2019

Generative Adversarial Networks for Online Visual Object Tracking Systems

ghsoun zin
zinx1710@mylaurier.ca

Generative Adversarial Networks for Online Visual Object Tracking Systems

By

Ghsoun Zin

THESIS

Submitted to the Department of Physics and Computer Science

Faculty of Science

in partial fulfilment of the requirements for

Degree of Master of Applied Computing

Wilfrid Laurier University

August 2019

Abstract

Object Tracking is one of the essential tasks in computer vision domain as it has numerous applications in various fields, such as human-computer interaction, video surveillance, augmented reality, and robotics. Object Tracking refers to the process of detecting and locating the target object in a series of frames in a video. The state-of-the-art for tracking-by-detection framework is typically made up of two steps to track the target object. The first step is drawing multiple samples near the target region of the previous frame. The second step is classifying each sample as either the target object or the background. Visual object tracking remains one of the most challenging task due to variations in visual data such as target occlusion, background clutter, illumination changes, scale changes, as well as challenges stem from the tracking problem including fast motion, out of view, motion blur, deformation, and in and out planar rotation. These challenges continue to be tackled by researchers as they investigate more effective algorithms that are able to track any object under various changing conditions. To keep the research community motivated, there are several annual tracker benchmarking competitions organized to consolidate performance measures and evaluation protocols in different tracking subfields such as Visual Object Tracking VOT challenges and The Multiple Object Tracking MOT Challenges [1, 2].

Despite the excellent performance achieved with deep learning, modern deep tracking methods are still limited in several aspects. The variety of appearance changes over time remains a problem for deep trackers, owing to spatial overlap between positive samples. Furthermore, existing methods require high computational load and suffer from slow running speed.

Recently, Generative Adversarial Networks (GANs) have shown excellent results in solving a variety of computer vision problems, making them attractive in investigating their potential use in achieving better results in other computer vision applications, namely, visual object tracking.

In this thesis, we explore the impact of using Residual Network ResNet as an alternative feature extractor to Visual Geometry Group VGG which is commonly used in literature. Furthermore, we attempt to address the limitations of object tracking by exploiting the ongoing advancement in Generative Adversarial Networks. We describe a generative adversarial network intended to improve the tracker's classifier during the online training phase. The network generates adaptive masks to augment the positive samples detected by the convolutional layer of the tracker's model in order to improve the model's classifier by making the samples more difficult. Then we integrate this network with Multi-Domain Convolutional Neural Network (MDNet) tracker and present the results.

Furthermore, we introduce a novel tracker, MDResNet, by substituting the convolutional layers of MDNet that were originally taken from Visual Geometry Group (VGG-M) network with layers taken from Residual Deep Network (ResNet-50) and the results are compared.

We also introduce a new tracker, Region of Interest with Adversarial Learning (ROIAL), by integrating the generative adversarial network with the Real-Time Multi-Domain Convolutional Network (RT-MDNet) tracker. We also integrate the GAN network with MDResNet and MDNet and compare the results with ROIAL.

This is for you, my beloved Anas & Hamza

Without your support and love, none would have come to be.

Acknowledgement

Foremost, I would like to thank my supervisor Dr. Safaa Bedawi for her support and guidance throughout my research. Your valuable directions and constructive suggestions helped me with the development of this research.

A special thanks to Dr. Chinh Hoang for the sponsorship he provided me with for Compute Canada which was critical to the success of this thesis. Thank you.

My deepest appreciation goes to ISOW and Jusoor and to everyone involved in giving me this once-in-a-lifetime opportunity. I would have never achieved any of this without your enablement and support. Thanks for believing in me and setting me up for success.

Finally, I would like to thank my beloved husband for supporting me throughout this journey, and my beloved son for bearing with me throughout this experience. They were always there at all times providing me with the emotional support I needed.

Thank you.

Contents

List of Figures

List of Equations

List of Tables

Chapter 1

1. INTRODUCTION

1.1 OVERVIEW

Computer Vision is considered one of the most interesting fields in computer science and artificial intelligence. The pursuit of providing machines with a sense of sight that is as good or better than that of humans is keeping researchers busy and motivated. The fundamental goal of this field is to reduce the gap between what the machine can understand, that is pixels and numbers, and what the actual meaning of the image is [3].

There is a wide range of problems with active research within the domain of computer vision, such as facial recognition, object classification, and scene recognition. In this thesis, we focus on Visual Tracking.

Visual Tracking is a domain within computer vision that has become a very active research zone, thanks to the abundance of computing power, availability of high-quality cameras, and the increasing need for automated video analysis. Visual Object Tracking has become a fundamental building block of various cutting edge technologies such as autonomous cars and traffic monitoring [4, 5]. It has also enabled great advancements in video surveillance [6] and sports analysis [7]. Visual Object Tracking also contributes to numerous other applications including human-

computer interaction, augmented reality, robotics, film production, and measurements in various fields such as medicine, biology, and meteorology [8, 9, 10, 11, 12].

Visual Tracking can be defined as the process of estimating and detecting the localization of one or more target objects in a sequence of frames [13, 6, 14]. In other words, it is the process of capturing a specific object or objects while in motion.

Visual Object Trackers can be classified as discriminative or generative, single or multi-object, with its learning algorithms either offline or online. The generative model represents the target object and ignores the background while the discriminative model trains a classifier to distinguish the target object from the background [15]. The single tracker is designed to track only one target in the sequence, regardless of the number of objects within the sequence. However, multi-object trackers are designed to track all existing objects in a sequence [16, 14]. Offline trackers are trained on specific objects, while online trackers require an initialization phase to learn the target object from the first frame. The target is determined by a bounding box around its region in the image in the first frame. All types of trackers are trained offline [6].

Several methods and algorithms have been explored to solve the problem of object tracking, such as point tracking, kernel tracking, and silhouette tracking. The review of these traditional tracking algorithms is approached in [13]. One of the kernel tracking approaches is tracking-by-detection, which is a discriminative approach that uses a classifier to distinguish between the target object and the background. One cannot mention classification without mentioning deep learning, which has recently shown significant results with respect to this problem [14, 17].

Deep Learning uses deep neural networks, which are artificial neural networks with many layers between the input and output layers [18, 19]. Deep neural networks, particularly Convolutional

Neural Networks (CNN), were successfully used to improve the performance of visual tracking by a wide margin. There are extensive surveys that present many methods utilizing neural networks for object tracking [14, 20, 21]. Currently, VGG-Net is considered the most preferred choice in the literature for extracting features from images [22]. Another recent model has received increasing attention to extract features from images which is called residual learning framework (Res-Net model). Res-Net deals with "Vanishing gradient" problem by using residual blocks [23].

Despite the notable efforts made by researchers in the field of Visual Object Tracking, there is still plenty of room for improvements due to many challenges such as occlusion, background clutter, illumination changes, scale variation, low resolution, fast motion, out of view, motion blur, deformation, and in and out of plane rotation [24, 25].

One emergent technology that has been shown to produce excellent results in various computer vision problems is Generative Adversarial Networks (GANs) [10, 26, 27]. GANs build upon the success of Deep Neural Networks on content generation. The basic idea behind GANs is the use of two deep neural networks, one for content generation and another for classification. The content generation network, usually called the generator network, is used to generate content based on the training set. The classification network, usually called the discriminator network, is used to score the result of the generator. The score is used to update both networks, the generator and the discriminator, until convergence.

The GANs framework is suitable for unsupervised learning tasks. However, the problem of visual tracking is classified as a supervised task. Furthermore, the tracking-by-detection approach depends on a classifier, which is trained using supervised methods. It is possible, however, to adapt

the GANs framework to solve supervised and semi-supervised problems such as object detection, localization and object tracking [28, 29, 30].

In this thesis, we examine the impact of changing the feature extractor on the performance of the tracker framework. We compare two models; one is based on VGG-Net and the other is based on Res-Net. We also explore the impact of using Generative Adversarial Networks (GANs) with state-of-the-art Visual Object Tracking frameworks in addition to our introduced framework which is based on Res-Net. We describe an adaptation of a GAN framework to augment each tracker's architecture and update their algorithms accordingly. The performance of each tracker is first captured without using a generative adversarial network, and then after integrating the generative adversarial network, the performance of each tracker is captured again. Finally, we present our findings in detail and provide a comprehensive analysis of the results.

1.2 MOTIVATION

Visual Object Tracking impact is potentially very significant. Its applications will one day transform the way we live in our homes and build our cities. It is one of the fundamental building blocks for self-driving systems and a major component in various smart systems where automatic content analysis is required.

Deep Learning remains an active area of research due to its promising results in various tasks. The emergent Generative Adversarial Deep Networks continues to attract researchers to explore their potentials across several classes of problems. Although both of these technologies were used to produce better results in visual object tracking, existing frameworks are still limited in several aspects because of the variety of appearance changes over time.

Additionally, there is always an imbalance between available data about the target and available data about the background in each frame, which affects the classification negatively. Furthermore, current methods require high computational load and suffer from low running speed.

In this thesis, we attempt to address these limitations by exploiting the ongoing advancement in Generative Adversarial Networks to integrate this new method into tracking frameworks.

1.3 OBJECTIVES

This thesis investigates the impact of integrating an adopted GANs framework into existing state-of-the-art deep visual trackers. The integrated framework should perform better than the original without an increased computational complexity.

Furthermore, we compare the performance of proposed trackers based on one-pass evaluation protocol for overlap and precision metrices. We trained the trackers on VOT dataset and evaluated their performance on OTB datasets to be consistent with other researches in this field. Additionally, we reinitialized the trackers and trained them on ImageNet-Vid dataset and evaluated them again on OTB datasets.

We provide a qualitative analysis for the most interesting results in addition to a comprehensive analysis for all of the results of our experiments. The results are fine grained in terms of various attributes and discussed to determine the conditions in which each tracker performs the best or worst.

1.4 THESIS OUTLINE

This thesis is divided into five chapters including this introductory chapter. The second chapter presents a literature review to briefly describe the technologies, methods and frameworks mentioned throughout the thesis. In chapter 3, detailed descriptions are given on the proposed frameworks, the adopted generative adversarial framework, and the integration of this GANs framework into various trackers. The chapter includes the architecture, algorithms and notable implementation details. Chapter 4 starts with a description of the environment used to run the experiments, which are subsequently explained. The chapter follows with a detailed analysis of the results, including a comprehensive and qualitative analysis. Finally, we present our conclusions in chapter 5 along with future research interests and a summary of the challenges and difficulties faced throughout this research.

Chapter 2

2. BACKGROUND AND LITERATURE REVIEW

In this chapter, we briefly describe the technologies, methods and frameworks mentioned throughout the thesis.

2.1 OBJECT TRACKING

Object tracking is the process of finding the location of a moving object in a sequence of images. Figure 2.1 illustrates the general model for any tracking algorithm, which includes four stages [31]:

- Target initialization

  In this stage, the initial state of the target is defined by drawing a bounding box around it to estimate the localization of the target in the first frame of the sequence.

- Appearance modelling

  In this stage, the visual features of the target are extracted and represented in a model.

- Motion estimation

  In this stage, the goal is to predict the most probable region that the target will appear in.

- Target localization

  In this stage, the location of the target object is estimated based on the extracted visual features and the predicted region.

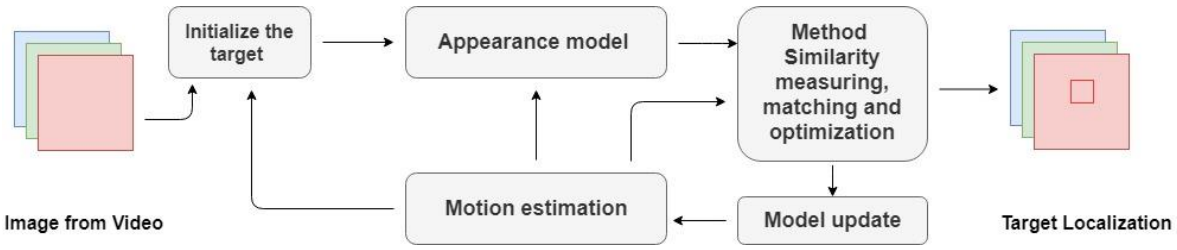


Figure 2.1: The General model for object tracking
Object tracking starts with target initialization to extract useful information for appearance modelling. The appearance model is then used to estimate the motion of the object in order to localize its position in the next frame.

Many traditional algorithms have been proposed to solve the object tracking problem, such as boosting, support vector machine, Naive Bayes, random forest, multiple instance learning, metric learning, structured learning, latent variable learning, correlation filter and others. A review of such methods is available in [14].

Recently, Deep Learning algorithms have achieved impressive success for tracking tasks, following their success in feature extraction and classification tasks. These algorithms consider the tracking task as a classification between two classes: the target and the background. The basic idea is to use a feature extractor and build a classifier to distinguish between the features of the target pixels and the features of the background pixels. This method is called Tracking-by-Detection [21].

The following sections will give a brief introduction about deep learning and some related techniques for performance improvements.

2.2 DEEP LEARNING

Lately, deep learning techniques have been used to make significant progress in artificial intelligence overall and in computer vision in particular, which aims to simulate the functionality of the human eye and the components in the brain responsible for the sense of sight. Deep Learning uses deep neural networks, which are artificial neural networks with several hidden layers between the input and output layers [17]. The term "deep" usually refers to the high number of hidden layers included in the neural network. Usually deep networks can have up to 150 layers. Neural networks are motivated by the goal of building systems that learn and think like humans' brains. Deep Learning models can learn the representation of data and their features by being trained using large sets of labeled data [18].

One of the most popular architectures that are based on deep learning are convolutional neural networks, which are explained next.

2.3 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Network (CNN) architecture is one of the most popular Deep Artificial Neural Networks. It has shown very effective results in areas such as image classification and recognition. It is made up of an input layer, an output layer, and several hidden layers of different types [32]. The Convolutional Neural Network was named due to being built based on the "convolution" operation. The convolution operation for a function f with another function g in the continuous or discrete domain is defined as follows:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x) \cdot g(t - x) dx$$

Equation 1: Convolution operation in the continuous domain

$$(f * g)(t) = \sum_{x=-\infty}^{\infty} f(x) \cdot g(t - x)$$

Equation 2: Convolution operation in the discrete domain

There are several types of layers within a deep network. The following is a description of these types.

- Convolutional layer

  In math, the term convolution refers to the combination of two functions to produce a third function. In the context of convolutional neural networks, the operation of convolution means multiplying a set of weights with an input image to produce a feature map.

- Fully connected layer

  The function of this layer is to aggregate information from previous layers and produce a final classification. Each neuron in this layer takes input from every node of the preceding layer. The output is an N dimensional vector where N is the number of classes.

- Activation layer

  This layer is applied after each convolutional layer to introduce nonlinearity to the system. There are many nonlinear functions such as $tanh$ and $sigmoid$. However, using Rectified Liner Units (ReLU) shows better results because of their ability to train

the network faster in addition to maintaining a good rate of accuracy. The ReLU layer applies the function in Equation 3 to all input values. A good explanation of how this works is provided in [33].

$$f(x) = max(0, x)$$

Equation 3: ReLU activation function

- Pooling layer

  This type of layer is also known as a down-sampling layer. The objective of this layer is to reduce the dimension of the input representation. This also reduces the computational cost by reducing the number of parameters. Additionally, this helps with over-fitting by extracting the abstracted form of the representation. The operation of the pooling layer is to apply max function or avg function using a kernel over the input representation [18, 34].

- Dropout layer

  Dropout is an approach to regularization in neural networks that aims to reduce the interdependent learning and reduce overfitting. This layer drops up a random set of activations from the previous layer by setting them to zero to force the network to learn more robust features [18, 34].

Deep learning frameworks require large datasets for training due to the large number of weights embedded in their architecture and their ability to learn rich features. The following is a description of some of the methods that attempt to address limited amounts of training data.

2.4 DATA AUGMENTATION

Data augmentation is a technique that generates new data from existing limited data. The aim of this technique is to expand the dataset and diversify its attributes in order to improve the performance of the model being trained on it and its ability to generalize and deal with overfitting.

Image data augmentation is the most popular type of data augmentation. The basic idea is to construct transformed versions of images in the training dataset via simple operations from the field of image manipulation, such as translation, flipping, shifting, rotation, and occlusion [35]. The new position for each pixel in the new image (x', y') is a function of its position in the original image (x, y).

Researchers have obtained good results by applying data augmentations techniques. However, the selection of the specific data augmentation technique must be done carefully and with knowledge of the problem category to avoid bad scenarios such as labeling images incorrectly. For example, if we consider a vertical flip of the image of a cat, that will confuse the model because it is unlikely to see a photo of an inverted cat. Another example is that of MNIST dataset [36]; if we apply a rotation to the image "9", it will look like the image "6" with an incorrect label.

Since deep networks require large datasets for training, the training procedure is often long and slow because of the complexity of the architecture. One method to work around this issue is described next.

2.5 TRANSFER LEARNING

Training a deep network on a very large dataset using a significant amount of computing power is an expensive procedure. Transfer learning was introduced as a way of reusing existing representations that were previously learned. It is suitable for tasks related to that for which the original model was trained.

Computer vision tasks commonly extract low-level features from visual data such as edges, shapes, corners and intensity. The representation of such features can be reused across various other kinds of tasks that can benefit from similar features. In other words, we can transfer the weights of these layers to compatible layers of the new model and fine-tune the model based on a dataset specific to the problem at hand. Transfer learning is described in detail in [37, 38].

We have introduced deep networks with their architecture and some of the techniques used to build and train these networks. Next, we will introduce existing state-of-the-art frameworks that are based on deep networks specifically targeted at object detection and object tracking-by-detection.

2.6 OBJECT DETECTION

As mentioned in the past section, deep learning considers the tracking task as a classification task between the target and background [3]. To do so, the tracking method pre-trains the network on object detection datasets. Even though object tracking is different from object detection, both tasks are highly correlated [14, 39] and many tracking algorithms depend on big part of object detection frameworks such as [40] and [41].

Object detection refers to detecting all known objects in an image. By detecting we mean identifying the classes of these known objects and localizing their location within the image. This means that there are two main tasks for object detection:

- Detect the class of the object (classification)
- Detect the position of the object in the image (localization)

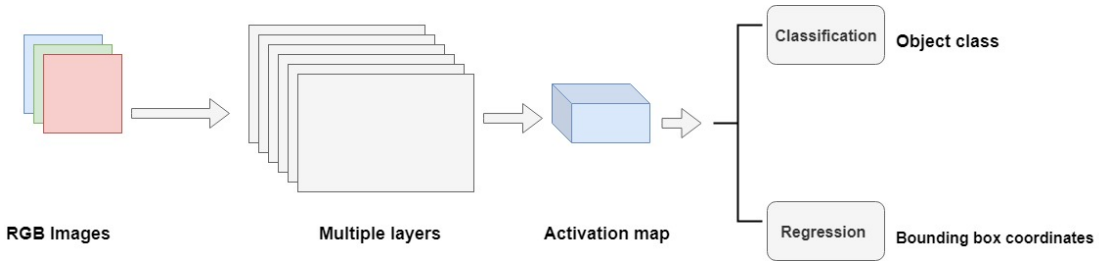


Figure 2.2: Conventional architecture for object detection
Classification task is used to identify the class of the object. Localization task is used to find the location of the object within the image.

The state of the art approaches for object detection can be divided into two categories [3]; Region Proposal Based Frameworks and Regression/Classification Based Frameworks.

Region Proposal Based Framework simulates the mechanism of the human brain by scanning the whole scene first, then concentrating on regions of interest. The following key algorithms are the most popular region proposal-based methods.

- Regions with Convolutional Neural Network (R-CNN)

  This algorithm [42] proposes a set of bounding boxes in the image and checks if these boxes include the target object. It depends on a selective search to extract these candidates' boxes, which represent the Regions of Interest (ROI). Then, all of these regions are reshaped (warped or cropped) and passed to CNN to extract the features of each region. The final classification step is performed using Support Vector Machines

(SVMs).

Finally, a bounding box regression model is used to produce the bounding boxes for each specific region.

- Fast R-CNN

  This algorithm [43] reduces the expensive computations and slowness of the R-CNN algorithm. In this approach, the input image is fed into the CNN once to generate the Regions of Interest. The extracted regions then are reshaped into fixed sizes using ROI pooling layer. Then it can be fed into a fully connected layer to classify the features.

- Faster R-CNN

  Faster-RCNN [44] is the modified version of Fast-RCNN. The main difference between these two algorithms is the method of generating the interests. Faster R-CNN uses Region Proposal Network (RPN) instead of selective search to make the algorithm much faster. This network takes an image of any size as input and computes a set of object proposals.

- Mask R-CNN

  Mask R-CNN [45] adds a new branch of a fully connected layer to Faster R-CNN to address the problem of fine segmentation of the object in the bounding box. The new branch of Fully Convolutional Network concentrates on predicting pixel-level segmentation masks in each region of interest. In addition, Mask R-CNN introduces a new layer called Region of Interest Alignment (RoIAlign) instead of RoI pooling layer to solve the problem of misalignments between the ROI and the extracted features.

Regression/Classification Based Framework takes the whole image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The most common algorithms for this framework are the following.

- YOLO (You Only Look Once)

  In this algorithm, the entire image is pushed as an input. The input is divided into an $S \times S$ grid. Within each cell of this grid, the algorithm predicts a number of bounding boxes and the class probability for each bounding box. YOLO is considered the first real-time object detector that is much faster than other object detection algorithms [46].

- SSD (Single Shot Detector)

  This algorithm is similar to YOLO algorithm. It predicts the bounding boxes and class probabilities once with an end-to-end CNN architecture. The model passes the input image through multiple convolutional layers with several filters. SSD uses a collection of default anchor boxes with various aspect ratios and scales to represent the output space of the bounding box [47].

Popular object detection algorithms were explored in this section. The following section introduces generative adversarial networks and their applications in object tracking.

2.7 Object Tracking With Generative Adversarial Networks

2.7.1 Generative Adversarial Networks (GANs)

One of the recent and most popular methods based on deep learning is called Generative Adversarial Networks. This method builds upon the success of using deep neural networks in content generation. Following the success of using GANs in solving and improving several problems, the process has been described as the best idea in machine learning in recent years [26].

The basic idea behinds GANs is to use two deep neural networks: one for content generation and another for classification. The content generation network, usually called the generator network, is used to generate content based on the training set to test and improve the results of the data analysis network [27, 10].

In the context of computer vision, the generator network is used to generate images and the classification network, usually called the discriminator, is used to analyze the images for the target result. GANs were found to be especially well suited for image generation tasks [27, 48].

The term 'generative' refers to the fact that these networks can learn how to produce data samples that are similar to real ones in the training dataset. The term 'adversarial' refers to the type of optimization procedure used to train the network. The two deep neural networks work against each other in a zero-sum game framework, where the generator attempts to generate a fake image and the discriminator attempts to detect a real one. Successful training of a GAN requires reaching an equilibrium state between two opposing objectives, unlike CNN or Long Short-Term Memory (LSTM) where the training objective is to minimize or maximize the value of a single loss function [27].

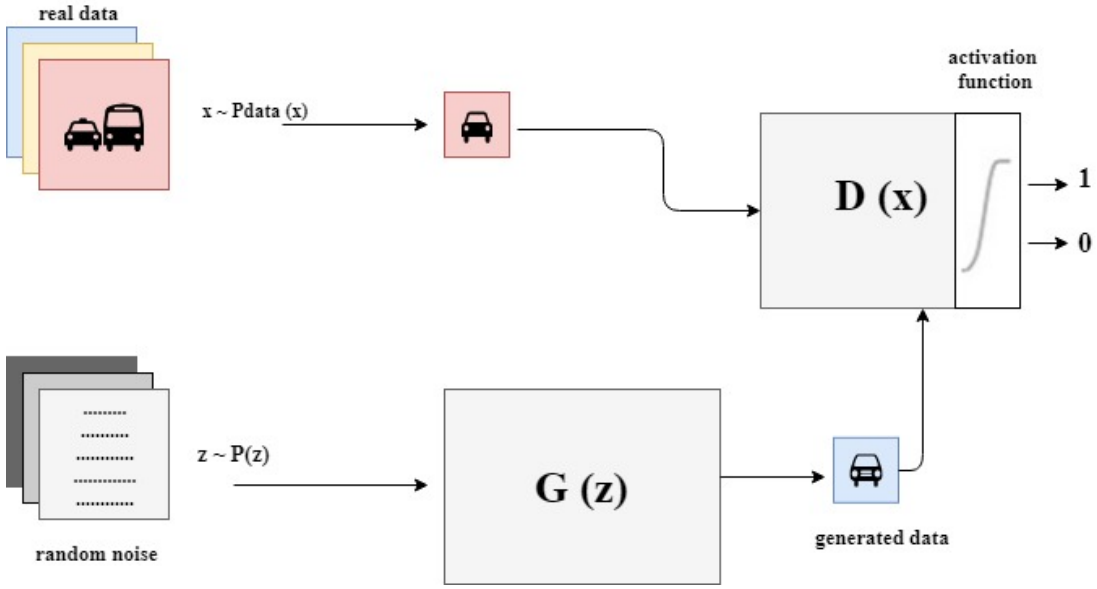


Figure 2.3: GAN framework structure
GAN framework consists of two networks: Discriminator (D) and Generator (G)

GANs can be seen as functions that map from one manifold to another. The discriminator, which is usually constructed as a standard convolutional classifier, takes an image and applies down-sampling techniques, such as max pooling, to generate a probability. The generator recreates an image by taking a random noise vector and up-sampling it to an image [27]. Figure 2.3 shows an overview of a GAN framework. The discriminator $D(x)$ accepts input from either the real

distribution data $Pdata(x)$, or the generator $G(z)$. The generator $G(z)$ accepts input from a random noise distribution $P(z)$ and generates an image as its output. The generated image is fed into the discriminator network $D(x)$, which attempts to classify the image as real or generated by G. The result of the classification is backpropagated to the generator to help it learn how to produce images with a closer representation of the input data [26].

The loss function used in training the networks is formulated as [27]:

$$L_{adv} = \min_{G} \max_{D} E_{x \in X}(\log D(x)) + E_{z \in Z}\left(\log\left(1 - D\big(G(z)\big)\right)\right)$$

Equation 4: Adversarial loss function

GANs are typically used in unsupervised and semi-supervised learning tasks such as image synthesis and image editing, image classification, and image to image translation [26].

However, there is an emerging trend towards using Generative Adversarial Networks with supervised learning tasks using existing objective functions. In the following section, we provide a brief overview of this emerging trend.

2.7.2 Integrating GANs with Object Tracking

The recent success of generative adversarial networks to solve unsupervised tasks has encouraged researchers to consider using the GANs framework to solve supervised and semi-supervised computer vision tasks like object detection, localization, and object tracking. Several attempts have been made to modify the loss function in Equation 4 to introduce various extensions of GANs to narrow the gap between GANs and supervised/semi-supervised learning with notable applications in Object Detection and Object Tracking problems [30, 49, 50, 28, 29]. A summary for many

trackers frameworks that are using GANs to solve some problems related to tracking task is shown in Table 1.

Table 1: Architecture comparison of object tracking using GANs

Framework	**Generator**	**Discriminator**	**Backbone**	**Loss Function**
VITAL [30]	Adaptive feature dropout net	Classification net	VGG M model	$L = L_{adv} + L_{cost\ sensitive}$
SINT++ [51]	Transformation and deformation nets	SINT	VGG model	$L = L_{SINT++}$
TGGAN [52]	Transformation net	Siamese and classification nets	Siamese	$L = L_{adv} + L_{siam}$
AFSL [53]	Adaptive feature dropout net	Classification net	VGG-16 model	$L = L_{AFSL}$
SANT [54]	Similarity or dissimilarity feature generator	Classification net	Siamese	$L = L_{SANT}$

To address the lack of diversity in datasets, [30] introduced the first tracking system that integrates adversarial learning to improve tracking robustness. The system, which is called Visual Tracking via Adversarial Learning (VITAL), generates masks randomly and uses them to drop out input features adaptively in order to simulate a variety of appearance changes. The backbone tracker is CNN tracking-by-detection, which is made up of a feature extractor and a classifier. The discriminator D is the classifier and the generator G is added between the classifier and the feature extractor. The generator augments positive training samples by randomly generating weight masks and applying them to the input features. Each mask represents a particular type of appearance variation in the temporal domain.

In another attempt that also claims to address high computational load and low running speed, [53] presented a tracking framework based on adversarial learning called Adversarial Feature Sampling Learning (AFSL). The general idea is to use one cropped image around the target object and randomly generate masks and use them to drop out input features adaptively to catch an assortment of appearance variations over the temporal domain. The framework uses CNN to produce the feature map from the cropped and resized image. The discriminator D is the classifier and the generator G is added between the classifier and the feature extractor.

Another notable attempt is [51] which presented an approach to improve tracking performance by generating hard positive training samples. The system is called SINT++ based on its backbone tracker which is the Siamese INstance search Tracker (SINT). It also includes two networks that are used to augment input to the tracker to increase its robustness. The first network is the Positive Sample Generation Network (PSGN), which is used to inflate the number of training samples by using a variational auto-encoder (VAE). It augments the input by introducing more variations such as deformation and motion blur. The second network is the Hard-Positive Transformation Network (HPTN), which is trained using deep reinforcement learning technique to occlude the target object with a patch from the background to generate a difficult positive training sample. The Siamese network compares the initial target in the first frame with the candidates in a new frame and elects the most identical one by learned matching function.

To address drift-over-time limitation, [52] proposed the first method of learning the general appearance distribution for a target in a video and a robust Siamese matching network. The system, which is called Task-Guided Generative Adversarial Network (TGGAN), treats the tracking problem as a template matching problem using the Siamese network, and uses conditional GAN architecture as an online adaption strategy to avoid drifting. The Siamese network is considered

the discriminator D, which is prepared to learn a general similarity function between two images. The function extracts the final feature from both low-level features and high-level semantic features and matches the score of these features. The generator G takes a patch from the ground truth in the first frame and random vectors as input. The output consists of the generated templates for the target object. The generated template that best suits the current target appearance is selected for the tracking task.

The Siamese Adversarial Network Tracker (SANT) is proposed in [54]. The network learns the resemblance between the target object and its temporary changes as well as the contrast between the target object and the background. The Siamese network is used to extract features from the input image and the search image. The generator G takes the features produced from the input image and produces either of two sets: either a similar set, which includes features of the object; or a dissimilar set, which includes features of the background. The discriminator D takes both input sets, similar set and dissimilar set, and distinguishes between original and generated images. The output of the discriminator is used to make the generator learn the temporal mapping of the target features to SAN features, which are then used by the Siamese network for tracking. Table 2 presents the performance comparison of these trackers with two metrics: success rate and precision rate. The success rate is the ratio of the frames whose tracked box has more overlap with the ground-truth box than the threshold. Similarly, a precision rate shows the ratio of successful frames whose tracker output is within the given threshold from the ground-truth, measured by the center distance between bounding boxes.

Table 2: Performance comparison of object tracking using GAN

Framework	**Dataset**	**Success Rate**	**Precision Rate**
VITAL [30]	OTB-2013	0.710	0.950
	OTB-2015	0.682	0.917
SINT++ [51]	OTB-50	0.624	0.839
	OTB-100	0.574	0.768
TGGAN [52]	OTB-50	0.576	0.773
	OTB-100	0.618	0.818
AFSL [53]	OTB-2013	0.685	0.928
	OTB-2015	0.661	0.892
SANT [54]	OTB-100	0.654	0.867

Chapter 3

3. METHODOLOGY

3.1 PROBLEM FORMULATION

The focus of this thesis is to build a visual object tracking framework using generative adversarial networks to track any moving target in a sequence without prior training on the target. The tracker is *monocular*, *single-object*, *casual*, *model-free*, and *short-term* [31]. The meanings of these terms are as follows:

- Monocular: The sequence comes from a single camera.
- Single-object: The tracker can only track one object at a time.
- Casual: The tracker does not have any knowledge of future frames while handling the current frame.
- Model-free: The model does not track a particular target. Instead, it is given a bounding box around the target object in the first frame of the sequence.
- Short-term: If the tracker loses the object in the current frame, it is not possible to redetect the object. Instead, this is considered a failure.

Various tracking-by-detection were studied recently with excellent results. However, their performance remains restricted by the imbalance between positive and negative samples and the high spatial overlap between positive samples.

Those trackers focused on capturing and emphasizing the most discriminative features for positive samples. However, since the framework tracks a target in a sequence of images, this target will move, and the most discriminative features will change frame by frame. Although the algorithms used try to mitigate this issue by retraining the classifier online, the classifier's performance degrades over time owing to the changing features of the target object, including occlusion, scaling, rotation and blur.

In this thesis, we study the impact of using a generative adversarial network to augment the positive samples used to train the classifier online. The augmentation is done by preparing different masks to occlude some features from the feature map to encourage the classifier to increase its ability to detect the target.

3.2 THESIS ARCHITECTURE

The scheme of this thesis is as follows:

- **Experiment 1:**
    1. Evaluate benchmark tracking framework that fits the description in section 3.1, in order to establish a baseline for comparison. The chosen framework is Multi-Domain Convolutional Neural Network (MDNet) [40].
    2. Evaluate another benchmark framework that fits the description in section 3.1, which is Real-Time Multi-Domain Convolutional Neural Network (RT-MDNet) [41].
- **Experiment 2:**

1. Modify the architecture of the baseline framework from the first tracker in the experiment 1 by inserting a generative adversarial network and jointly training it with the classifier online [30]. We will refer to this framework as MDGaNet (Multi-Domain Generative Adversarial Neural Network).
2. Modify the architecture of the framework RT-MDNet from experiment 1 by inserting a generative adversarial network and jointly training it with the classifier online. We will refer to this network as ROIAL-MDNet (Region of Interest and Adversarial Learning for a Multi-Domain Neural Network).

- **Experiment 3:**

1. Modify the architecture of the baseline framework the first tracker in the experiment 1 by changing the backbone network from VGG-M to ResNet-50 [55]. We will refer to this framework as MDResNet (Multi-Domain Deep Residual Neural Network).
2. Modify the architecture of the framework MDResNet from experiment 3 by inserting a generative adversarial network and jointly training it with the classifier online. We will refer to this framework as MDResGaNet (Multi-Domain Deep Residual Generative Adversarial Neural Network).

We will compare and discuss the results of all the aforementioned frameworks.

The network architecture, offline training algorithm, and online tracking algorithms of each of these frameworks are explained with relevant illustrations in the following sections. The scheme of this thesis is summarized in Figure 3.1

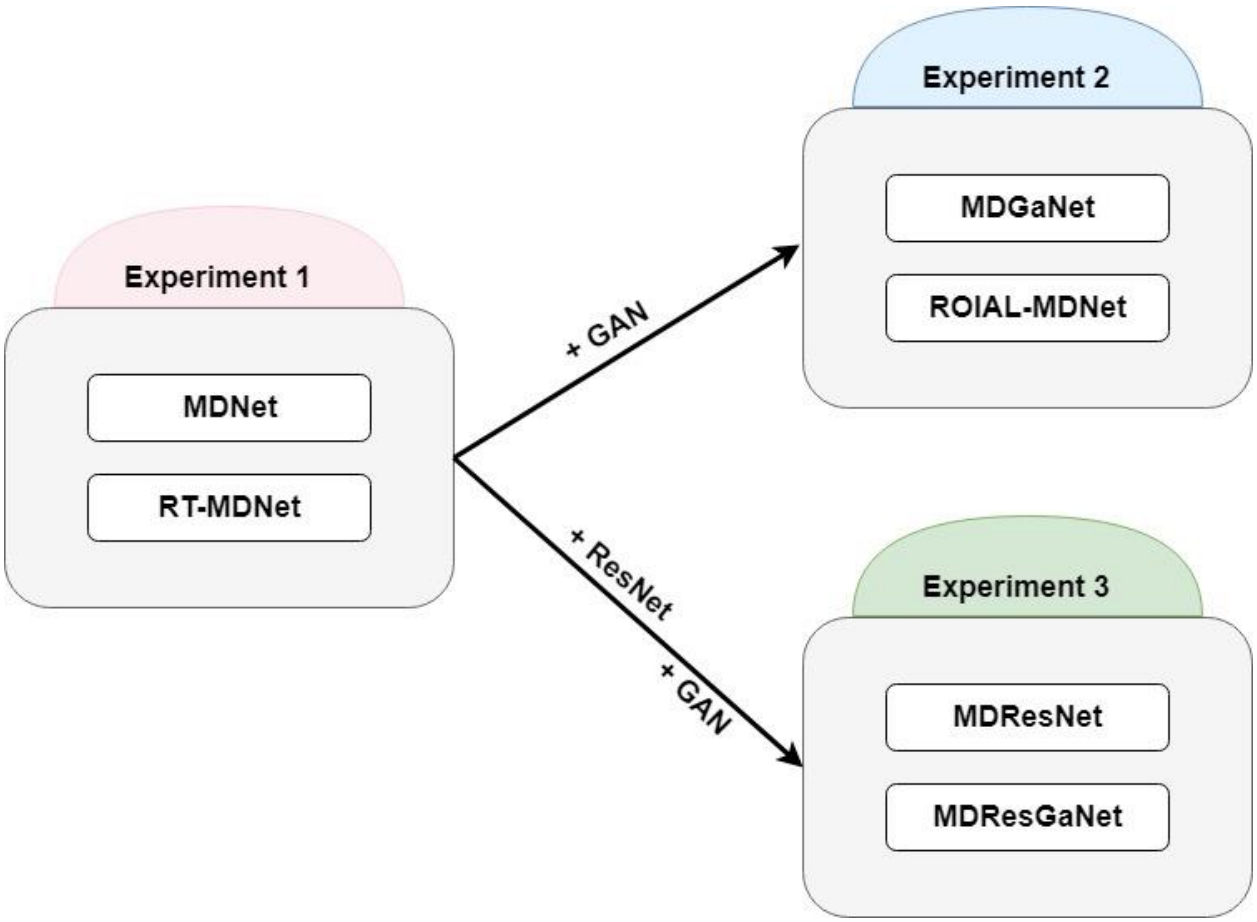


Figure 3.1: Work scheme for the experiments
Experiment1 consists of evaluating benchmark trackers: MDNet and RT-MDNet. Experiment 2 shows integrating GAN with benchmark trackers. Finally, experiment 3 includes modifying MDNet architecture and integrating the new tracker with GAN.

3.3 EXPERIMENT 1

In this experiment, we evaluate two benchmark frameworks of the type of tracker explained in section 3.1.

3.3.1 MDNet Tracker

MDNet [40] uses a convolutional neural network (CNN) to obtain a generic representation of the target object and distinguish between the target and the background. The pre-training process uses a large set of annotated videos. Each video in the training set is associated with a file that includes the coordinates of the bounding box for the annotated target in each frame.

The fundamental problem with free-model object trackers is that the target in one video could be the background in another, which would and does confuse the convolutional neural network. The MDNet tracker framework mitigates this problem by proposing a network architecture that consists of two parts: shared layers and branches of domain-specific layers. We chose MDNet to be the baseline for our experiments due to being the state-of-the-art visual tracker based on a CNN trained on a large set of tracking sequences, and the winner tracker of The VOT2015 Challenge.

The performance for MDNet comparing to other frameworks shows in Figure 3.2 The state-of-the-art for Visual Object TrackersFigure 3.2.

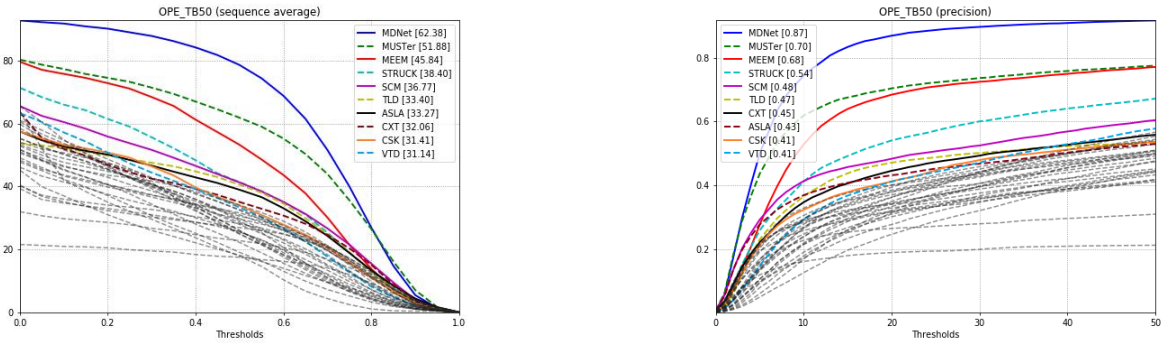


Figure 3.2 The state-of-the-art for Visual Object Trackers

3.3.1.1 MDNet Network Architecture

MDNet architecture consists of three main parts: the convolutional part, the fully connected part and the domain-specific part. Figure 3.3 depicts the architecture of this framework.

- The convolution part consists of three layers (conv1, conv2, conv3) that are similar to the first three layers of the Visual Geometry Group (VGG-M) network. Transfer learning is used to initialize the convolutional layers from a pre-trained VGG-M model.
- Each of the two fully connected layers (fc4, fc5) is combined with rectified linear units (ReLU) and dropouts.
- The last fully connected layer (fc6) has as many branches as the number of training sequences. Each branch contains a binary classification layer with SoftMax cross entropy loss function.

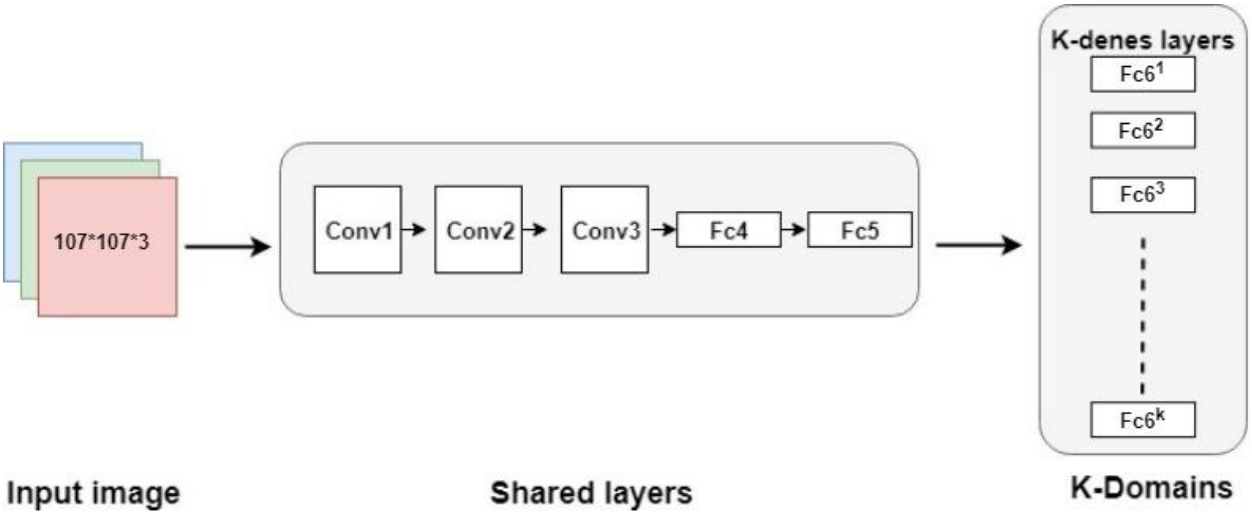


Figure 3.3: MDNet architecture
It includes the shared layers, three convolutional layers and two fully connected layers, and k-domains part which consists of k branch of fully connected layers.

3.3.1.2 MDNet Offline Learning Algorithm

Each domain in the training set is considered an independent training sequence. The network is therefore trained iteratively over K domains, where for each domain it learns to classify between the target object and the background using a binary classification layer with SoftMax cross entropy

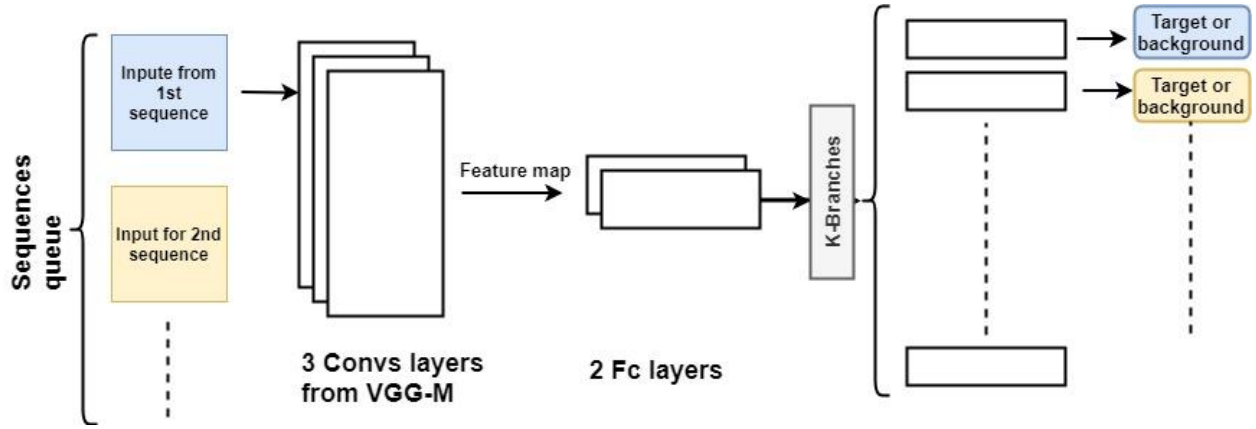


Figure 3.4: Offline learning process for tracking with MDNet
Each domain in the training dataset is considered an independent training sequence. All layers are updated based on the classification score of each domain independently to learn the generic representation of all objects in the dataset.

loss as shown in Equation 5. The weights of the shared layers are updated during the training of each domain, which helps the network learn the generic representation of the domains.

$$\mathcal{L} = -(\mathrm{y}\log(\mathrm{p}) + (1-\mathrm{y})\log(1-\mathrm{p}))$$

Equation 5: SoftMax cross entropy for visual object tracker

Figure 3.4 illustrates the offline learning process.

3.3.1.3 MDNet Online Finetuning and Tracking Algorithm

In the tracking phase, the domain specific binary layers are removed and replaced with one binary classification layer. The last binary classification layer is trained online on the domain at hand while also fine-tuning the weights for the preceding two fully connected layers.

To determine the bounding box of the object in the current frame, N candidates are sampled around the bounding box of the previous frame $x^1, \dots, x^N$. The candidates are then evaluated using the network to obtain their positive scores $f^+(x^i)$ and negatives scores $f^-(x^i)$. The candidate with the maximum positive score is selected:

$$x^* = \underset{x^i}{\operatorname{argmax}} f^+(x^i)$$

Equation 6: Candidate selection

A bounding box regression model is used to predict the accurate location of the target using the features of the third convolution layer. All steps of the online tracking are shown as algorithm 1.

Algorithm 1: MDNet Online Training and Tracking

Input: Pretrained convolutional layers $\{w_1, w_2, w_3\}$ and two dense layers $\{w_4, w_5\}$

Initial bounding box x_1

Output: Estimated bounding boxes x_t^*

01. Randomly initialize the last binary classification layer w_6
02. Train a bounding box regression model using w_3 output from x_1
03. Generate positive samples S_1^+ and negative samples S_1^- from x_1
04. Fine-tune $\{w_4, w_5, w_6\}$ using the samples S_1^+ and S_1^-
05. Keep track of recent positive samples: $\mathrm{T}_s \leftarrow \{1\}, \mathrm{T}_l \leftarrow \{1\}$
06. Repeat for every frame $t > 1$
07. Generate candidates $x_t^i, i \in \{1, .., N\}$
08. Repeat for every candidate x_t^i
09. Get the convolutional features and their scores $f^+(x_t^i)$

```
Find best candidate x_t^* using Equation 6
If f^+(x_t^*) > 0.5 then
    Generate S_t^+ and S_t^-
    T_s ← T_s ∪ {t}, T_l ← T_l ∪ {t}
    If |T_s| > τ_s then T_s ← T_s \ {min_{v∈T_s} v}
    If |T_l| > τ_l then T_l ← T_l \ {min_{v∈T_l} v}
    Adjust x_t^* using bounding box regression
If f^+(x_t^*) < 0.5 then
    Update {w_4, w_5, w_6} using S^+_{v∈T_s} and S^-_{v∈T_s}
else if t mod 10 = 0 then
    Update {w_4, w_5, w_6} using S^+_{v∈T_l} and S^-_{v∈T_l}
```

3.3.2 RT-MDNet Tracker

MDNet tracker, which is inspired by R-CNN, computes the convolutional features and their score for each candidate independently. This is not optimal due to redundant computations caused by the overlap between proposed candidates.

RT-MDNet tracker [41], which is inspired by Fast R-CNN, improves the situation by using the proposed candidates to crop the image into compatible patches, compute a shared convolutional feature map for these patches, and use a Region of Interest Alignment layer to extract relevant features for each original candidate from the shared feature map.

Unlike Fast R-CNN, which uses Region of Interest Pooling (RoIPooling), RT-MDNet uses Region of Interest Alignment (RoIAlign) via bilinear interpolation to achieve better object localization.

3.3.2.1 RT-MDNet Network Architecture

Similar to MDNet architecture, RT-MDNet architecture starts with three convolutional layers and ends with three dense layers. RT-MDNet inserts an additional RoIAlign layer between the convolutional and dense layers. Additionally, the max pooling between the second and third convolutional layers is removed to retain high resolution information, which is useful to the RoIAlign layer. Figure 3.5 illustrates RT-MDNet architecture.

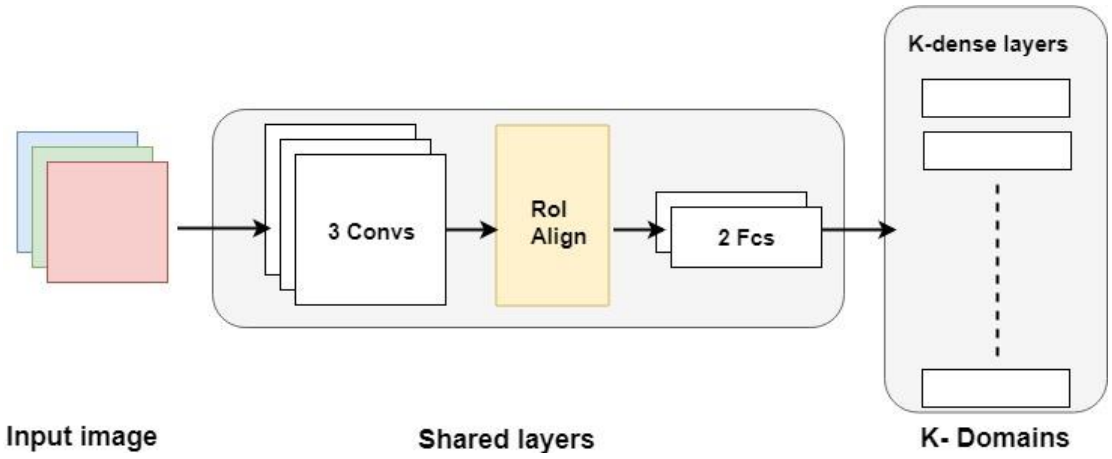


Figure 3.5: RT-MDNet architecture
RoIAlign layer is added between the third conv layer and the first fully connected layer

3.3.2.2 RT-MDNet Offline Learning Algorithm

The learning procedure for RT-MDNet is implemented in exactly the same way as that of MDNet.

3.3.2.3 RT-MDNet Online Finetuning and Tracking Algorithm

RT-MDNet reduces the computational complexity by avoiding redundant computations due to the overlap between target candidates. In MDNet, each candidate x^i is processed independently. In RT-MDNet, the candidates are used to crop the shared parts of the image and compute convolutional features of these parts to produce a shared feature map. RoIAlign is then used to

extract each candidate's corresponding features from the shared feature map. Finally, the best candidate is selected using Equation 6.

The online tracking algorithm for RT-MDNet is described as algorithm 2.

Algorithm 2: RT-MDNet Online Training and Tracking

Input: Pretrained convolutional layers $\{w_1, w_2, w_3\}$ and two dense layers $\{w_4, w_5\}$

Initial bounding box x_1

Output: Estimated bounding boxes x_t^*

Randomly initialize the last binary classification layer w_6

Train a bounding box regression model using w_3 output from x_1

Generate positive samples S_1^+ and negative samples S_1^- from x_1

Fine-tune $\{w_4, w_5, w_6\}$ using the samples S_1^+ and S_1^-

Keep track of recent positive samples: $\mathrm{T}_s \leftarrow \{1\}, \mathrm{T}_l \leftarrow \{1\}$

Repeat for every frame $t > 1$

    Generate candidates $x_t^i, i \in \{1, .., N\}$

    $t' \leftarrow$ crop frame t using positional data from x_t^i

    Split t' into patches of compatible size with the model

    Get the convolutional features F of t'

    Using F and x_t^i, extract each candidate's features using RoIAlign

    Get each candidate's score f^+

    Find best candidate x_t^* using Equation 6

    If $f^+(x_t^*) > 0.5$ then

        Generate S_t^+ and S_t^-

        $\mathrm{T}_s \leftarrow \mathrm{T}_s \cup \{t\}, \mathrm{T}_l \leftarrow \mathrm{T}_l \cup \{t\}$

        If $|\mathrm{T}_s| > \tau_s$ then $\mathrm{T}_s \leftarrow \mathrm{T}_s \setminus \left\{\min_{v \in \mathrm{T}_s} v\right\}$

```
        If |T_l| > τ_l then T_l ← T_l \ {min_{v∈T_l} v}
        Adjust x_t^* using bounding box regression
    If f^+(x_t^*) < 0.5 then
        Update {w_4, w_5, w_6} using S^+_{v∈T_s} and S^-_{v∈T_s}
    else if t mod 10 = 0 then
        Update {w_4, w_5, w_6} using S^+_{v∈T_l} and S^-_{v∈T_l}
```

3.4 EXPERIMENT 2

In this experiment, we modify the architecture of the trackers that explained in experiment 1 by inserting a generative adversarial network and jointly training it with the classifier online [30].

3.4.1 MDGaNet Tracker

By considering the last three dense layers in MDNet to be a discriminator network D as described in section 2.7.1, we can design a generator network G [30] to improve the performance of the tracker's classifier during online tracking and learning.

The generator learns a weight mask that gradually identifies the discriminative features through adversarial learning. It starts with several random masks; each one represents a specific type of appearance variation. By applying the mask to the feature space before forwarding the features to the classifier, the generator G gradually identifies the features that degrade the classifier the most. At the same time, the classifier D is gradually updated using more robust features over a long temporal span.

$$\mathcal{L} = \min_{G} \max_{D} \mathbb{E}_{(C,M)\sim P(C,M)}(\log D(M \cdot C)) + E_{C\sim P(C)}\big(\log\big(1 - D(G(C) \cdot C)\big)\big)$$

$$+ \lambda \mathbb{E}_{(C,M)\sim P(C,M)} \big|\big| G(C) - M \big|\big|^2$$

Equation 7: Adversarial loss with object tracking

The symbols in Equation 7 are explained in the following table.

Table 3: Symbol meanings in adversarial loss with object tracking equation

Symbol	**Meaning**
$\boldsymbol{\mathcal{L}}$	Cost sensitive loss value
$\boldsymbol{G}$	Generator network
$\boldsymbol{D}$	Discriminator network
$\boldsymbol{C}$	Input
$\boldsymbol{M}$	Optimal mask
$\boldsymbol{G(C)}$	Predicted mask
$\boldsymbol{\lambda}$	Constant
$\boldsymbol{P(C,M)}$	Distribution of input and optimal masks
$\boldsymbol{P(C)}$	Distribution of input
$\mathbb{E}_{(\boldsymbol{C},\boldsymbol{M})}$	The entropy of the discriminator classifying the input after applying the mask
$\mathbb{E}_{\boldsymbol{C}}$	The entropy of the discriminator classifying the input

3.4.1.1 MDGaNet Network Architecture

The architecture of MDGaNet is the same as MDNet, but has an additional generator network between the convolutional networks and the dense networks. The generator network G is made up of two fully connected layers combined with dropouts and a rectified linear unit between the two.

During online training, the feature map that is produced by the convolutional networks is augmented using the predicted mask that is generated by the generator network, which is then fed into the classifier. However, during the tracking phase, the feature map is fed into the classifier untouched by the generator. Thus, the role of the generator network is to improve the finetuning process of the classifier during online learning.

Figure 3.6 illustrates the architecture of MDGaNet.

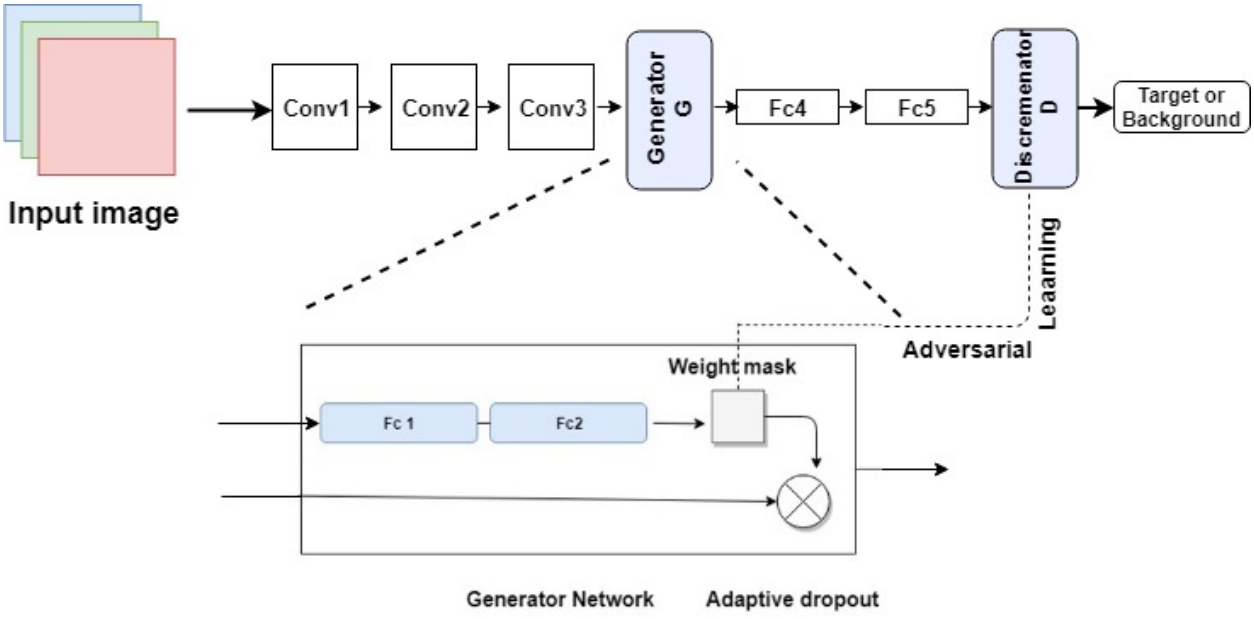


Figure 3.6: MDGaNet architecture
It contains three convolutional layers, the generator (G), two fully connected layers and the discriminator (D) which is the classification layer

3.4.1.2 MDGaNet Offline Learning Algorithm

The implementation is exactly the same as the MDNet framework. The generator network is not used during offline training, but rather during online finetuning of the classifier.

3.4.1.3 MDGaNet Online Fine-tuning and Tracking Algorithm

MDGaNet algorithm is a modified version of the MDNet algorithm as shown in algorithm 3. It adds three additional steps: initializing the generator network G, augmenting the feature map during online tracking, and jointly training the generator and the classifier D during online training.

Upon generating the positive samples S^+, the generator network is initialized and pretrained on these samples. To pretrain the generator, a set of random masks is generated and applied to the positive features iteratively to find the optimal mask M, which is the one that produces the lowest score from the classifier. The generator is then iteratively updated to learn M.

To update G, the positive features C are forwarded through G to obtain the predicted optimal mask $G(C)$, which is then applied to the input before forwarding the result $G(C) \cdot C$ through the classifier D to obtain the positive score $f^+(G(C) \cdot C)$, which is then used along with the score $f^+(M \cdot C)$ to update G using Equation 7.

During online training, the same process is used to jointly update G as well as D. To Update D, the augmented features $G(C) \cdot C$ are used to obtain the classifier's score instead of just C.

Algorithm 3: MDGaNet Online Training and Tracking

Input: Pretrained convolutional layers $\{w_1, w_2, w_3\}$ and two dense layers $\{w_4, w_5\}$

Initial bounding box x_1

Output: Estimated bounding boxes x_t^*

01. Randomly initialize the last binary classification layer w_6
02. Randomly initialize the generator network G
03. Train a bounding box regression model using w_3 output from x_1
04. Generate positive samples S_1^+ and negative samples S_1^- from x_1
05. Fine-tune $\{w_4, w_5, w_6\}$ using the samples S_1^+ and S_1^-
06. Pretrain G using S_1^+
07. Keep track of recent positive samples: $\mathrm{T}_s \leftarrow \{1\}, \mathrm{T}_l \leftarrow \{1\}$
08. Repeat for every frame $t > 1$
09. Generate candidates $x_t^i, i \in \{1, .., N\}$
10. Find best candidate x_t^* using Equation 6
11. If $f^+(x_t^*) > 0.5$ then
12. Generate S_t^+ and S_t^-
13. $\mathrm{T}_s \leftarrow \mathrm{T}_s \cup \{t\}, \mathrm{T}_l \leftarrow \mathrm{T}_l \cup \{t\}$
14. If $|\mathrm{T}_s| > \tau_s$ then $\mathrm{T}_s \leftarrow \mathrm{T}_s \setminus \left\{\min_{v \in \mathrm{T}_s} v\right\}$
15. If $|\mathrm{T}_l| > \tau_l$ then $\mathrm{T}_l \leftarrow \mathrm{T}_l \setminus \left\{\min_{v \in \mathrm{T}_l} v\right\}$
16. Adjust x_t^* using bounding box regression
17. If $f^+(x_t^*) < 0.5$ then
18. Update $\{w_4, w_5, w_6\}$ using $S^+_{v \in \mathrm{T}_s}$ and $S^-_{v \in \mathrm{T}_s}$
19. else if t mod 10 = 0 then
20. Update $\{w_4, w_5, w_6\}$ using $S^+_{v \in \mathrm{T}_l}$, $S^-_{v \in \mathrm{T}_l}$ and G
21. Update G using $S^+_{v \in \mathrm{T}_l}$

3.4.2 ROIAL-MDNet Tracker

The impact of the generative adversarial learning is also studied on the tracker presented in section 3.3.3.

3.4.2.1 ROIAL-MDNet Network Architecture

In this tracker we combine the architecture of the RT-MDNet tracker presented in section 3.3.2 with the generative adversarial network. The generator network G is inserted between the alignment layer RoIAlign and the classification layer in order to improve the classification performance.

Figure 3.7 illustrates the architecture of ROIAL-MDNet.

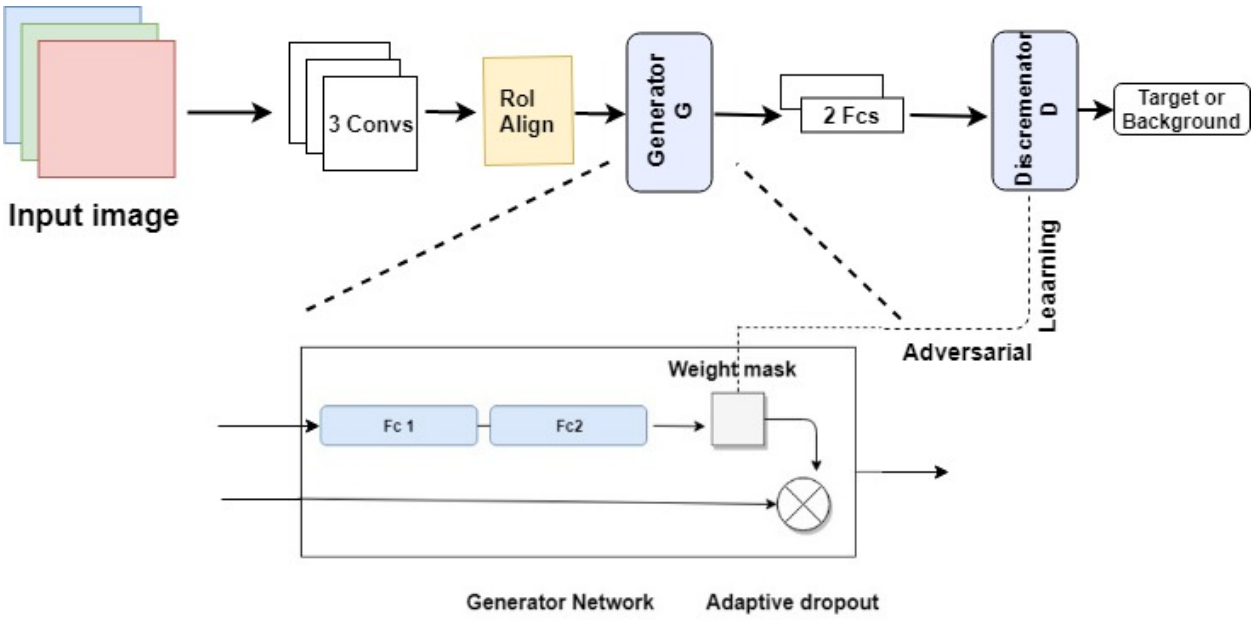


Figure 3.7: ROIAL-MDNet architecture
It starts with the convolutional layers, followed by RoIAlign layer, the generator (G), two fully connected layers, and the classifier (discriminator (D)).

3.4.2.2 RIOAL-MDNet Offline Learning Algorithm

The implementation is identical to that of the MDNet tracker presented in section 3.3.1.2.

3.4.2.3 ROIAL-MDNet Online Fine-tuning and Tracking Algorithm

The implementation is similar to that of MDGaNet presented in section 3.3.4.3. The generator is pretrained on the first frame's positive samples to learn an optimal mask. During online training, the positive features are augmented using the predicted mask of G to train D, and at the same time, G is fine-tuned using newer positive samples. Algorithm 4 is described the online tracking for ROIAL-MDNet tracker.

Algorithm 4: ROIAL-MDNet Online Training and Tracking

Input: Pretrained convolutional layers $\{w_1, w_2, w_3\}$ and two dense layers $\{w_4, w_5\}$
Initial bounding box x_1

Output: Estimated bounding boxes x_t^*

Randomly initialize the last binary classification layer w_6
Train a bounding box regression model using w_3 output from x_1
Generate positive samples S_1^+ and negative samples S_1^- from x_1
Fine-tune $\{w_4, w_5, w_6\}$ using the samples S_1^+ and S_1^-
Pretrain G using S_1^+
Keep track of recent positive samples: $\mathrm{T}_s \leftarrow \{1\}, \mathrm{T}_l \leftarrow \{1\}$
Repeat for every frame $t > 1$
    Generate candidates $x_t^i, i \in \{1, .., N\}$
    Get the shared feature map of x_t^i
    Extract each candidate's features using RoIAlign
    Get each candidate's score f^+
    Find best candidate x_t^* using equation {}
    If $f^+(x_t^*) > 0.5$ then
        Generate S_t^+ and S_t^-

15. $T_s \leftarrow T_s \cup \{t\}, T_l \leftarrow T_l \cup \{t\}$

16. If $|T_s| > \tau_s$ then $T_s \leftarrow T_s \setminus \left\{\min_{v \in T_s} v\right\}$

17. If $|T_l| > \tau_l$ then $T_l \leftarrow T_l \setminus \left\{\min_{v \in T_l} v\right\}$

18. Adjust x_t^* using bounding box regression

19. If $f^+(x_t^*) < 0.5$ then

20. Update $\{w_4, w_5, w_6\}$ using $S^+_{v \in T_s}$ and $S^-_{v \in T_s}$

21. else if t mod 10 = 0 then

22. Update $\{w_4, w_5, w_6\}$ using $S^+_{v \in T_l}$, $S^-_{v \in T_l}$ and G

23. Update G using $S^+_{v \in T_l}$

3.5 EXPERIMENT 3

In this experiment, we modify the architecture of MDNet tracker by changing the backbone network from VGG-M to ResNet-50. Then, inserting a generative adversarial network to the new tracker.

3.5.1 MDResNet Tracker

We construct this tracker by replacing the VGG-M convolutional layers of the MDNet tracker with three layers from Deep Residual Network ResNet-50. ResNet [55] was recently introduced as a deeper network that attempts to avoid the problem of vanishing gradients. It has shown excellent results in the image classification problem [23].

3.5.1.1 MDResNet Network Architecture

The architecture is similar to MDNet. It consists of the first three layers of ResNet-50, followed by two dense layers, and finally, a classification layer. We connect ResNet layers to the dense layers using a max pooling layer to encourage the dense layers to learn the generic features. Transfer learning is used to initialize ResNet layers using a pretrained model.

Figure 3.8 illustrates the architecture of MDResNet in comparison to the architecture of MDNet.

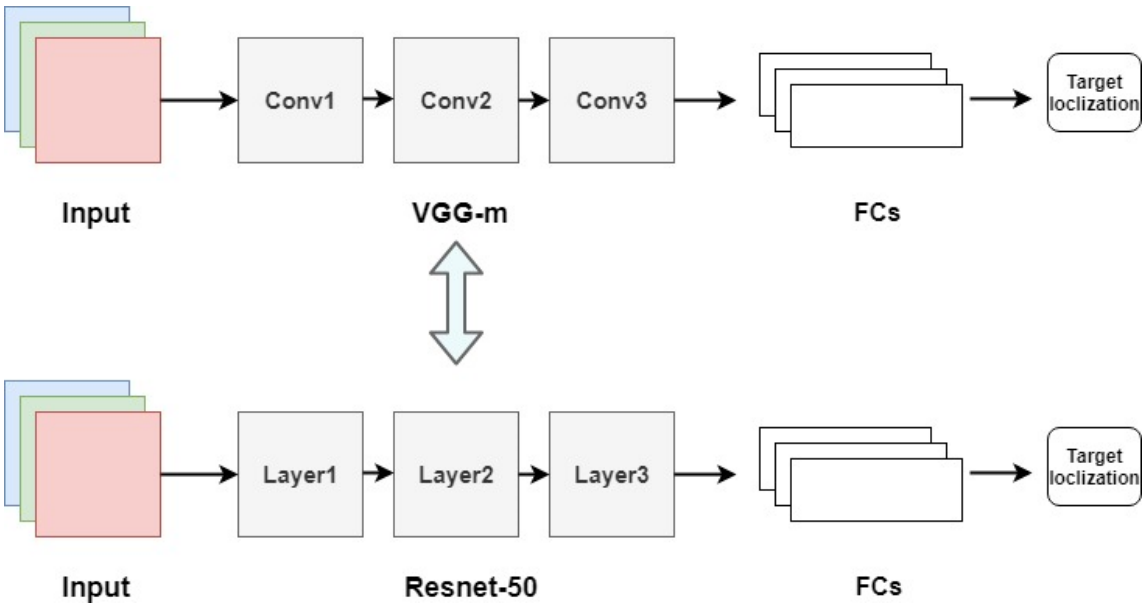


Figure 3.8: MDResNet compared to MDNet
MDResNet consists of three layers from pre-trained resnet-50, max pooling layer and three fully connected layers

3.5.1.2 MDResNet Offline Training

We use the same offline training implementation used in MDNet, and set the classification layer to include as many layers as the domains in the training set. Each sequence is considered an independent domain with its dedicated classifier. Based on the result of each classifier during the

iterative learning procedure, all the weights of the precedent layers are updated to obtain a generic representation of the various target objects in the training set.

3.5.1.3 MDResNet Online Learning and Tracking

Similar to MDNet, the classification layer in this phase is replaced with a single binary classifier. The classifier's weights are randomly initialized at the beginning and the whole network is fine-tuned on the target object specified in the first frame. The algorithm is identical to that of the MDNet tracker.

Since ResNet is able to capture richer features due to its deeper architecture, we expect this tracker to produce better average results across multiple domains and various conditions.

3.5.2 MDResGaNet Tracker

To study the impact of the generative adversarial framework on the tracker suggested in section 3.4.1, we present this tracker, Multi-Domain Deep Residual and Generative Adversarial Neural Network.

3.5.2.1 MDResGaNet Network Architecture

We update the suggested architecture in section 3.5.1 to augment it with a generative network that follows the same design presented in section 3.4.1 A generator network G is added between the residual layers and the classifier. The network G is made up of two fully connected layers combined with dropouts and a rectified linear unit between them.

3.5.2.2 MDResGaNet Offline Learning Algorithm

The implementation is exactly the same as that of the MDNet framework.

3.5.2.3 MDResGaNet Online Learning and Tracking Algorithm

Online learning and tracking implementation is identical to that of MDGaNet presented in section 3.4.1.

Chapter 4

4. EXPERIMENTS

In this chapter, we start by describing the environment and tools used to implement and run the experiments. This is followed by the datasets used for training and the evaluation of the proposed trackers in section 3.3. Finally, we present our findings and discuss the results.

4.1 ENVIRONMENT

In this section we will explain the software stack that we used to implement our experiments. In addition to describe the hardware stack.

4.1.1 Software Stack

To implement and evaluate our proposed trackers, we used the following technologies:

- Python 3.7 and a collection of libraries such as NumPY, SciPy, and OpenCV.
- PyTorch 1.0
- CUDA

A complete and detailed list of all Python dependencies used throughout our experiments is provided in Appendix A.

For implementations, we rewrote available implementations [56, 57] to be compatible with Python3 and the benchmarking framework.

4.1.2 Hardware Stack

Due to the lack of physical access to a proper workstation, we used the cloud services of Compute Canada in order to rent the necessary computational resources to run our experiments and produce the results.

Compute Canada [58] enables Canadian researchers to use Advanced Research Computing (ARC) strategies and access the computing power they need.

Once our account with Compute Canada was set up, the instructions available through the Compute Canada website were used to prepare our software stack using the requirements detailed in Appendix A.

For each of our experiments, one or more files were prepared to describe the computational resources we required from Compute Canada, and the programs we wanted to execute. Each one of these files is then used to schedule a job on a node on one of the clusters of Compute Canada. The results of these jobs are then written to disk and made available to download or for further processing.

The following hardware specifications were used to run each experiment:

- 1 GPU: NVIDIA P100-PCIE-12GB
- 16GB of RAM
- 1 CPU: Intel E5-2650 v4 Broadwell @ 2.2GHz

The following template was used to write the job description files.

```
#!/bin/bash
#SBATCH --acount=my-account
#SBATCH --gres=gpu:1
#SBATCH --mem=16G
#SBATCH --time=0-10:00
module load python/3.7 cuda
source ~/ENV/bin/activate
cd ~/trackers
python run_tracker.py -t MDNet -s tb100
```

4.2 DATASETS

4.2.1 Training Dataset

Two datasets were used to train the proposed trackers: VOT [1] and ImageNet-Vid [59].

- VOT 13, 14, and 15, excluding videos in OTB 2015 dataset. The total number of videos in these sets is 58. The sequences include various objects with challenging backgrounds. It is an adequately small, diverse, and well annotated dataset
- ImageNet-Vid 2015. This dataset is designed for object detection from video. It contains 3862 fully annotated sequences in 30 object categories. It is a huge dataset compared to VOT.

4.2.2 Testing Dataset

To evaluate each tracker, separate datasets are used in order to be consistent with other researchers in this field. These datasets are commonly used by researchers to evaluate and compare the results of their frameworks with each other. The datasets are:

- TB100: contains 100 sequences.
- TB50: contains 50 sequences.
- CVPR13: contains 50 sequences.

Each sequence in these datasets is tagged with specific attributes to describe the type of challenges contained in the sequence. These attributes are explained in Table 4 as seen in [24].

Table 4: Benchmark attributes

Attribute	**Description**
IV	Illumination Variation: illumination in the target region is significantly changing
SV	Scale Variation: the ration of the bounding box in the first frame and the current frame is out of range [1.5, 2]
OCC	Occlusion: the object is partially or fully occluded
DEF	Deformation: non-rigid object deformation
MB	Motion Blur: the target region is blurred due to motion of the target or camera
FM	Fast Motion: the motion of the ground truth is larger than 20 pixels
IPR	In-Plane Rotation: the target rotates in the image plane
OPR	Out-of-Plane Rotation: the target rotates out of the image plane
OV	Out-of-View: some portion of the target object leaves the view

BC	Background Clutter: the background near the target region includes colors or textures similar to the target object
LR	Low Resolution: the number of pixels in the ground truth bounding box is less than 400

4.3 EVALUATION METRICS

To evaluate the performance of the proposed trackers, we use the Online Tracking Benchmark (OTB) protocol [24]. In this protocol, the tracker is initialized with one bounding box in the initial frame. After that, the tracker is run through the whole sequence of the video to calculate a One-Pass Evaluation metric (OPE) for the success rate or precision rate.

The success rate is calculated based on bounding box overlap using the Area under Curve (AUC) score to show the performance of the tracker. The overlap score is defined as:

$$S = \frac{|r_t \cap r_g|}{|r_t \cup r_g|}$$

Equation 8: Overlap score

Where r_t is the tracked bounding box and r_g is the ground truth bounding box. $\cap$ and $\cup$ are the intersection and union of two regions respectively. $|r|$ denotes the number of the pixels in the region r.

The precision plots are computed by the center location error to measure the difference between the predicted center location and the ground truth center location in one frame. The distance is calculated as the average Euclidean distance.

Additionally, we measure the tracker's speed at processing each frame by calculating the duration from the input image to producing the bounding box. The speed is calculated using frames per second metric:

$$S = \frac{\sum x_t}{|T|}$$

Equation 9: FPS score

Where x_t is the number of seconds it took the tracker to process frame t, and T is the set of frames. $|T|$ denotes the number of frames in T and $t \in T$.

4.4 TRACKER PARAMETERS

All trackers were updated using Stochastic Gradient Descent (SGD) method. For offline representation learning, we trained all trackers for 500 epochs. The learning rate was set to 10^{-3} with 0.9 and 5×10^{-3} for momentum and weight decay respectively.

The trackers were initialized on the first frame using 50 iterations with a learning rate set to 10^{-3}, and finetuned every 10 frames using 15 iterations with a learning rate set to 10^{-2}.

The generative adversarial network (GAN) that was used in the proposed trackers in sections 3.3.4, 3.3.5, and 3.3.6 was initialized using 300 iterations with a learning rate set to 5×10^{-2}. Since it was jointly finetuned with the classifier, it was updated every 10 frames with a learning rate set to

5×10^{-3}, 10 times smaller than the initialization rate to retain the discriminative features representation over a long temporal span.

To train the generator network to predict an optimal mask, a 3×3 mask template is used to prepare 9 random masks, and a similar resolution is used for the input features. The masks are distinct from each other and cover all parts of the input. By applying each mask to the input independently, 9 different versions of each input feature are produced. The optimal mask is chosen as the mask corresponding to the version that produces the lowest score for the classifier. The generator is updated to learn this optimal mask. During the fine-tuning phase, the generator and discriminator are jointly finetuned every 10 frames.

4.5 Results and Discussion

Our experiments show in Figure 4.1, Figure 4. and Figure 4.3 that integrating GAN into all three of the deep trackers improved their performance on all three datasets. The results are consistent regardless of whether the model was trained on VOT or ImageNet as shown in Figure 4.6, Figure

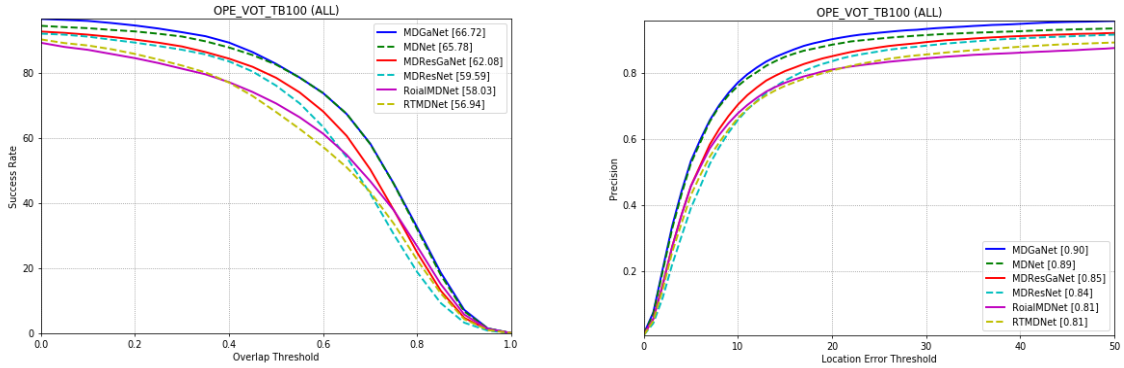


Figure 4.1: VOT TB100 Scores

4.6 and Figure 4.7. The comprehensive performance comparison is shown in Table 5.

Contrary to our expectations, MDResNet results did not compete with MDNet results. We believe this is due to the richer features extracted by ResNet which confuses the classifier as it does not learn the most discriminative features. This is supported by the results of adding a generative adversarial network that helps the classifier learn better discriminative features.

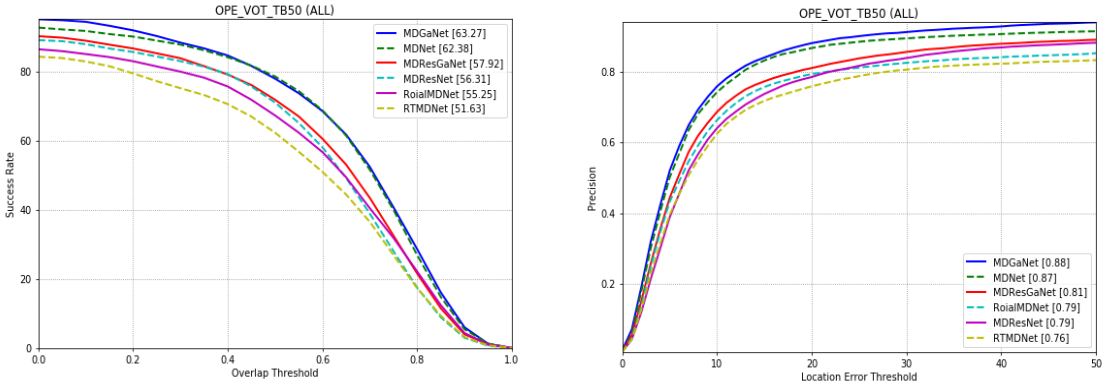


Figure 4.2: VOT TB50 Scores

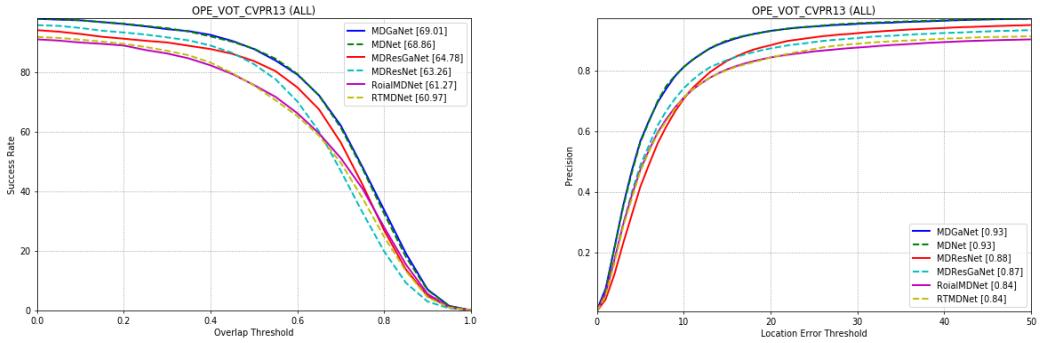


Figure 4.2: VOT CVPR13 Scores

Interestingly, the speed of MDResNet is comparable to that of MDNet, even though it has a deeper architecture. This is due to its residual nature which helps accelerate the forwarding process.

ROIAL-MDNet performed much better than RT-MDNet, but came at the cost of slower speed as shown in Figure 4.4. However, ROIAL-MDNet is still faster than all other trackers. Furthermore

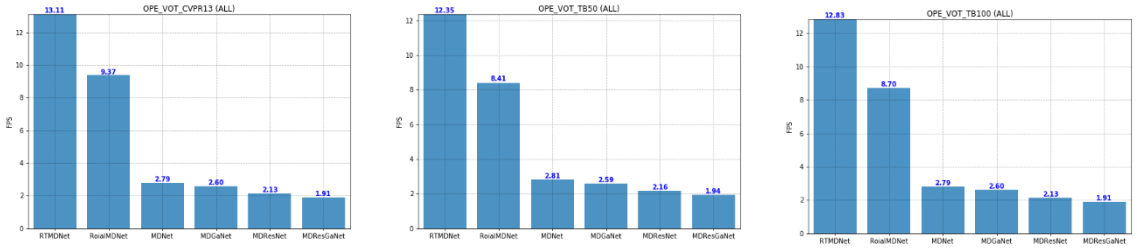


Figure 4.3: VOT FPS Scores

RT-MDNet remains the fastest and lowest performing tracker. Even though the alignment step significantly reduces the running complexity, it does not produce results that are as good.

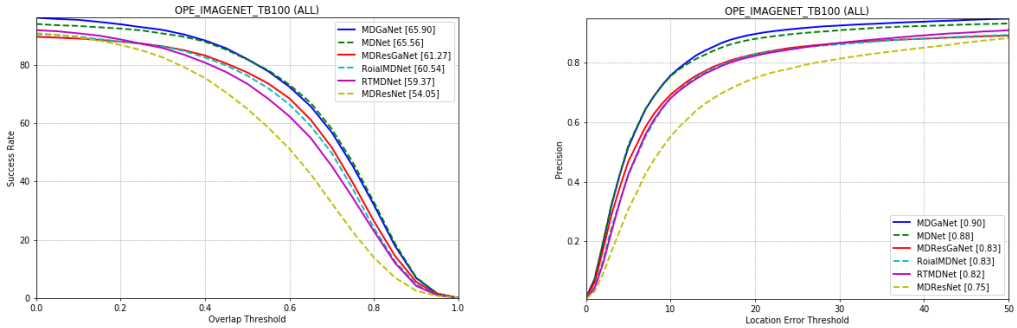


Figure 4.5: ImageNet TB100 Scores

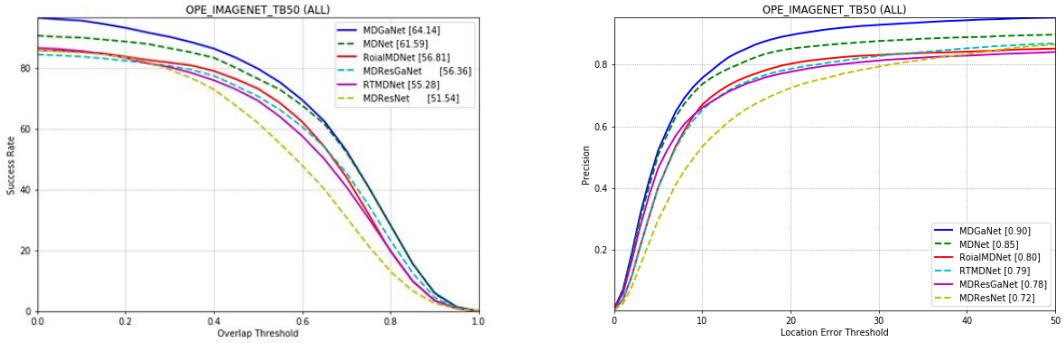


Figure 4.5: ImageNet TB50 Scores

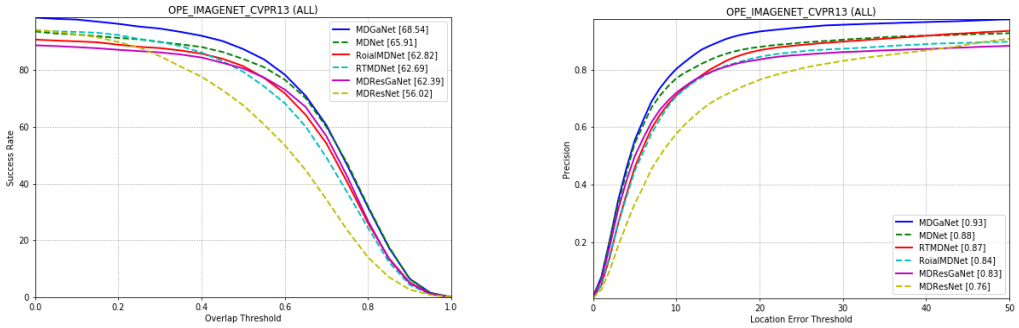


Figure 4.6: ImageNet CVPR13 Scores

Table 5 : Comprehensive performance comparison for trackers frameworks in experiment 1

Framework	**Training Dataset**	**Testing Dataset**	**Literature**			**Experiment 1**		
			Success Rate	Precision Rate	FPS	Success Rate	Precision Rate	FPS
MDNET	VOT	TB100	0.678	0.909	1	0.6578	0.89	2.8
		TB50	0.708	0.948		0.6238	0.87	
		CVPR13	--	--		0.68.86	0.93	
MDNET	ImageNet VID	TB100	--	--	--	0.6556	0.88	2.5
		TB50	--	--		0.6159	0.85	
		CVPR13	--	--		0.6591	0.88	
RT-MDNet	VOT	TB100	--	--	-	0.5694	0.81	**12.5**
		TB50	--	--		0.5163	0.79	
		CVPR13	--	--		0.6097	0.84	
RT-MDNet	ImageNet VID	TB100	0.65	0.88	25	0.5937	0.82	**12.8**
		TB50	--	--		0.5528	0.79	
		CVPR13	--	--		0.6269	0.87	

Table 6: Comprehensive performance comparison for trackers frameworks in experiment 2

Framework	**Training Dataset**	**Testing Dataset**	**Literature**			**Experiment 2**		
			Success Rate	Precision Rate	FPS	Success Rate	Precision Rate	FPS
MDGaNet	VOT	TB100	0.68	0.91	1.5	**0.6672**	**0.90**	2.6
		TB50	--	--		**0.6327**	**0.88**	
		CVPR13	0.71	0.95		**0.6901**	**0.93**	
MDGaNet	ImageNet VID	TB100	--	--	--	**0.659**	**0.90**	2.4
		TB50	--	--		**0.6414**	**0.90**	
		CVPR13	--	--		**0.6854**	**0.93**	
ROIAL-MDNet	VOT	TB100	--	--	--	**0.5803**	**0.81**	**9**
		TB50	--	--		**0.5525**	**0.79**	
		CVPR13	--	--		**0.6127**	**0.84**	
ROIAL-MDNet	ImageNet VID	TB100	--	--	--	**0.6054**	**0.83**	**9**
		TB50	--	--		**0.5681**	**0.80**	
		CVPR13	--	--		**0.6282**	**0.84**	

As Table 5, Table 6, Table 7 show, the success rate and precision rate improved by adding Generative Adversarial Networks for all base trackers. As Table 6, MDGaNet presented the highest scores in both success rate and precision rate while RT-MDNet displayed the highest score for speed as Table 5 showed. In Table 6 , ROIAL-MDNet improved success rate and precision rate scores for RT-MDNet while maintaining a high-speed score.

Table 7: Comprehensive performance comparison for trackers frameworks in experiment 3

Framework	**Training Dataset**	**Testing Dataset**	**Literature**			**Experiment 3**		
			Success Rate	Precision Rate	FPS	Success Rate	Precision Rate	FPS
MDResNet	VOT	TB100	--	--	-	0.5959	0.84	2.13
		TB50	--	--		0.5631	0.79	
		CVPR13	--	--		0.6326	0.88	
MDResNet	ImageNet VID	TB100	--	--	-	0.5405	0.75	2.12
		TB50	--	--		0.5154	0.72	
		CVPR13	--	--		0.5602	0.79	
MDResGaNet	VOT	TB100	--	--	--	0.6208	0.85	1.91
		TB50	--	--		0.5792	0.81	
		CVPR13	--	--		0.6478	0.87	
MDResGaNet	ImageNet VID	TB100	--	--	--	0.6127	0.83	2
		TB50	--	--		0.5636	0.78	
		CVPR13	--	--		0.6239	0.83	

When considering individual attributes from Table 4, we observed the best results from MDResNet and MDResGaNet on sequences where the object was partially out of view, had low resolution, or had motion blur as Figure 4.8, Figure 4.10 and Figure 4.9. We think this is due to the richer representation of the target object that is embedded in the deeper network, which helps in detecting the object when parts of the input are lost.

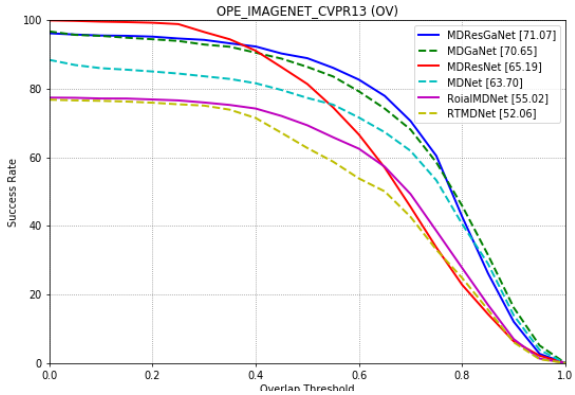


Figure 4.7: Out-of-View Scores

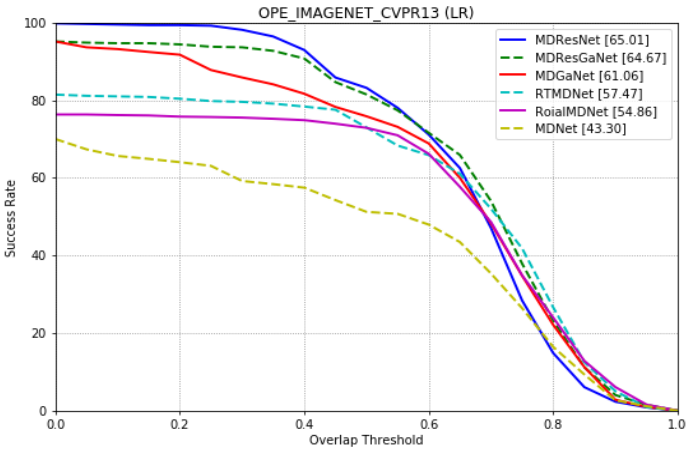


Figure 4.8: Low Resolution Scores

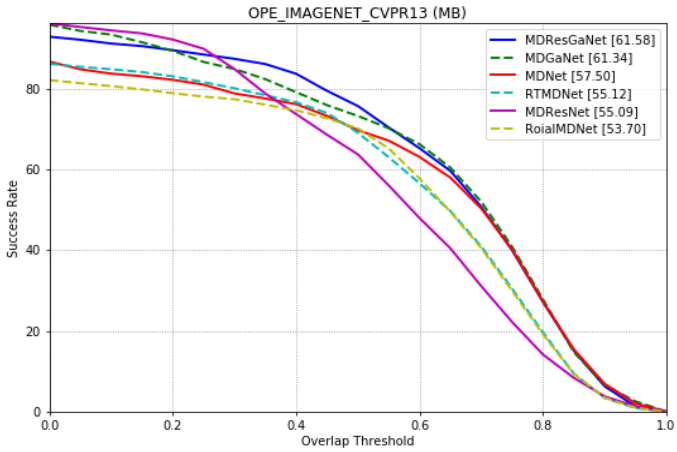


Figure 4.9: Motion Blur Scores

A comprehensive list of all attributes scores is provided in Appendix B.

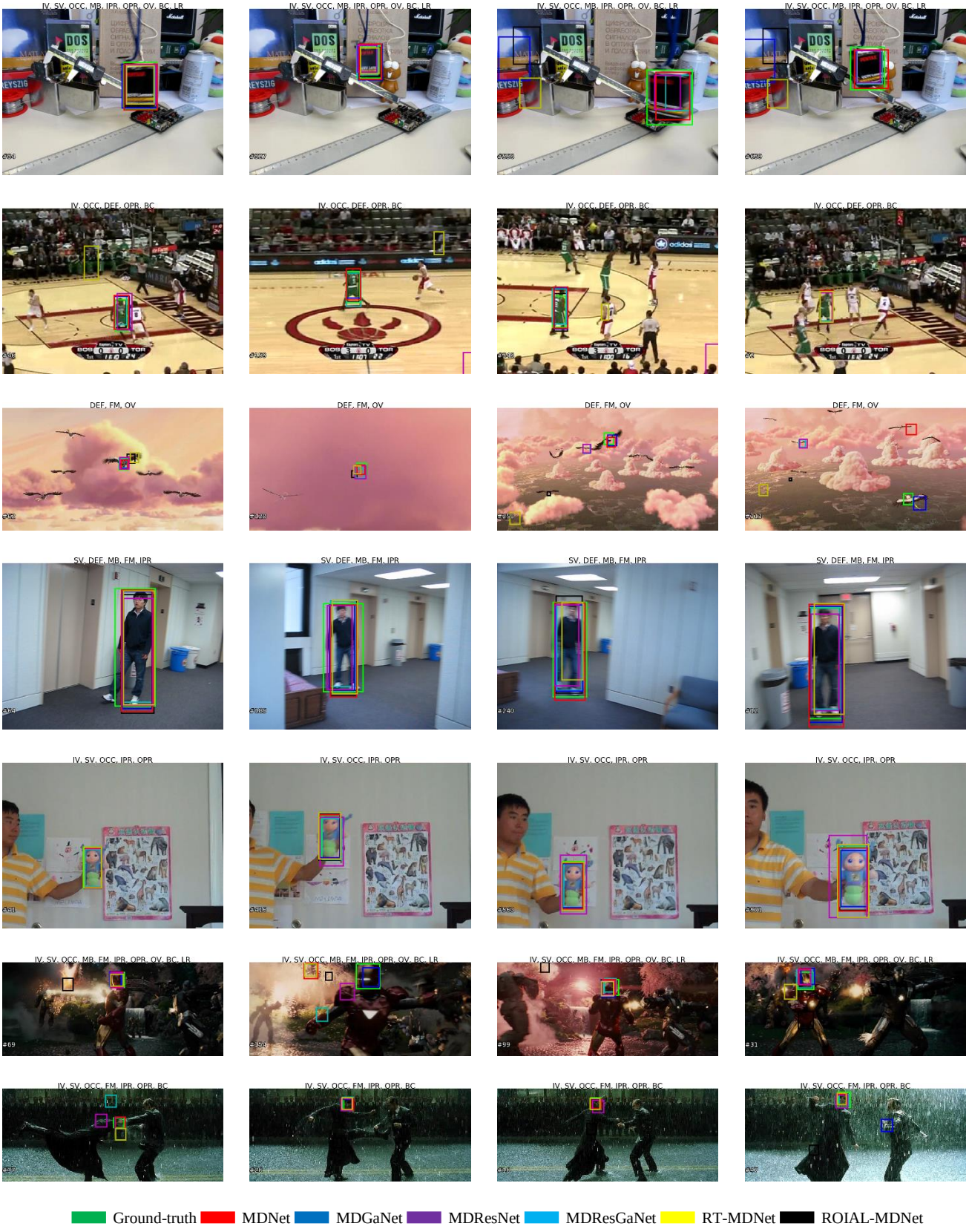
In general, there is a tradeoff between speed and accuracy that comes from the overhead of generating and testing of candidates, and based on that, no one tracker can be considered the best for all use cases. The optimal tradeoff between speed and accuracy depends on the application. The appropriate selection of an object tracker requires careful consideration of the requirements

of the situation. In real-world situations, it is important to consider the balance of multiple metrics. If the situation requires a faster tracker with acceptable robustness, then RT-MDNet and ROIAL-MDNet are preferred. Some applications demand a tracker can handle low resolution and blurred images perfectly. For this scenario, MDResNet and MDResGaNet are more suitable trackers.

To summarize,

- The fastest tracker is RT-MDNet by a wide margin with moderate robustness.
- The most robust tracker is MDGaNet.
- ROIAL-MDNet provides improved robustness with slightly reduced speed.
- MDResNet and MDResGaNet provide improved robustness with limited input (occlusion, low resolution and out-of-view).

Figure 4.10: Qualitative results of the proposed trackers on some challenging sequences



Chapter 5

5. CONCLUSION

5.1 SUMMARY OF RESEARCH

In this thesis, we introduced three novel deep visual object trackers; MDResNet, MDResGaNet and ROIAL-MDNet based on the popular MDNet tracker.

First, we evaluated two popular trackers, MDNet and RT-MDNet. Then, MDResNet is built by changing the backbone of MDNet from VGG-M to the first three layers of ResNet-50 in order to capture a richer representation of the target object.

Furthermore, we integrated a Generative Adversarial Network GAN into three deep visual object trackers including MDResNet tracker. The GAN network was used to augment the positive samples in order to encourage the classifier to learn more discriminative features for better classification performance.

Using the aforementioned GAN framework, three trackers were implemented by integrating that GAN into their architecture. The trackers are MDGaNet based on MDNet, MDResGaNet based on MDResNet, and ROIAL-MDNet based on RT-MDNet.

Finally, we evaluated the performance of all six trackers and discussed the results. We showed that the GAN framework improved the performance of all three trackers and achieved superior results under specific conditions including partial occlusion and low resolution.

5.2 CHALLENGES AND DIFFICULTIES

The chosen topic is an emergent topic with little published research. The most relevant publications referenced in this research were published in the last two years and some of them were published while developing this thesis. This made it very challenging to thoroughly understand the techniques and implementations used in related work.

Additionally, the domain of this research requires powerful computational resources to run experiments and produce results. The training and evaluation processes are both time consuming, making the iteration and development of the research difficult and slow.

Although Compute Canada provided the necessary resources, a non-trivial amount of time was spent waiting for resources to be available. In addition, the huge size of ImageNet-Vid dataset caused problems with Compute Canada storage quotas. In other words, setting up the working environment was in a significant learning curve.

Furthermore, Generative Adversarial Network frameworks work by reaching an equilibrium state between the generator and the discriminator. To achieve such state, the learning parameters must be optimized and this is typically done iteratively. This challenge was magnified when combined with the resources challenge.

5.3 FUTURE RESEARCH

Based on the insights discussed in section 4.5, particularly the performance of MDResNet and its property of being able to learn a rich representation, it would be interesting to build a modified version of RT-MDNet, which requires a rich representation of the input, by using ResNet-50 instead of VGG-M. It would also be interesting to integrate the GAN framework into such a tracker as well.

Furthermore, it would be worthwhile to explore an alternative GAN framework where positive samples are augmented with different transformations such as rotation and scaling [60].

A further improvement could be the use of the SSD object detector to generate regions of interest (RoIs) instead of R-CNN and Fast R-CNN, and integrating it with GANs in order to improve the performance of tracking.

6. APPENDIX A – PYTHON DEPENDENCIES

```
cycler==0.10.0
kiwisolver==1.1.0
matplotlib==3.0.3
numpy==1.16.3
opencv-python==3.4.4.19
Pillow==5.3.0
pyparsing==2.4.0
python-dateutil==2.8.0
PyYAML==3.13
scikit-learn==0.20.2
scipy==1.2.1
six==1.12.0
torch==1.1.0
torch-gpu==1.0.0
torchvision==0.3.0
```

7. APPENDIX B – PLOTS

Figure 7.6.1: Imagenet CVPR13 Success plots
The success plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on Imagenet-VID dataset and tested on CVPR13

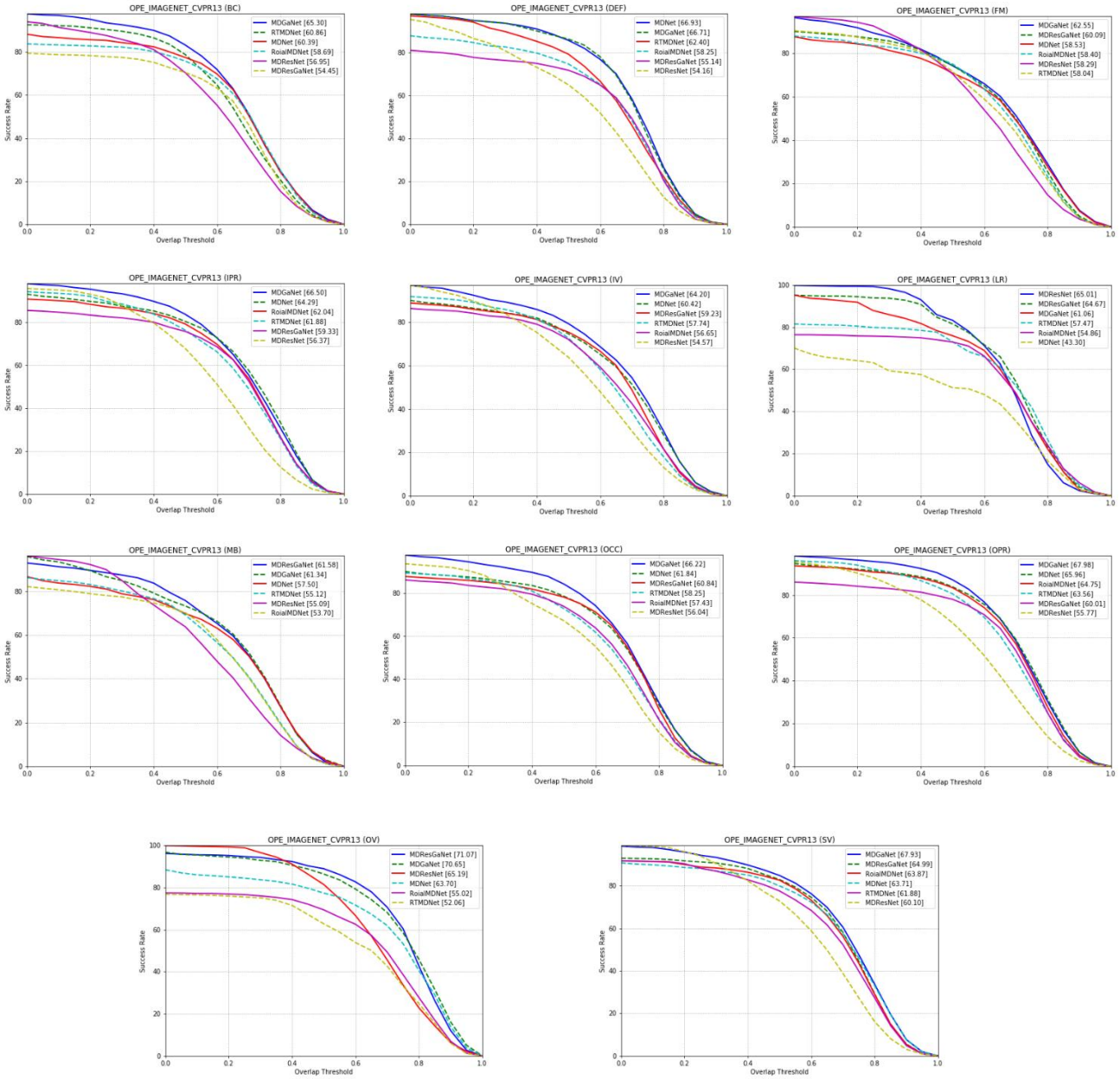


Figure 7.2: Imagenet TB50 Success plots
The success plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on Imagenet-VID dataset and tested on TB50

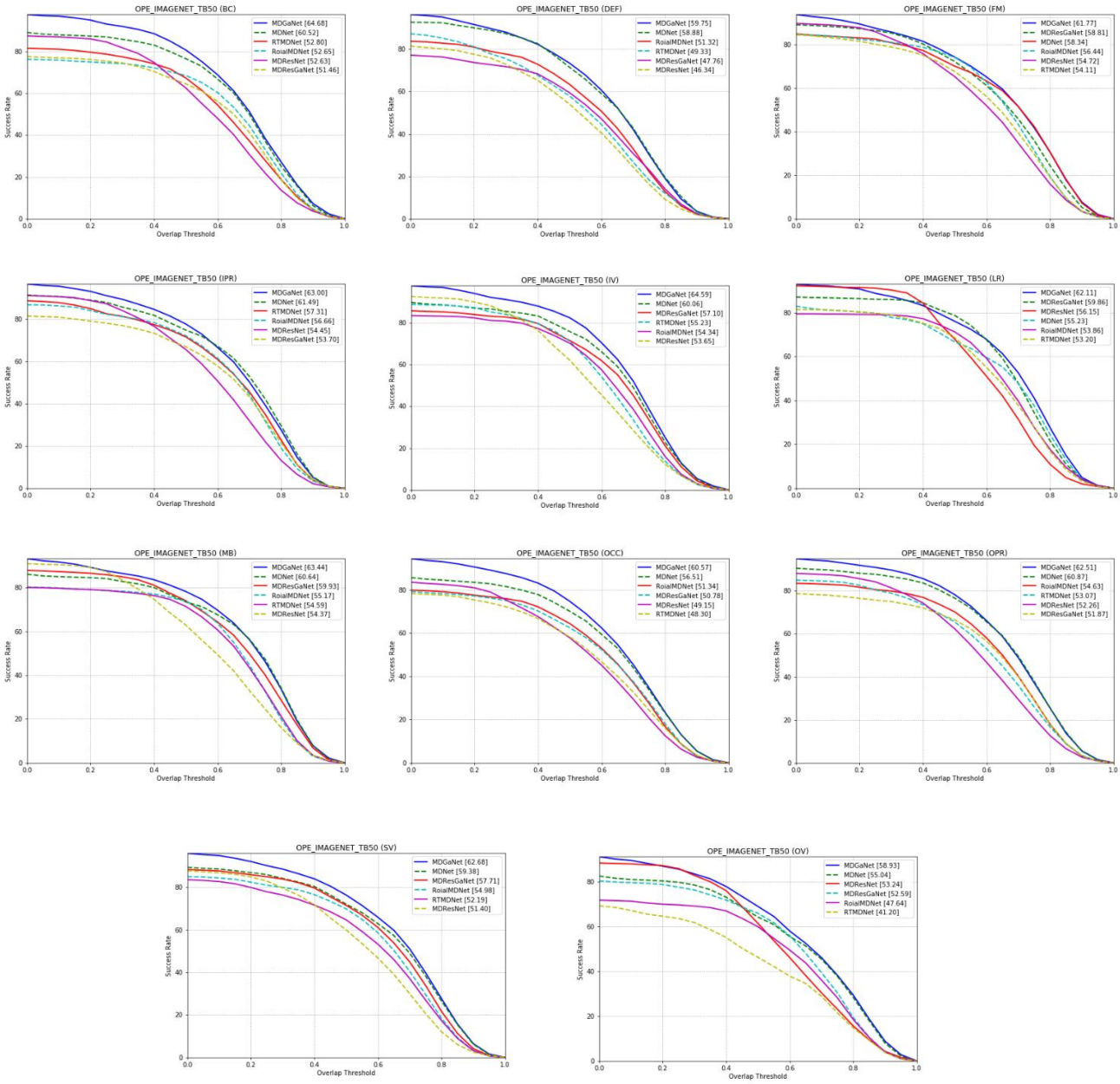


Figure 7.3: Imagenet TB100 Success plots
The success plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on Imagenet-VID dataset and tested on TB100

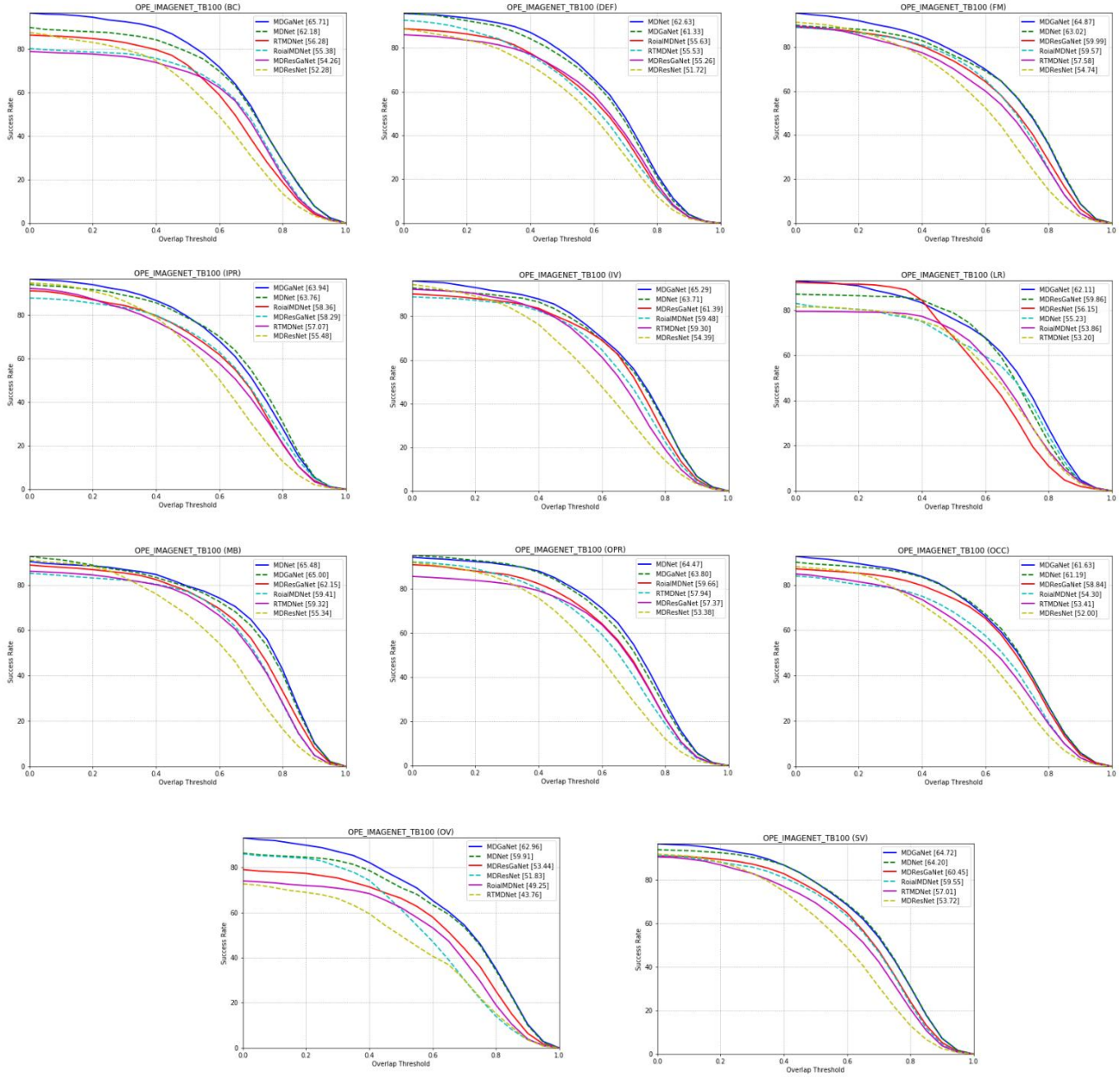


Figure 7.4: Imagenet CVPR13 Precision plots
The Precision plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on Imagenet-VID dataset and tested on CVPR13

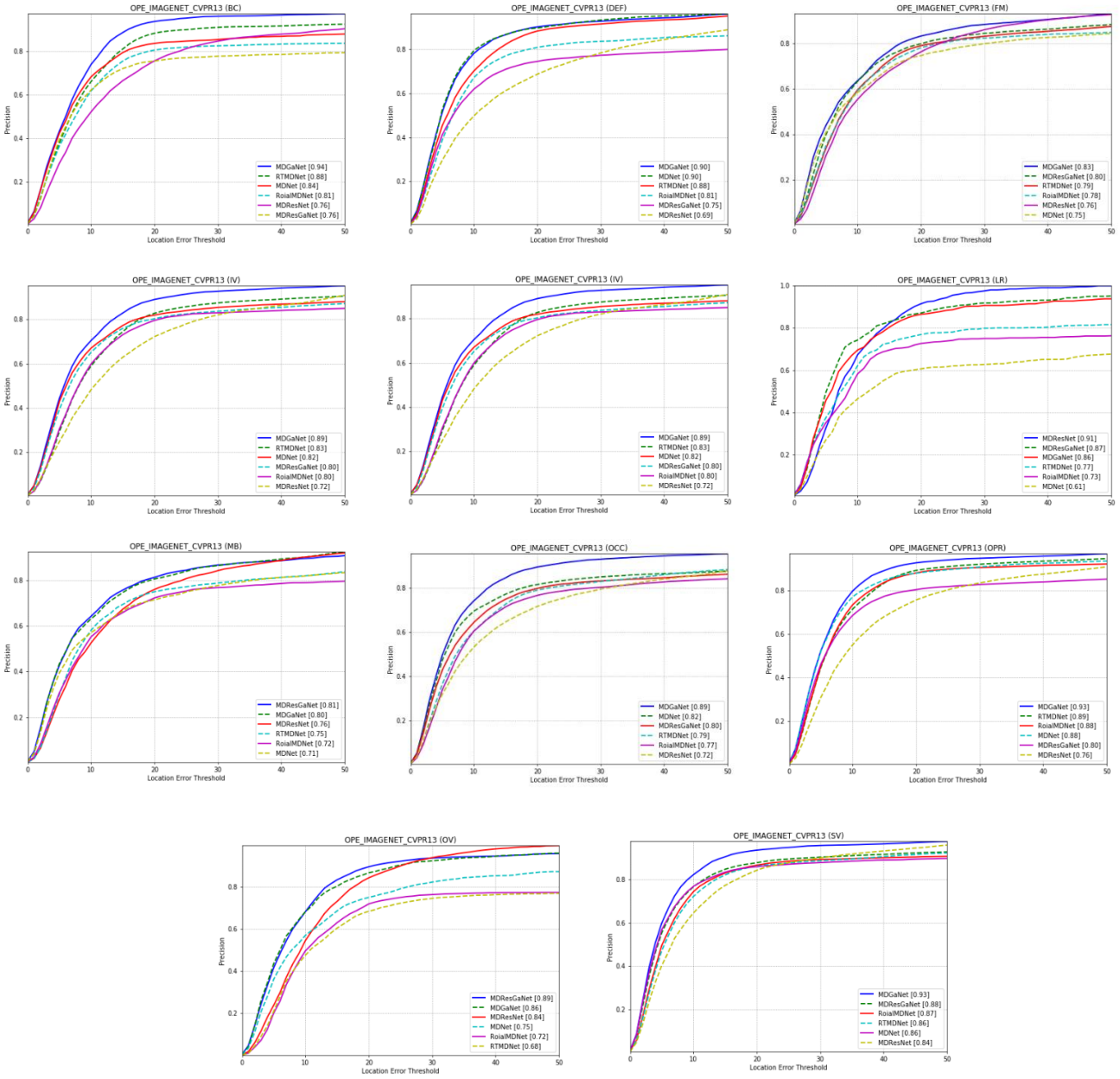


Figure 7.5: Imagenet TB50 Precision plots
The Precision plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on Imagenet-VID dataset and tested on TB50

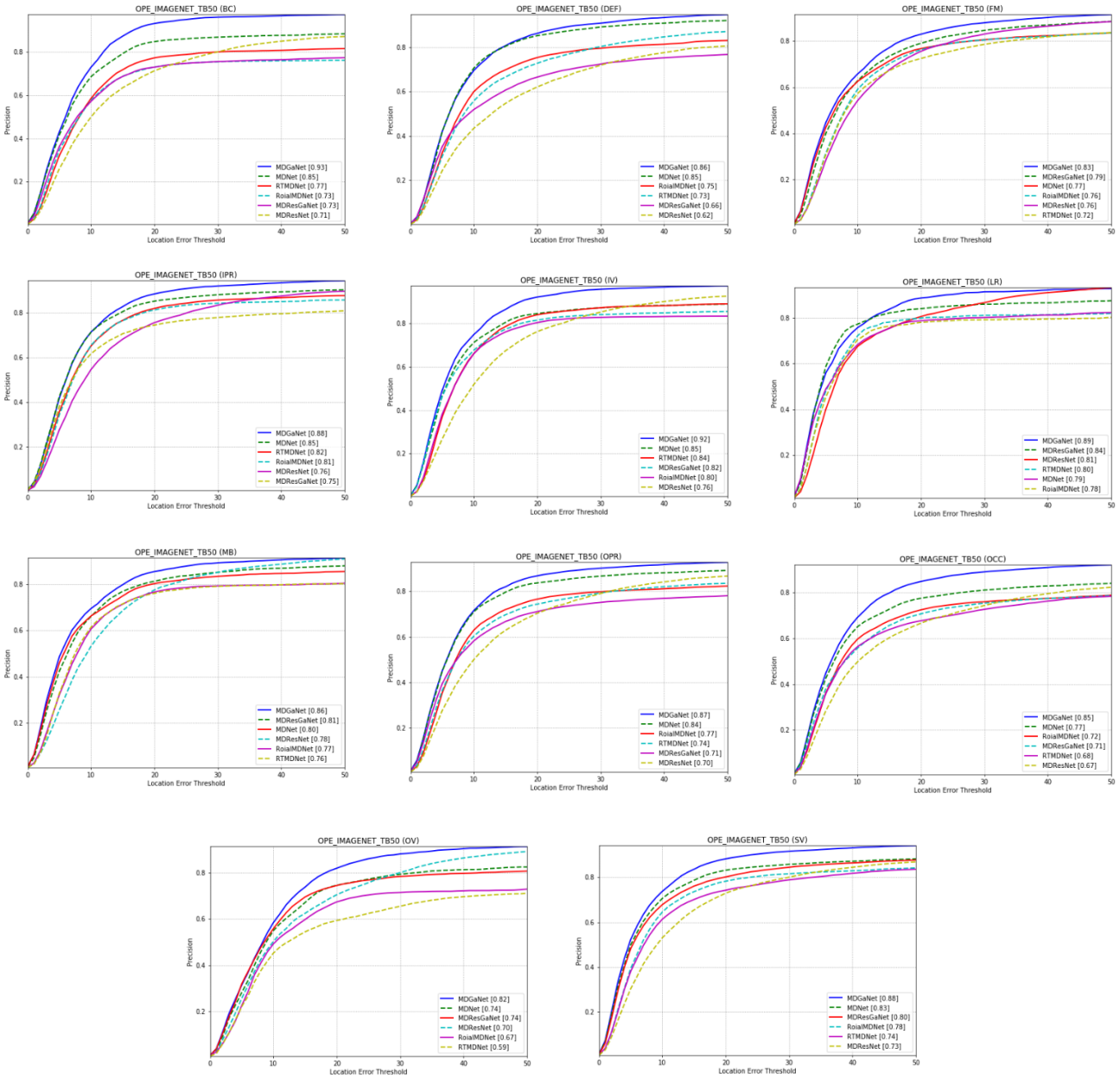


Figure 7.6 : Imagenet TB100 Precision plots
The Precision plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on Imagenet-VID dataset and tested on TB100

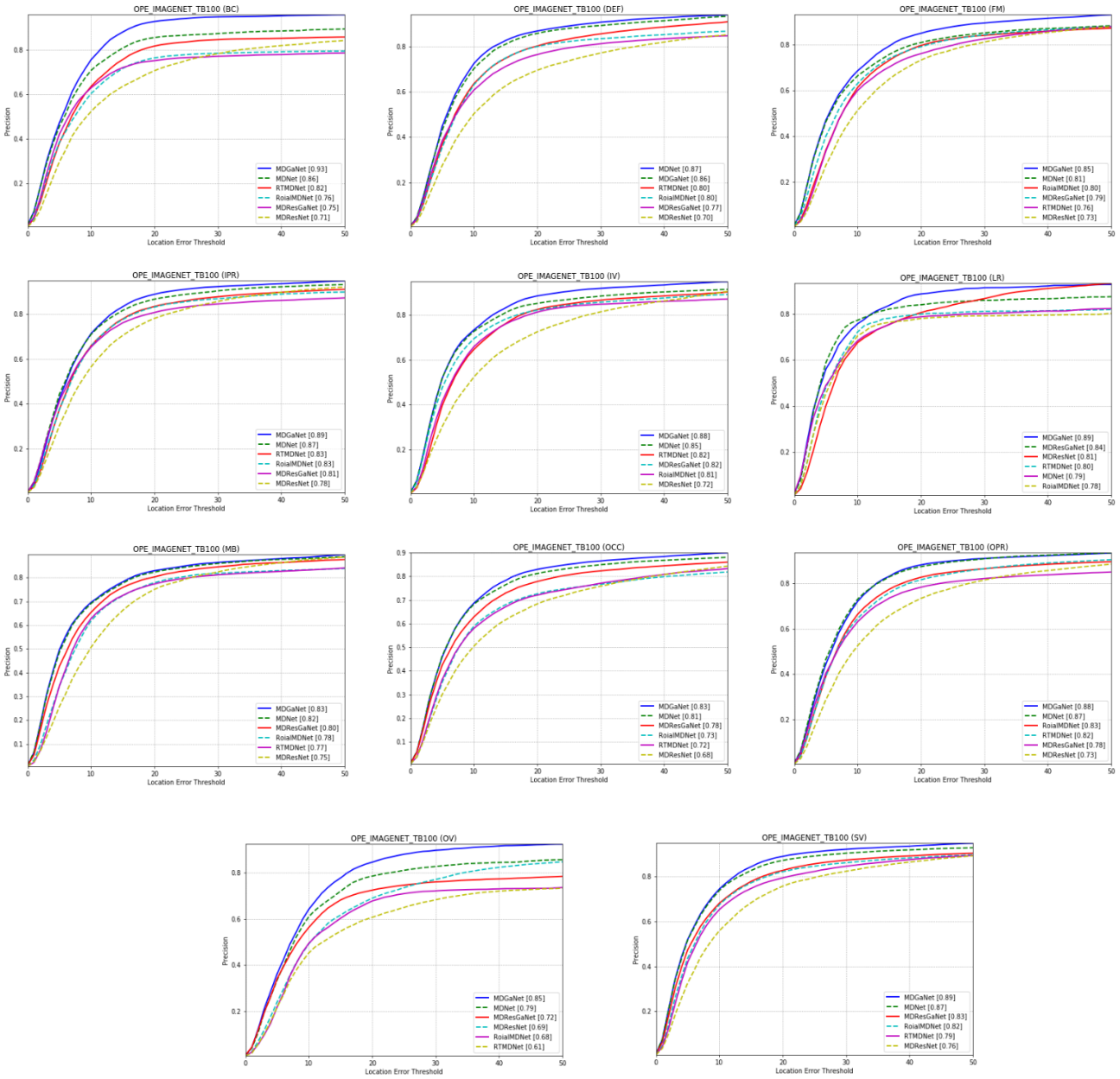


Figure 7.7: VOT CVPR13 Success plots
The success plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on VOT dataset and tested on CVPR13

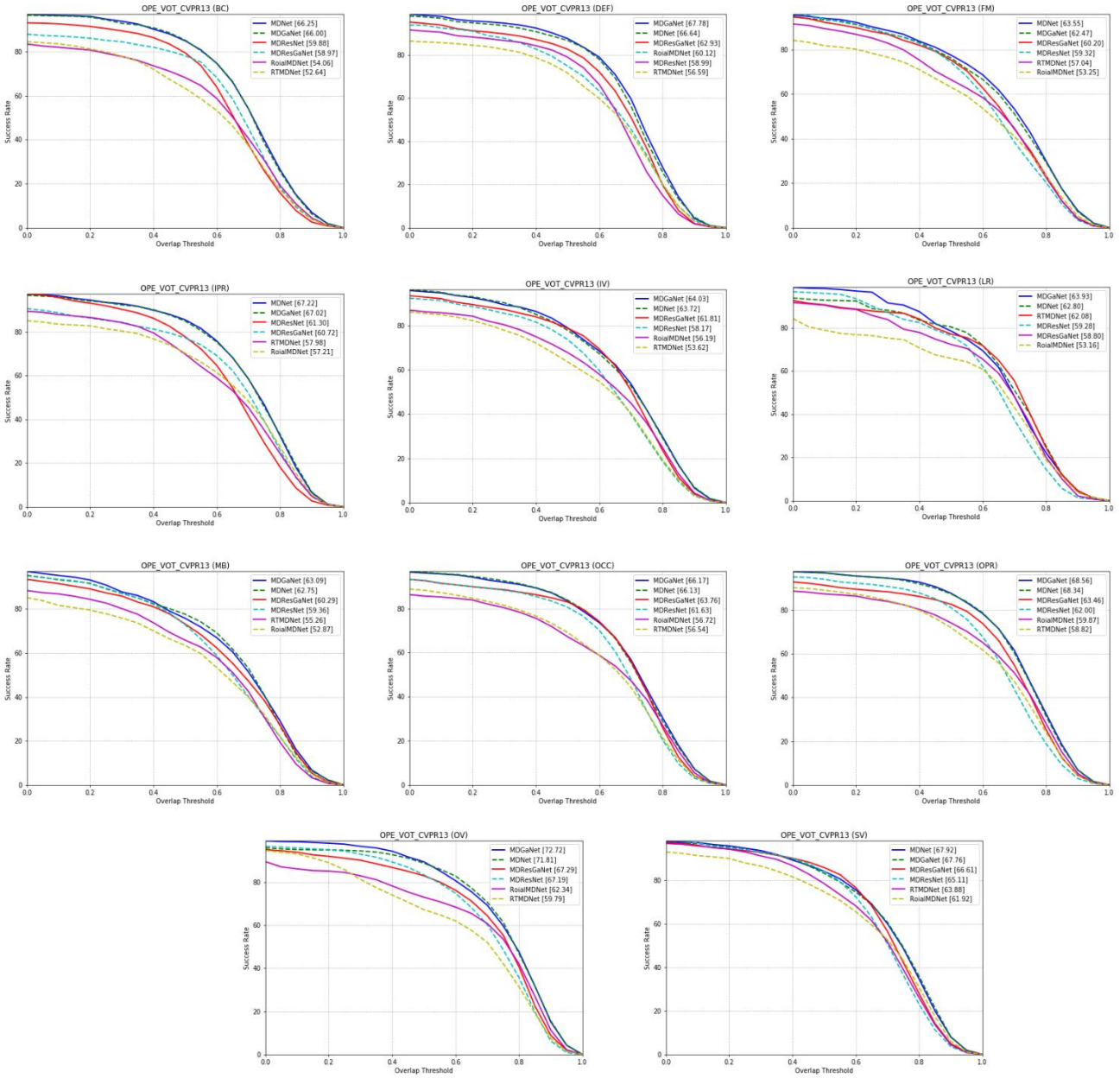


Figure 7.8: VOT TB50 Success plots
The success plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on VOT dataset and tested on TB50

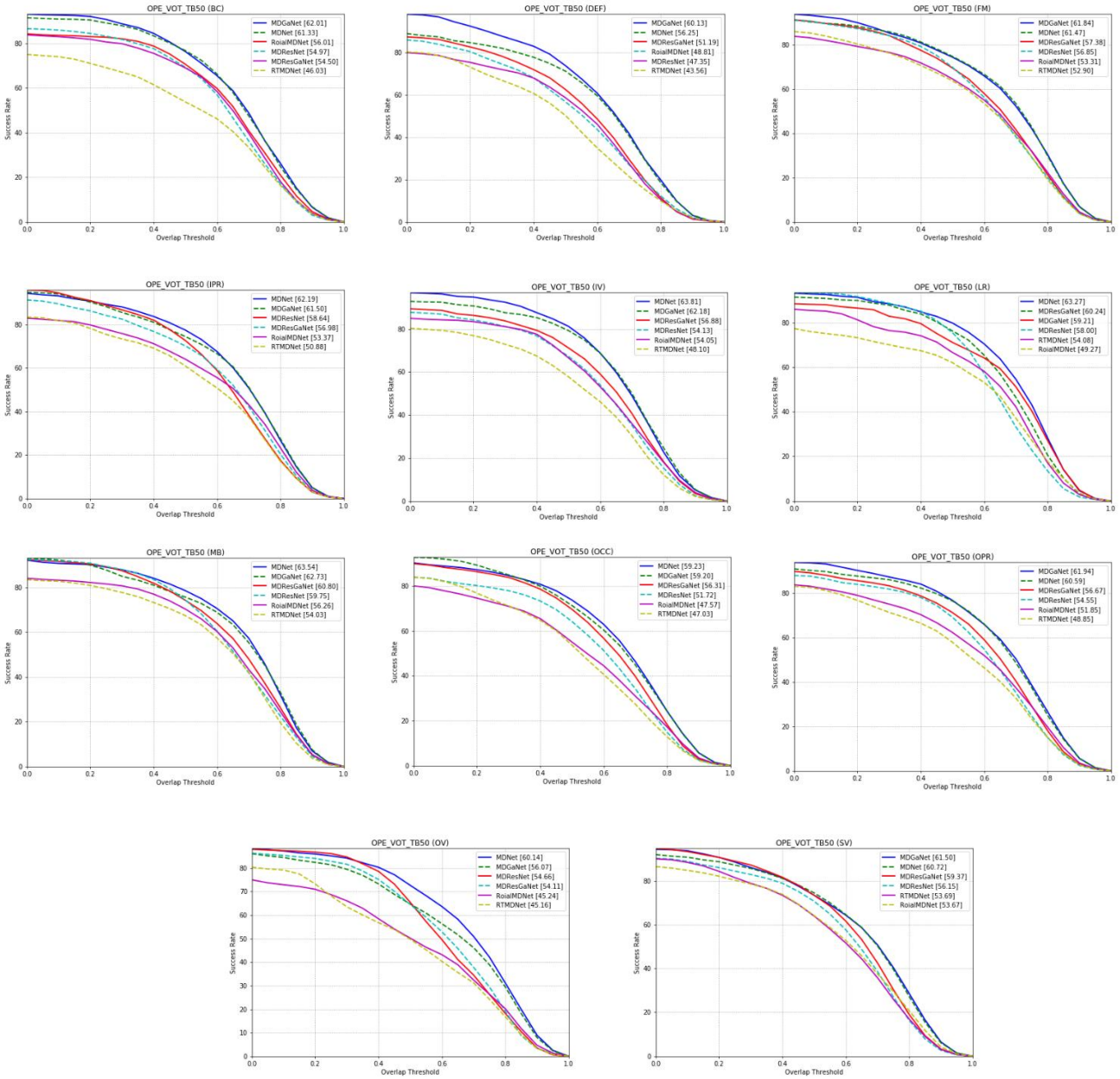


Figure 7.9: VOT TB100 Success plots
The success plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on VOT dataset and tested on TB100

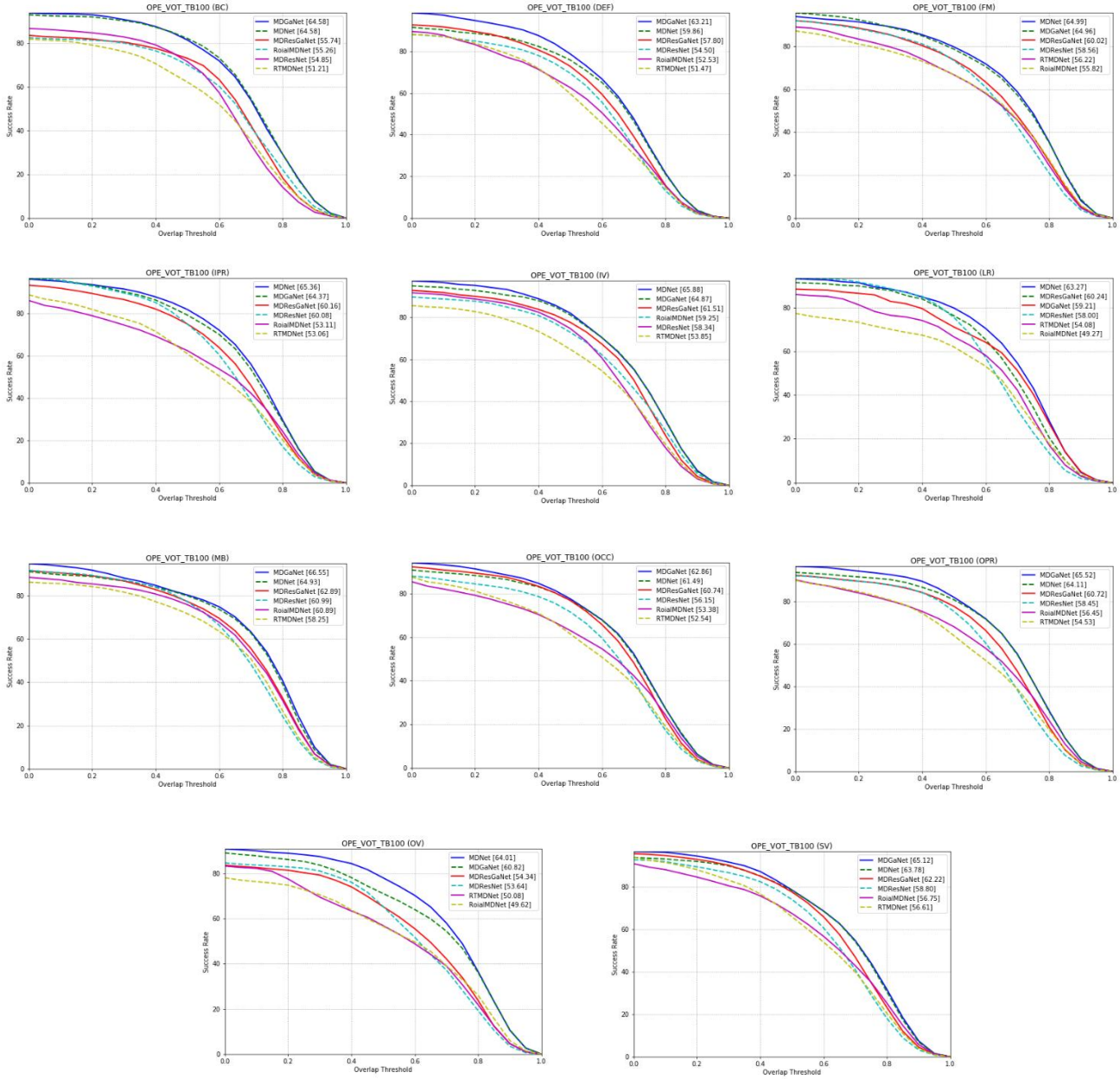


Figure 7.10 : VOT CVPR13 Precision plots
The Precision plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on VOT dataset and tested on CVPR13

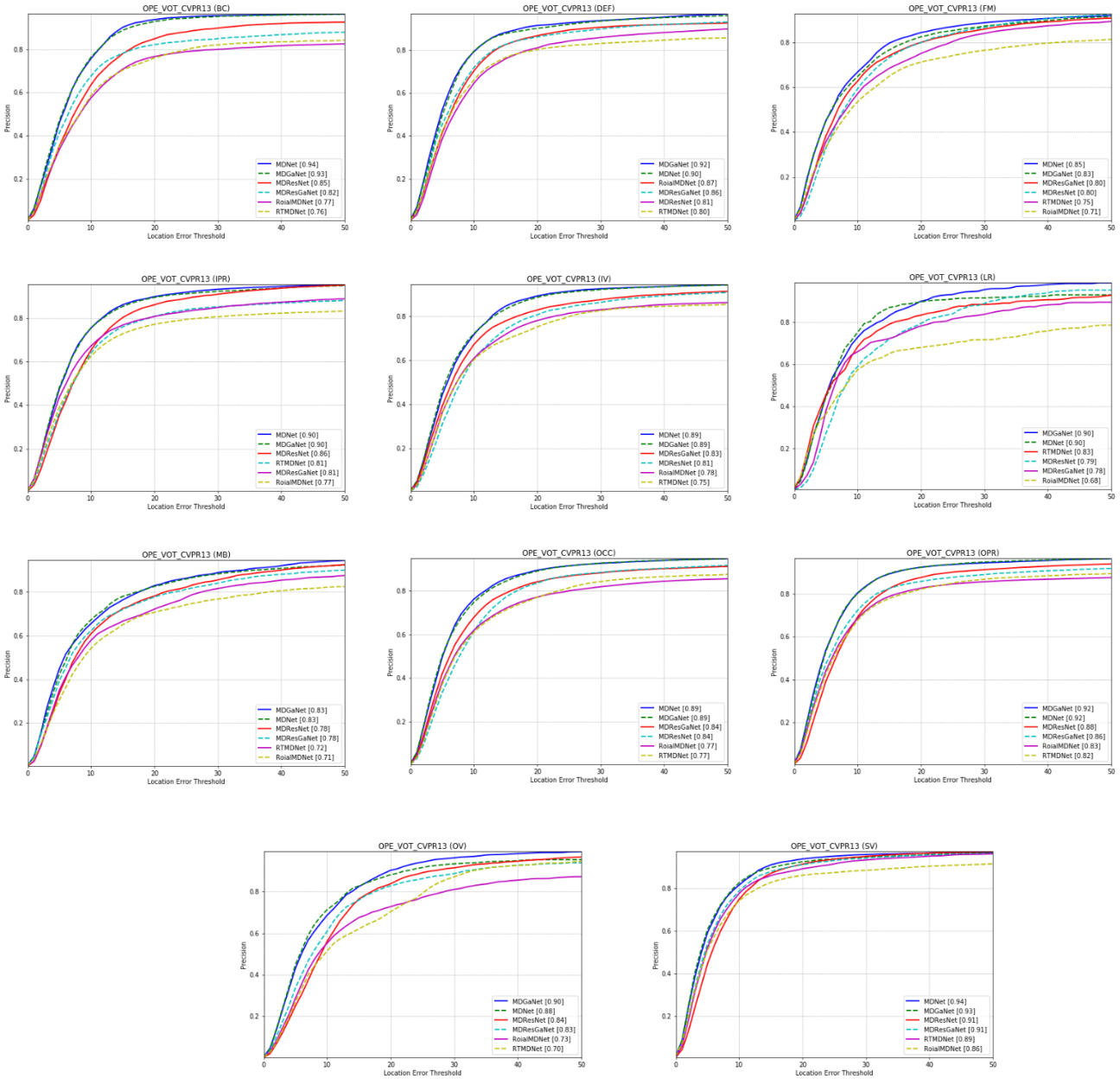


Figure 7.11: VOT TB50 Precision plots
The Precision plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on VOT dataset and tested on TB50

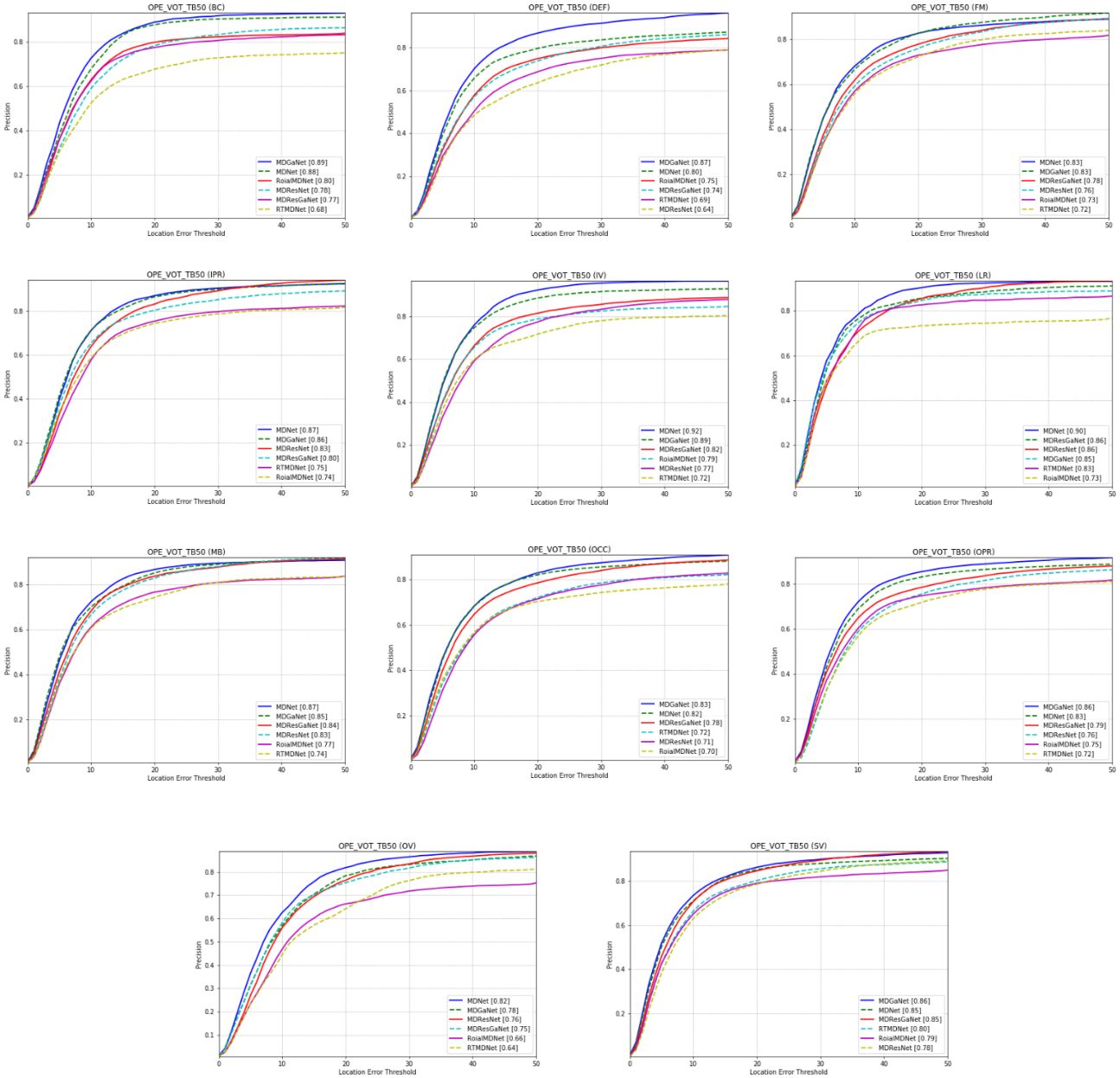
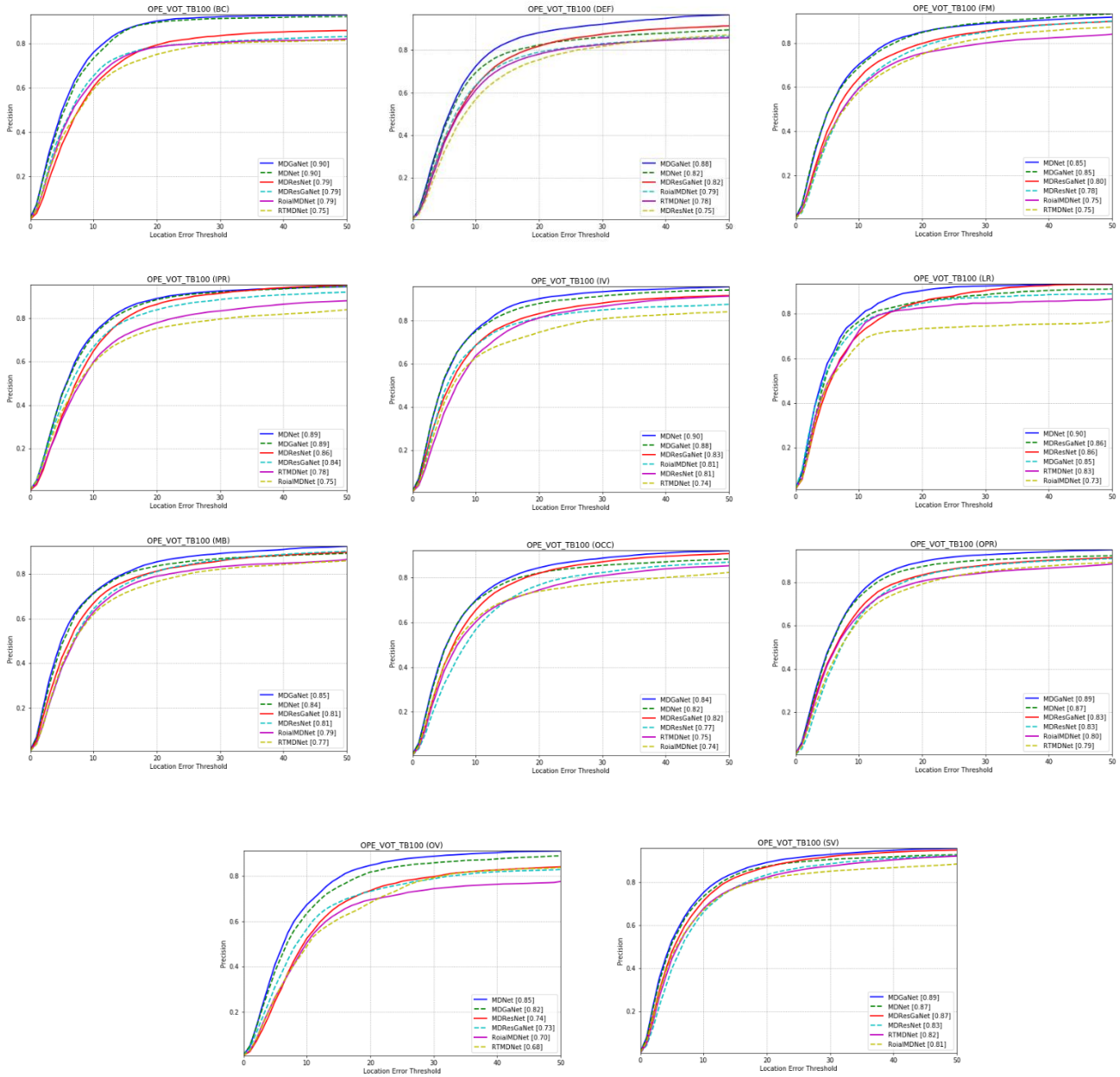


Figure 7.12: VOT TB100 Precision plots
The Precision plots for 11 challenge attributes described in Table 4 for all trackers. The trackers are trained on VOT dataset and tested on TB100



8. REFERENCES

[1] July 2019. [Online]. Available: http://votchallenge.net/challenges.html.

[2] "Mot challenges (2019)," [Online]. Available: https://motchallenge.net/.

[3] Z. Q. Zhao, P. Zheng, S. T. Xu and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[4] V. A. Laurense, J. Y. Goh and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *Proceedings of the American Control Conference*, 2017.

[5] B. Tian, Q. Yao, Y. Gu, K. Wang and Y. Li, "Video processing techniques for traffic flow monitoring: A survey," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2011.

[6] A. Ali, A. Jalil, J. Niu, X. Zhao, S. Rathore, J. Ahmed and M. Aksam Iftikhar, *Visual object tracking—classical and contemporary approaches*, 2016.

[7] J. K. Aggarwal and L. Xia, *Human activity recognition from 3D data: A review*, 2014.

[8] R. T. Azuma, *A survey of augmented reality*, 1997.

[9] F. Carrara, F. Falchi, R. Caldelli, G. Amato and R. Becarelli, "Adversarial image detection in deep neural networks," *Multimedia Tools and Applications*, 2019.

[10] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, *Generative Adversarial Networks: An Overview,* 2018.

[11] N. O. Bernsen and L. Dybkjær, Human-Computer Interaction Series, 2009.

[12] M. Klopschitz, G. Schall, D. Schmalstieg and G. Reitmayr, "Visual tracking for augmented reality," in *2010 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2010 - Conference Proceedings*, 2010.

[13] M. Al Najjar, M. Ghantous and M. Bayoumi, "Object tracking," *Lecture Notes in Electrical Engineering,* vol. 114, pp. 119-146, 2014.

[14] P. Li, D. Wang, L. Wang and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition,* 2018.

[15] L. Qin, H. Snoussi and F. Abdallah, "Object tracking using adaptive covariance descriptor and clustering-based model updating for visual surveillance," *Sensors (Switzerland),* 2014.

[16] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao and T.-K. Kim, "Multiple Object Tracking: A Literature Review," 26 9 2014.

[17] A. Voulodimos, N. Doulamis, A. Doulamis and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Computational Intelligence and Neuroscience,* 2018.

[18] J. Schmidhuber, *Deep Learning in neural networks: An overview,* 2015.

[19] T. M. T. Md Zahangir Alom, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," 2019.

[20] N. Hawkes, "Deep Learning for Generic Object Detection: A Survey," *BMJ (Online)*, 2014.

[21] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[22] K. Simonyan, "Very Deep Convolutional Networks for large-scale image recognition," 2015.

[23] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang and X. Tang, "Residual attention network for image classification," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[24] Y. Wu, J. Lim and M. H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013.

[25] Y. Wu, J. Lim and M. H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets (NIPS version)," *Advances in Neural Information Processing Systems 27*, 2014.

[27] X. Wu, K. Xu and P. Hall, "A survey of image synthesis and editing with generative adversarial networks," *Tsinghua Science and Technology*, 2017.

[28] Y. Shen, R. Ji, S. Zhang, W. Zuo and Y. Wang, "Generative Adversarial Learning Towards Fast Weakly Supervised Detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[29] X. Zhang, Y. Wei, J. Feng, Y. Yang and T. Huang, "Adversarial Complementary Learning for Weakly Supervised Object Localization," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[30] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau and M. H. Yang, "VITAL: VIsual Tracking via Adversarial Learning," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[31] M. Fiaz, A. Mahmood, S. Javed and S. K. Jung, "Handcrafted and Deep Trackers: Recent Visual Object Tracking Approaches and Trends," 6 12 2018.

[32] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and F. F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014.

[33] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[34] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015.

[35] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," 13 12 2017.

[36] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141-142, 19 10 2012.

[37] S. J. Pan and Q. Yang, *A survey on transfer learning*, 2010.

[38] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[39] H. Parekh, D. Thakore and U. Jaliya, "A Survey on Object Detection and Tracking Methods," *International Journal of Innovative Research in Computer and Communication Engineering*, 2014.

[40] H. Nam and B. Han, "Learning Multi-domain Convolutional Neural Networks for Visual Tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[41] I. Jung, J. Son, M. Baek and B. Han, "Real-Time MDNet," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[42] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley and J. Malik, "R-CNN," *1311.2524v5*, 2014.

[43] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[44] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[45] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[46] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg, "SSD: Single shot multibox detector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[48] W. Y. Zhaoqing Pan, "Recent Progress on Generative Adversarial Networks (GANs): A Survey," 2019.

[49] Y. Bai, Y. Zhang, M. Ding and B. Ghanem, "SOD-MTGAN: Small object detection via multi-task generative adversarial network," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[50] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng and S. Yan, "Perceptual generative adversarial networks for small object detection," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[51] X. Wang, C. Li, B. Luo and J. Tang, "SINT++: Robust Visual Tracking via Adversarial Positive Instance Generation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[52] J. Guo, T. Xu, S. Jiang and Z. Shen, "Generating Reliable Online Adaptive Templates for Visual Tracking," in *Proceedings - International Conference on Image Processing, ICIP*, 2018.

[53] Y. Yin, L. Zhang, D. Xu and X. Wang, "Adversarial Feature Sampling Learning for Efficient Visual Tracking," 12 9 2018.

[54] H.-I. Kim and R.-H. Park, "Siamese adversarial network for object tracking," *Electronics Letters*, vol. 55, no. 2, pp. 88-90, 20 11 2018.

[55] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[56] "MDNET Github," [Online]. Available: https://github.com/hyeonseobnam/MDNet. [Accessed 2018].

[57] "RT-MDNet-Github," [Online]. Available: https://github.com/IlchaeJung/RT-MDNet.

[58] July 2019. [Online]. Available: https://www.computecanada.ca/.

[59] July 2019. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/index.

[60] Z. T. F. Y. R. S. F. D. M. Xi Peng, "Jointly Optimize Data Augmentation and Network Training:Adversarial Data Augmentation in Human Pose Estimation," 2018.