Test11 in terms of success rate .                                    | 85 |
| Figure 4.32 | Comparison of model results in Test12 in terms of success and precision plots . . . . .             | 87 |
| Figure 4.33 | Comparison of model results in Test12 in terms of success rate .                                    | 87 |
| Figure 4.34 | Comparison of models utilized ILSVRC dataset in terms of success and precision plots . . . . .      | 89 |
| Figure 4.35 | Comparison of models utilized TrackingNet dataset in terms of success and precision plots . . . . . | 90 |

## LIST OF ABBREVIATIONS

### ABBREVIATIONS

|         |                                                          |
|---------|----------------------------------------------------------|
| IOU     | Intersection Over Union                                  |
| CFB     | Correlation Filter Based                                 |
| MOSSE   | Minimum Output Sum of Squared Error                      |
| ADMM    | Alternating Direction Method of Multipliers              |
| UDT     | Unsupervised Deep Tracking                               |
| AUC     | Area Under Curve                                         |
| ILSVRC  | ImageNet Large Scale Visual Recognition Competition      |
| OTB     | Object Tracking Benchmark                                |
| DCF     | Discriminant Correlation Filter                          |
| FFT     | Fast Fourier Transform                                   |
| RPN     | Region Proposal Network                                  |
| MIL     | Multiple Instance Learning                               |
| GMM     | Gaussian Mixture Model                                   |
| OPE     | One Pass Evaluation                                      |
| TRE     | Temporal Robustness Evaluation                           |
| SRE     | Spatial Robustness Evaluation                            |
| SRDCF   | Spatially Regularized Discriminative Correlation Filters |
| RPN     | Region Proposal Network                                  |
| SiamRPN | Siamese Region Proposal Network                          |
| NMS     | Non Maximum Suppression                                  |
| RNN     | Recurrent Neural Network                                 |
| JSU     | Joint Supervised-Unsupervised                            |
| SOTA    | State of the Art                                         |





## **CHAPTER 1**

### **INTRODUCTION**

Modern visual surveillance and autonomous systems desire visual analysis of the target. Such systems try to convert sensor outputs to semantic meanings according to their sub-tasks. Visual object tracking is one of these sub-tasks that has been one of the most visited problems in computer vision with wide range of applications. These applications include motion-based recognition, pedestrian flow analysis, anomaly detection and vehicle navigation. The trajectory information of a pedestrian can help analyzing flow on the streets. It also helps to recognize the activity of an object to indicate whether there is an abnormal situation or not. On autonomous systems, tracking the target has great importance. These systems adjust their movements according to the tracked target in order to reach their goal or prevent a collusion. For instance, autonomous vehicles track the objects around them, and try to estimate their location in future. By doing so, they can act before an accident happens.

Visual object tracking is required by most of these systems, therefore, attracted most researchers' attention. The problem seeks to find the location of an object of interest throughout the video sequence frames with help of the visual appearance model of the object. In visual tracking, only available information is the raw pixels of the video frames. Therefore, there is a difficulty for obtaining distinctive and robust models for the appearance of the target object by using the only available visual data.

Computer vision and machine learning based algorithms try to obtain best visual appearance model in order to distinguish the target object from background and other objects around. Obviously, the main input for these algorithms is the region of interest or image region of the target object. These regions contain raw pixels representing object of interest that are seen as raw patterns. Computer vision based solutions

represent them in high dimensional feature space, while machine learning based approaches try to obtain best model for the object appearance while performing tracking operation.

## 1.1 Challenges of Visual Object Tracking

Visual object tracking on videos is an open-ended and far-reaching problem in computer vision and nowadays, in machine learning area. This thesis specifically focuses on single object tracking on video frames. There are many challenges that should be taken into consideration while developing a single object tracker. One of these challenges, namely *scale change*, is the appearance change which can be described as any object appearance difference between a template frame and a search frame. Scale changes that result from objects moving towards to or away from the camera cause considerable visual information loss. Therefore, the tracker should be able to estimate the object size in multiple scales. *Deformations* on the object due to the various motion types is another reason for object appearance change, thus, the tracker should extract features that are invariant to deformations.

Another critical challenge for visual tracking is *motion blur effect*. Non-focused and non-stationary cameras, and fast moving objects causes blurred frames on the video. In such situations, localization performance of the tracker degrades significantly since the object appearance changes dramatically. The *illumination change* due to environmental effects, similar to the motion blur, causes challenges on trackers. Usually the illumination change occurs abruptly when the object of interest goes into the shaded area. Hence, the tracker should cope with this problem as well by obtaining illumination invariant object appearance model.

*Abrupt motion* is yet another problem for visual tracking. When the camera or object moves abruptly, the tracker may fail if the search region is not large enough to contain the object of interest. The erroneous result on the test frame causes incorrect update of the appearance model, and causes drift from the actual object position. Another reason for this incorrect update is the full or partial *occlusion* of the object. This update can occur on the undesired part of the occlusion, which eventually may cause

drift from the true location. Therefore, the tracker should be designed with a good search area and update rule strategy to overcome these type of challenges.

With the recent developments on deep network architectures, tracker algorithms start adopting deep networks for feature extraction due to their superior performance. Some existing deep tracking methods either use off-the-shelf deep network models (e.g., VGG [1]) or off-line training of a specific CNN model for feature extraction. The Siamese trackers [2, 3, 4] utilize similarity matching for tracking task, and use deep networks as feature extractor as well. In order to train these deep networks, visual tracking algorithms use labeled data for supervised learning. Neural networks usually learn very well on their training sets, and obtain a good convergence on those sets. However, they do not perform well when they encounter data having very different properties such as resolution, quality, field-of-view, aspect ratio. Although there are several tracking data sets having full labels for each video, it is always desired to use many of these sets so that the network can learn representative coefficients and get good generalization ability. If a tracker is expected to perform best on a camera, it is a good option to collect video frames from that camera and annotate them before using on a training set. Nevertheless, annotating video frames desires heavy human resources. Automating annotation is also a research area, and applied to several data sets (e.g. TrackingNet [5]). In TrackingNet, forward-backward tracking is used for data annotation and tracker evaluation. Unfortunately, these annotations include many inaccurate labels which is not desired.

Computer vision and machine learning areas aim to solve those aforementioned challenges above. Many methods in these fields try to obtain discriminative appearance model from object of interest using various feature extractors for improving robustness against occlusions and deformations [3, 4]. These techniques also utilize ensemble of trackers method to mitigate the tracking failures caused by appearance changes of the object of interest [6]. Unsupervised training methodology [7] is also developed to overcome annotation problem for visual object tracking.

## 1.2 Scope of the Thesis and Contributions

In this work, instead of designing new network architectures in order to increase the accuracy and robustness of visual trackers against most of the aforementioned challenges, the goal is to focus on the training strategy and investigate the effect of incomplete supervision on the performance. The reason of analyzing training strategy rather than the network architecture is the rapid developments on the deep learning areas has resulted in rapid performance increase on the trackers as well. The research efforts mainly focused on mostly supervised or sometimes unsupervised learning methodologies, and they have reached to a point that seems challenging to take a step forward. Therefore, it is noteworthy to pay attention to semi supervision and joint supervision for visual tracking problem.

There are numerous supervised and unsupervised visual object trackers. DCFNet [8] and UDT [7] are two well-known algorithms from the literature and in this thesis, they are adopted for examining the effect of incomplete and joint supervision. DCFNet [8] is a correlation filter based tracker having light-weight CNN backbone as feature extractor. DCFNet is trained in supervised manner by using NUS-PRO [9], TempleColor128 [10] and UAV123 [11] data sets, and tested by OTB100 [12]. On the other hand, UDT is the unsupervised trained version of the DCFNet tracker that utilizes ILSVRC2015 [13] dataset. Similar network architecture and training settings are utilized for CNN backbone part for both CFNet and UDT; moreover, the correlation filter part is exactly the same as well. Therefore, DCFNet tracking algorithm becomes a perfect sample for incomplete and joint supervision trials.

In this thesis, two different incomplete supervision approaches are experimented. One of these techniques is performed in such a way that the training batch samples are selected either from labeled version or unlabeled version of the data set according to a predetermined probability. In other words, if the probability is set to 0.3, supervised samples are selected from the labeled instances with probability being equal to 0.3 . Otherwise, the samples are selected from the unlabeled instances. As an example, if there are 100 iterations during training, 30 of them are labeled instances, and 70 of them are unlabeled instances. Whenever the labeled samples are selected, the error calculation is performed according to the supervised learning scheme. On the other

hand, the error is calculated according to the unsupervised learning scheme, if the unlabeled samples are selected. In the rest of this thesis, the explained method is denoted as *weakly supervised training* in the document. For the other tested methodology, the probability value is swept from value 1.0 to 0.0 during training. This experiment is also divided into two parts. While the probability value is continuously being decrease in *SWU\_C*, the other approach decreases the probability value in a discontinuous manner, namely *SWU\_P*. As a typical example for *SWU\_C* methodology, the probability value is set to 1.0 in the first epoch. The probability value is decreased at each epoch in a manner that it becomes 0.0 at the final epoch. However, *SWU\_P* method sets the value to 1.0 for first  $N$  epoches, and then evenly decrease it after each  $N$  epoches so that the value becomes 0.0 for last  $N$  epoches.

In addition to the incomplete supervision, joint supervised and unsupervised trainings are also performed to investigate their effect on the tracker performance. The loss values of the supervised and unsupervised learning schemes are combined by a weighting parameter together to obtain a final loss value during a training iteration. These method is also divided into two parts. One is called *JSU* where control parameter value is fixed throughout training, whereas the other one ,named as *JSU\_SWU*, has gradually decreased the control parameter value after each epoch during training. *JSU\_SWU* method is also split into two. One has continuously decreased the value after each epoch and denoted as *JSU\_SWU\_C*. Other one has partially decreased the value similar to the one in incomplete supervision, and denoted as *JSU\_SWU\_P*.

The semi supervised and joint supervised experiments are conducted separately on two datasets, which are ILSVRC2015 and TrackingNet. Tests are performed on the OTB50 and OTB100 benchmark sets.

Our contributions can be summarized as follows:

- To the best of our knowledge, the proposed methods are the first incomplete supervision training techniques, which are investigated for visual object tracking task
- Improved and robust feature extraction of a correlation filter based tracker
- Obtaining better results with incomplete supervision than the fully supervised

### **1.3 The Outline of the Thesis**

Chapter 1 includes the motivation for the problem and problem statement, which is followed by the contribution to the problem.

The literature survey about the concepts and the terms of visual tracking are introduced in Chapter 2. The relevant scientific efforts that attack the challenges regarding the visual single object tracking are also examined in detail in this chapter.

Chapter 3 presents the architecture, training methodology, implementation details and some preliminary results of DCFNet and UDT trackers, which are the algorithms mainly used during the experiments.

Chapter 4 starts with providing information about purpose of the experiments, utilized network architecture, datasets and evaluation methodology. It continues with detailing the experiment groups. Finally, results and the analysis for the supervised, unsupervised, semi supervised and joint supervised trained models are presented.

Chapter 5 concludes the thesis by emphasizing the importance of the problem and summarizing the work done.

## CHAPTER 2

### LITERATURE SURVEY AND RELATED WORK

Single or multiple object tracking has been the main purpose of visual object tracking task, however, the purpose of this study is confined to single visual object tracking. This research aims developing new training technique to alleviate the weaknesses of current algorithms for single object tracking.

One of the most important developments in recent visual tracking research was extensive adoption of discriminative methods. The main component of most modern trackers is a discriminative classifier, aiming to distinguish between the target and the surrounding environment. Given an initial image patch containing the target, the goal is to learn a classifier to discriminate between its appearance and that of the environment.

In this chapter, correlation filter based and regression based trackers are explained in detail. Their solution to the tracking problem and how they attack to mitigate challenges of it are investigated. After that, semi supervised training methodology is described.

#### 2.1 Correlation Filter Based Trackers

Correlation filter based (CFB) tracking methods rely on template matching idea that is based on matched filter theory. CFB trackers aim to find a good template that represents the object of interest in feature space, which is also called as correlation filter. The main goal is to output a good correlation plane when the correlation operation between this filter and the candidate object instance is performed. The expected result

of this correlation (or equivalently convolution) operation is a peaky-shaped signal. The peak location should occur at the true center of the object location. Correlation filter is perceived as discriminative regressor model since it is developed according to a desired correlation plane.

Large images or long signals can be problematic in terms of computation since it requires  $\mathcal{O}(n^2)$  multiplications ( $n$  is the signal length). Therefore, circular correlation is commonly used instead of linear correlation. The circular correlation can be performed in frequency domain with element-wise multiplications according to the Fast Fourier Transform (FFT) and Convolution Theorem. The correlation operation becomes much faster in terms of computation since FFT has  $\mathcal{O}(n \log n)$  complexity[14].

Before going into details about the correlation filters, the correlation and convolution theorem is reminded as follows:

$$z[n] = \sum_m x[m]y[n - m] = \mathcal{F}^{-1}\{X \odot Y\} \quad (21)$$

$$z[n] = \sum_m x[m]y[n + m] = \mathcal{F}^{-1}\{X^* \odot Y\} \quad (22)$$

For two one-dimensional discrete signals,  $x$  and  $y$  having equal length, equations (21) and (22) show the correlation and convolution theorem, respectively. Lower case letters denote the signals in the spatial domain whereas capital letters represent the signals in frequency domain unless stated otherwise.  $\mathcal{F}$  and  $\mathcal{F}^{-1}$  are Discrete Fourier Transform (DFT) and inverse DFT, respectively, while the symbol  $\odot$  shows element-wise multiplication. The subscript  $*$  represents the element-wise complex conjugate, and  $x[n - m]$  is considered to be circularly translated version of  $x[n]$  by an integer amount  $m$  to the right as in the case of these equations. The theorem can be extended into two-dimensional case because DFT has a separable 2-dimensional definition.

Discriminative correlation filter based tracking methods are discussed in the upcoming subsections by detailing their solution to variety of challenges in visual object tracking.



### 2.1.1 Discriminative Correlation Filters

A pioneering study on discriminative correlation filters is published in [15] for visual tracking by Bolme *et al.*, known as Minimum Output Sum of Squared Error (MOSSE). In MOSSE, the correlation filter is calculated by minimizing the cost function which is the sum of squared errors between the desired correlation response and the correlation of the filter for each training sample. Closed form solution for the correlation filter exists since the circular correlation and Convolution Theorem make it possible to operate in frequency domain. In MOSSE, the foreground and background are learned discriminatively since correlation responses at all possible circular translations of each training samples are utilized. Because of the existence of closed form solution, the computation is performed efficiently in frequency domain. This makes MOSSE algorithm run much faster than real time.

In the following part, the computation of basic correlation filter computation for tracking task is described. When the tracking process is started, it is assumed that there are set of training examples  $\{x_i\}_{i=1}^N$ , where  $N$  represents the number of examples. Each  $x_i$  denotes an  $M \times M$  two dimensional object patch. Furthermore, for the training samples and a filter  $w$ ,  $\{y_i\}_{i=1}^N$  are the desired correlation responses.  $y$  is desired to be centered when the object  $x$  is centered regarding to the bounding box of the target. Moreover, when  $x$  is shifted from the center by some amount,  $y$  should be shifted from the center at the same amount as well. The desired response  $y$  is assumed to be a peaky-shaped Gaussian. The peak location is expected to be shifted from center as  $x$  is shifted. In order to enforce this resemblance, the following equation is utilized:

$$w_{opt} = \arg \min_w \sum_{i=1}^N \|w \circledast x_i - y_i\|^2 \quad (23)$$

where  $w_{opt}$  is the derived correlation filter and  $\circledast$  is the circular correlation operation. This circular correlation operation in (23) can be written in the frequency domain according to the correlation theorem as follows:

$$W_{opt} = \arg \min_W \sum_{i=1}^N \|W^* \odot X_i - Y_i\|^2 \quad (24)$$

The term inside *argmin* in (24) can be expressed as in (25) due to the element-wise multiplication operations.

$$\mathcal{L} = \sum_{i=1}^N W^* \odot W \odot X_i^* \odot X_i + Y_i^* \odot Y_i - W^* \odot X_i \odot Y_i^* - W \odot X_i^* \odot Y_i \quad (25)$$

When the derivative of function in (25) is computed regarding to  $W^*$  for all of its elements, the derived equation is:

$$\frac{\partial \mathcal{L}}{\partial W^*} = \sum_{i=1}^N W \odot X_i^* \odot X_i - X_i \odot Y_i^* \quad (26)$$

When we set the equation (26) to zero,  $W$  is computed as:

$$W = \frac{\sum_{i=1}^N X_i \odot Y_i^*}{\sum_{i=1}^N X_i^* \odot X_i} \quad (27)$$

To find the object location, when the filter  $W_t$  is obtained at the  $t^{th}$  frame, the incoming frame  $T_{t+1}$  is transformed to frequency domain as  $X_{t+1}$ . Afterwards, it is element-wise multiplied by  $W_t$  resulting in obtaining the response  $Y_{t+1}$  on frequency domain. The calculated response  $y_{t+1}$  is obtained basically by taking its inverse DFT. The pixel location where  $y_{t+1}$  has the highest value indicates the shift from the center of object.

In frequency domain,  $W$  is the correlation filter which minimizes the cost in (23). Because of the Fast Fourier Transform algorithm, the calculation of  $W$  has complexity of  $\mathcal{O}(M^2 \log(M))$ .  $M^2 \times M^2$  matrix operations would be required if the operations were done on time domain. This makes correlation filter a suitable solution for tracking problem that can run real-time even on weak hardware. A standard correlation filter based tracking is visualized in Figure 2.1.

### 2.1.2 Update Rule

While performing tracking operation, the target object usually goes under appearance transformation when it goes through dissimilar lighting conditions, by changing its

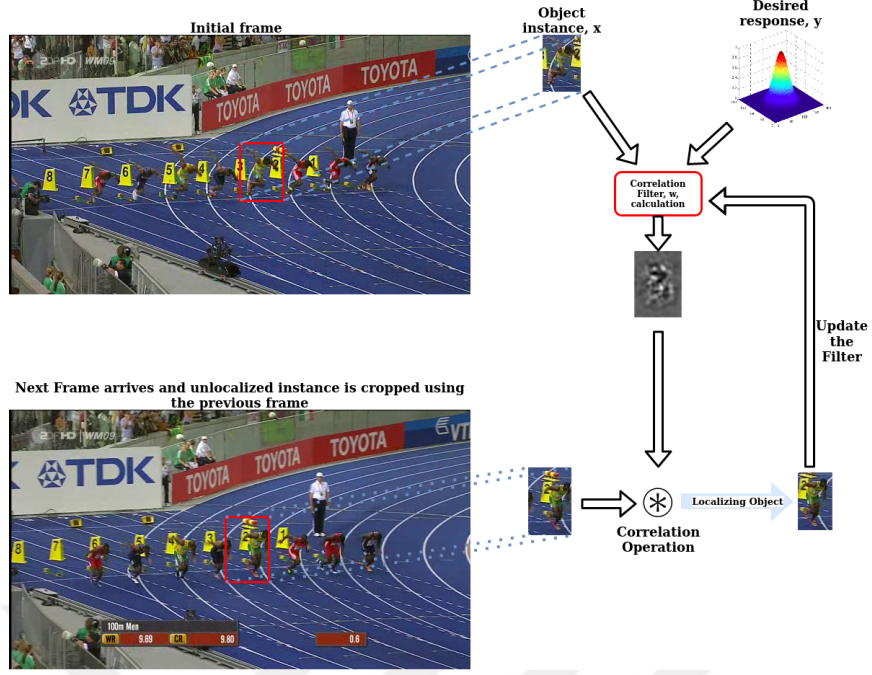


Figure 2.1: Standard CFB tracking workflow. Main purpose is to obtain a correlation filter whose correlation response with the unlocalized object will result in a Gaussian response where its peak value is at the true location.

rotation, pose, scale, or by exposing non-rigid deformation. Therefore, correlation filters must swiftly change itself in order to follow objects correctly. An efficient algorithm adopted by MOSSE[15] is running average method described as follows:

$$\begin{aligned} A_{t+1} &= (1 - \alpha)A_t + \alpha(X_{t+1} \odot Y_{t+1}^*), \\ B_{t+1} &= (1 - \alpha)B_t + \alpha(X_{t+1} \odot X_{t+1}^*), \end{aligned} \quad (28)$$

$$W_{t+1} = \frac{A_{t+1}}{B_{t+1}} \quad (29)$$

where  $W_{t+1}$  is the updated correlation filter at frame  $t+1$ ,  $X_{t+1}$  is feature map of cropped object patch and  $Y_{t+1}$  is the desired centered Gaussian response in the frequency domain.

MOSSE was the path-breaking method in the CFB visual tracking algorithms, and it can run beyond real-time. Nonetheless, MOSSE algorithm is not suitable for sig-

nificant scale changes on the object. Furthermore, utilizing single feature map is not sufficient for proper tracking when the appearance change damages the representation property of the filter. Henceforth, to solve these problems, methods that utilize multi-scale search steps and include multiple feature maps [16, 17] are proposed. To embed the multi-channel property, the feature maps of training example  $x \{x^1, \dots, x^d\}$ , which is composed of particular feature maps having the dimension same as the object patch is utilized. In addition to that, a regularization term is joined to the loss function to prevent correlation filter coefficients to become large. When these properties are added to the loss function, it becomes as follows:

$$L(w) = \sum_{k=1}^t \left\| \left( \sum_{l=1}^d w^l \otimes x_k^l \right) - y_k \right\|^2 + \lambda \sum_{l=1}^d \|w^l\|^2 \quad (210)$$

The equation (210) has also close form expression in the Fourier domain but for only one training example. After necessary calculations similar to single channel version, the correlation filter in frequency domain becomes as follows:

$$W^l = \frac{X^l \odot Y^*}{\sum_{m=1}^d X^m \odot X^{m*} + \lambda} \quad (211)$$

Here  $\lambda$  is the  $L2$  regularization term parameter of the filter. The update rule is also changed slightly as stated in (212):

$$W_{t+1}^l = \frac{A_{t+1}^l}{B_{t+1} + \lambda} \quad (212)$$

where nominator (A) and denominator (B) calculation are changed as well:

$$\begin{aligned} A_{t+1}^l &= (1 - \alpha)A_t^l + \alpha(X_{t+1}^l \odot Y_{t+1}^*), \forall l \in \{1, 2, \dots, d\}, \\ B_{t+1} &= (1 - \alpha)B_t + \alpha \sum_{l=1}^d X_{t+1}^l \odot X_{t+1}^{l*} + \lambda \end{aligned} \quad (213)$$

### 2.1.3 Scale Estimation

Scale estimation is another challenge for visual object tracking. One of the solutions is to create the input at multiple but different scales and compute correlation between the filter and the inputs to find the scale change on target dimensions. The samples  $\{x_r\}_{r \in \{\lceil \frac{1-S}{2} \rceil, \dots, \frac{S-1}{2} \}}$  are brought out at the scales  $a^r$  according to the current target object scale and centered at the previous target object location.  $S$  represents the number of scales and  $a$  is the scale increment factor. The sample  $x_r$  is generated by re-scaling the input with respect to  $a^r$  before the feature extraction.

Similar to aforementioned solution, discriminative scale space tracking (DSST [18]) uses  $d$  multi-channel feature maps for  $S$  scale factors. With addition to the standard correlation filter  $w$  for localization, they have introduced scale correlation filter  $w_{scale}$  for calculation of appropriate scale. The computation of scale filter  $w_{scale}$  is performed using scale samples  $w_s \in R^{dxS}$ . The suitable scale is calculated by finding the scale having the largest number on the response map of the scale filter and test sample. By doing so, the exhaustive scale search is prevented. Hence, the method manage to operate real-time.

### 2.1.4 Spatial and Temporal Regularization

Discriminatively learned correlation filters, which assume periodicity on the training image patches to learn a classifier on all patches in the target surroundings efficiently, have been successfully used to solve the tracking problem. Nevertheless, the periodic assumption also comes with undesired boundary effects, which damage the tracking quality. The authors of spatially regularized discriminative correlation filters (SRDCF [19]) proposed to penalize correlation filter coefficients on the boundaries during learning process by introducing spatial regularization component. With the addition of spatial regularization term, the loss function becomes as:

$$L(w) = \sum_{k=1}^t \left\| \left( \sum_{l=1}^d w^l \otimes x_k^l \right) - y_k \right\|^2 + \lambda \sum_{l=1}^d \|f \cdot w_t^l\|^2 \quad (214)$$

where  $f$  is the regularization term. Although the additional term has violated the

closed form solution and increases complexity, authors managed to solve it efficiently using iterative Gauss-Seidel method in the frequency domain. The introduced spatial regularization component led to more discriminative appearance model for correlation filter. Henceforth, SRDCF has outscored the many of the DCF based tracker when it was published in 2015.

During online tracking, when a new frames come one after another, the algorithm first tries to find the location of target object, then updates the filter based on the new location. However, when occlusion occurs on the object of interest, the update process causes the filter to learn occluding object instead of the target object, which results in the tracking failure. In order to prevent this problem, STRCF [20], namely spatial-temporal regularized correlation filters, proposes temporal regularization in addition to the spatial regularization. The main purpose is to prevent filter to change significantly, therefore, they have utilized the following loss function:

$$w_{opt} = \arg \min_w \frac{1}{2} \left\| \left( \sum_{d=1}^D w^d \otimes x_t^d \right) - y \right\|^2 + \frac{1}{2} \sum_{d=1}^D \|f \cdot w^d\|^2 + \frac{\mu}{2} \|w - w_{t-1}\|^2 \quad (215)$$

where  $w_{t-1}$  represents the correlation filter in the  $(t - 1)^{th}$  frame, and  $\mu$  represents the regularization parameter. Although there is no direct closed form solution, the function in (215) is convex, and globally optimal solution can be obtained by minimizing it via alternating direction method of multipliers (ADMM). Unfortunately, the iterative solution due to ADMM has decreased the efficiency of the algorithm significantly. However, the performance increase in terms of accuracy and robustness has shown its effectiveness.

### 2.1.5 Feature Extractors

DCF based trackers have used variety of features to learn discriminative filters for increasing the performance of trackers. First introduced MOSSE tracker used only gray-scale samples to train the filter. For multi-channel filter on multi-dimensional features, Histogram of Oriented Gradients (HOG [21]) and Color-Names [22] have utilized. Following the developments on deep learning algorithm, convolutional fea-

tures which are trained independently from other tasks like image classification are started to be used on feature extraction for correlation filter training in tracking. This has greatly enhanced tracking performance. Illustrations of HOG and deep network features are done in Figure 2.2 and Figure 2.3, respectively.

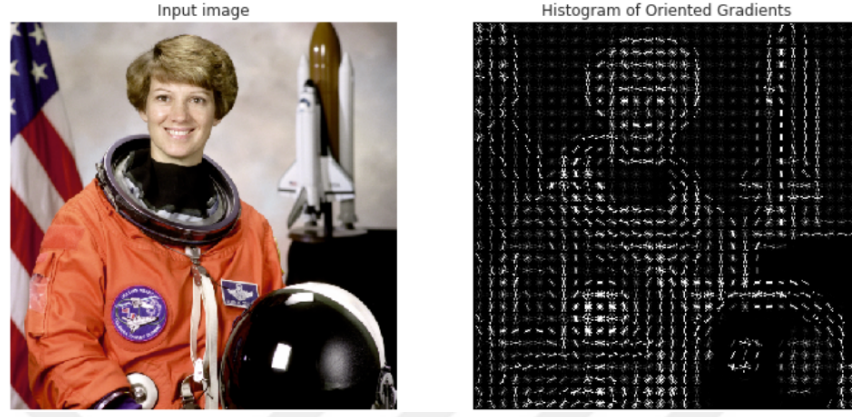


Figure 2.2: Visualization of Histogram of Oriented Gradient features.

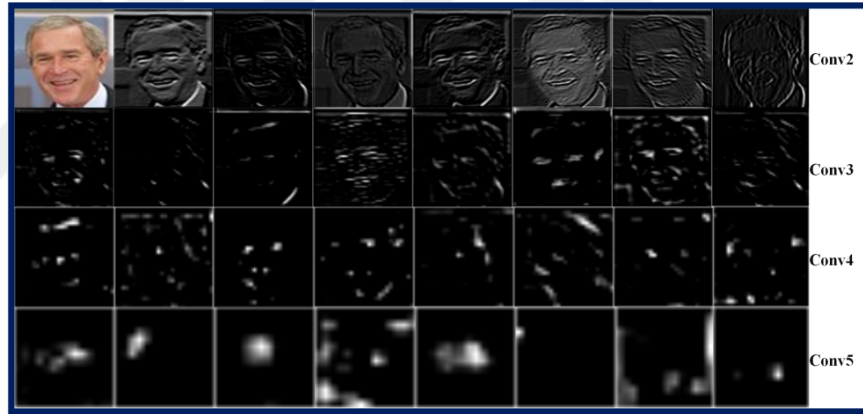


Figure 2.3: Illustration of deep features extracted from Conv2, Conv3, Conv4 and Conv5 layers of VGG network [23]

One of the first studies on end-to-end representation learning using convolutional filters for tracking was done by DCFNet [8]. With the emergence of vast labelled tracking datasets, instead of using pretrained networks which are hand-picked and heavy-weight, DCFNet tried to learn a very light-weight network while preserving correlation filter properties. Utilizing the standard DCF loss function for backward propagation, DCFNet trains light-weight convolutional network to learn features in an end-to-end style while being suitable for CFB tracking. After an offline learn-

ing process, online tracking is used for evaluation. The algorithm first extracts the feature maps of target object in the reference frame using trained convolutional neural network. The correlation filter is trained using these feature maps with standard multi-channel formulation. After that, the feature extraction process continues on the predetermined search region of incoming target frames. Finally, the correlation between the filter and target feature maps indicates the location of object.

## 2.2 Regression Based Trackers

In the recent years, deep convolutional networks trained from large supervised data sets have been adopted to solve most of the problems in computer vision. For visual tracking problem, in order to extract discriminant features for the correlation filter, deep networks are exploited. Either pretrained networks or end-to-end trained networks have been utilized. The fully convolutional solution to tracking problem was proposed in SiamFC [2]. They proposed a different approach to the correlation filter, and advocated a general similarity learning property. Specifically, they trained a Siamese network to find the location of object of interest in a template image within a wider search image by computing cross-correlation of the features of these inputs as it is shown in Figure 2.4.

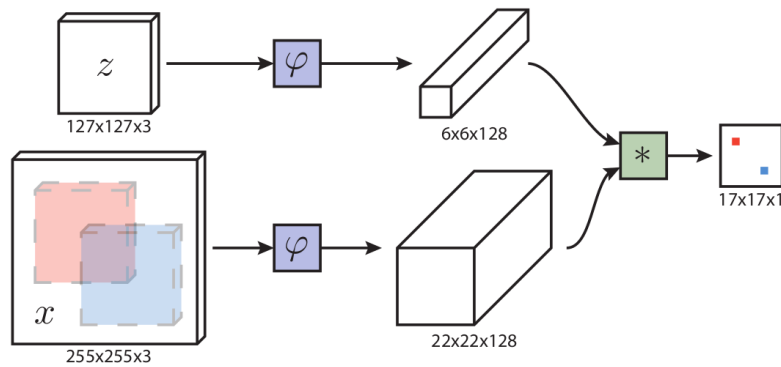


Figure 2.4: The overview of SiamFC architecture[2]

Fully convolutional property makes it possible to use much larger search image size instead of same size as template image as in the case of correlation filters. By this way, in one evaluation, all translated sub windows are computed on a dense grid. To



perform this operation, feature extractor function  $\phi$  is used and the resulting feature maps are cross-correlated:

$$f(z, x) = \phi(z) * \phi(x) + b \quad (216)$$

As shown in Figure 2.4, the output of this network is a kind of heat map defined on a finite grid  $D \subset Z^2$ . Search frame is cropped according to the previous size of the target and center location. The maximum value on the score map is taken into account during online tracking. The position of the value relative to the center indicates the positional shift of the target object on the search frame. The amount of shift multiplied by the stride of the network shows the actual displacement on the search frame. To detect scale changes, mini-batch of multiple scaled images are combined and given to the network as a single input. The output is also a mini-batch of score maps corresponding to the different scales. The score map that gives the maximum score is chosen to be current scale.

Logistic loss is employed to train the network with using both negative and positive sample pairs:

$$l(y, v) = \log(1 + \exp(-yv)) \quad (217)$$

where  $v$  is the score of a sample pair and  $y \in \{+1, -1\}$  is the ground-truth label. Since the output of SiamFC is a 2D score map, the logistic loss function can be extended as follows:

$$L(y, v) = \frac{1}{\|D\|} \sum_{u \in D} l(y[u], v[u]) \quad (218)$$

where  $D \rightarrow \mathbb{R}$  is number of scores in the score map. The network part is trained using Stochastic Gradient Descent (SGD) and logistic loss described in (218). The elements of the score map are considered belonging to a positive sample if they are

within radius  $R$  of the center:

$$y[u] = \begin{cases} +1 & \text{if } \|u - c\| \leq R \\ -1 & \text{otherwise.} \end{cases} \quad (219)$$

SiamFC is first of its kind in terms of fully-convolutional model and similarity learning but not the last. Although the algorithm performs competitively against its rivals when it was published, the bounding box regression still depends on the multi-scale search. The problem with the multi-scale search is that the aspect ratio never changes throughout the tracking process although the target object changes significantly in terms of aspect, which eventually degrades the tracking performance. In the upcoming subsections the two solutions to this problem are explained.

### 2.2.1 Region Proposal Networks

With the emergence of SiamFC-like trackers, it is observed that deep learning based trackers can beat the state-of-art CFB methods. In order to solve scale and aspect ratio problems, inspiring from object detection algorithms, SiamRPN [3] introduces region proposal network for bounding box regression alongside with the object classification. Similar to SiamFC, SiamRPN includes template and detection branches. The network part of SiamRPN is trained offline in an end-to-end fashion using large scale datasets.

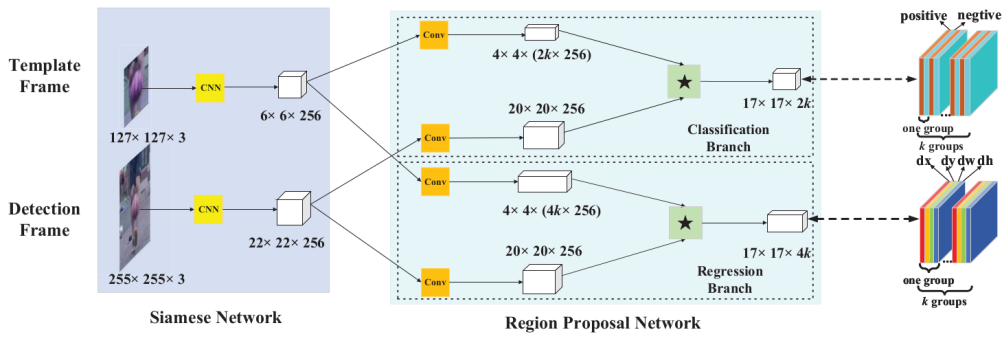


Figure 2.5: The overall framework of Siamese-RPN[3]

As it is shown in Figure 2.5, the proposed method consists of a Siamese feature extraction and region proposal (RPN) subnetworks. The two branches in RPN subnet-

work are responsible for proposal refinement and foreground-background classification. The overall system is trained end-to-end using sample pairs comprehending the target objects. The Siamese feature extraction subnetwork consists of modified AlexNet [24] in which the paddings are removed, and the groups from conv2 and conv4 are extracted. The template and detection branches share parameters of feature extractor network. For convenience, two output feature maps are represented as  $\phi(z)$  and  $\phi(x)$ .

The RPN part of SiamRPN is composed of supervision and correlation sections. The supervision part has two branches for classification and proposal regression. The network is designed to output  $4k$  channels for regression and  $2k$  channels for classification when there are  $k$  number of anchors. Therefore, the correlation parts firstly expand the number of channels of feature map of template branch, which have  $2k$  and  $4k$  times in channel, namely  $[\phi(z)]_{cls}$  and  $[\phi(z)]_{reg}$ , using two convolutional layers. The feature map from detection branch is also fed into two different convolution layers that preserves the number of channels, namely  $[\phi(x)]_{cls}$  and  $[\phi(x)]_{reg}$ . The correlation is performed on both branches as follows:

$$\begin{aligned} A_{wxh2k}^{cls} &= [\phi(x)]_{cls} \star [\phi(z)]_{cls} \\ A_{wxh4k}^{reg} &= [\phi(x)]_{reg} \star [\phi(z)]_{reg} \end{aligned} \quad (220)$$

where  $\star$  represents the convolution operation.

The feature maps of template branch are utilized as a convolution kernel.  $2k$  channel vector in  $A_{wxh2k}^{cls}$  represent the positive and negative activation for each anchor in relative position of original feature map. Similarly,  $4k$  channel vectors in  $A_{wxh4k}^{reg}$  indicate  $dx$ ,  $dy$ ,  $dw$ ,  $dh$  which are used to calculate distance between corresponding ground truth and that anchor. For supervision on the classification branch, softmax loss is utilized. Cross-entropy loss is adopted for classification branch error calculation. For regression, smooth  $L_1$  loss with normalized coordinates is used.

Assuming that  $A_x, A_y, A_w, A_h$  indicate center location and shape of the anchor boxes, and for ground truth boxes,  $T_x, T_y, T_w, T_h$  are utilized, then normalized distance can

be written as:

$$\begin{aligned}\delta[0] &= \frac{T_x - A_x}{A_w}, \delta[1] = \frac{T_y - A_y}{A_h} \\ \delta[2] &= \ln \frac{T_w}{A_w}, \delta[3] = \ln \frac{T_h}{A_h}\end{aligned}\tag{221}$$

They are fed into the smooth  $L_1$  loss which is as follows:

$$smooth_{L_1}(x, \sigma) = \begin{cases} 0.5\sigma^2 x^2, & |x| < \frac{1}{\sigma^2} \\ |x| - \frac{1}{2\sigma^2}, & |x| \geq \frac{1}{\sigma^2} \end{cases}\tag{222}$$

While  $L_{cls}$  indicates the cross-entropy loss, the regression loss can be written as follows:

$$L_{reg} = \sum_{i=0}^3 smooth_{L_1}(\delta[i], \sigma)\tag{223}$$

Finally, the total loss can be written as follows:

$$loss = L_{cls} + \lambda L_{reg}\tag{224}$$

where  $\lambda$  is hyper-parameter to control the effect of two loss functions.

During the training, image pairs from Youtube-BB[25] and ILSVRC[13] are randomly chosen. Feature extraction part is pretrained using ImageNet [26]. Positive and negative instances are chosen according to a strategy. The anchors which have  $IoU > 0.6$  are defined as positive instances, while the anchors having  $IoU < 0.3$  with the ground truth boxes are denoted as negative instances. Furthermore, number of samples for each pair is limited to 64 samples in which at most 16 of them are positive.

No parameter update is done during online tracking, and the classification branch is exposed to the cosine windows multiplication in order to prevent large displacement.

Also, to penalize large size and ratio changes, the following penalty is applied:

$$penalty = e^{k * \max(\frac{r}{r'}, \frac{r'}{r}) * \max(\frac{s}{s'}, \frac{s'}{s})} \quad (225)$$

where  $k$  is a hyper parameter,  $r$  and  $r'$  indicate ratio of height and width of the proposal and that of last frame, respectively.  $s$  shows the overall scale of the proposal and  $s'$  denotes that of last frame.

After performing these operations, the anchor result on regression output map that is corresponding to the one having largest confidence score on the classification output map of positive part is chosen to be used. Target location and size are updated accordingly. This algorithm have beaten almost all of the state-of-the-art trackers when it was published in 2018, and it was considered as a milestone in visual tracking since the gap between object detection and tracking has been decreased significantly. Moreover, the scale and aspect ratio of the object bounding box are computed much better than the other visual trackers.

After this astonishing work, tracking community tried to utilize deeper neural networks in order to increase discrimination ability of the trackers. However, using deeper networks for feature extraction did not result in better tracking performance as expected. The reason behind this is that localization ability of the algorithms decreased as the network goes deeper due to increasing receptive fields of neurons. SiamRPN++ [4] has managed to utilize ResNet50 [27] as a feature extractor by modifying the network so that the receptive fields do not become too large. SiamRPN++ also utilized feature from middle layers for classification and regression as it is illustrated in Figure 2.6.

These additional features made SiamRPN++ one of state-of-the-art algorithm, and still to be beaten up to date.

### 2.2.2 Intersection Over Union Net

The developments on the object localization methods for object detection continue to inspire visual object tracking algorithms. One of the object detection algorithms,

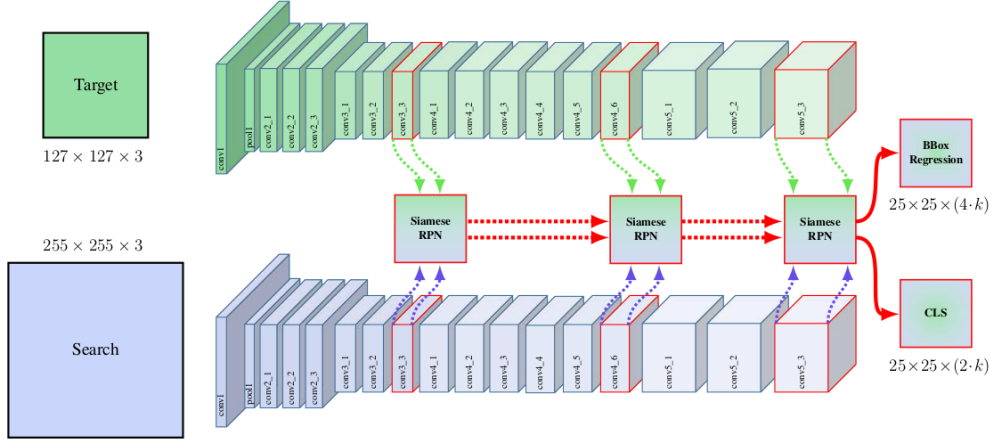


Figure 2.6: The general view of SiamRPN++ framework[4].

known as IOU-Net [28], claims that modern object detectors, using non-maximum suppression (NMS) and bounding box regression to localize objects, pay attention to the classification confidence while the localization confidence is not available. They indicate that this makes accurately detected bounding boxes vanished during NMS. IOU-Net substitutes classification confidence with the predicted IOU. This technique, namely IOU-guided NMS, helps to prevent the suppression of accurately detected bounding boxes due to the misleading classification confidences. They proposed an optimization-based bounding box refinement method that predicts the IOU between the detected bounding box and ground truth box. By solving the IOU optimization with gradient descent, they tried to increase the IOU value between detected box and ground truth.

After the inspiration on IOU-Net methodology, the ATOM [29] tracker proposed to use similar IOU optimization for target object localization. The overview of the ATOM tracker is shown in Figure 2.7. Similar to the Siamese trackers, this algorithm also extracts feature maps for reference and test images using a deep network, namely ResNet-18[27]. IOU-predictor network is used for target location estimation. IOU-predictor network takes four inputs: bounding box of the target object in the reference image, backbone feature maps from reference image, bounding box proposals in the test image and backbone feature maps from test image. It outputs the predicted IOU score for each of the test image bounding box proposals. During online tracking, the bounding box estimation of the target object in the test image is calculated using

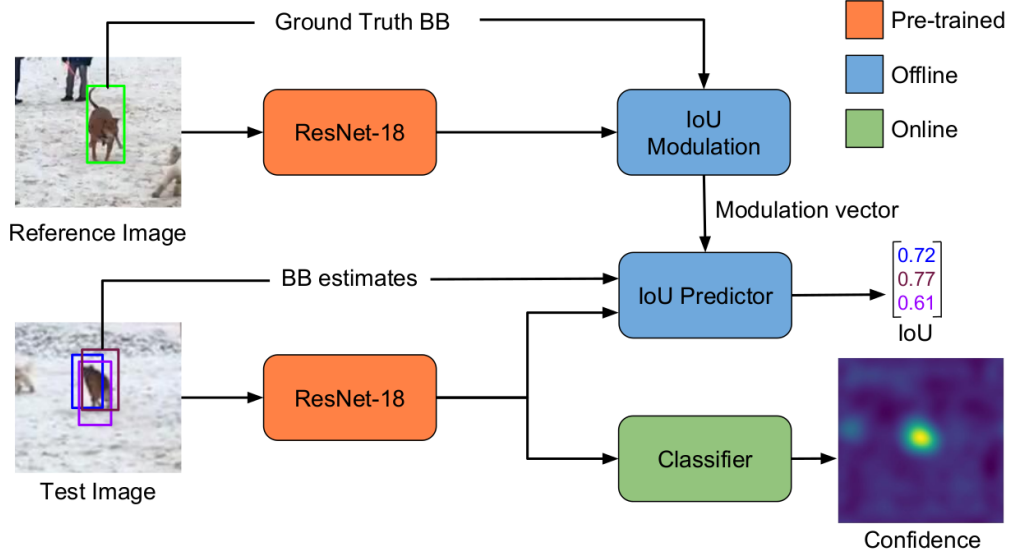


Figure 2.7: Overview of network architecture of ATOM[29].

gradient descent by maximizing the IOU score.

Although the bounding box estimation is done in a good way, the target estimation module has difficulties to distinguish between background objects and target object. Therefore, a second head on the target estimation module ('Classifier' block in Figure 2.7) is employed to improve the estimation module, whose main purpose is to carry out this discrimination. The goal of classification module is to provide a rough 2D-location of the object. Although other parts of the network are trained offline, the classification part is trained in an online manner. The target classification module is 2-layer fully convolutional neural network, formally defined as:

$$f(x; w) = \phi_2(w_2 * \phi_1(w_1 * x)) \quad (226)$$

where  $*$  indicates multi-channel convolution,  $\{w_1, w_2\}$  are network parameters,  $x$  is the feature map resulting from backbone network, and  $\phi_1, \phi_2$  denote activation functions. Following the success of the CFB approaches, a similar objective function is

used for classification error:

$$L(w) = \sum_{j=1}^m \gamma_j \| (f(x_j; w) - y_j) \|^2 + \sum_k \lambda \| w_k \|^2 \quad (227)$$

where  $x_j$  is feature map of training sample,  $y_j \in R^{W \times H}$  is a Gaussian function,  $\gamma_j$  is the weight on samples and  $\lambda_k$  is the regularization term for the network parameters  $w_k$ .

Due to slow convergence of stochastic gradient descent (SGD) based optimization approaches, authors proposed more sophisticated optimization strategy that is faster, therefore, more suitable for online learning during tracking. The solution includes Gaussian-Newton approximation and Conjugate Gradient method. Since the full explanation is outside the scope of this thesis, it is not explained in detail (see details in [30]).

Similar to SiamRPN++, the improvement on the bounding box regression and classification part made ATOM algorithm to become one of the state-of-the-art trackers in the modern era.

### 2.3 Semi Supervised Training

Machine learning has tried to solve various problems using supervised learning techniques, and achieved great success particularly in classification and regression. Training datasets, that contain large amount of training examples, are used to learn predictive models. However, it is difficult to obtain strong supervision for every task due to high cost of labeling process. Therefore, developing weak supervision methods for machine learning algorithms is an important task. In this section, weakly supervision techniques used in machine learning algorithms are going to be investigated. In general, three types [31] of weak supervision are going to be reviewed, which are incomplete, inexact and inaccurate supervision. In Figure 2.8, these three types are illustrated with their subsections.

Inexact supervision deals with the situation where there are labels for supervision but



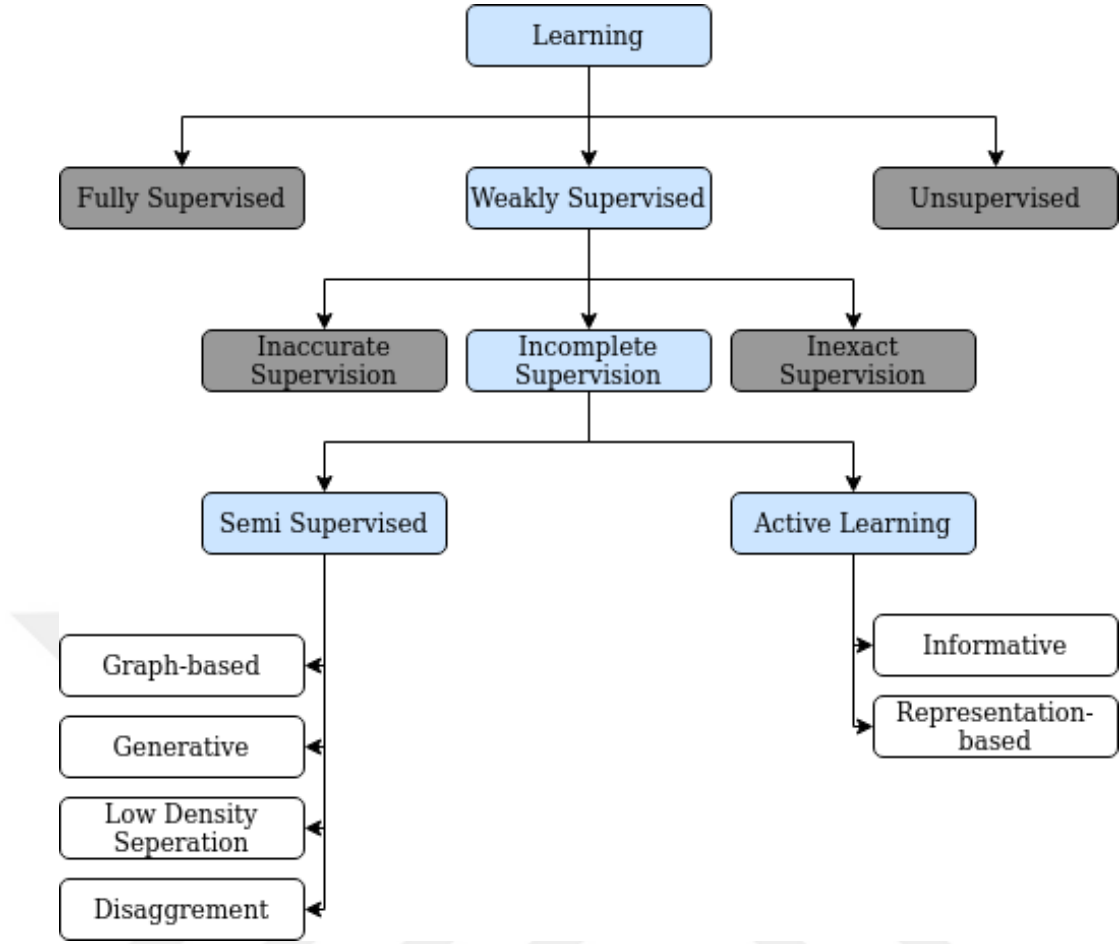


Figure 2.8: Illustration of three typical types of weak supervision[31].

not as exact as desired. For example, it is exhausting to label every frame where an anomaly occurs on anomaly videos. However, it is known that the video contains at least one anomaly type. Formally, the task is to learn if a video segment contains abnormal situation or not. [32] treats anomalous and normal videos as video segments and bags. Video segments on an anomalous video constitute a positive bag whereas segments on normal video generate a negative bag. This methodology is also called Multiple Instance Learning (MIL). Video segments in each positive and negative bags pass through a classifier network that generates a score for each segment. MIL related loss function tries to maximizes the difference between the maximum score on positive and negative bags. This type of solutions using MIL has been successfully applied to various tasks, such as image/text categorization/retrieval/annotation [33, 34, 35], object detection [36], object tracking [37] etc. In these tasks, it is a design choice for what to use as an instance (image, text, video segment) in a bag.

Inaccurate supervision tries to solve the problem of the supervision information that is not always ground truth. More precisely, some labels may contain errors. The erroneous labels may result from incapable human expert annotators for the labeling task. There may also exist adversaries who annotate incorrect labels deliberately. A basic idea is trying to identify the potentially wrong labels, and then try to do correction on them. A data-editing [38] approach is developed for this purpose. Another scenario of inaccurate supervision is *crowdsourcing*. It is used to outsource unlabeled instances to a large number of annotators. A famous crowdsourcing system is Amazon Mechanical Turk (AMT). To infer the ground truth from the crowd, many methods are developed, such as majority voting strategy with ensemble methods [39], expectation maximization (EM [40]), the minimax entropy principle[39] etc.

Incomplete supervision deals with training data having only a subset of it is labeled while the remaining part is unlabeled. There are two major methodologies exploiting incomplete supervision, i.e. active learning [41] and semi-supervised learning. Active learning tries to classify unlabeled instances whether they are suitable for training or not. By doing so, the labeling cost for training can be minimized. Active learning utilizes human experts or some algorithms as 'oracle' for choosing most valuable unlabeled samples to be labeled with ground truth. Generally, active learning algorithms utilize two selection methods, i.e. informativeness and representativeness. Representativeness tries to predict whether an instance is beneficial to constitute structure of main task better. Nonetheless, informativeness is interested in decreasing the uncertainty of the model by measuring the unlabeled sample.

Active learning tries to utilize human interaction to exploit unlabeled data, whereas semi supervised learning does not require any human intervention. It makes use of both labeled and unlabeled instances to train a model. Semi supervised learning has a special type called transductive learning. Different to the semi supervised learning, it assumes that unlabeled data is actually the test data, tries to maximize the performance on it. However, unlabeled data is not assumed to be test data in actual semi supervised learning. The workflow of active, transductive and semi supervised learning methodologies are shown in Figure 2.9.

Semi supervised learning can be investigated in four methods. First one is gener-

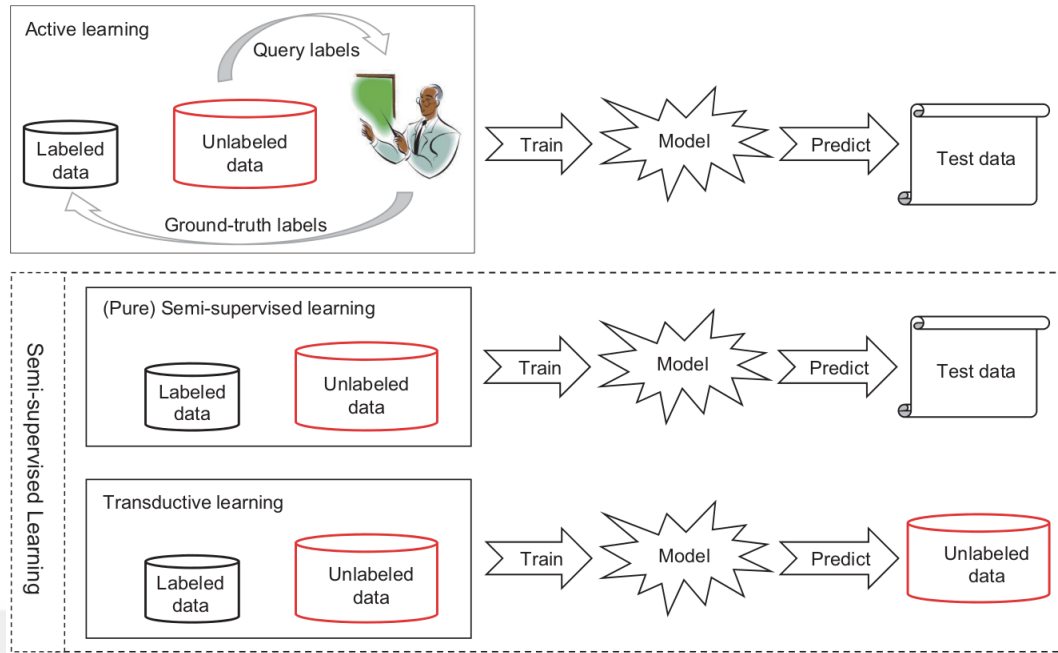


Figure 2.9: Illustration of three incomplete supervision methods[31].

active methods that expect unlabeled and labeled data to be composed in same domain and shows similar properties. Generative methods try to estimate unlabeled samples by using algorithms like EM (Expectation Maximization) [42]. In order to make generative methods work, domain specifications must be known in advance. Second method is called as graph based methods. These methods generate graphs where nodes are training samples and edges are similarity measure between them, such as distance. The efficiency of these methods are generally depends on structure of the graph. Computational complexity of graphs becomes usually very high when the number of samples are large. Moreover, it is hard to settle in unlabeled samples without reconstructing the graph. Third approach to semi supervised learning is low density separation. The purpose of this method is to make the classification boundary pass through low density region in the sample space. The well known method for this approach is S3VM [43] that attempts to draw boundary line on less dense region while classifying the labels accurately. S3VM is a sophisticated optimization problem that depends on the efficiency of it. Fourth method is denoted as disagreement based method. Multiple models participate to utilize unlabeled samples. A model, having very confident result on an unlabeled sample, sets its output as a label to the unlabeled sample, and transfer this information for training of another model. This

methodology is also seen as an automated active learning.

### 2.3.1 Object Detection and Semantic Segmentation

In this section, a short literature review on weakly supervised object detection and segmentation is described.

With the success of data-driven object detection on the literature, benchmark datasets are getting larger and larger. As the number of sets enlarges, the annotation workload also increases. That results in heavy human resource, a lot of time and cost. Therefore, researches try to create weakly supervised object detector models, *e.g.* using only category labels instead of bounding boxes while learning a detector. Most of the works on the literature utilizes the MIL paradigm [44, 45, 46]. A hypothesis states that among the bounding box proposals the actual one should contribute to image classification more than the others. Therefore, MIL based studies try to put larger weights to the proposals which are consistent with the classification labels. With recent developments on deep learning, MIL based methods are combined with deep networks [47, 48, 49]. This has significantly enhanced the accuracy of MIL based methods.

In WSDDN [47], two branch detection is proposed. One branch is responsible for proposal selection, while other branch acts as proposal classifier. The method starts with ImageNet pretrained existing network such as AlexNet. After dividing image into regions  $R$ , the region-level features are extracted from the top of the CNN convolutional layers using spatial pyramid pooling. Subsequently, the network is divided into two branches, denoted as recognition and detection. On the recognition branch, pooled region-level features are associated with a class score. On the other hand, the detection branch tries to compute a probability distribution over region-level features to decide the region having most salient structure. Finally, two branches are merged in order to classify the whole image, which is used to provide supervision during learning. The overview of WSDDN is shown in Figure 2.10.

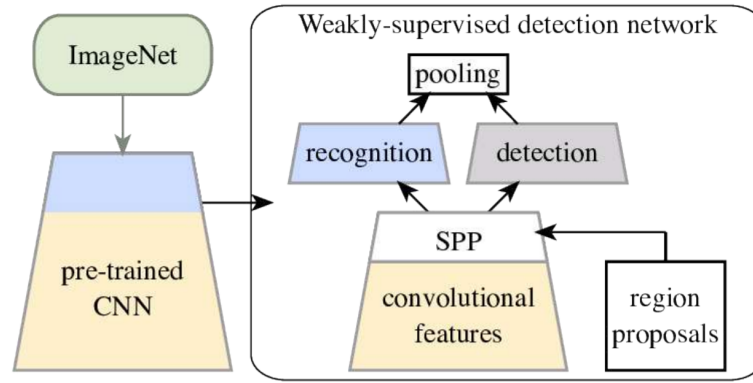


Figure 2.10: The overview of Weakly Supervised Deep Detection Network[47].

After promising results of WSDDN, most of the latter works continue to improve the algorithm. With the addition of recursive refinement branches in [48], WSDDN is further enhanced. Another promising and effective study is performed in [50] by learning models from easy to hard progressively and using continuation optimization. In addition to these single-stage models, multiple-stage models are also developed in which the single-stage detection results are used as pseudo labels. WSDDN is also used as a image difficulty estimator in [51] in order to train Fast-RCNN [52] with curriculum learning strategy.

Semantic segmentation is also another way to localize objects. In [53], in order to learn category labels for segmentation map in a weakly supervised manner, segmentation map from output of a network is taken and merged with global classification prediction. [54] tries to add positive and negative evidence into aggregation function in order to represent both presence and absence of the target class. To drive the network for generation of responses on the undetected parts, [55] proposes masking the response region of the former forward passes with recurrent adversarial learning strategy. Segmentation is also utilized in weakly supervised object detection since there are relations between detection and segmentation. Map resulting from segmentation branches is used to eliminate false positives on detection branches [56]. [57] proposes utilizing weakly supervised segmentation models for both generation and elimination of proposals. More recent work [58] proposes a better collaboration model using generative adversarial localization structure, and obtains state-of-the-art results on PASCAL VOC 2007 [59] and 2012 [60] datasets.

### 2.3.2 Pose and Depth Estimation

In this section, a brief related work is given about weakly supervised pose and depth estimation methods.

Various studies have emerged for unsupervised and weakly supervised depth estimation in the literature. Garg et al. [61] proposed that ego-motion and depth can be learned at the same time using the known inter-view displacement and predicted depth to compose reverse warping of the target frame. [62] illustrated a fully differentiable solution to predict ego-motion and depth utilizing deep networks. Wang et al. [63] used direct visual odometry with differentiable implementation to train a depth CNN predictor. In [64], randomized layer normalization was proposed during training, and occlusions were attacked differentiably and geometrically. In [65], features of left and right stereo pairs were extracted using Siamese network. These methods performed learning without any labeled data on depth estimation, however, the performances were lower than the supervised methods. Some methods tried to solve this task utilizing weakly labeled pixel pairs. Chen *et al.* [66] tried to generate relative depth maps using hourglass network trained on weakly labeled pixel pairs. In [67], structure-from-motion algorithm was used to create depth ground-truth for landmarks, and then generated weak pixel pair labels from landmarks and foregrounds. Most recent work SUW-Learn [68] purposed joint supervised, unsupervised and weakly supervised learning scheme for depth estimation. Supervised learning is performed in a conventional way such that the network tries to estimate depth, and loss is generated using the ground truth depth. Weakly supervision is performed by utilizing weakly labeled pixel pairs similar to [67]. The unsupervised learning is staged for ego motion estimation. It simply takes 3 successive frames and depth estimation of the network as inputs, and tries to estimate the camera motion. Loss values of these three learning models are summed for total loss estimation, and the network is optimized accordingly. The aforementioned methodology is illustrated in Figure 2.11. The results show that the network has better generalization ability leading the results being better than supervised learning only.

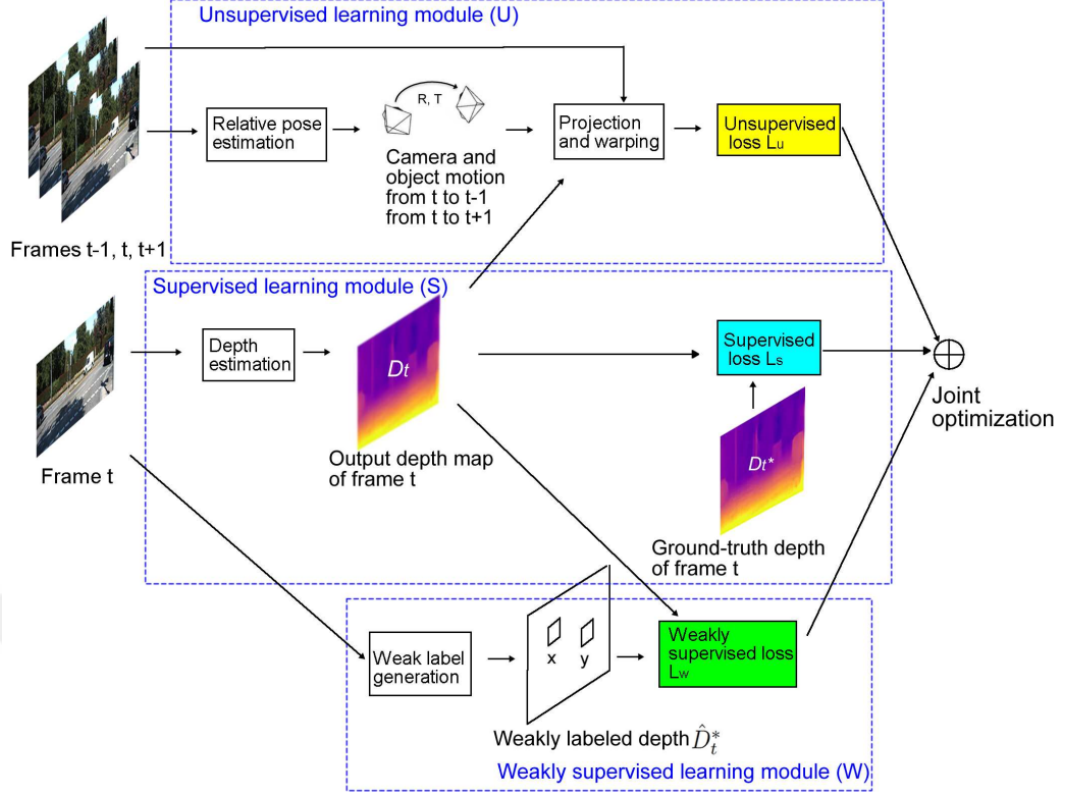


Figure 2.11: The overview of learning schemes used in SUW-Learn[68]

Similarly, there exist several semi-supervised learning methods for human pose estimation as well. However, due to the difficulty of this work, designs are limited to the simpler tasks. For example, using only walking part of the human pose sequences, 3D pose models are trained using manifold regularization and GMM-based clustering to learn unlabeled data in [69] and [70]. Also, huge amount of synthetic images are generated to overcome data shortage. In [71], the pose estimation of the network is evaluated using action-specific pose model if there is no label for that specific instance. True positives are regarded as pose annotations and re-learned.





## CHAPTER 3

### SUPERVISED AND UNSUPERVISED DCF BASED TRACKERS

#### 3.1 DCFNet: DCF Network For Visual Tracking

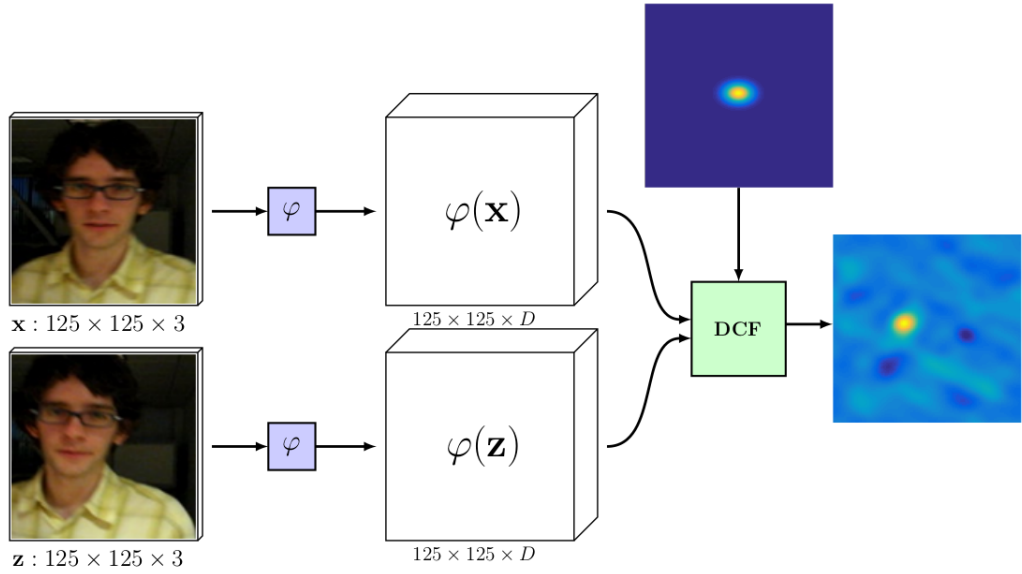


Figure 3.1: The general DCFNet architecture [8]

DCFNet tracker, whose architecture is shown in Figure 3.1, trains discriminative correlation filters on the features of the target patch  $\phi(x) \in \mathbb{R}^{M \times N \times D}$  and the ideal output  $y \in \mathbb{R}^{M \times N}$  that is a peaky-shaped Gaussian function at the center. By minimizing the output ridge loss, the desired filter  $w$  is obtained using:

$$\epsilon = \left\| \sum_{l=1}^D w^l \otimes \phi^l(x) - y \right\|^2 + \lambda \sum_{l=1}^D \|w^l\|^2 \quad (31)$$

where the constant  $\lambda \geq 0$  is regularization coefficient,  $\circledast$  means circular correlation and  $w^l$  indicates the channel  $l$  of filter  $w$ . As described in section 2.1.2 of the thesis, the solution becomes as follows:

$$\hat{w}^l = \frac{\hat{\phi}^l(x) \odot \hat{y}^*}{\sum_{k=1}^D \hat{\phi}^l(x) \odot (\hat{\phi}^l(x))^* + \lambda} \quad (32)$$

where the letters with hat denotes their discrete Fourier transform version,  $\odot$  represents the Hadamard product and  $y^*$  indicates the complex conjugate of a complex number  $y$ . During the detection process, a search patch is cropped and the features  $\phi(z)$  is obtained in the new frame. The movement of the object can be calculated by searching the peak value of correlation response map  $g$ :

$$g = F^{-1}(\sum_{l=1}^D \hat{w}^{l*} \odot \hat{\phi}^l(z)) \quad (33)$$

The convolutional neural network part responsible for feature extraction is very lightweight, and mainly consists of conv1 from VGG [1]. The pooling layer are extracted and number of output channels is made 32. The network is trained offline and fixed during online tracking. The objective function to train the network is formulated as:

$$L(\theta) = \|g - \tilde{g}\|^2 + \gamma \|\theta\|^2 \quad (34)$$

An explicit regularization is added in order to prevent objective function to get a non-convergence condition. Stochastic gradient descent (SGD) is used as optimizer, and the weight decay method is utilized to implicit this regularization. Moreover, at the end of the convolutional layers, a local response normalization (LRN) layer is added to avoid large magnitude of feature map values.

The filters  $w$  are updated during online tracking. The update rule is similar to the one described in (213), however, it is extended to time series. The optimization problem

defined in (31) can be rewritten with additional incremental mode as:

$$\epsilon = \sum_{t=1}^p \beta_t \left\| \sum_{l=1}^D w_p^l \circledast \phi^l(x) - y \right\|^2 + \lambda \sum_{l=1}^D \|w_p^l\|^2 \quad (35)$$

Utilizing the (35), the closed form solution in (32) becomes as:

$$\hat{w}_p^l = \frac{\sum_{t=1}^p \beta_t \hat{y}^* \odot \hat{\phi}^l(x_t)}{\sum_{t=1}^p \beta_t (\sum_{k=1}^D \hat{\phi}^k(x) \odot (\hat{\phi}^k(x))^* + \lambda)} \quad (36)$$

Due to aforementioned time series properties, DCFNet can be seen as an RNN network.

During training of DCFNet, TempleColor [10], NUS-PRO [9] and UAV123 [11] sets are used. The videos that have overlap with the test set are excluded. The resulting training set includes 166643 frames. Each pair of frames is selected within the range of 10 frames. Subsequently, the target patches are cropped with  $1.5x$  padding size using the ground truth labels. The cropped patches are resized to  $125 \times 125$ . SGD with momentum of 0.9 is used to train the network from scratch. The learning rate is set to  $1e-5$ . The weight decay  $\gamma$  is set to  $5e-4$ . Training lasts 20 epoch with batch size of 16. The regularization coefficient  $\lambda$  and online learning rate  $\beta_t$  are set to  $1e-4$  and  $8e-4$ , respectively. During both offline training and online tracking, Gaussian spatial bandwidth is fixed to 0.1. For scale change, patch pyramid with 3 scales where value  $a$  (see 2.1.3) is set to 1,0375.

OTB2013 [72] and OTB2015 [12], also named as OTB50 and OTB100 (see section 4.3 for details), are utilized for testing the performance of the DCFNet tracker. Success plots of DCFNet reported along with other correlation filter based and other trackers in Figure 3.2. The numbers stated on the figure legend indicate the AUC (area under curve) value of trackers.

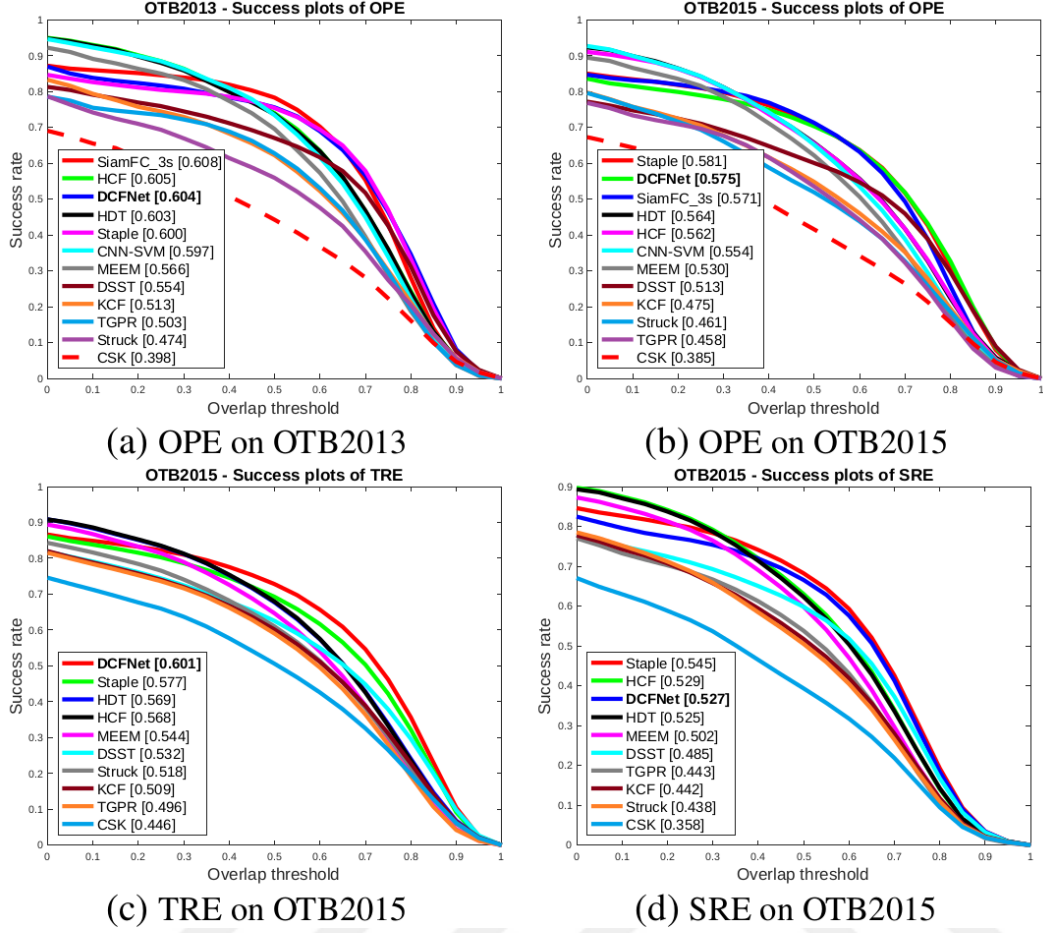


Figure 3.2: OTB2013 and OTB2015 success plots of DCFNet along with other correlation based trackers and others: KCF[73], SiamFC[2], CNN-SVM[74], CSK[75], DSST[18], MEEM[76], HCF[77], TGPR[78], HDT[79], Staple[80].The figure is taken from [8]

### 3.2 UDT: Unsupervised Deep Tracking

UDT [7] claims that visual object tracking utilizing deep networks for feature extraction can be learned using forward and backward tracking without ground truth labels. The comparison between the supervised and proposed unsupervised learning is illustrated in Figure 3.3 Forward tracking is performed by initially annotating the randomly drawn region on the first frame, and tracking the object on the subsequent frames. The predicted location in the last frame is used for tracker initialization, and the tracker is run from last frame towards the first frame during backward tracking.

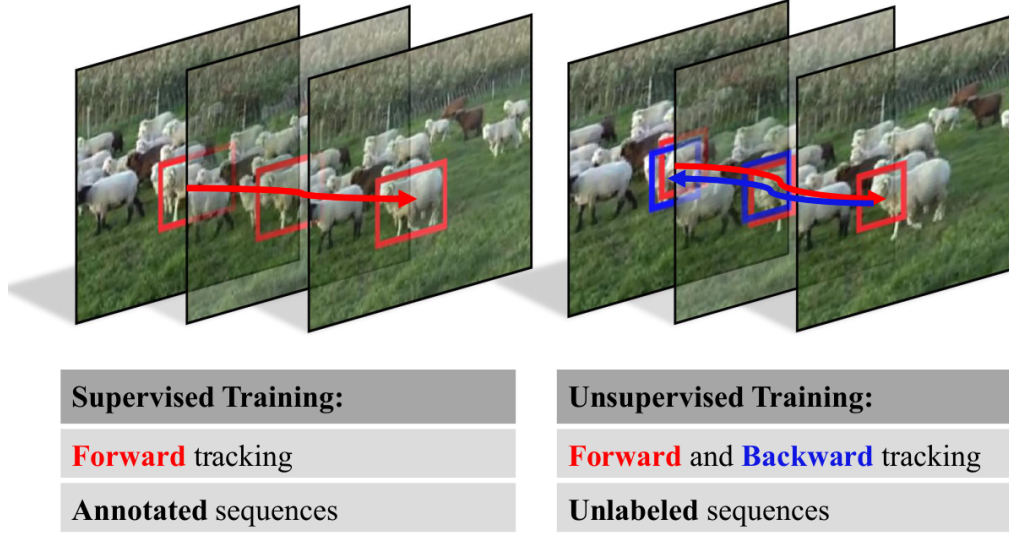


Figure 3.3: Supervised and unsupervised learning comparison. UDT tracker is trained using forward and backward tracking without heavyweight annotations [7].

The target position computed in the first frame using backward tracking is desired to be same position with the initial randomly drawn region. Consistency loss is used to measure the difference between backward and forward trajectories. The architecture of UDT is shown in Figure 3.4.

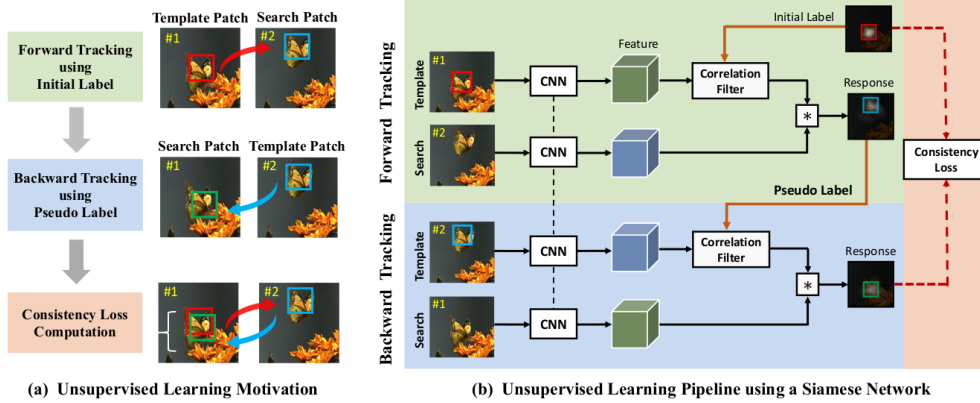


Figure 3.4: General unsupervised deep tracking algorithm. In (a) forward-backward tracking methods is illustrated. In (b) unsupervised training procedure of Siamese correlation filter network is shown [7].

During forward tracking, two consecutive frames  $P_1$  and  $P_2$  are cropped to template patch  $\mathbf{T}$  and search patch  $\mathbf{S}$ , respectively. Randomly drawn object in  $\mathbf{T}$  is used for generating the required Gaussian label ( $\mathbf{Y}_T$ ) centered at bounding box center for initialization of the Siamese tracker. The correlation filter  $\mathbf{W}_T$  generated using  $\mathbf{Y}_T$  is computed using close form solution (see section 2.1) as follows:

$$W_T = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\phi(T)) \odot \mathcal{F}^*(Y_T)}{\mathcal{F}^*(\phi(T)) \odot \mathcal{F}(\phi(T)) + \lambda}\right) \quad (37)$$

After computation of  $\mathbf{W}_T$ , the target is tracked on search patch. The output response map of the tracker, denoted as  $\mathbf{R}_S$ , is calculated as:

$$R_S = \mathcal{F}^{-1}(\mathcal{F}^*(W_T) \odot \mathcal{F}(\phi(S))) \quad (38)$$

Using the Gaussian response map of the tracker output  $\mathbf{R}_S$  for frame  $P_1$ , a pseudo Gaussian label, denoted as  $\mathbf{Y}_S$ , centered on the maximum value of  $\mathbf{R}_S$  to be used for backward tracking is generated. In the backward tracking,  $\mathbf{S}$  is treated as template patch and  $\mathbf{T}$  is seen as search patch. The tracker is reinitialized using pseudo label  $\mathbf{Y}_S$ . The backward tracking is performed from original search patch to initial template patch. Note that the network parameters are fixed during these tracking steps. The Gaussian response of the tracker on original template patch is denoted as  $\mathbf{R}_T$ . Ideally,  $\mathbf{R}_T$  should be as similar as the originally given label  $\mathbf{Y}_T$  that is Gaussian label with peak located at the target position. By using this requirement, the feature extractor network  $\phi(\cdot)$  can be trained by minimizing the reconstruction loss computed as:

$$L_{un} = \|R_T - Y_T\|_2^2 \quad (39)$$

After loss computation, the network weight parameters are updated with the back-propagation of the calculated error.

The tracker may sheer away from the target in the forward tracking in practice, however, it still returns to the initial location after backward tracking. In this case, the proposed loss function returns small numbers which prevents penalizing this behaviour. To overcome this problem, multiple frame validation is proposed in [7]. It involves

another frame  $P_3$  which is a subsequent frame after  $P_2$ . The forward tracking and backward validation are performed on three frames in the same manner. As shown in Figure 3.5, the forward tracking is firstly performed from template patch to search patch #1. Subsequently, according to the output response map, the tracker is reinitialized on that patch using the pseudo Gaussian label. After that, forward tracking from search patch #1 to #2 is performed. Finally, backward tracking from search patch #2 to template patch is done using newly calculated pseudo Gaussian label considering the tracker output on search patch #2. The difference between initial Gaussian label and final response map is computed using (39), and training of the network is realised. With utilization of multiple frame validation, the verification failures are expected to be reduced. The reconstruction error and the computed loss tend to be amplified in case of failure, which will penalize the network. The Figure 3.5 illustrates the multiple frame validation method.

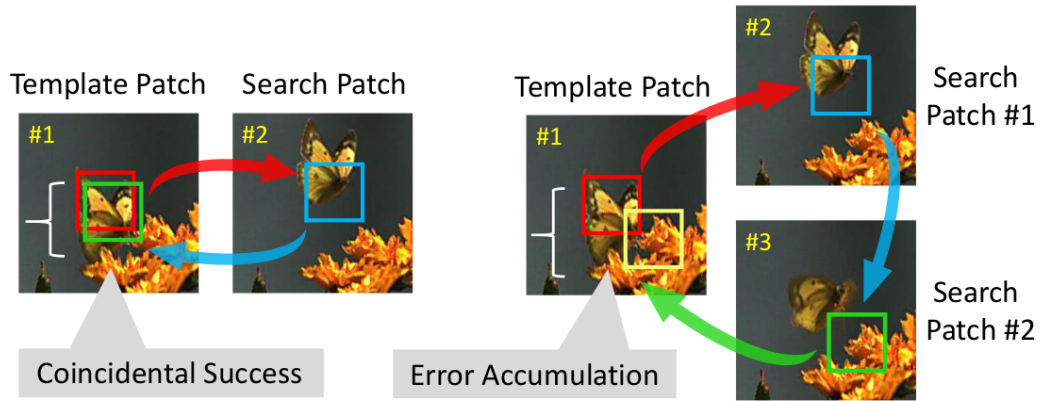


Figure 3.5: Single frame validation on the left and multiple frames validation on the right. The incorrect forward tracking in single frame validation may not be realized. With usage of multiple frames, localization error can build up to prevent the prediction consistency during forward tracking and backward verification.[7]

During training of UDT, the noisy training pairs may appear since the randomly drawn bounding box can be a noisy background content or an occluded target. This causes significantly large losses, and prevents the network from convergence. Therefore, 10% of the overall training sample pairs containing large error values are excluded in order to prevent noisy pairs. The exclusion is executed during a mini batch training including multiple training pairs. Loss value of each training pair is multiplied with

$A_{drop}^i$ , which is 0 for 10% of mini batch having largest loss value, and 1 for the remaining pairs.

The authors of UDT [7] also claim that objects having larger motion are more effective for training the network. Since targets are randomly chosen from frames, they may include static objects such as soil, grass, tree or sky. These objects do not move, therefore, contribute less on the training. In order to boost the effect of largely moving objects, motion weight vector  $A_{motion}^i$  is assigned to all pairs in mini batch. Its computation is as follows:

$$A_{motion}^i = \|R_{S_1}^i - Y_T^i\|_2^2 + \|R_{S_2}^i - Y_{S_1}^i\|_2^2 \quad (310)$$

where  $Y_T^i$  is the initial label,  $Y_{S_1}^i$  is the pseudo label generated on search patch #1.  $R_{S_1}^i$  and  $R_{S_2}^i$  are the output response maps of the tracker for search patch #1 and #2, respectively. As  $A_{motion}^i$  becomes large, it means that the target object makes large displacement on multiple frame tracking. It is a hard sample pair that the network should put more attention for better training. For size of  $n$  mini batch, the normalization on the binary and motion weight vector is performed as:

$$A_{norm}^i = \frac{A_{drop}^i * A_{motion}^i}{\sum_{i=1}^n A_{drop}^i * A_{motion}^i} \quad (311)$$

After the aforementioned modifications on the reconstruction loss, the final loss for a mini batch can be calculated as:

$$L_{un} = \frac{1}{n} \sum_{i=1}^n A_{norm}^i \|R_T^i - Y_T^i\|_2^2 \quad (312)$$

Similar to DCFNet [8], a very lightweight network structure is used for feature extraction. The network properties of UDT is shown in Table 3.1 including layers and their properties. As training dataset, ILSVRC2015 [13] is used without any preprocessing on the data. The samples are created by simply cropping half of the whole image from the center of each frame. The cropped patches are then resized to  $125 \times 125$  before going into to the network for feature extraction. Three frames for template and



Table 3.1: Network configuration of UDT

| Layer Type | Number of Filters | Kernel Size | Stride |
|------------|-------------------|-------------|--------|
| Conv1      | 32                | 3           | 1      |
| Relu       | -                 | -           | -      |
| Conv2      | 32                | 3           | 1      |
| LRN        | -                 | -           | -      |

search patches are chosen from successive 10 frames in a video so that the the object on the cropped region does not go out of scope. In Figure 3.6, some of sample patches cropped from the center of frames are illustrated. It can be seen that the content can include less meaningful objects (e.g. images on the last row of the figure).

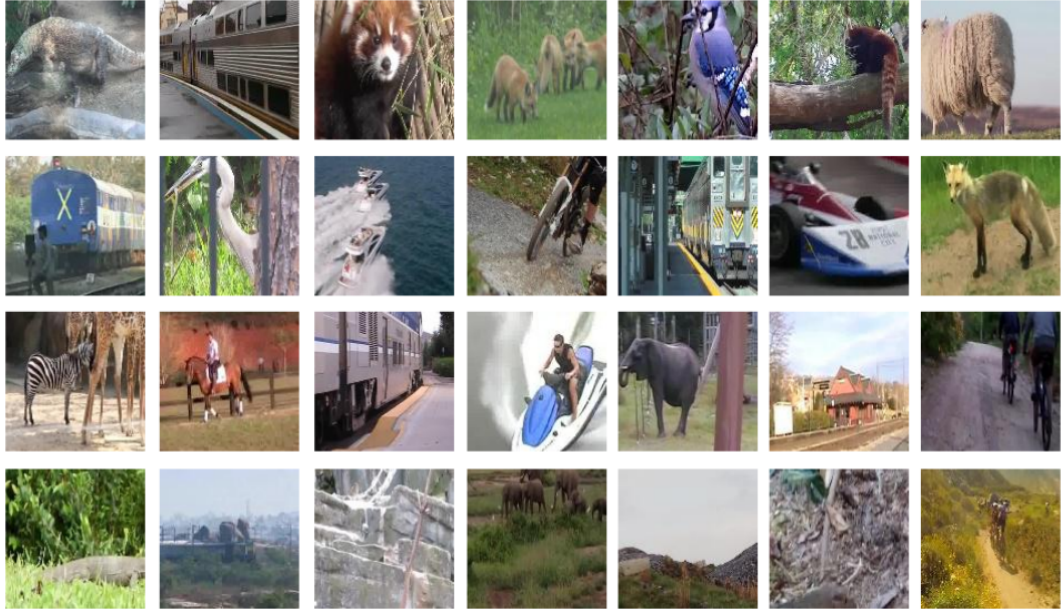


Figure 3.6: Randomly cropped images from the center of frames in ILSVRC2015 dataset for UDT training [7].

During online tracking, the filters  $w$  are updated with an update rule similar to the one described in 213:

$$W_t = (1 - \alpha_t)W_{t-1} + \alpha_t W \quad (313)$$

In addition to the incremental model update, scale change is estimated similar to the one in DCFNet [8] (see section 3.1). Three scales are searched for this purpose with  $\alpha$  equals to 1.015. In [7], authors also proposed an improved tracker called UDT+. It uses advanced model update and different DCF formulation according to the [81].

The optimization during training is done by using stochastic gradient descent (SGD). The weight decay and momentum are set to  $5e-4$  and 0.9, respectively. The learning rate is set to  $1e-2$  initially, and decays until  $1e-5$  throughout 50 epochs. The mini batch size  $n$  is fixed to 32. Tests are performed on OTB2015 [12] and Temple-Color [10] datasets. Success and precision plots for both test sets are shown in Figure 3.7 and Figure 3.8, respectively. They are compared with other trackers such as SiamFC[2], DSST[18], KCF[73], ACT[82], CFNet[83], ACFN[84], SCT[85], CSRDCF[86]. The number on the legend of the figures indicates the area under curve (AUC) value of each tracker.

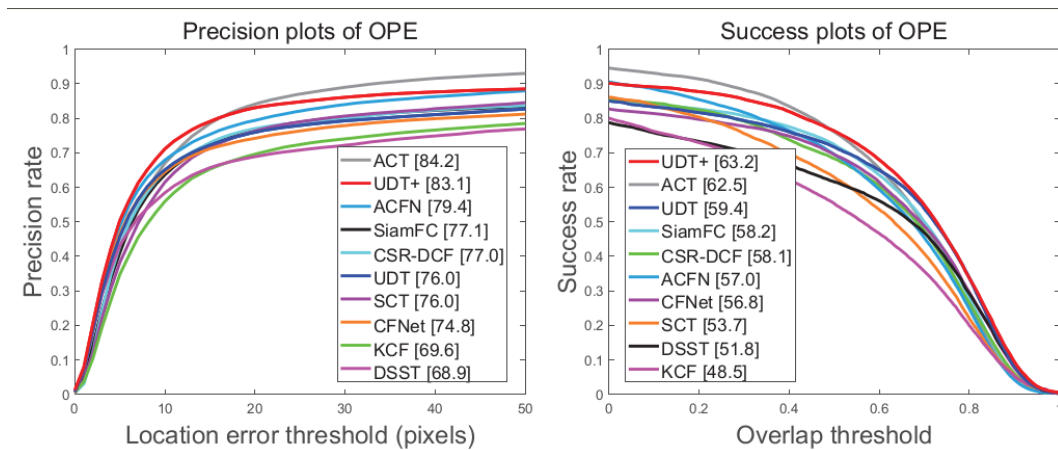


Figure 3.7: OTB2015 test results of UDT and UDT+ trackers in terms of success and precision plots with other trackers [7].

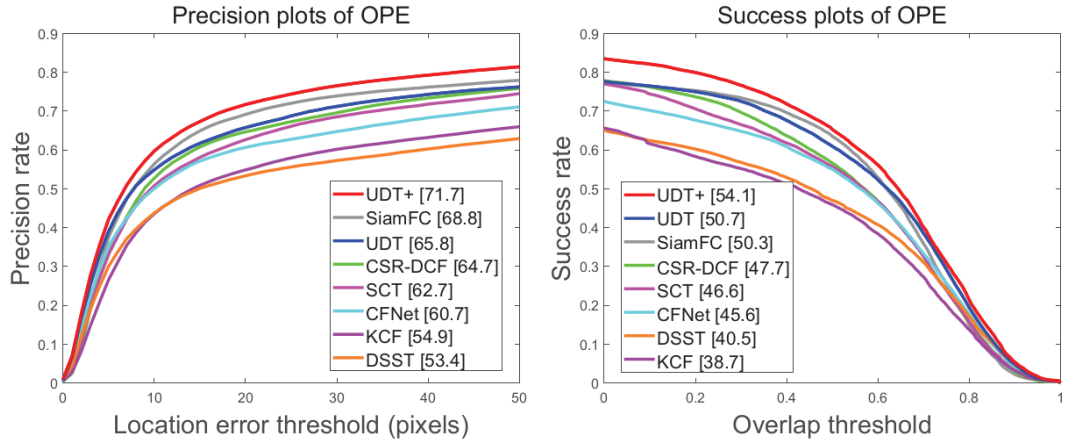


Figure 3.8: TempleColor[10] test result of UDT and UDT+ trackers in terms of success and precision plots with other trackers [7].

Results indicate the effectiveness of UDT and UDT+ trackers when they are trained in an unsupervised manner. Due to its very light weight network, UDT can run faster than real time. Authors in [7] reports 70 fps for UDT and 55 fps for UDT+ with a computer having NVIDIA GTX 1080Ti GPU and 4.00GHz Intel Core I7-4790K.



## CHAPTER 4

### SEMI SUPERVISED TRAINING FOR VISUAL OBJECT TRACKING

#### 4.1 Proposed Methods

The main aim of this thesis is to investigate the effects of incomplete supervision strategy using semi-supervised training of CFB trackers that rely on deep networks for feature extraction.

Incomplete supervision is applied using two different methods. Firstly, labeled images are provided to the network with a predefined probability  $p$  during training. In other words, during a batch training of an epoch, full supervision is performed with  $p$  probability using ground truth labels, otherwise unlabeled samples are provided for training. As an example, during training, assume that an epoch consist of 100 iterations and  $p$  is chosen to be 0.75. The 75 of the forward and backward propagation is performed using labeled samples and supervised learning loss while 25 of them are done using unlabeled samples and unsupervised learning cost function. Aforementioned methodology is called *weakly supervised* training on the rest of this thesis document.

The second method that is applied for incomplete supervision is called *supervised-weakly-unsupervised* (SWU) training. Basically, the training is started with above-mentioned probability value set to 1.0. After that, for each epoch probability value is gradually decreased, and at final epoch it becomes 0.0. This method is divided into two parts, denoted as SWU\_C and SWU\_P trainings. In SWU\_C, the probability values is decreased continuously. It starts with 1.0 and decreased until 0.0 with equal intervals throughout 50 epochs. In SWU\_P, however, the probability value is decreased in a discretized manner. For example, assume that the total number of

epoch is 50. The value  $p$  is set to 1.0 for first 10 epoch. For the second 10 epoch,  $p$  is set to 0.75. The third 10 epoch uses the value as 0.5. The fourth one uses 0.25, and the final 10 epoch utilizes 0.0 probability value which means the training is performed completely unsupervised for the last 10 epochs. For better understanding, the probability value variation for SWU\_C and SWU\_P methods with respect to epoch number is illustrated in Figure 4.1.

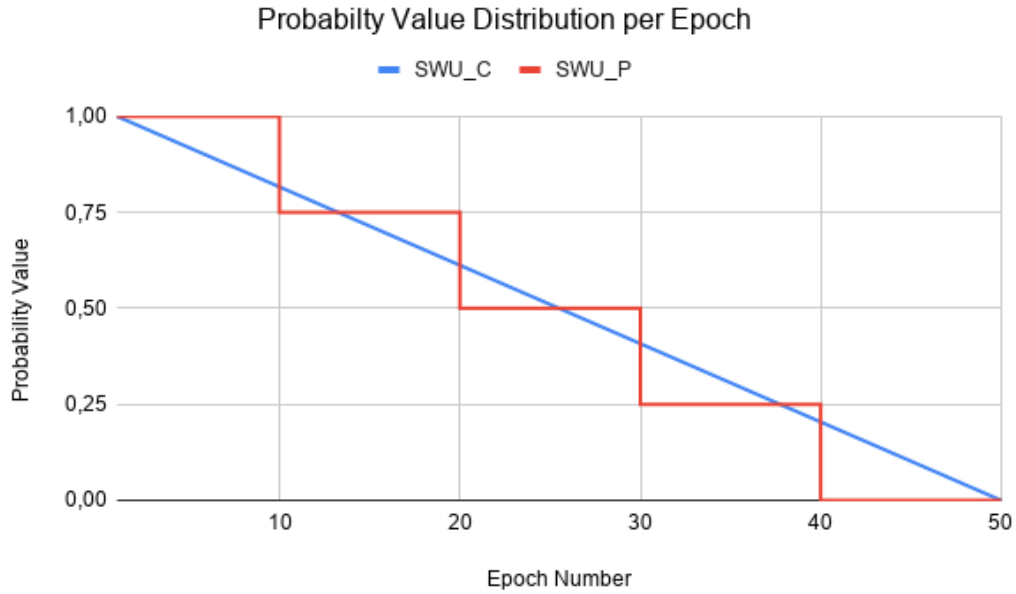


Figure 4.1: The probability value distribution for SWU\_C and SWU\_P semi supervised training methods.

DCFNet [8] has supervised trained deep feature extractor and is a correlation filter based tracker, hence, its unsupervised version UDT [7] are assumed to be perfectly appropriate to the main aim of this thesis. The supervised part of DCFNet and unsupervised part of UDT is merged for *weakly* and *SWU* model trainings. The overall flow of the training steps are presented in Figure 4.2.

In addition to the aforementioned incomplete supervision, the joint supervised unsupervised (JSU) training is also conducted. For each iteration, both labeled and unlabeled samples are used for forward tracking, instead of selecting one of them during training. Labeled samples are used on supervised model to generate error value, in the mean time, the unlabeled samples are also utilized on unsupervised model to com-

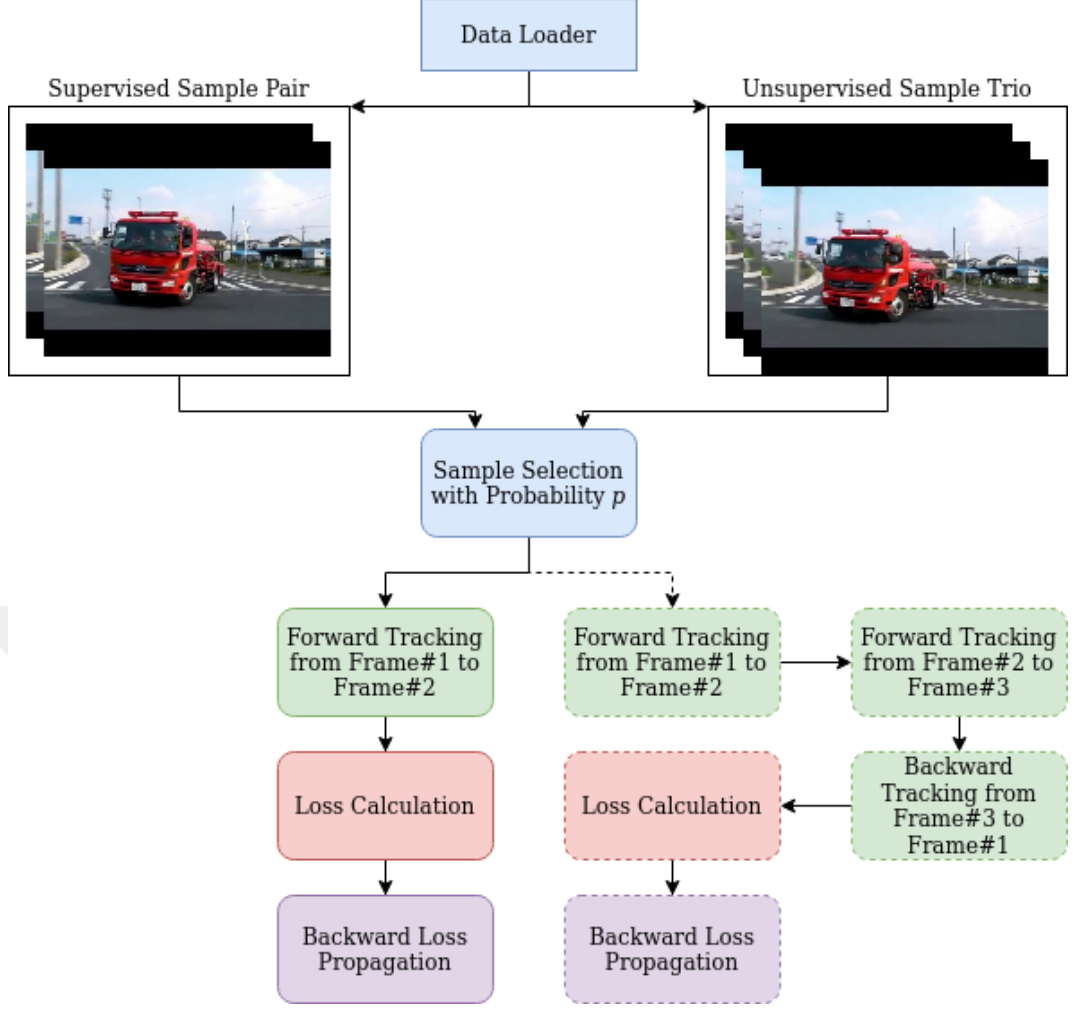


Figure 4.2: The overall flow of semi-supervised trainings.

pute unsupervised loss. The computed loss values are added, and back propagation is performed accordingly. The computation of final loss function is as follows:

$$Joint\_loss = Loss_{su} * \alpha + Loss_{uns} * (1 - \alpha) \quad (41)$$

where  $\alpha$  is the control parameter,  $Loss_{su}$  is the error value generated using labeled sample pairs on supervised model, and  $Loss_{uns}$  is the error value obtained by using unlabeled sample trios on unsupervised model. Similar to the weakly supervised training, JSU training is also composed of two methods. Firstly, the value of the control parameter  $\alpha$  is set to a fixed number throughout the training, which is denoted as *JSU*. In the second method, the value of  $\alpha$  is set to 1.0 at the start of the train-

ing. After that, it is decreased gradually. This method is also divided into two parts, denoted as *JSU\_SWU\_C* and *JSU\_SWU\_P* trainings. On *JSU\_SWU\_C*,  $\alpha$  value is decreased continuously after each epoch analogous to the *SWU\_C* trainings. Similar to the *SWU\_P* training, the value of  $\alpha$  is partially decreased on *JSU\_SWU\_P*.

Intuitively, the incomplete supervision described before is very similar to the JSU method. Therefore, it is important to see the effect on the tracking performance. The flow of the training of JSU is shown in Figure 4.3.

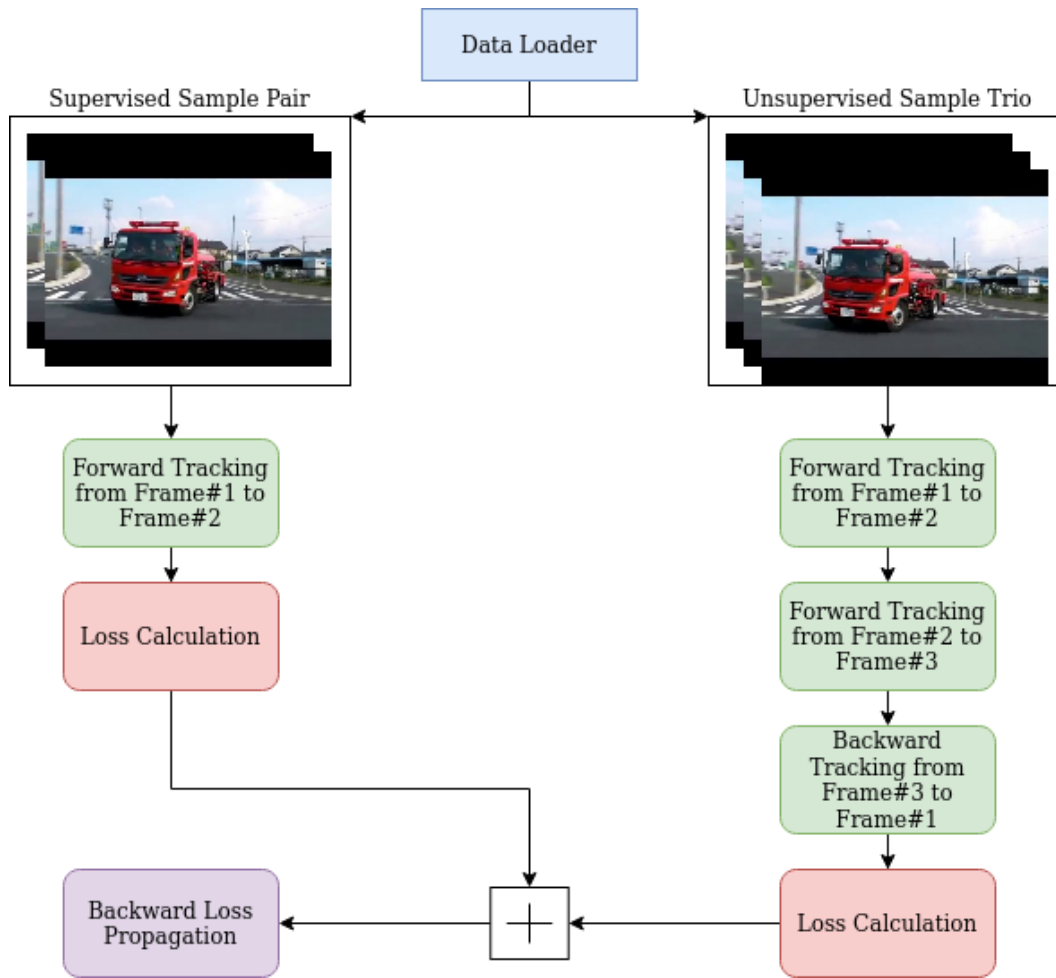


Figure 4.3: The training workflow of joint supervised unsupervised trainings.



## 4.2 Network Configuration and Training Settings

The overall architecture used in the model trainings is completely same with the one described in Section 3.2, and DCF formulation is implemented using (211). The deep network responsible for feature extraction consists of two convolutional layers with properties stated in Table 3.1. As described in section 4.1, trainings include both supervised and unsupervised samples. The objective function stated in (34) is used when supervised samples appear during training as a loss function. Similarly, the objective function described in (312) is utilized for unsupervised samples. SGD is employed for optimization purpose. Momentum and weight decay value are set to 0.9 and  $5 \times 10^{-5}$ , respectively. Learning rate is started from 0.01, and decreased exponentially until  $10^{-5}$ . Sample pairs are randomly selected from video frames having at most 10 frame difference. In other words, sample pairs can be successive frames or 10 frames apart. The frame difference range between the frames in a pair is chosen randomly between 1 and 10 during training as well.

During online tracking and testing, the model update is carried out using (313), and  $\alpha_t$  parameter is set to 0.01. Scale estimation is done according to the multi-scale search methodology described in section 2.1.3. Pyramid patches on three scales are utilized by setting  $\alpha$  parameter to 1.0275.

## 4.3 Datasets

Training and testing of models are performed using four datasets, namely OTB50[72], OTB100[12], ILSVRC2015[13], TrackingNet[5]. ILSVRC2015 and TrackingNet are used for training the deep feature extractor network part of the tracker, separately. OTB50 and OTB100 are used for evaluation of the trained models.

### 4.3.1 OTB50 and OTB100

OTB50 [72], namely object tracking benchmark or OTB2013, is one of the first datasets that is specialized for single object tracker evaluation. It consists of 50 fully annotated video segments. OTB50 includes 11 attributes which are occlusion, illumi-

nation variation, motion blur, scale variation, deformation, fast motion, out-of-plane rotation, in-plane rotation, background clutter, out-of-view, and low resolution. Any sequence can be annotated with multiple attributes, and some attribute may occur more frequently than others. OTB50 test set also includes attribute based subsets to be used for attribute specific evaluations. For example, occlusion (OCC) subset includes 29 sequences that can be utilized to investigate the robustness of trackers to handle occlusion. In Figure 4.4, the distribution of the sequences with occlusion and attribute distribution of whole test set are illustrated.

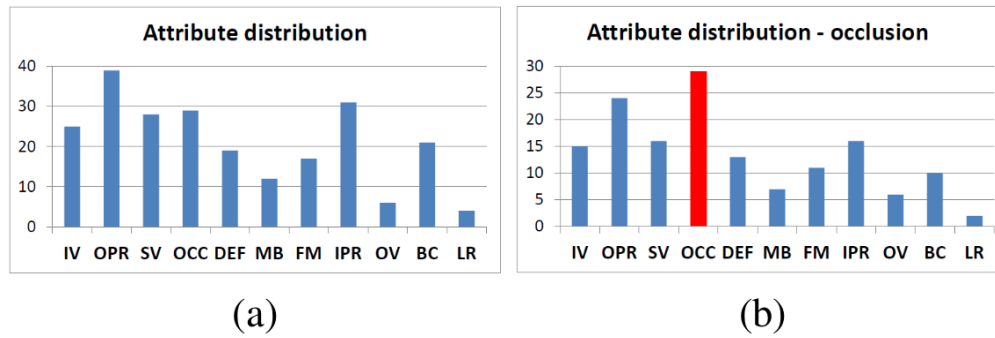


Figure 4.4: OTB50: (a) Distribution of attributes on the whole dataset, and (b) the distribution of occlusion (OCC) attribute [72].

OTB100 (OTB2015[12]) is a complementary version of OTB50. In addition to the 50 targets on OTB50, another 50 difficult and representative objects are added to the test set. However, the total number of videos is less than 100 since a few sequences have multiple targets in them.

The evaluation methodology is described in Section 4.4.

### 4.3.2 ILSVRC 2015

As part of the object detection from video challenge, 2015 edition of ImageNet Large Scale Visual Recognition Challenge (ILSVRC2015 [13]) is introduced. Dataset includes 30 various classes of vehicles and animals for participants to classify and locate them. Almost 4500 videos are available for training and validation sets together, containing a total of more than one million annotated frames. When compared with

VOT[87] and OTB[72] datasets in terms of the number of labelled sequences, which are together less than 500 videos, this number is remarkably attractive. The tracking community has shown great interest to this data set, since it was published because of not only for its vast size, but also variety of objects and scenes inside the videos. Therefore, it is logical to use it to train a deep network model for tracking without over-fitting to the domain of videos used in these benchmarks.

### **4.3.3 TrackingNet**

TrackingNet[5] dataset is the first large scale dataset and benchmark for object tracking in the wild. It includes 30,643 videos with an average duration of 16.6s. 14431266 frames are annotated with a single upright bounding box. 30132 videos with 14205677 frames are selected as training sequence, whereas 511 videos with 225589 frames are chosen as test sequence. Both training and testing videos include 27 different object classes. Figure 4.5 compares the TrackingNet dataset with other datasets in terms of the number of annotations, videos and length of the videos. Training set of TrackingNet is split into 12 training subsets where each of them contains 2511 videos and preserves the 27 object class distribution. TrackingNet includes 15 attributes that characterize videos, which are illumination variation, aspect ratio change, scale variation, out-of-view, fast motion, low resolution, background clutter, camera motion, motion blur, in-plane rotation, similar object, deformation, partial occlusion and full occlusion. The comparison of TrackingNet with other datasets is illustrated in Figure 4.5.

TrackingNet dataset owners claim that they have better control over the number of frames per video in the dataset, with a more contained variation with respect to the other datasets. Furthermore, the bounding box resolution distribution is more diverse in it. They also claim that TrackingNet has more natural distribution in terms of motion speed, aspect ratio and object types.

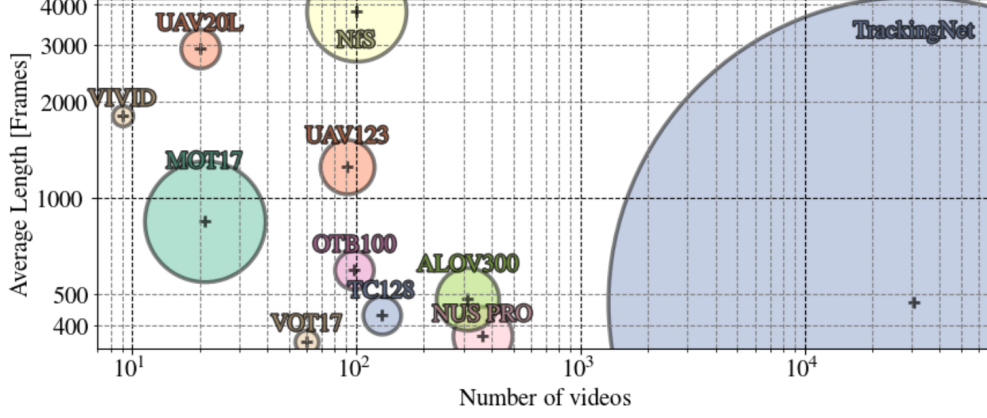


Figure 4.5: Comparison of datasets in terms of the length and number of videos [5].

#### 4.4 Evaluation Metrics for Tracking Experiments

Evaluation of our proposed semi supervised and joint supervised techniques is performed using the methods described in OTB [72]. Mainly, there are three metrics used for tracker performance evaluation, which are precision plot AUC (Area Under Curve) score, success plot AUC score and success rate.

One way to measure tracker precision is the center location error, namely, the average Euclidean distance between the manually labeled ground truths and the center locations of the tracked targets. Performing this calculation overall frames of a sequence summarizes the overall performance for that sequence. This computation is also known as root-mean-square-error (RMSE):

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^N \|F_k^G - F_k^T\|} \quad (42)$$

where  $k$  is the frame number,  $F_k^T$  is the predicted center location by the tracker and  $F_k^G$  is the ground truth center location. Nevertheless, the average error value may not measure the tracking performance correctly when tracker loses the target. Authors of OTB suggest that using 20 pixels as threshold brings representative precision score for each tracker. Subsequently, precision plot shows the percentage of frames whose estimated location is within the given threshold distance of the ground truth when the

threshold value is swept between two certain values. The area under curve (AUC) value of this plot is also used as performance indicator for tracking algorithms, and calculated as:

$$\text{Precision Plot AUC score} = \frac{1}{50} \sum_{l=0}^{50} \frac{1}{N} \sum_{k=0}^N y(k, l) \quad (43)$$

$$y(k, l) = \begin{cases} 1 & (\|F_k^G - F_k^T\| < l) \\ 0 & \text{otherwise.} \end{cases} \quad (44)$$

where  $t$  is the pixel error threshold.

Another evaluation metric is the bounding box overlap. IOU value in terms of number of pixels is calculated between the tracker output and the ground truth boxes, and calculated as follows:

$$\phi_k = \frac{|BB_k \cap GB_k|}{|BB_k \cup GB_k|} \quad (45)$$

where  $BB_k$  is bounding box output of the tracker and  $GB_k$  is the bounding box of ground truth at the frame  $k$ .

The performance is measured by calculating the number of successful frames whose IOU value is higher than the given threshold on a sequence of frames. Similar to the precision plot, success plot is also obtained by sweeping the threshold value. It illustrates the proportion of successful frames where the thresholds are varied from 0 to 1. The AUC score of the success plot is also used for ranking the tracking algorithms. Moreover, the success value at a specific threshold 0.5 is used for robustness evaluation as well, and denoted as success rate. The AUC score on success plot is computed as follows:

$$Success\ Plot\ AUC\ score = \int_{i=0}^1 \frac{1}{N} \sum_{k=0}^N t(k, i) \quad (46)$$

$$t(k, i) = \begin{cases} 1 & (\phi_k > i) \\ 0 & otherwise. \end{cases} \quad (47)$$

The success and precision plots can be generated using three different evaluation methods. The more conventional way is one-pass evaluation (OPE), which is performed by initializing the tracker on the first frame with ground truth position and running it throughout a test sequence. A tracker may be sensitive to the initialization, therefore, two other evaluation methods are introduced by OTB [72], which are spatial robustness evaluation (SRE) and temporal robustness evaluation (TRE). Temporal robustness evaluation is applied by initializing the tracker from one initial frame with ground truth of target and running to the end of sequence. The performance of tracker is measured on each segment, and the whole statistics are merged. Spatial robustness evaluation is performed by sampling the initial bounding box in the first frame by scaling or shifting the ground truth. The tracker is evaluated multiple times for different shift and scale values.

The test results for OTB50 and OTB100 are compared using success plot, precision plot and success rate with one-pass evaluation.

#### 4.5 Test Groups

The proposed methodologies are tested using 12 test groups. 6 of these groups are weakly supervised, supervised-weakly-unsupervised (SWU) and joint supervised-unsupervised (JSU) tests using ILSVRC dataset as training set. The other 6 of the tests belong to weakly supervised, SWU and JSU models utilizing TrackingNet as training dataset. The training and validation parts of the ILSVRC dataset is employed as train set during trainings, whereas the only 1 of 11 training parts of TrackingNet is utilized in order to be comparable with ILSVRC dataset. The total number of epoch is set 50 for each training.

#### 4.5.1 Weakly Supervised Test Groups

Weakly supervision is performed similar to the incomplete supervision. During an unsupervised training, supervised samples are added to the batch iteration according to a probability value  $p$  between 0 and 1. For example, if  $p$  is set to 0.1, 10% of the training batches are chosen from the supervised samples, and loss is calculated accordingly. The value  $p$  is predefined before the training starts. The UDT tracker is trained using ILSVRC and TrackingNet datasets separately in order to prove the concept for different datasets. Different values of  $p$  are tested to see the effect of it. In Table 4.1, properties of test group Test1 are shown with model names, training dataset, number of samples per epoch and predefined probability values.

Table 4.1: Properties of models used in Test1 are shown.

| Model Name | # of Pairs per Epoch | Dataset | Probability Value |
|------------|----------------------|---------|-------------------|
| DCFNet     | 1291104              | ILSVRC  | 1.0               |
| UDT        | 1291104              | ILSVRC  | 0.0               |
| UDT_W_0.75 | 1291104              | ILSVRC  | 0.75              |
| UDT_W_0.5  | 1291104              | ILSVRC  | 0.5               |
| UDT_W_0.3  | 1291104              | ILSVRC  | 0.3               |
| UDT_W_0.1  | 1291104              | ILSVRC  | 0.1               |

Properties of Test2 are stated in Table 4.2 with similar properties. The only difference between these groups is the number of samples per epoch. The letter 'S' at the end of the model names for Test2 indicates the word 'small'.

Table 4.2: Properties of models in Test2 are shown.

| <b>Model Name</b> | <b># of Pairs per Epoch</b> | <b>Dataset</b> | <b>Probability Value</b> |
|-------------------|-----------------------------|----------------|--------------------------|
| UDT_S             | 537536                      | ILSVRC         | 0.0                      |
| DCFNet_S          | 537536                      | ILSVRC         | 1.0                      |
| UDT_W_S_0.75      | 537536                      | ILSVRC         | 0.75                     |
| UDT_W_S_0.5       | 537536                      | ILSVRC         | 0.5                      |
| UDT_W_S_0.3       | 537536                      | ILSVRC         | 0.3                      |
| UDT_W_S_0.1       | 537536                      | ILSVRC         | 0.1                      |

With similar properties but having different training dataset, properties of Test3 and Test4 are shown in Table 4.3 and Table 4.4, respectively.

Table 4.3: Properties of models in Test3 are shown.

| <b>Model Name</b> | <b># of Pairs per Epoch</b> | <b>Dataset</b> | <b>Probability Value</b> |
|-------------------|-----------------------------|----------------|--------------------------|
| DCFNet_TN         | 1305696                     | TrackingNet    | 1.0                      |
| UDT_TN            | 1305696                     | TrackingNet    | 0.0                      |
| UDT_TN_W_0.75     | 1305696                     | TrackingNet    | 0.75                     |
| UDT_TN_W_0.5      | 1305696                     | TrackingNet    | 0.5                      |
| UDT_TN_W_0.3      | 1305696                     | TrackingNet    | 0.3                      |
| UDT_TN_W_0.1      | 1305696                     | TrackingNet    | 0.1                      |

Table 4.4: Properties of models in Test4 are shown.

| <b>Model Name</b> | <b># of Pairs per Epoch</b> | <b>Dataset</b> | <b>Probability Value</b> |
|-------------------|-----------------------------|----------------|--------------------------|
| DCFNet_TN_S       | 539840                      | TrackingNet    | 1.0                      |
| UDT_TN_S          | 539840                      | TrackingNet    | 0.0                      |
| UDT_TN_W_S_0.75   | 539840                      | TrackingNet    | 0.75                     |
| UDT_TN_W_S_0.5    | 539840                      | TrackingNet    | 0.5                      |
| UDT_TN_W_S_0.3    | 539840                      | TrackingNet    | 0.3                      |
| UDT_TN_W_S_0.1    | 539840                      | TrackingNet    | 0.1                      |



#### 4.5.2 SWU: Supervised-Weakly-Unsupervised Test Groups

For SWU based trainings, the probability value  $p$  described in section 4.5.1 is decreased from 1.0 to 0.0 during training. SWU models are tested in two groups. First one is called as SWU\_C where the attenuation of  $p$  is performed continuously at each epoch step by step. The second one, namely SWU\_P, is also performed by decreasing the  $p$  value partially after each 10 epoch. The Figure 4.1 illustrates the probability attenuation for SWU\_C and SWU\_P tests at each epoch. Test5 is composed of SWU\_C and SWU\_P models on ILSVRC dataset, and shown in Table 4.5 with their properties.

Table 4.5: Properties of models used in Test5 are shown.

| Model Name | # of Pairs per Epoch | Dataset | Probability Value |
|------------|----------------------|---------|-------------------|
| DCFNet     | 1291104              | ILSVRC  | 1.0               |
| UDT        | 1291104              | ILSVRC  | 0.0               |
| UDT_SWU_C  | 1291104              | ILSVRC  | Continuous        |
| UDT_SWU_P  | 1291104              | ILSVRC  | Partial           |

Test6 is generated with smaller number of samples of Test5, and its properties are described in Table 4.6.

Table 4.6: Properties of models used in Test6 are shown.

| Model Name  | # of Pairs per Epoch | Dataset | Probability Value |
|-------------|----------------------|---------|-------------------|
| DCFNet_S    | 537536               | ILSVRC  | 1.0               |
| UDT_S       | 537536               | ILSVRC  | 0.0               |
| UDT_SWU_C_S | 537536               | ILSVRC  | Continuous        |
| UDT_SWU_P_S | 537536               | ILSVRC  | Partial           |

TrackingNet trained version of Test5 and Test6 are denoted as Test7 and Test8, and their properties are shown in Table 4.7 and Table 4.8, respectively. In addition to SWU models, DCFNet and UDT trainings with similar properties are also added to the tests for better comparison.

Table 4.7: Properties of models used in Test7 are shown.

| Model Name   | # of Pairs per Epoch | Dataset     | Probability Value |
|--------------|----------------------|-------------|-------------------|
| DCFNet_TN    | 1305696              | TrackingNet | 1.0               |
| UDT_TN       | 1305696              | TrackingNet | 0.0               |
| UDT_TN_SWU_C | 1305696              | TrackingNet | Continuous        |
| UDT_TN_SWU_P | 1305696              | TrackingNet | Partial           |

Table 4.8: Properties of models used in Test8 are shown.

| Model Name     | # of Pairs per Epoch | Dataset     | Probability Value |
|----------------|----------------------|-------------|-------------------|
| DCFNet_TN_S    | 539840               | TrackingNet | 1.0               |
| UDT_TN_S       | 539840               | TrackingNet | 0.0               |
| UDT_TN_SWU_C_S | 539840               | TrackingNet | Continuous        |
| UDT_TN_SWU_P_S | 539840               | TrackingNet | Partial           |

#### 4.5.3 JSU: Joint Supervised-Unsupervised Test Groups

JSU based models have leveraged both supervised and unsupervised loss functions at each iteration of the training process. Two different methods are trialed about the usage of control parameter defined in (41). First one is denoted as JSU tests, which uses fixed value of control parameter throughout the trainings. The models, having different but fixed control parameter and used ILSVRC as training set along side with original DCFNet and UDT trainings, compose the Test9. Model names, used dataset, number of samples per epoch and predefined control parameter  $\alpha$  values of Test9 is shown in Table 4.9.

Table 4.9: Properties of models used in Test9 are shown.

| <b>Model Name</b> | <b># of Pairs per Epoch</b> | <b>Dataset</b> | <b>Alpha Value</b> |
|-------------------|-----------------------------|----------------|--------------------|
| DCFNet_S          | 537536                      | ILSVRC         | 1.0                |
| UDT_S             | 537536                      | ILSVRC         | 0.0                |
| UDT_JSU_S_0.1     | 537536                      | ILSVRC         | 0.1                |
| UDT_JSU_S_0.3     | 537536                      | ILSVRC         | 0.3                |
| UDT_JSU_S_0.5     | 537536                      | ILSVRC         | 0.5                |
| UDT_JSU_S_0.75    | 537536                      | ILSVRC         | 0.75               |

Test10 is the TrackingNet version of Test9, and shown in Table 4.10 with their properties.

Table 4.10: Properties of models used in Test10 are shown.

| <b>Model Name</b> | <b># of Pairs per Epoch</b> | <b>Dataset</b> | <b>Alpha Value</b> |
|-------------------|-----------------------------|----------------|--------------------|
| DCFNet_TN_S       | 539840                      | TrackingNet    | 1.0                |
| UDT_TN_S          | 539840                      | TrackingNet    | 0.0                |
| UDT_TN_JSU_S_0.1  | 539840                      | TrackingNet    | 0.1                |
| UDT_TN_JSU_S_0.3  | 539840                      | TrackingNet    | 0.3                |
| UDT_TN_JSU_S_0.5  | 539840                      | TrackingNet    | 0.5                |
| UDT_TN_JSU_S_0.75 | 539840                      | TrackingNet    | 0.75               |

The second method, called as JSU\_SWU, is divided into two parts. One of them is called JSU\_SWU\_C tests. On these models, the control parameter is set to 1.0 at first epoch, and decreased gradually until 0.0 at the last epoch. Second method, however, decreases the value of control parameter partially, and denoted as JSU\_SWU\_P. In addition to the original DCFNet and UDT trainings, the ILSVRC trained JSU\_SWU\_C and JSU\_SWU\_P generate Test11 group. The TrackingNet version of Test11 is named as Test12. The training properties of Test11 and Test12 are shown in Table 4.11 and 4.12, respectively.

Table 4.11: Properties of models used in Test11 are shown.

| Model Name      | # of Pairs per Epoch | Dataset | Alpha Value |
|-----------------|----------------------|---------|-------------|
| DCFNet_S        | 537536               | ILSVRC  | 1.0         |
| UDT_S           | 537536               | ILSVRC  | 0.0         |
| UDT_JSU_SWU_C_S | 537536               | ILSVRC  | Continuos   |
| UDT_JSU_SWU_P_S | 537536               | ILSVRC  | Partial     |

Table 4.12: Properties of models used in Test12 are shown.

| Model Name         | # of Pairs per Epoch | Dataset     | Alpha Value |
|--------------------|----------------------|-------------|-------------|
| DCFNet_TN_S        | 539840               | TrackingNet | 1.0         |
| UDT_TN_S           | 539840               | TrackingNet | 0.0         |
| UDT_TN_JSU_SWU_C_S | 539840               | TrackingNet | Continuos   |
| UDT_TN_JSU_SWU_P_S | 539840               | TrackingNet | Partial     |

In Table 4.13, general properties of all aforementioned test groups are described.

Table 4.13: Properties of all test groups are shown.

| Test ID | Supervision Type | Tranining Dataset | # of Pairs per Epoch | $p$ Value Strategy | $\alpha$ Value Strategy | Loss Calculation |
|---------|------------------|-------------------|----------------------|--------------------|-------------------------|------------------|
| Test1   | Weakly           | ILSVRC            | 1291104              | Fixed              | NA                      | Single           |
| Test2   | Weakly           | ILSVRC            | 537536               | Fixed              | NA                      | Single           |
| Test3   | Weakly           | TrackingNet       | 1305696              | Fixed              | NA                      | Single           |
| Test4   | Weakly           | TrackingNet       | 539840               | Fixed              | NA                      | Single           |
| Test5   | SWU              | ILSVRC            | 1291104              | Decreasing         | NA                      | Single           |
| Test6   | SWU              | ILSVRC            | 537536               | Decreasing         | NA                      | Single           |
| Test7   | SWU              | TrackingNet       | 1305696              | Decreasing         | NA                      | Single           |
| Test8   | SWU              | TrackingNet       | 539840               | Decreasing         | NA                      | Single           |
| Test9   | JSU              | ILSVRC            | 537536               | NA                 | Fixed                   | Joint            |
| Test10  | JSU              | TrackingNet       | 539840               | NA                 | Fixed                   | Joint            |
| Test11  | JSU_SWU          | ILSVRC            | 537536               | NA                 | Decreasing              | Joint            |
| Test12  | JSU_SWU          | TrackingNet       | 539840               | NA                 | Decreasing              | Joint            |

## 4.6 Experimental Results and Analysis

Pytorch implementations of both DCFNet and UDT by their original authors are considered as baseline implementations of these algorithms. The experiments also performed using Pytorch framework by making required changes on the published code blocks. In this section, the experimental results of 12 test groups are shared in terms of success rate, success plots and precision plots of OTB50 and OTB100 benchmark sets. Fully supervised and unsupervised models, namely DCFNet and UDT, are independently trained three times, and the best performing models are taken into account for comparison.

### 4.6.1 Weakly Supervised Test Groups

#### 4.6.1.1 Experimental Results for Test1 and Test2

The OTB50 and OTB100 success and precision plots for Test1 of weakly supervised trainings on ILSVRC dataset are illustrated on Figure 4.6. The success rate comparison is shown in Figure 4.7. The best model of the trainings are chosen according to a model having largest success plot AUC value. The precision plots and success rate are drawn according to the result of that model.

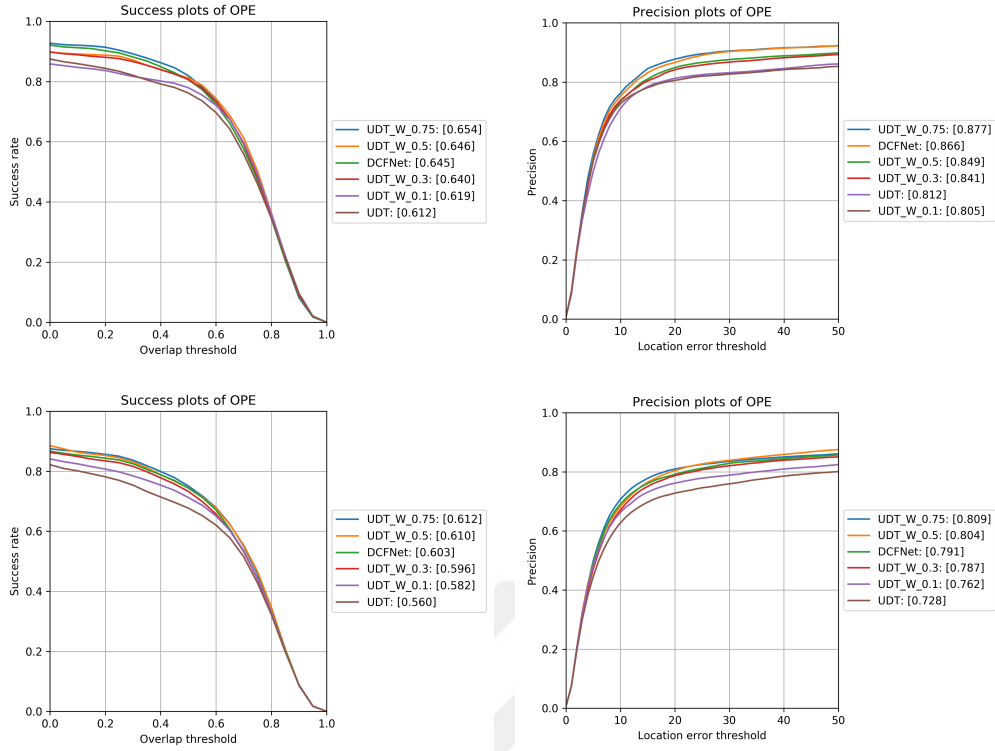


Figure 4.6: Comparison of model results in Test1 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicate the AUC value of that specific plot. Best viewed in color.

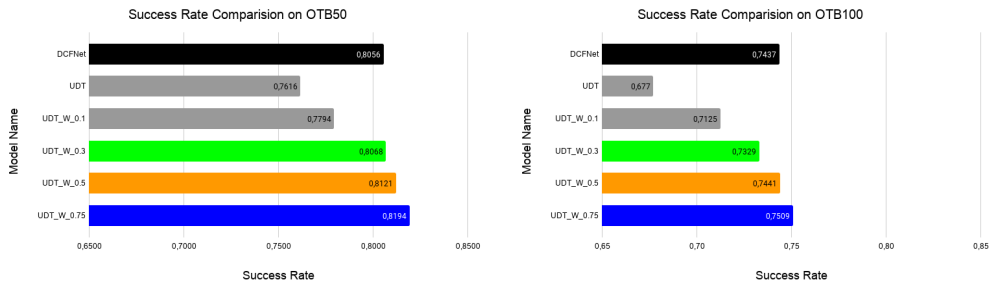


Figure 4.7: Comparison of model results in Test1 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

When Figure 4.6 and Figure 4.7 are examined, it can be observed that weakly supervised model with probability value  $p$  equals to 0.75, namely UDT\_W\_0.75, beats the fully supervised DCFNet model on both test sets. There exist 0.9% increase on success plot AUC value, 1.1% increase on precision plot AUC value, and 1.4% increase on success rate value for OTB50 test set.

Similarly, on OTB100, UDT\_W\_0.75 surpasses DCFNet with 0.9% increase on success plot AUC value and 1.8% increase on precision plot AUC value and 0.7% increase on success rate. Another deduction can be made according to the aforementioned plots is that UDT\_W\_0.5 model also has performance compatible with fully supervised one. In another words, using a dataset whose only half of its frames are labeled has a similar result with a version having all of its frames are labeled in the case of DCFNet tracking architecture. Naturally, the supervised and unsupervised learning scheme has also effect on this results. The results also indicates that using even 10% of labeled data during training has significant effect on the performance when compared with fully unsupervised model. UDT\_W\_0.1 is an example for this statement. When it is compared with UDT, it has 3.4% gain on precision plot AUC value, 2.2% gain on success plot AUC value and 3.5% gain on success rate.

The purpose of the Test2 is to observe the weakly supervised training on smaller samples per epoch. In Figure 4.8, success and precision plots on OTB50 and OTB100 test sets for Test2 are illustrated. Success rate comparison is shown in Figure 4.9.

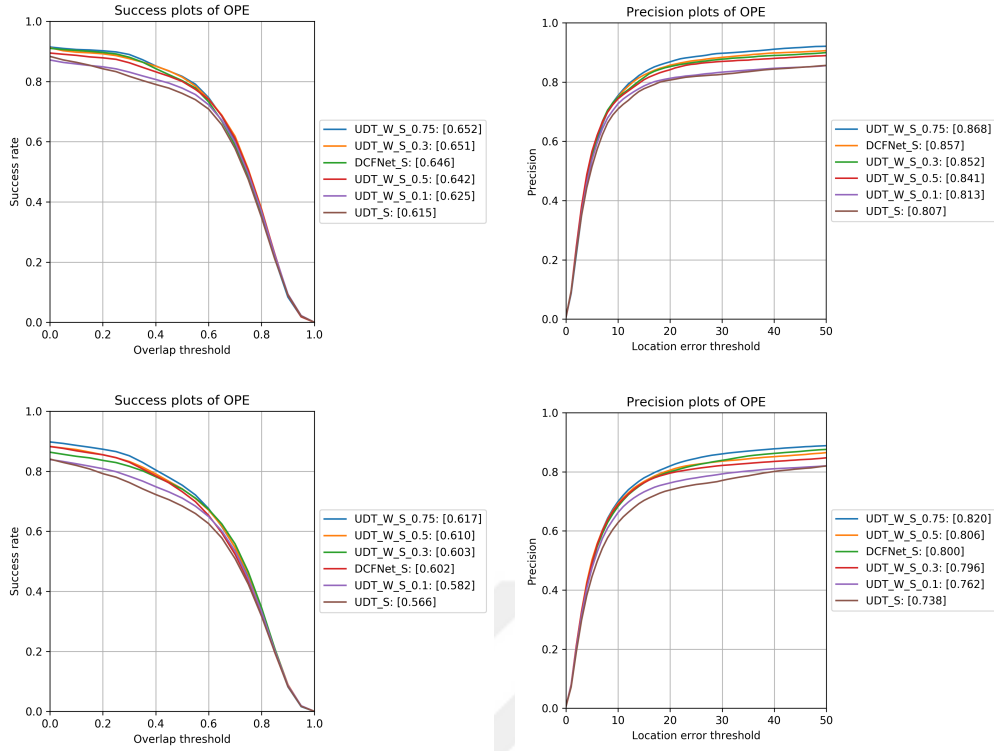


Figure 4.8: Comparison of model results in Test2 in terms of success and precision plots. Two graphs upper-side of the figure shows OTB50 test set results while the other two graphs below shows OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

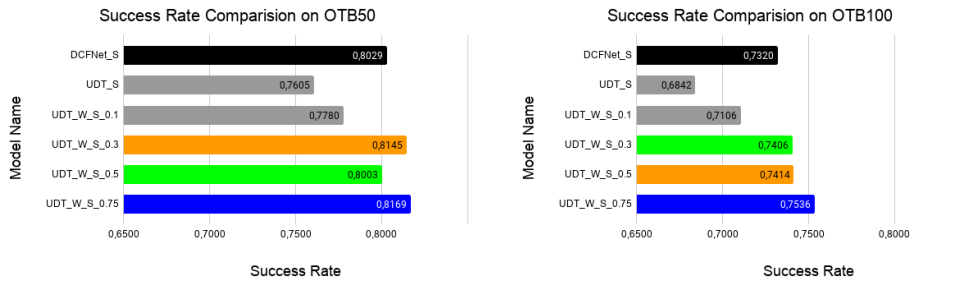


Figure 4.9: Comparison of model results in Test2 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.



The models in Test2 show similar performance gains with Test1. According to the results shown in Figure 4.8 and Figure 4.9, the UDT\_W\_S\_0.75 model again improves over fully supervised DCFNet\_S for every evaluation parameter on both test sets. Specifically, there occurs 1.5% increase on success plot AUC value, 2.0% increase on precision plot AUC value and 2.1% increase on success rate for OTB100 test. Similarly on OTB50 test set, there exists 0.6% increase on success plot AUC value, 1.1% increase on precision plot AUC value and 1.4% increase on success rate. Analogous to Test1, setting probability value to 0.5 has compatible results with fully supervised model, and using even small number of labeled instances boosts the performance of the tracker.

Test1 and Test2 show that mixing supervised and unsupervised learning schemes with some amount make the DCFNet tracker more accurate and robust than using only supervised learning when trained with ILSVRC dataset. The tracker is gained more generalization ability with incomplete supervision. The unsupervised learning model has important effect at aforementioned improvement, since it is more appropriate to the real world scenarios. Moreover, it prevents the tracker to memorize on to the training dataset. However, it is not enough to train a proper tracker by itself as the results of UDT and UDT\_S models indicate.

In order to observe the effect of using different size of samples, Test1 and Test2 are compared with each other, and presented in Figure 4.10 together. The notation 'S' on the model names indicates that small number of samples are used on that model. In terms of OTB50 results, models that have larger samples (UDT\_W\_##) than others (UDT\_W\_S\_##) are slightly in front at the performance race. However, this behaviour is vice versa for OTB100. From this results, one can deduct that using large number of samples does not always result in better accuracy when ILSVRC dataset is utilized.

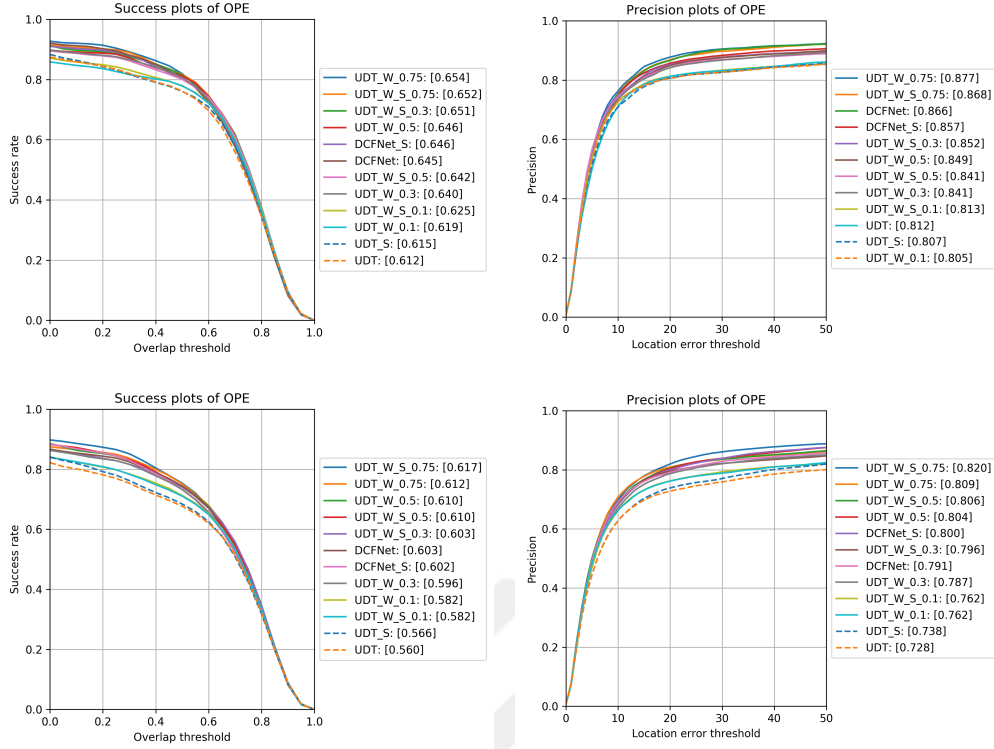


Figure 4.10: Comparison of model results in Test1 and Test2 in terms of success and precision plots. Two graphs upper-side of the figure shows OTB50 test set results while the other two graphs below shows OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

#### 4.6.1.2 Experimental Results for Test3 and Test4

The OTB50 and OTB100 test results for the test group Test3 of weakly supervised trainings on TrackingNet dataset are illustrated on Figure 4.11. The success rate comparison is shown in Figure 4.12. The best model of the trainings are selected according to a model having largest success plot AUC value. The precision plots and success rate are drawn according to the result of that model.

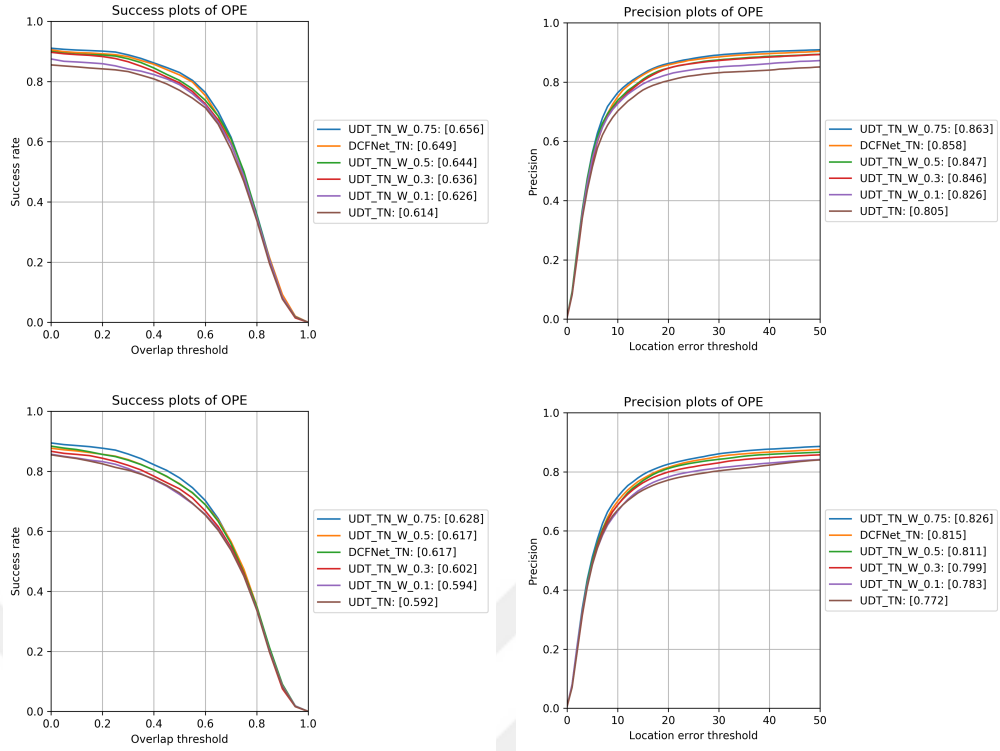


Figure 4.11: Comparison of model results in Test3 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

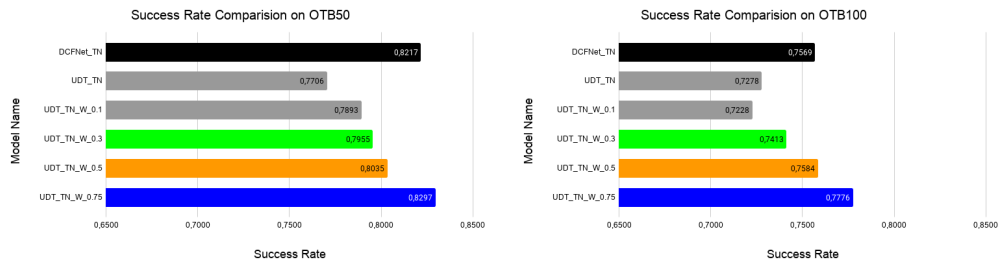


Figure 4.12: Comparison of model results in Test3 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

One of the first deductions that can be made from the performance results is that

DCFNet and UDT trainings on TrackingNet have better performance values than the ones trained using ILSVRC. This is an expected result, since standard trackers have shown an increase on performance between 1% and 2%, when trained using TrackingNet as stated in [5]. This is due to the more diverse and challenging videos of TrackingNet dataset than ILSVRC. Despite that increase, weakly supervised model with probability value  $p$  equals to 0.75, namely UDT\_TN\_W\_0.75, again beats the fully supervised DCFNet\_TN on both test set as it can be observed from Figure 4.11 and Figure 4.12. There is 0.7% increase on success plot AUC value, 0.5% increase on precision plot AUC value, 0.8% increase on success rate for OTB50 test set. Similarly, on OTB100 test set, there exists %1.1 increase on success plot AUC value, 1.1% increase on precision plot AUC value, 2.1% increase on success rate value.

Although the performance increase on OTB50 is not significant, OTB100 results show much better accuracy gains. The reason for this behaviour is that OTB100 has more diverse and compelling videos than OTB50. This makes OTB100 test set more representative of the real world. Moreover, the light weight architecture of the DCFNet can reach a maximum at each set due to its own design. While the TrackingNet increases the performance of DCFNet and UDT models, the model become closer to that maximum value. Therefore, the effect of incomplete supervision become less visible on OTB50.

Test4 is composed in the same way as Test3; however, it has smaller number of samples per epoch. The purpose of this test is again to examine the effect of using small number of samples during training. The success and precision plots of Test4 for OTB50 and OTB100 are presented in Figure 4.13. Success rate comparison is shown in Figure 4.14.

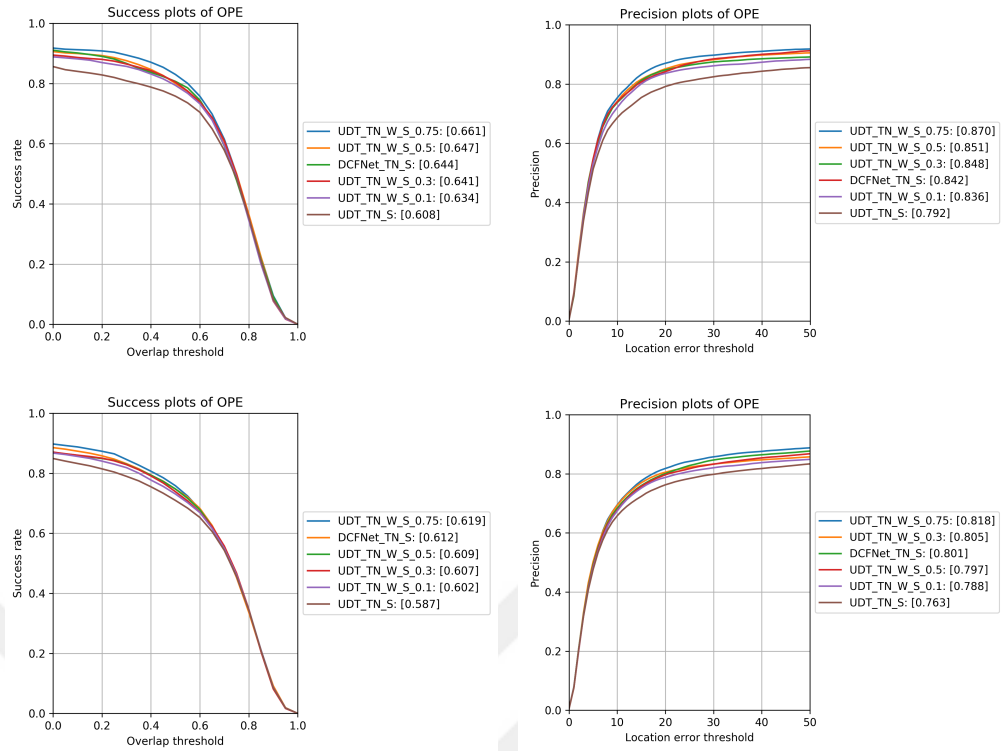


Figure 4.13: Comparison of model results in Test4 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

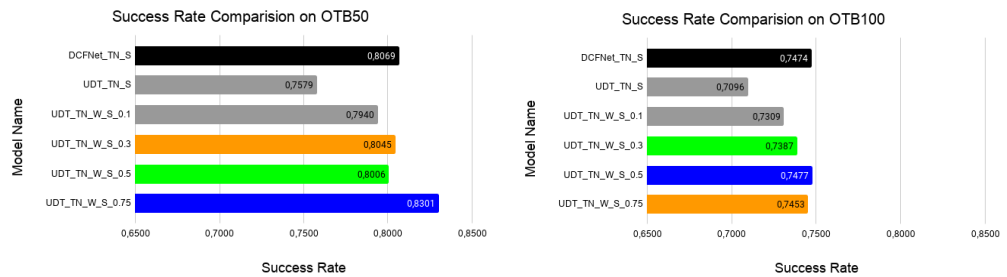


Figure 4.14: Comparison of model results in Test4 in terms of success rate. Two graph on the left of the figure show success rate values on OTB50 while the graph on the right show success rate on OTB100. Best viewed in color.

The results of Test4 shows that weakly supervised model with value  $p$  equals to 0.75, denoted as UDT\_TN\_W\_S\_0.75, has better performance than fully supervised trained DCFNet\_TN\_S. It is observed that there occurs 1.7% increase on success plot AUC value, 2.8% increase on precision plot AUC value and 2.4% increase on success rate value when OTB50 results are considered. When OTB100 performance is compared, it is clear that there is 0.7% increase on success plot AUC value and 1.7% increase on precision plot AUC value, however, there is 0.2% decrease on success rate. Similar to the weakly supervised test groups Test1 and Test2 on ILSVRC set, using a dataset whose half of its instances are labeled has a compatible performance with a version having all of its frames are labeled. Furthermore, by using only 10% labeled instances overall dataset has enhanced the performance significantly when compared with the fully unsupervised trained UDT\_TN\_S. When performance gains of Test3 and Test4 on the fully supervised models are compared, it is observed that the positive effect of incomplete supervision become more obvious when smaller number of samples are utilized during training in case of utilizing TrackingNet dataset.

In Figure 4.15, the performance results of Test3 and Test4 are plotted together in order to observe the effect of using different size of samples during training. In terms of OTB50 results, models that have smaller samples (UDT\_TN\_W\_S\_##) than others (UDT\_TN\_W\_##) are slightly in front at the performance race. However, this behaviour is vice versa for OTB100. This situation is exact opposite to the one occurred on ILSVRC based models, which shows the effect of using a dataset having different properties.

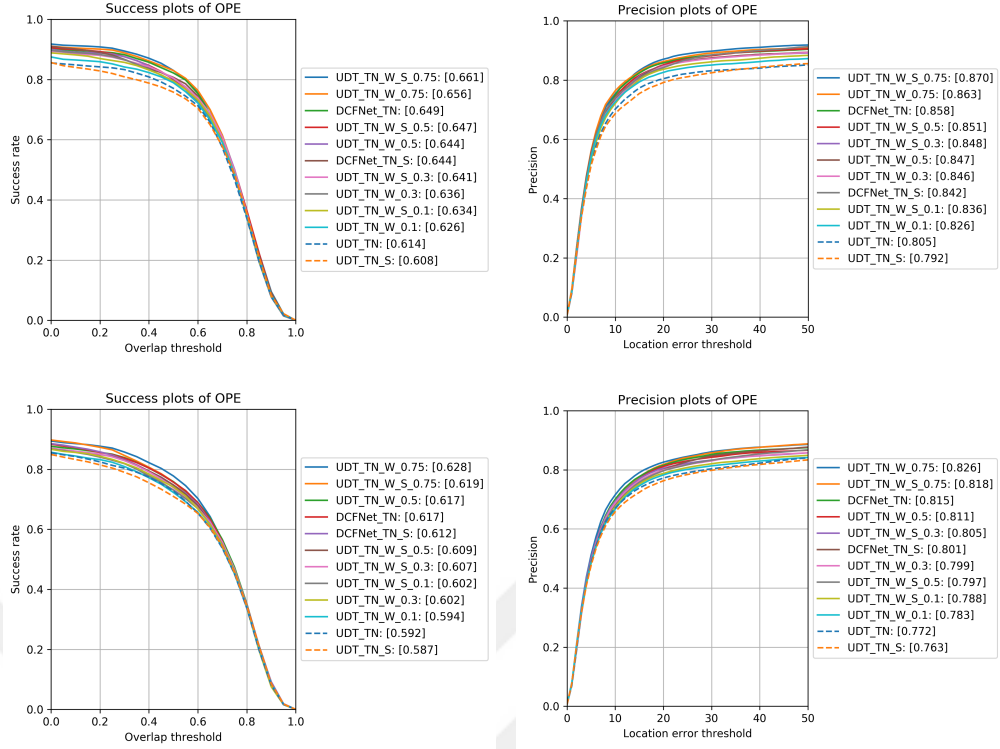


Figure 4.15: Comparison of model results of Test3 and Test4 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

## 4.6.2 SWU Test Groups

### 4.6.2.1 Experimental Results for Test5 and Test6

The OTB50 and OTB100 success and precision plots for the test group Test5 of SWU trainings on ILSVRC dataset are illustrated in Figure 4.16. The success rate comparison is shown in Figure 4.17. The best model of the trainings are chosen according to a model having largest success plot AUC value. The precision plots and success rate are drawn according to the result of that model.

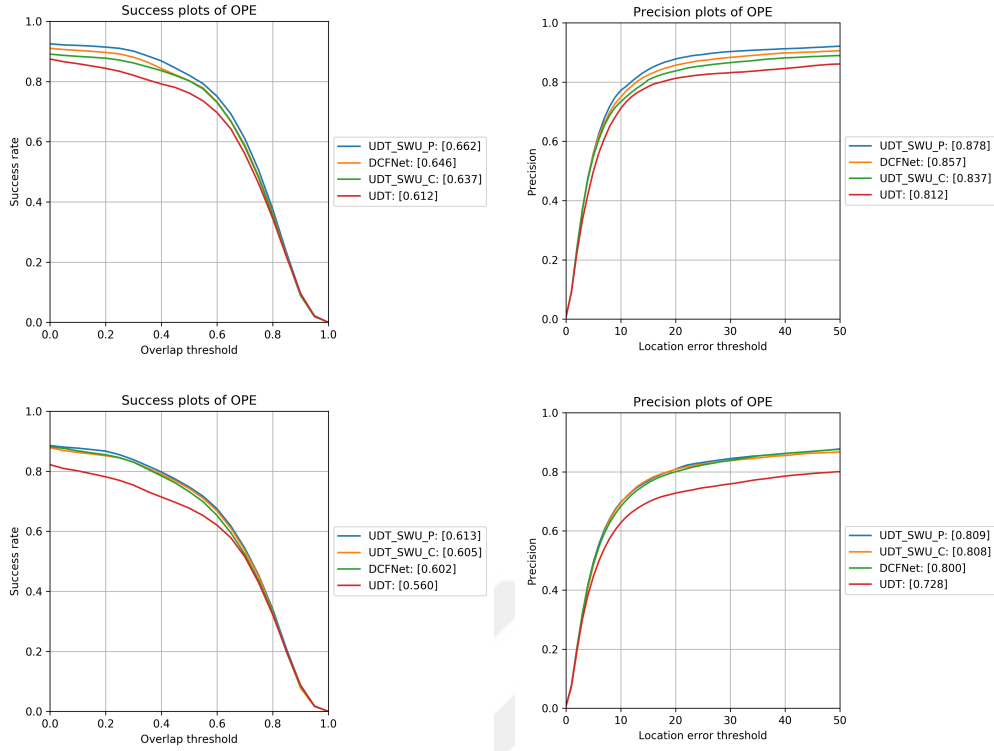


Figure 4.16: Comparison of model results in Test5 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

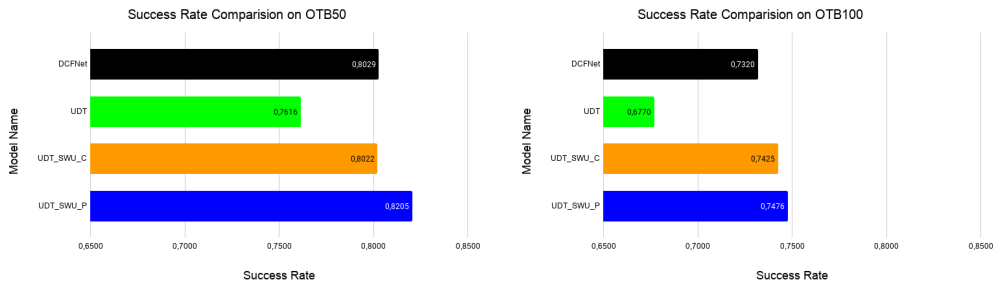


Figure 4.17: Comparison of model results on Test5 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.



From Figure 4.16 and 4.17, it is clear that partial SWU training, namely SWU\_P, has better performance than fully supervised version. When the UDT\_SWU\_P and DCFNet models are compared, it is observed that there is 1.6% increase on success plot AUC value, 2.1% increase on precision plot AUC value and 1.8% increase on success rate in terms of OTB50 test results. Similarly on OTB100 test set, there exist 1.1% increase on success plot AUC value, 0.9% increase on precision plot AUC value and 1.5% increase on success rate. When UDT\_SWU\_C model results are analyzed, the performance increase is not observed. This result indicates that decreasing probability value partially has better effect on training when the model is trained on ILSVRC dataset. In other words, initially supervising the network, and mixing unsupervised learning later on make the DCFNet model learn more robust features. However, unsupervised learning starts to affect negatively when the probability value decreased too much. An interesting result observed for SWU\_P model is that the best models occur when the  $p$  value is around 0.75. For example, the best model of UDT\_SWU\_P on OTB100 is the one generated after epoch 14.

Test6 is also generated to see the effect of using small number of samples for SWU\_C and SWU\_P trainings. Success and precision plots of Test6 are shown in Figure 4.18, and success rate comparison is illustrated in Figure 4.19 for both test sets.

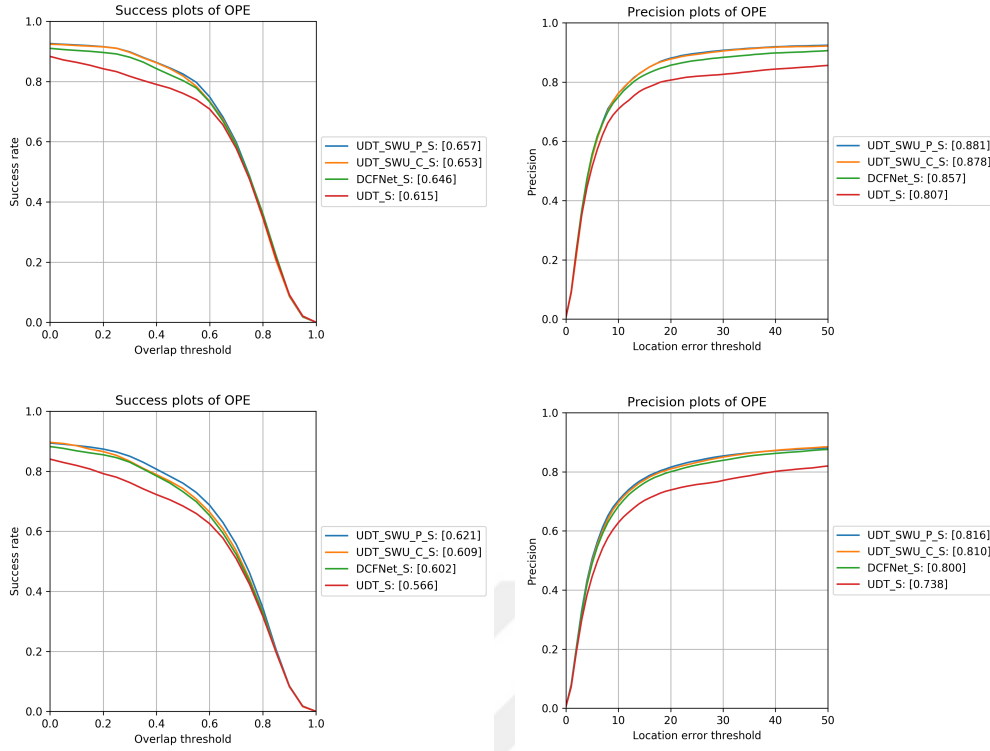


Figure 4.18: Comparison of model results in Test6 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below show OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

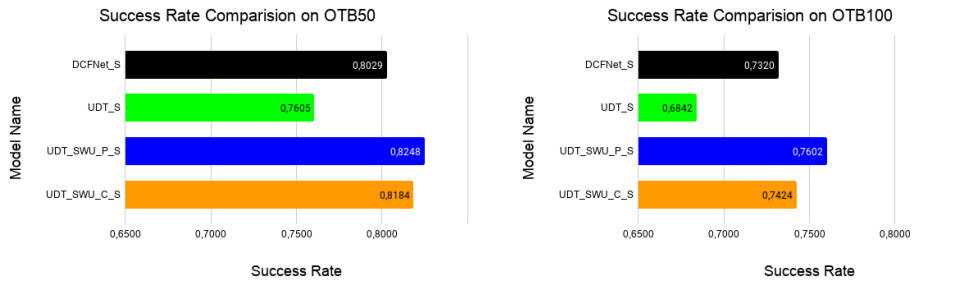


Figure 4.19: Comparison of model results in Test6 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

The results show that both UDT\_SWU\_C\_S and UDT\_SWU\_P\_S beat DCFNet at all evaluation parameters of test sets. Moreover, UDT\_SWU\_P\_S model surpasses UDT\_SWU\_C\_S in the same way. When UDT\_SWU\_P\_S and DCFNet is compared, it is clear that there is 1.1% increase on success plot AUC value, 2.4% increase on precision plot and 2.2% increase on success rate for OTB50 test. Identically for OTB100 test set, there occurs 1.9% increase on success plot AUC value, 1.6% increase on precision plot AUC value and 2.8% increase on success rate value. In contrast to the results of Test5, this time continuously decreasing  $p$  value after each epoch (UDT\_SWU\_C\_S) manages to outperform fully supervised model.

The results of Test5 and Test5 again show that when smaller number of samples are utilized during training, the effect of incomplete supervision become more clear. In addition to the results shown above, Test5 and Test6 are compared together and plotted in Figure 4.20. Similar to the weakly supervised models trained on ILSVRC dataset, the performance of SWU\_P model having different number of samples during training are similar to each other on OTB50 test set, whereas the models having small number of samples are much better in terms of OTB100 performance evaluation parameters.

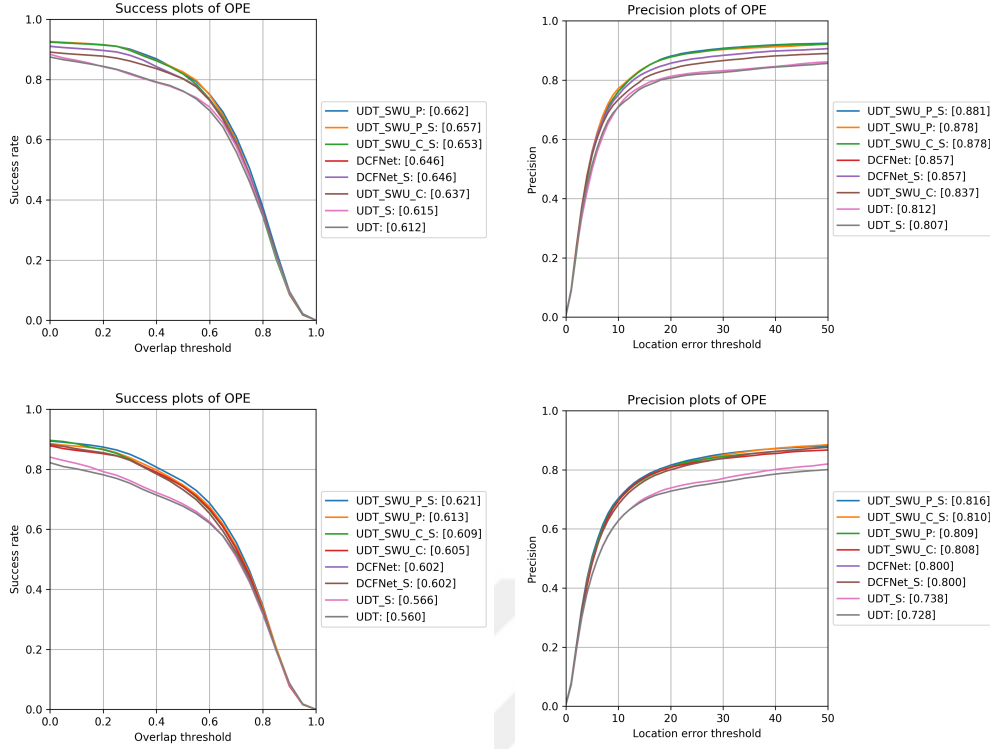


Figure 4.20: Comparison of model results of Test5 and Test6 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

#### 4.6.2.2 Experimental Results for Test7 and Test8

The OTB50 and OTB100 success and precision plots for Test7 of SWU trainings on TrackingNet dataset are illustrated on Figure 4.21. The success rate comparison is shown in Figure 4.22. The best model of the training results are selected according to a model having largest success plot AUC value. The precision plots and success rate are drawn according to the result of that model.

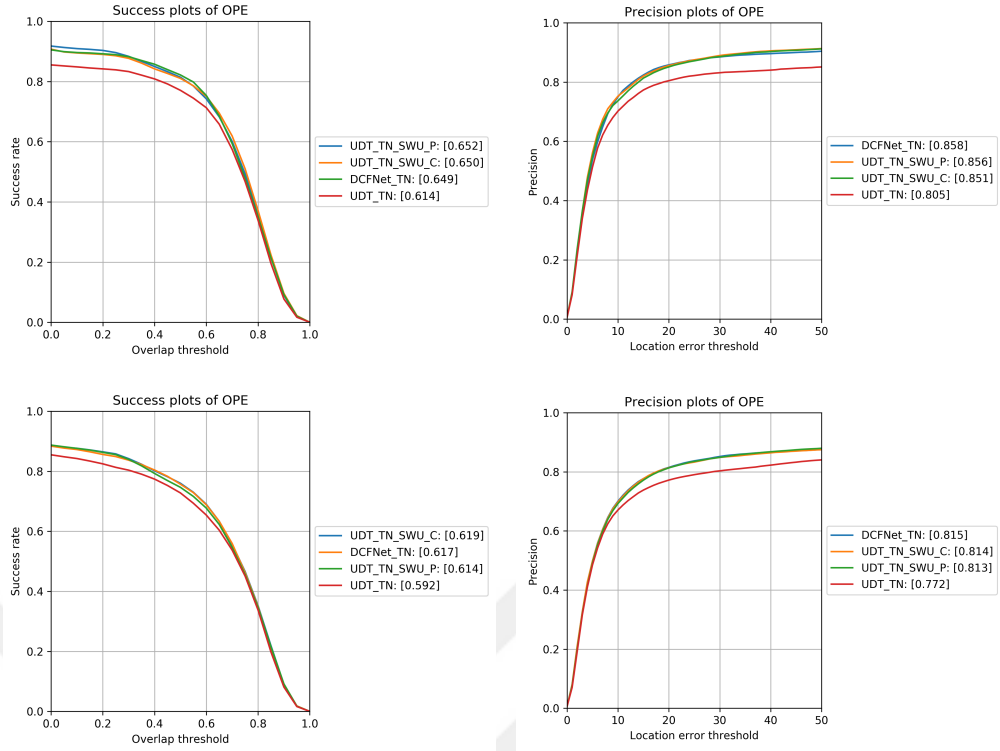


Figure 4.21: Comparison of model results in Test7 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below show OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

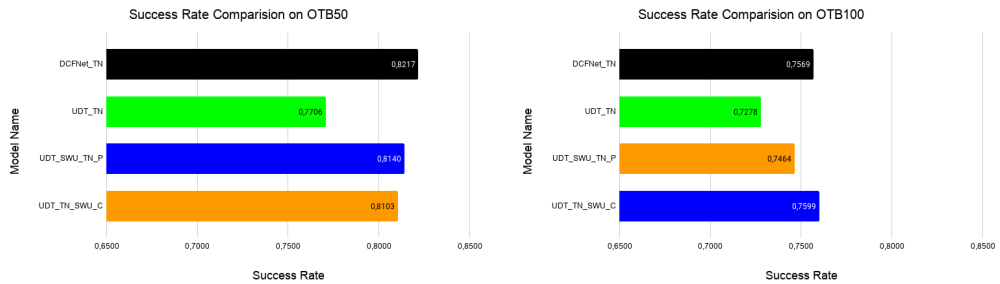


Figure 4.22: Comparison of model results in Test7 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

Performance results of Test7 show that SWU\_C and SWU\_P models on TrackingNet have similar result with fully supervised DCFNet\_TN. Although there are small improvements on success plot, it is hard to say that SWU models on TrackingNet perform better than DCFNet model.

Test8 is also generated to see the effect of using smaller number of samples than the one in Test7. Success and precision plots of Test8 are illustrated in Figure 4.23, and success rate comparison is shown in Figure 4.24.

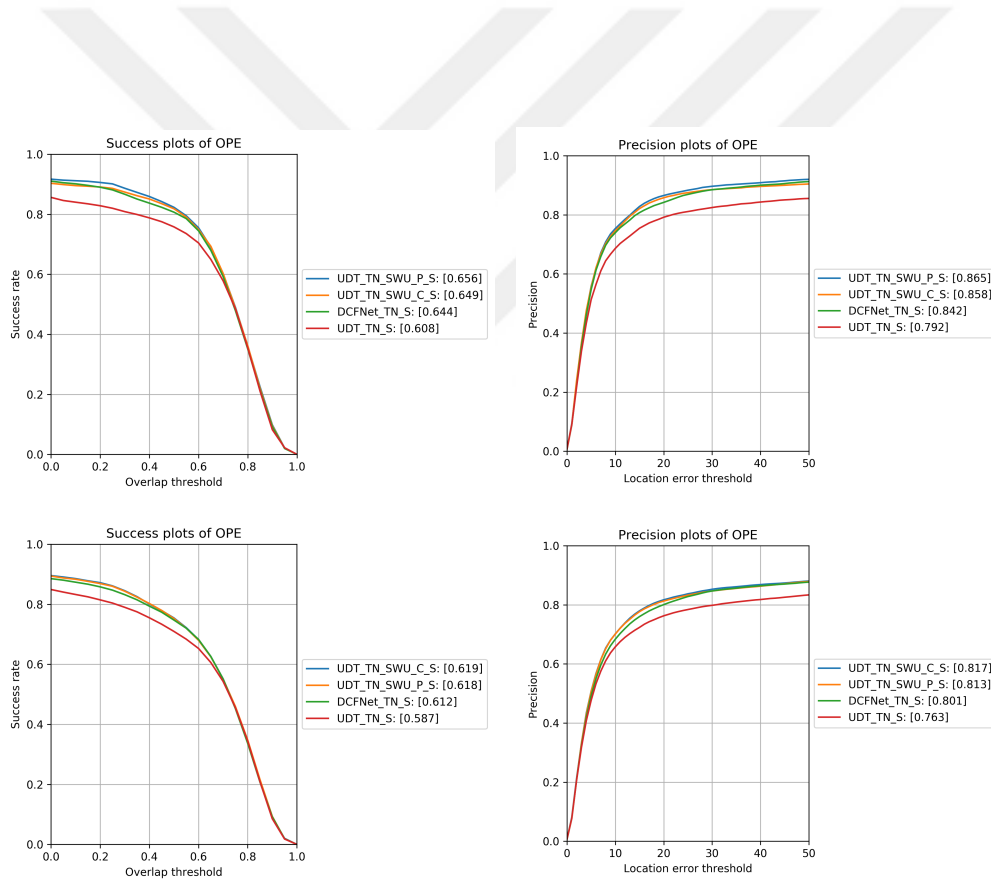


Figure 4.23: Comparison of model results in Test8 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below show OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

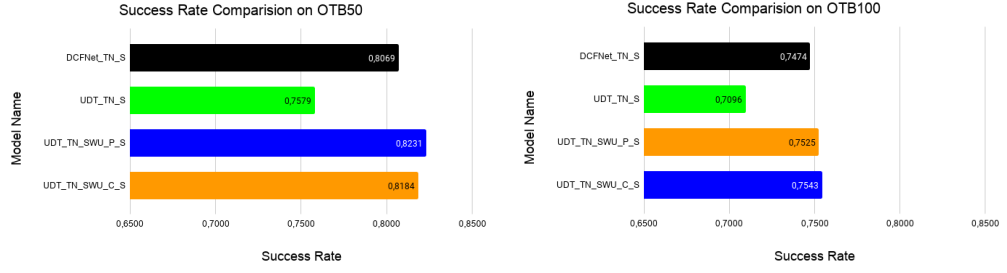


Figure 4.24: Comparison of model results in Test8 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

In contrast to the Test7, SWU\_C and SWU\_P models in Test8 manage to beat the fully supervised version. When UDT\_SWU\_P\_S and DCFNet\_TN\_S are compared, it is seen that there is 1.2% increase on success plot AUC value, 2.3% increase on precision plot AUC value and 1.6% increase on success rate in terms of OTB50 performances. For OTB 100, it is observed that there is 0.6% increase on success plot AUC value, 1.2% increase on precision plot AUC value and 0.6% increase on success rate.

From the results of Test7 and Test8 test groups, it can be again deducted that using large number of samples is decreasing the effect of weak supervision on TrackingNet. The dataset is composed of many different objects with 15 attributes, therefore, the optimum model can be obtained without any need of unsupervised modelling where there exist enough samples for training.

In Figure 4.25, performance results of Test7 and Test8 are shown together in order to see the effect of using different size of training samples. However, it is seen that the results are very close to each other to make any deduction. Considering the TrackingNet dataset, the benefit of SWU\_C and SWU\_P training methods is only visible when the number of training instances is small enough, and they are not as effective as weakly supervised models.

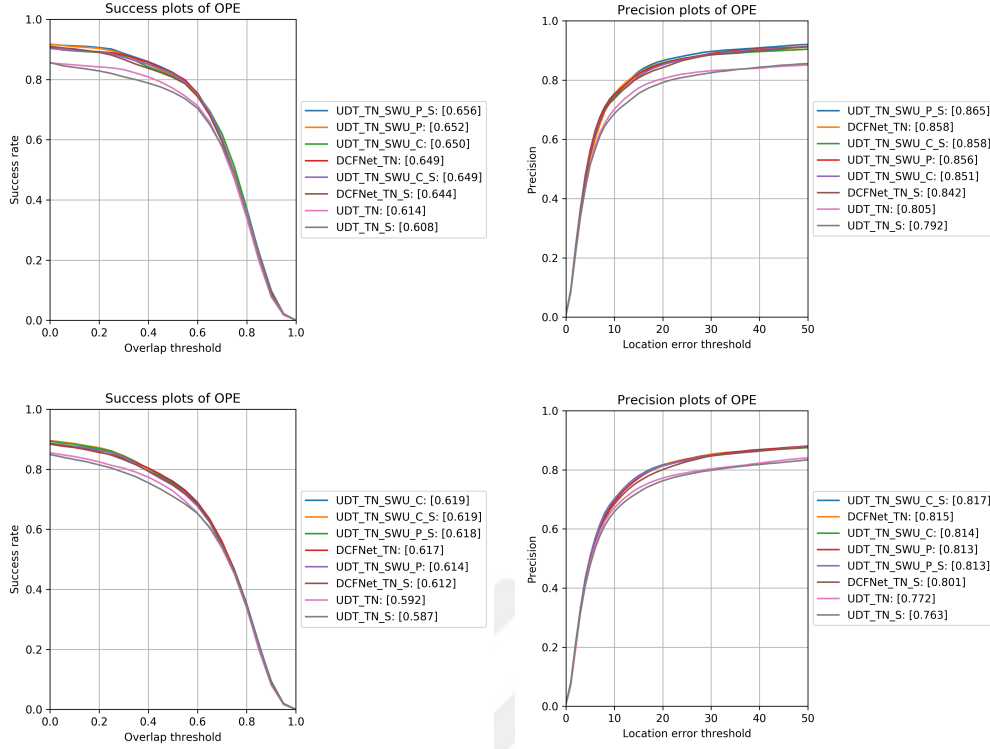


Figure 4.25: Comparison of model results of Test7 and Test8 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below show OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

### 4.6.3 JSU Test Groups

#### 4.6.3.1 Experimental Results for Test9 and Test10

The OTB50 and OTB100 success and precision plots for Test9 of JSU trainings on ILSVRC dataset are illustrated in Figure 4.26. The success rate comparison is shown in Figure 4.27. The best model of the trainings are selected according to a model having largest success plot AUC value as before. The precision plots and success rate are drawn according to the result of that model.



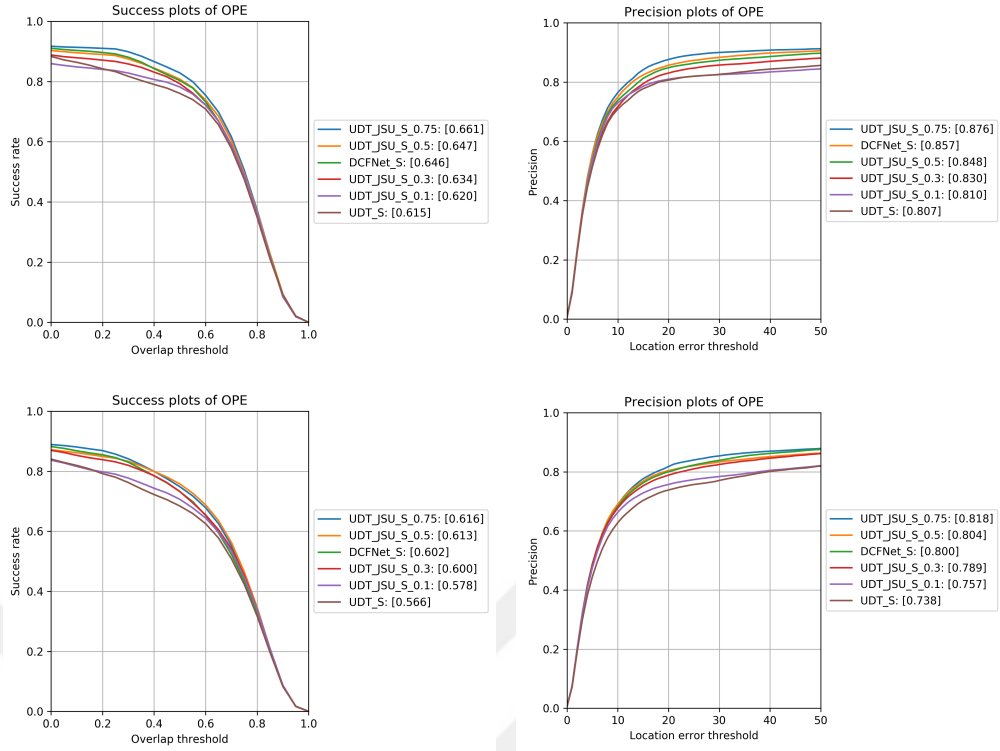


Figure 4.26: Comparison of model results in Test9 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

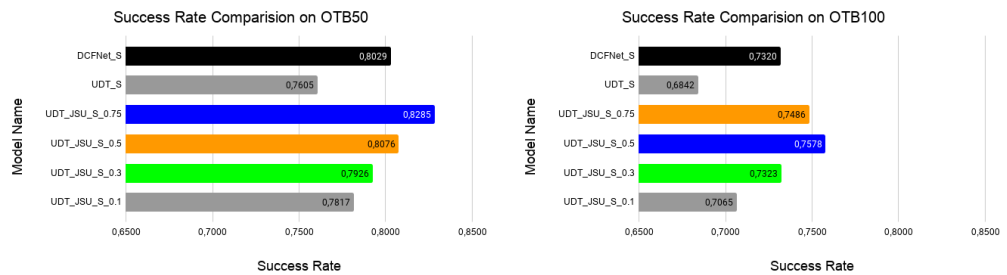


Figure 4.27: Comparison of model results in Test9 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

When the results shown in Figure 4.26 and Figure 4.27 are analyzed, it is observed that UDT\_JSU\_S\_0.75 model has better performance results than fully supervised model DCFNet\_S. When UDT\_JSU\_S\_0.75 and DCFNet\_S are compared, it is observed that there exist 1.5% increase on success plot AUC value, 1.9% increase on precision plot AUC value and 2.6% increase on success rate in terms of OTB50 results. For OTB 100, it is observed that there is 1.4% increase on success plot AUC value, 1.8% increase on precision plot AUC value and 1.7% increase on success rate. These results show that joint supervised and unsupervised learning has a similar effect on the tracker performance as incomplete supervision. In other words, the generalization ability of the tracker has increased with joint supervision methodology. Moreover, it is concluded that increasing the effect of unsupervised loss on the training results in performance loss.

Test10 is composed to see the effect of TrackingNet dataset when it is used as training set. Success and precision plots of Test10 are illustrated in Figure 4.28, and success rate comparison is shown in Figure 4.29.

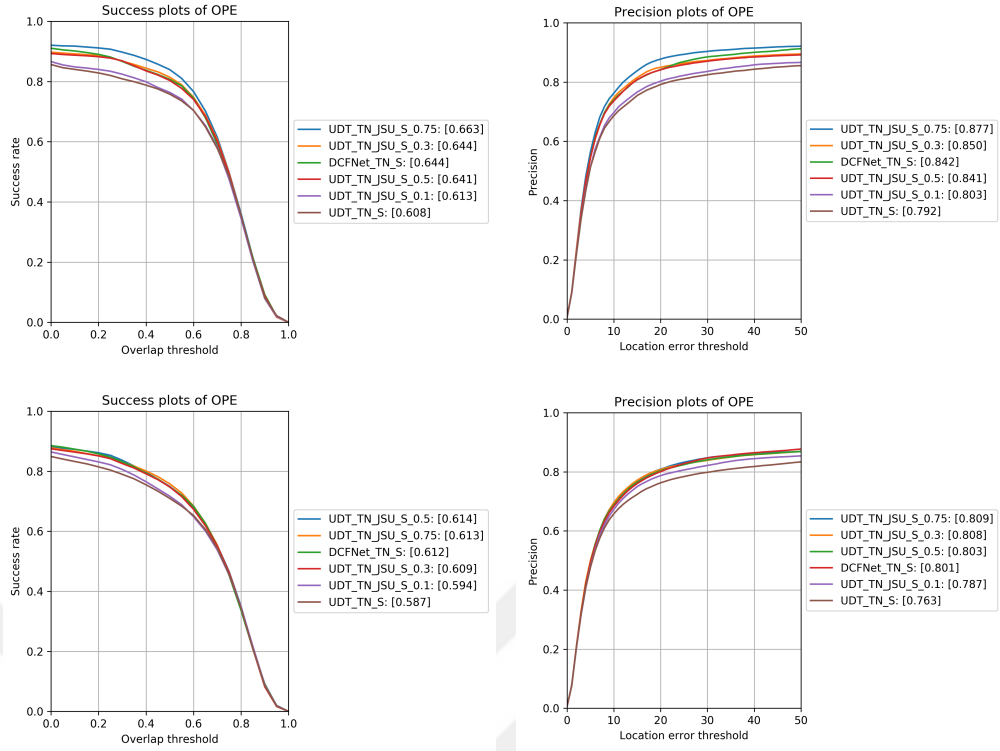


Figure 4.28: Comparison of model results in Test10 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

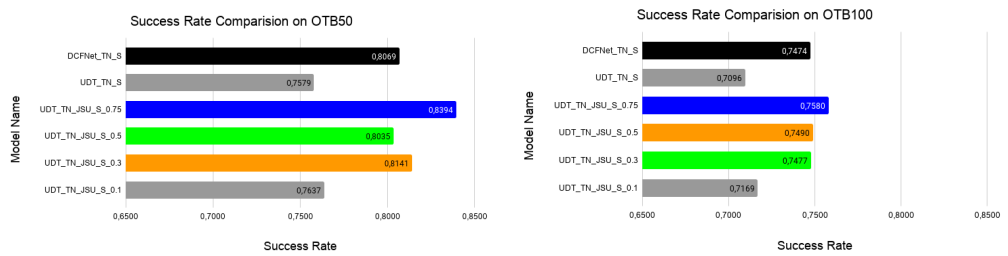


Figure 4.29: Comparison of model results in Test10 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

The results of Test10 shown in Figure 4.28 and Figure 4.29 indicates that the JSU

model with alpha equals to 0.75, namely UDT\_TN\_JSU\_S\_0.75, has very promising results on OTB50 benchmark set. However, the same effect is not observed on OTB100 set. When it is compared with DCFNet\_S, it is seen that there is 1.9% increase on success plot AUC value, 3.5% increase on precision plot AUC value and 3.3% increase on success rate in terms of OTB50 performances. For OTB 100, it is observed that there is 0.2% increase on success plot AUC value, 0.8% increase on precision plot AUC value and 1.1% increase on success rate. Also, UDT\_TN\_JSU\_S\_0.5 model, where alpha value is set to 0.5, has exhibited compatible performance on all evaluation parameters with respect to DCFNet\_TN\_S.

From the results of Test9 and Test10 test groups, it can be seen that joint supervised-unsupervised model with fixed alpha value (0.75) has better results than fully supervised methodology when ILSVRC dataset is utilized during training. However, the performance increase is not same when TrackingNet is used for training. Although the results on OTB50 set are spectacular, the OTB100 results have very small advantage against fully supervised model. Since the only difference between these models is the training dataset, the logical explanation of this behaviour can be derived from the properties of TrackingNet dataset.

#### **4.6.3.2 Experimental Results for Test11 and Test12**

The OTB50 and OTB100 success and precision plots for Test11 of JSU\_SWU\_C and JSU\_SWU\_P trainings on ILSVRC dataset are illustrated on Figure 4.30. The success rate comparison is shown in Figure 4.31. The best model of the trainings are chosen according to a model having largest success plot AUC value. The precision plots and success rate are drawn according to the result of that model.

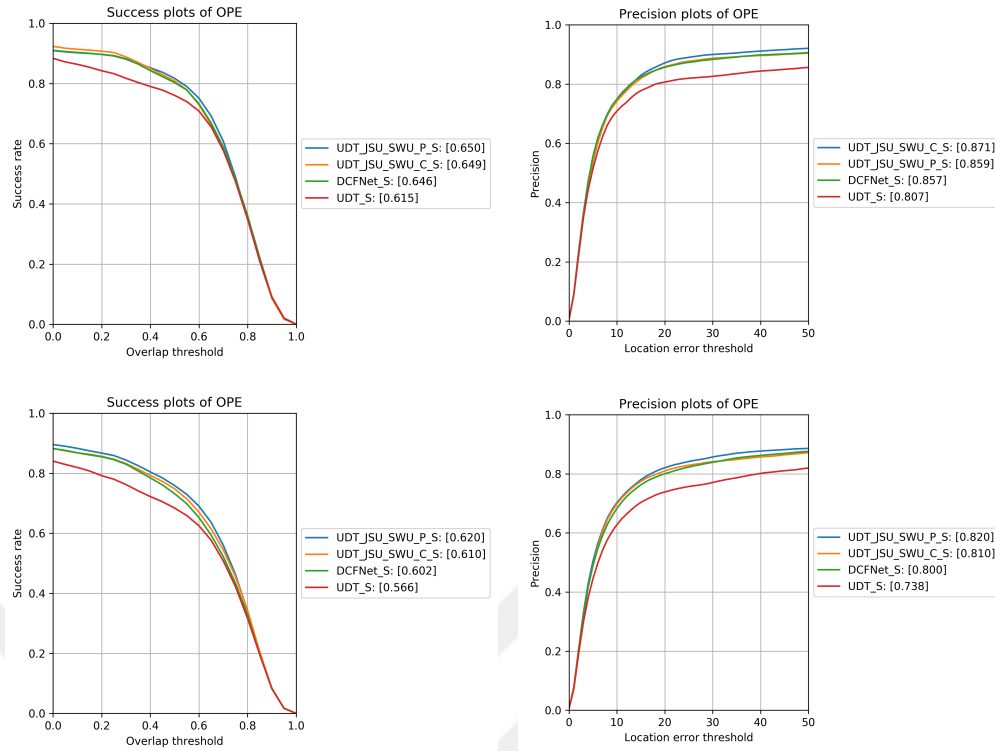


Figure 4.30: Comparison of model results in Test11 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

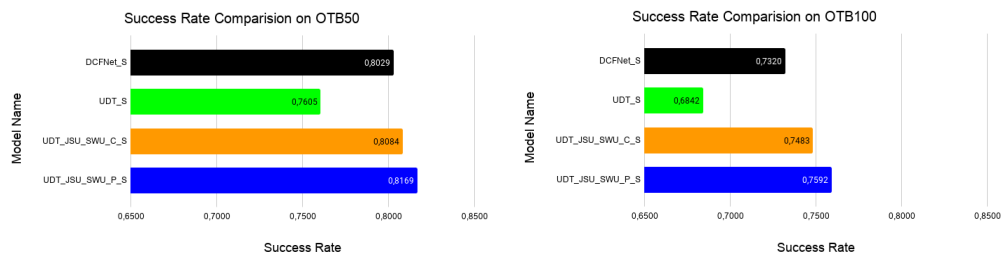


Figure 4.31: Comparison of model results in Test11 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

When results shown in Figures 4.30 and 4.31 are analyzed in detail, it is observed

that both UDT\_JSU\_SWU\_S and UDT\_JSU\_SWU\_P\_S models have better performance results than fully supervised DCFNet\_S. When UDT\_JSU\_SWU\_P\_S and DCFNet\_S are compared, it is seen that UDT\_JSU\_SWU\_P\_S has performance gain of 1.4% increase on precision plot AUC value, 0.4% increase on success plot AUC value and 1.4% increase on success rate in terms of OTB50 results. For OTB 100, it is observed that there is 1.8% increase on success plot AUC value, 2.0% increase on precision plot AUC value and 2.7% increase on success rate. Similarly, when UDT\_JSU\_SWU\_C\_S and DCFNet\_S are analyzed, the performance difference is not much but noticeable. It is observed that UDT\_JSU\_SWU\_C\_S outperforms it with 0.2% increase on success plot AUC value, 0.3% increase on precision plot AUC value and 0.6% increase on success rate in terms of OTB50 results. For OTB 100, there exists 0.8% increase on success plot AUC value, 1.0% increase on precision plot AUC value and 1.6% increase on success rate. These aforementioned results show that partially decreasing alpha value during training is better than continuously decreasing it when ILSVRC training set is concerned.

Test12 is generated to see the effect of TrackingNet dataset when it is used as training set. Success and precision plots of Test12 are illustrated in Figure 4.32, and success rate comparison is shown in Figure 4.33.

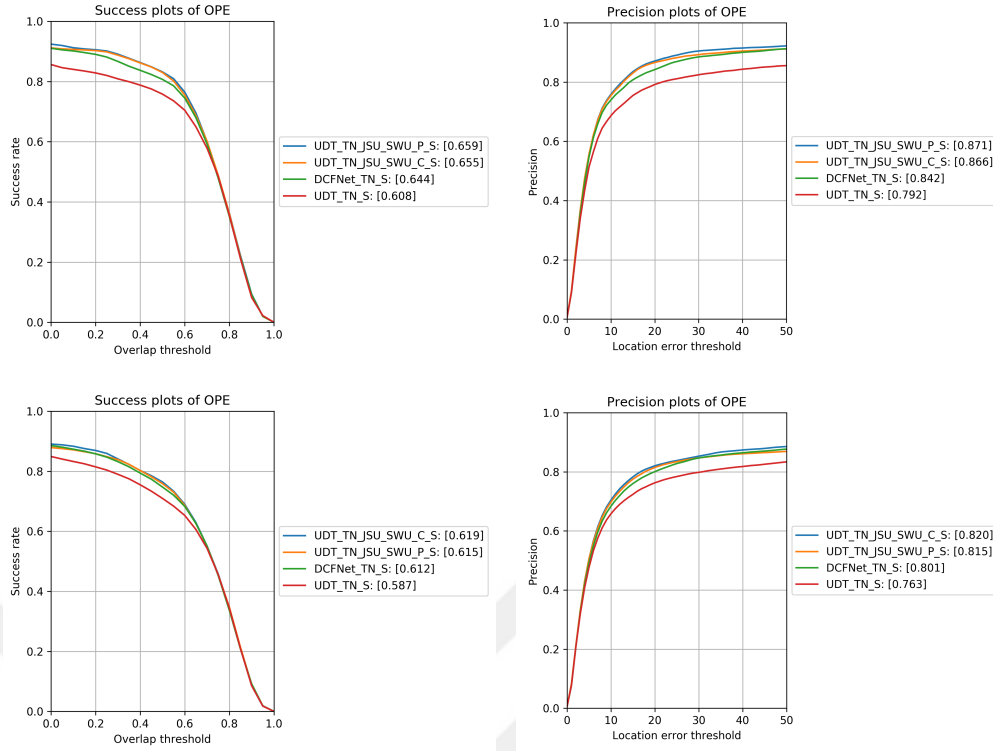


Figure 4.32: Comparison of model results in Test12 in terms of success and precision plots. Two graphs upper-side of the figure show OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

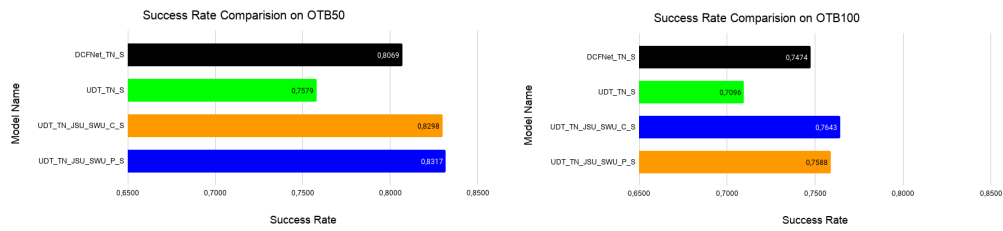


Figure 4.33: Comparison of model results in Test12 in terms of success rate. The graph on the left of the figure shows success rate values on OTB50 while the graph on the right shows success rate on OTB100. Best viewed in color.

The results of Test12 which are displayed in Figure 4.32 and Figure 4.33 show that

UDT\_TN\_JSU\_SWU\_C\_S model has good results on OTB50 benchmark set. The performance increase on OTB100 test set is also good to be noticed, however, it is not as powerful as OTB50 results. When it is compared with DCFNet\_S, it is seen that there is 1.1% increase on success plot AUC value, 2.4% increase on precision plot AUC value and 2.3% increase on success rate in terms of OTB50 performances. For OTB 100, it is observed that there is 0.7% increase on success plot AUC value, 1.9% increase on precision plot AUC value and 1.7% increase on success rate. In contrast to the ILSVRC trained models of Test11, partially decreasing alpha value is not always better than continuously decreasing it.

#### 4.6.4 Overall Comparison of Test Groups Trained on ILSVRC

OTB50 and OTB100 success and precision plots of models which are trained using small number of samples on ILSVRC dataset (Test2, Test6, Test9, Test11) are shown on Figure 4.34. These models include all of the proposed methods such as weakly supervised, SWU\_C, SWU\_P, JSU, JSU\_SWU\_C and JSU\_SWU\_P.

Figure 4.34 shows that two incomplete supervision methods (SWU and Weakly) and two joint supervision methods (JSU and JSU\_SWU) have shown better results than fully supervision method when ILSVRC dataset is used for training. When the OTB50 results are considered, it is observed that JSU\_S model with fixed alpha value (0.75) has the best score on success plot AUC value, and is followed by SWU\_P\_S and SWU\_S models. However, the situation is reversed on precision plot where SWU\_P\_S and SWU\_S models are ahead of JSU\_S\_0.75. Weakly supervised models are also showing good results when compared to the fully supervised one. Nevertheless, they are not as strong as other methods. In terms of OTB100 results, weakly supervised UDT\_W\_S\_0.75 model enters to the top three on the graph for both success and precision plots, and shows very competitive performance. On the other hand, SWU\_P\_S model occupies the top of the list on success plot of OTB100 where JSU\_SWU\_P\_S is the second best. For ILSVRC trained models, one can say that decreasing probability and alpha value partially instead of continuously on incomplete and joint supervision has better effect on learning robust features.



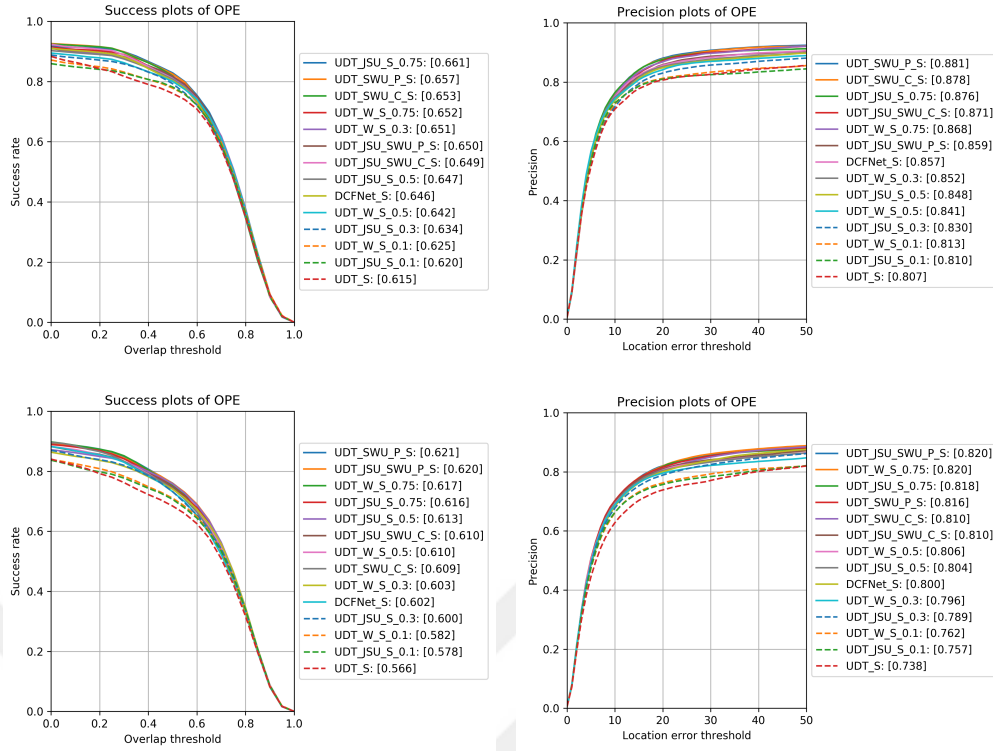


Figure 4.34: Comparison of models utilized ILSVRC dataset in terms of success and precision plots. Two graphs upper-side of the figure shows OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

#### 4.6.5 Overall Comparison of Test Groups Trained on TrackingNet

OTB50 and OTB100 success and precision plots of models trained using small number of samples on TrackingNet dataset (Test4, Test8, Test10, Test12) are shown on Figure 4.35. These models include all of the proposed methods such as weakly supervised, SWU\_C, SWU\_P, JSU, JSU\_SWU\_C and JSU\_SWU\_P.

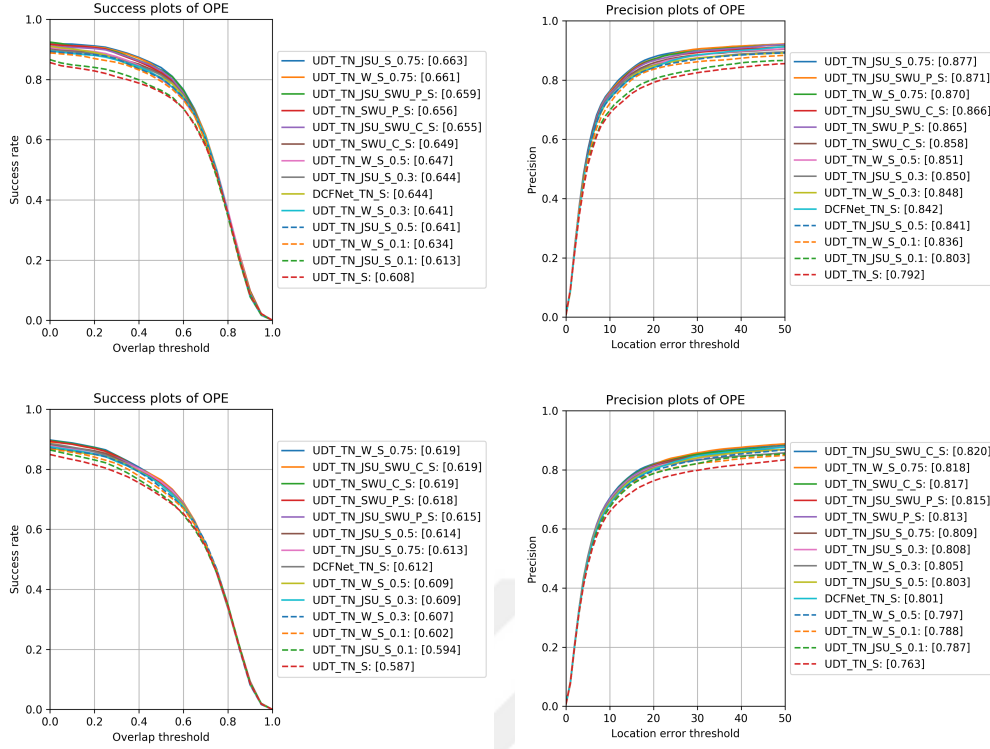


Figure 4.35: Comparison of models utilized TrackingNet dataset in terms of success and precision plots. Two graphs upper-side of the figure shows OTB50 test set results while the other two graphs below belong to OTB100 results. The number on the legend part of the graphs indicates the AUC value of that specific plot. Best viewed in color.

Figure 4.35 shows that two incomplete supervision methods (SWU and Weakly) and two joint supervision methods (JSU and JSU\_SWU) have shown better results than fully supervision method when TrackingNet dataset is used for training. However, the performance gains over fully supervised model are not as eye-catching as the ones in ILSVRC trained models. This behaviour results from the properties of the TrackingNet dataset since it is more diverse and representative of real world than ILSVRC. When OTB50 benchmark set results are taken into account, it is observed that JSU\_S\_0.75 has the best score on both precision and success plot AUC value, and is followed by JSU\_SWU\_P and W\_S\_0.75. However, JSU\_S model has lost its superiority on OTB100 test set, and fallen down on the list. The top three performers of OTB100 become W\_S\_0.75, JSU\_SWU\_C\_S, and SWU\_C\_S models. In contrast

to results on ILSVRC, decreasing probability and alpha value partially on incomplete and joint supervision does not always perform better than the one decreasing the values continuously.

#### 4.6.6 Comparison of SOTA Trackers and Proposed Models

In Table 4.14, OTB100 performance results of SOTA trackers and top performing proposed models together with fully supervised models are exhibited. Although the DCFNet architecture is couple of years old, our proposed methodologies decrease the performance gap between DCFNet and state of the art tracker models.

Table 4.14: OTB100 performance results of state-of-the-art trackers and top performing proposed models together with fully supervised models

| Tracker Name     | OTB100                 |                          |
|------------------|------------------------|--------------------------|
|                  | Success Plot AUC Score | Precision Plot AUC Score |
| SiamRPN++[4]     | 0.690                  | 0.900                    |
| ATOM[29]         | 0.670                  | 0.870                    |
| DaSiamRPN[88]    | 0.650                  | 0.880                    |
| UDT_TN_W_0.75    | 0.628                  | 0.826                    |
| UDT_SWU_P_S      | 0.621                  | 0.816                    |
| UDT_JSU_SWU_P_S  | 0.620                  | 0.820                    |
| UDT_TN_W_S_0.75  | 0.619                  | 0.804                    |
| UDT_TN_JSU_SWU_S | 0.619                  | 0.820                    |
| UDT_W_S_0.75     | 0.617                  | 0.820                    |
| DCFNet_TN        | 0.617                  | 0.815                    |
| DCFNet           | 0.600                  | 0.790                    |



## CHAPTER 5

### CONCLUSIONS

Visual object tracking with numerous real-world applications is a challenging computer vision problem. It is being used at target analysis, action recognition, augmented reality. It aims to find the target object position in a video frame given that there is a bounding box annotation in the first frame. In the past years, correlation filters have produced excellent results in terms of accuracy and, they have been popular because of their low computational complexity. The main purpose of correlation filter based tracker is to find a filter that can generate high values around the actual object location but low values for the locations far from the object. Feature properties which are utilized by these filters plays important role on the performance.

Learning discriminative features is the main goal of the algorithms. With the recent developments on deep network architectures, tracker algorithms started to adopt deep neural networks for feature extraction due to their superior performance. The Siamese trackers formulate the tracking task as a similarity matching process, and use deep networks as feature extractor as well. In order to train these deep networks, visual tracking algorithms use labeled data for supervised training. Neural networks usually learn very well on their training sets, and obtain a good convergence on those sets. However, they do not perform well when they encounter a frame having very different properties such as resolution, quality, field-of-view, aspect ratio. Although there are several tracking data sets having full labels for each video, it is always desired to use many of these sets so that the network can learn representative coefficients and get good generalization ability. Nevertheless, annotating video frames desires heavy human resources. Therefore, many unsupervised and semi supervised techniques are developed to train deep networks. Despite the fact that there are remarkable process

on object detection, segmentation, pose/depth estimation etc. for semi supervision, visual object tracking problem has few studies related to the unsupervised and semi supervised training. Therefore, it is aimed to explore the effect of very simple semi supervision technique on tracking task and compare with fully supervised and joint supervised techniques in this thesis.

Instead of designing new network architectures in order to increase the precision and robustness of visual trackers, the goal is to focus on the training strategy and investigate the effect of incomplete supervision on the performance. Two different incomplete supervision methods are experimented. One of them is performed in a manner that the training batch samples are either chosen from labeled version or unlabeled version of the data set with predetermined probability. In other words, if the probability is set to 0.3, supervised samples are chosen from the labeled set with 0.3 probability. Otherwise the samples are selected from the unlabeled samples. This method is called weakly supervised training. Other method, called as SWU, sweeps the probability value from 1.0 to 0.0 throughout the training. This method is investigated in two parts, denoted as SWU\_C and SWU\_P. SWU\_C training uniformly decreases the probability value after each epoch. However, the probability value is quantized on SWU\_P trainings. For example, assume that the total number of epoch is 50. The probability value is set to 1.0 for first 10 epoch. For the second 10 epoch, it is set to 0.75. The third 10 epoch uses the value as 0.5. The fourth one uses 0.25, and the final 10 epoch utilizes 0.0 probability value which means the training is performed completely unsupervised for last 10 epoches. In addition to the incomplete supervision methods, joint supervision is also investigated. During a training iteration, loss values resulting from supervised and unsupervised learning schemes are summed with a control parameter  $\alpha$ . Analogous to the probability value on weakly supervised method, the control parameter is set to a fixed number during training, and different values are trialed. This methodology is denoted as JSU. Furthermore, the value of control parameter is continuously and partially decreased at each epoch of a training, which are called JSU\_SWU\_C and JSU\_SWU\_P methods.

To perform the experiments, fully supervised trained DCFNet tracker and its unsupervised version UDT tracker are adopted. Several different probability values are tested for weakly supervision. In addition to that, two different number of samples

per epoch are used to see the effect of it. The proposed semi supervision techniques are trained on ILSVRC and TrackingNet dataset, and tested on widely used OTB50 and OTB100 tracking benchmark datasets. It is observed that weakly supervision with probability value 0.75 on both training datasets with small number of samples has beaten fully supervised version in terms of benchmark parameters of OTB50 and OTB100, which are success plot, precision plot and success rate. Incomplete supervision on ILSVRC2015 dataset leads up-to 1.1% gain on success plot AUC value, 2.4% increase on precision plot AUC value and 2.2% increase on success rate in terms of OTB50 benchmark parameters. Similarly on OTB100 test set, 1.9% gain on success plot AUC value, 2.0% on precision plot AUC value and 2.8% increase on success rate are observed. Meanwhile, incomplete supervision on TrackingNet results in up-to 1.7% increase on success plot AUC value, 2.8% increase on precision plot AUC value and 2.3% gain on success rate on OTB50 benchmark set. Furthermore, it leads up-to 0.7% increase on success plot AUC value, 1.7% increase on precision plot AUC value and 0.7 increase on success rate in terms of OTB100 evaluation parameters. In addition to these outcomes, it is observed that SWU\_P models performed better than other incomplete supervision techniques, which means that initially supervising the model to some point and then increasing the effect of self-supervision is a good choice to train DCFNet on ILSVRC set. However, it is seen that this deduction is not applicable for TrackingNet trained models due to different dataset properties.

Joint supervision model where control parameter is fixed to 0.75 shows similar results with incomplete supervision on OTB50 success plots for both training datasets. However, the same superiority is not seen on OTB100 dataset. This shows that JSU models learns basic tracking properties, however, struggles to overcome the challenging conditions. This situation become vice versa when control parameter is gradually decreased. JSU\_SWU models become competitive on OTB100 benchmark set as well.

The aforementioned semi supervised methodologies are also tested with larger number of training samples. ILSVRC trained weakly supervised and SWU models with large training instances also beat fully supervised model by a noticeable margin. However, TrackingNet trained SWU models do not manage to surpass DCFNet\_TN model. From these observations, we can deduct that weakly supervised and SWU

models are losing its performance superiority as the labeled training set enlarges and becomes more generalized.

After evaluating several incomplete supervised and joint supervised training methodologies, this thesis proposes that utilization of unsupervised samples on a standard supervised training for correlation filter based visual object tracking, the tracker generalizes itself better on the real world. As the training dataset becomes more universal with having many objects and attributes, the positive effect of incomplete and joint supervision start disappearing. However, it is firmly believed that these methods are expected to increase their performances with better unsupervised learning models.





## REFERENCES

- [1] K. Simonyan and A. Zisserman, “Very deep convolutional neural networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “Fully-convolutional siamese networks for object tracking,” *In European Conference on Computer Vision*, pp. 850–865, 2016.
- [3] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, “High performance visual tracking with siamese region proposal network,” *CVPR*, 2018.
- [4] B. Li, W. Wu, Q. Wang, F. Zang, J. Xing, and J. Yan, “SiamRPN++: Evolution of siamese visual tracking with very deep networks,” *CVPR*, 2019.
- [5] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem, “Trackingnet: A large-scale dataset and benchmark for object tracking in the wild,” *ArXiv*, vol. abs/1803.10794, 2018.
- [6] E. Gundogdu and A. Alatan, “Good features to correlate for visual tracking,” *IEEE Transactions on Image Processing*, vol. 27, pp. 2526–2540, 2018.
- [7] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, “Unsupervised deep tracking,” *arXiv preprint arXiv:1904.01088*, 2019.
- [8] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu, “DCFNet: Discriminant correlation filters network for visual tracking,” *arXiv preprint arXiv:1704.04057*, 2017.
- [9] A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan, “Nus-pro: A new visual tracking challenge,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 335–349, 2016.
- [10] P. Liang, E. Blasch, and H. Ling, “Encoding color information for visual tracking: Algorithms and benchmark,” *IEEE Transactions on Image Processing*, vol. 24, pp. 5630–5644, 2015.

- [11] M. Müller, N. Smith, and B. Ghanem, “A benchmark and simulator for UAV tracking,” in *ECCV*, 2016.
- [12] Y. Wu, J. Lim, and M.-H. Yang, “Object tracking benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1834–1848, 2015.
- [13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *IJCV*, 2015.
- [14] R. W. Oppenheim, A. V. Schafer, and J. R. Buck, eds., *Discrete-time Signal Processing (2nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 1999.
- [15] D. Bolme, J. R. Beveridge, B. Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” *In IEEE Conference on CVPR*, p. 2544–2550, June 2010.
- [16] Y. Li and J. Zhu, “A scale adaptive kernel correlation filter tracker with feature integration,” *In IEEE ECCV*, p. 254–265, 2014.
- [17] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” *BMVC*, 2014.
- [18] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, “Discriminative scale space tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1561–1575, 2017.
- [19] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, “Learning spatially regularized correlation filters for visual tracking,” *ECCV*, 2015.
- [20] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, “Learning spatial-temporal regularized correlation filters for visual tracking,” *CVPR*, 2018.
- [21] N. Dalal and B. Triggs, “Histograms of gradients for human detection,” *CVPR*, 2005.
- [22] J. van de Weijer, C. Schmid, J. Verbek, and D. Larlus, “Learning color names for real-world applications,” *TIP*, 2009.

- [23] J. Zhao, J. Han, and L. Shao, “Unconstrained face recognition using a set-to-set distance measure on deep learned features,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, pp. 2679–2689, 2018.
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *International Conference on Neural Information Processing Systems*, 2012.
- [25] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, “Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video,” *arXiv preprint arXiv:1702.00824*, 2017.
- [26] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *CVPR 2009*, 2009.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CVPR*, 2016.
- [28] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, “Acquisition of localization confidence for accurate object detection,” *ECCV*, 2018.
- [29] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, “ATOM: Accurate tracking by overlap maximization,” *CVPR*, 2019.
- [30] J. R. Shewchuk, “An introduction to the conjugate gradient method without the agonizing pain,” *Technical report, Pitts-burgh*, 1994.
- [31] Z.-H. Zhou, “A brief introduction to weakly supervised learning,” *National Science Review*, vol. 5, pp. 44–53, 2018.
- [32] W. Sultani, C. Chen, and M. Shah, “Real-world anomaly detection in surveillance videos,” *CVPR*, pp. 6479–6488, 2018.
- [33] Y. Chen and J. Z. Wang, “Image categorization by learning and reasoning with regions,” *Journal of Machine Learning Research* 5, pp. 913–939, 2004.
- [34] Q. Zhang, S. A. Goldman, W. Yu, and J. E. Fritts, “Content-based image retrieval using multiple-instance learning,” in *ICML*, 2002.

- [35] J. Tang, H. Li, G.-J. Qi, and T.-S. Chua, “Image annotation by graph-based inference with integrated multiple/single instance representations,” *IEEE Transactions on Multimedia*, vol. 12, pp. 131–141, 2010.
- [36] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan, “Object detection with discriminatively trained part based models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1627–1645, 2009.
- [37] B. Babenko, M.-H. Yang, and S. J. Belongie, “Robust object tracking with on-line multiple instance learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1619–1632, 2011.
- [38] F. Muhlenbach, S. Lallich, and D. A. Zighed, “Identifying and handling mislabelled instances,” *Journal of Intelligent Information Systems*, vol. 22, pp. 89–109, 2004.
- [39] Z.-H. Zhou, “Ensemble methods: Foundations and algorithms,” 2012.
- [40] V. C. Raykar, S. Yu, L. H. Zhao, G. Hermosillo, C. Florin, L. Bogoni, and L. Moy, “Learning from crowds,” *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, 2010.
- [41] B. Settles, “Active learning literature survey,” technical report 1648, Department of Computer Sciences, University of Wisconsin at Madison, Wisconsin, WI, 2010.
- [42] A. D. E. Altri, “Maximum likelihood from incomplete data via the EM algorithm,” 1977.
- [43] O. Chapelle and A. Zien, “Semi-supervised classification by low density separation,” in *AISTATS*, 2005.
- [44] H. Bilen, M. Pedersoli, and T. Tuytelaars, “Weakly supervised object detection with posterior regularization,” 2014.
- [45] M. B. Blaschko, A. Vedaldi, and A. Zisserman, “Simultaneous object detection and ranking with weak supervision,” in *NIPS*, 2010.
- [46] T. Deselaers, B. Alexe, and V. Ferrari, “Localizing objects while learning their appearance,” in *ECCV*, 2010.

- [47] H. Bilen and A. Vedaldi, “Weakly supervised deep detection networks,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2846–2854, 2016.
- [48] P. Tang, X. Wang, X. Bai, and W. Liu, “Multiple instance detection network with online instance classifier refinement,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3059–3067, 2017.
- [49] Y. Wei, Z. Shen, B. Cheng, H. Shi, J. Xiong, J. Feng, and T. Huang, “Ts2c: Tight box mining with surrounding segmentation context for weakly supervised object detection,” *ArXiv*, vol. abs/1807.04897, 2018.
- [50] F. Wan, P. Wei, Z. Han, J. Jiao, and Q. Ye, “Min-entropy latent model for weakly supervised object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 2395–2409, 2019.
- [51] X. Zhang, J. Feng, H. Xiong, and Q. Tian, “Zigzag learning for weakly supervised object detection,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4262–4270, 2018.
- [52] R. B. Girshick, “Fast r-cnn,” *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.
- [53] A. Kolesnikov and C. H. Lampert, “Seed, expand and constrain: Three principles for weakly-supervised image segmentation,” *ArXiv*, vol. abs/1603.06098, 2016.
- [54] T. Durand, T. Mordan, N. Thome, and M. Cord, “Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5957–5966, 2017.
- [55] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, “Object region mining with adversarial erasing: A simple classification to semantic segmentation approach,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6488–6496, 2017.

- [56] A. Diba, V. Sharma, A. M. Pazandeh, H. Pirsiavash, and L. V. Gool, “Weakly supervised cascaded convolutional networks,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5131–5139, 2017.
- [57] W. Ge, S. Yang, and Y. Yu, “Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1277–1286, 2018.
- [58] X. Li, M. Kan, S. Shan, and X. Chen, “Weakly supervised object detection with segmentation collaboration,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9734–9743, 2019.
- [59] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2009.
- [60] M. Everingham, S. Eslami, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International Journal of Computer Vision*, vol. 111, pp. 98–136, 2014.
- [61] R. Garg, B. V. Kumar, G. Carneiro, and I. Reid, “Unsupervised cnn for single view depth estimation: Geometry to the rescue,” in *ECCV*, 2016.
- [62] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6612–6619, 2017.
- [63] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, “Learning depth from monocular videos using direct methods,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2022–2030, 2018.
- [64] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, “Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8976–8985, 2019.

- [65] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. P. C. Valentin, and S. Izadi, “Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction,” in *ECCV*, 2018.
- [66] W. Chen, Z. Fu, D. Yang, and J. Deng, “Single-image depth perception in the wild,” in *NIPS*, 2016.
- [67] Z. Li and N. Snavely, “Megadepth: Learning single-view depth prediction from internet photos,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2041–2050, 2018.
- [68] H. Ren, A. Raj, M. El-Khamy, and J. Lee, “Suw-learn: Joint supervised, unsupervised, weakly supervised deep learning for monocular depth estimation,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3235–3243, 2020.
- [69] R. Navaratnam, A. Fitzgibbon, and R. Cipolla, “Semi-supervised learning of joint density models for human pose estimation,” in *BMVC*, 2006.
- [70] A. Kanaujia, C. Sminchisescu, and D. Metaxas, “Semi-supervised hierarchical models for 3d human pose reconstruction,” *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.
- [71] N. Ukita and Y. Uematsu, “Semi- and weakly-supervised human pose estimation,” *Comput. Vis. Image Underst.*, vol. 170, pp. 67–78, 2018.
- [72] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” *CVPR*, 2013.
- [73] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 583–596, 2015.
- [74] S. Hong, T. You, S. Kwak, and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network,” in *ICML*, 2015.
- [75] J. F. Henriques, R. Caseiro, P. Martins, and J. P. Batista, “Exploiting the circulant structure of tracking-by-detection with kernels,” in *ECCV*, 2012.

- [76] J. Zhang, S. Ma, and S. Sclaroff, “Meem: Robust tracking via multiple experts using entropy minimization,” in *ECCV*, 2014.
- [77] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, “Hierarchical convolutional features for visual tracking,” *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3074–3082, 2015.
- [78] J. Gao, H. Ling, W. Hu, and J. Xing, “Transfer learning based visual tracking with gaussian processes regression,” in *ECCV*, 2014.
- [79] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, “Hedged deep tracking,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4303–4311, 2016.
- [80] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, “Staple: Complementary learners for real-time tracking,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1401–1409, 2016.
- [81] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, “Eco: Efficient convolution operators for tracking,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6931–6939, 2017.
- [82] B. Chen, D. Wang, P. Li, S. Wang, and H. Lu, “Real-time ‘actor-critic’ tracking,” in *ECCV*, 2018.
- [83] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “End-to-end representation learning for correlation filter based tracking,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5000–5008, 2017.
- [84] J. Choi, H. J. Chang, S. Yun, T. Fischer, Y. Demiris, and J. Y. Choi, “Attentional correlation filter network for adaptive visual tracking,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4828–4837, 2017.
- [85] J. Choi, H. J. Chang, J. Jeong, Y. Demiris, and J. Y. Choi, “Visual tracking using attention-modulated disintegration and integration,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4321–4330, 2016.



- [86] A. Lukezic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan, “Discriminative correlation filter with channel and spatial reliability,” in *CVPR*, 2017.
- [87] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojir, G. Häger, G. Nebehay, and R. Pflugfelder, “The visual object tracking VOT2015 challenge results,” *ICCV*, 2015.
- [88] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, “Distractor-aware siamese networks for visual object tracking,” in *ECCV*, 2018.

