UTILIZING VIDEO COLORIZATION AS A SELF-SUPERVISED AUXILIARY TASK FOR OBJECT TRACKING

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

BY

ENGIN FIRAT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2021

Approval of the thesis:

UTILIZING VIDEO COLORIZATION AS A SELF-SUPERVISED AUXILIARY TASK FOR OBJECT TRACKING

submitted by **ENGIN FIRAT** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Assist. Prof. Dr. Emre Akbaş
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assist. Prof. Dr. Emre Akbaş
Computer Engineering, METU _____

Assoc. Prof. Dr. Aykut Erdem
Computer Engineering, Koç University _____

Date: 08.02.2021

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

Name, Surname: Engin Fırat

Signature :

ABSTRACT

UTILIZING VIDEO COLORIZATION AS A SELF-SUPERVISED AUXILIARY TASK FOR OBJECT TRACKING

Fırat, Engin
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Emre Akbaş

February 2021, 78 pages

In this thesis work, we studied combining an object tracker, which uses siamese networks, with another model that is trained by using the self-supervised learning paradigm. We define grayscale video colorization as a pretext task for self-supervised learning and we select the similarity based object tracking as a downstream task.

Both the siamese network based object tracker and the colorization network model use the similarity between subsequent video frames. The spatio-temporal coherence between the frames of a video enables the network to learn this similarity.

We study different ways of combining the two networks. Since colorization framework uses similarity learning as its basis, we cross correlate output features of colorization network as in siamese network based tracker. Then, we combine two different methods by taking the weighted average of their score maps in order to obtain a combined score map. We search for the optimal value of this weight by conducting several experiments. In addition, we conducted experiments with different neural network architectures for the colorization framework.

Our experimental results show that utilizing the self-supervised pretext task improves

the overall success rate when the combined network is further trained in a supervised manner. In addition, we also show that self-supervised video colorization network offers an alternative way for using modern and deeper networks in siamese architectures by alleviating the strict translational invariance restriction needed by siamese architectures.

Keywords: object tracking, siamese architectures, self-supervised learning

ÖZ

ÖZ-GÖZETİMLİ VİDEO RENKLENDİRMENİN YARDIMCI BİR SİSTEM OLARAK NESNE TAKİBİNDE KULLANIMI

Fırat, Engin
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Emre Akbaş

Şubat 2021 , 78 sayfa

Bu tez çalışmasında, siyam ağlarını kullanan bir obje takipçisi ile öz-gözetimli öğrenme ile eğitilen başka bir modelin birleştirilmesi üzerinde çalıştık. Bahsi geçen öz-gözetimli eğitimde, siyah beyaz videoların renklendirilmesi yolu ile benzerlik ölçümü tabanlı bir obje takipçisi elde edilmeye çalışıldı.

Hem siyam ağları temelli obje takipçisi hem de renklendirme modeli ardışık video kareleri arasındaki benzerliği kullanır. Video kareleri arasındaki uzay-zamansal uyumluluk her iki metodun da bahsi geçen benzerliği öğrenmesini mümkün kılar.

Bu çalışmada, bahsi geçen iki yöntemin birleştirilmesi için farklı yöntemler üzerinde çalıştık. Renklendirme metodu da temelinde benzerlik öğrenimi kullandığından, çıktı olarak elde edilen öznitelikler, siyam ağları temelli obje takipçisinde olduğu gibi çapraz-korelasyon operasyonuna girdi olarak verilmiştir. Sonrasında her iki yöntemden çıkan skor haritalarının ağırlıklı ortalamaları alınarak, birleştirilmiş skor haritası elde edilmiştir. Bir takım deneyler ile en uygun ağırlık aranmıştır. Buna ek olarak, renklendirme metodu için farklı yapay sinir ağları kullanılarak da bir takım deneyler

gerçekleştirilmiştir.

Yaptığımız deneyler, öz-gözetimli video renklendirmenin yardımcı bir sistem olarak obje takipçisinde kullanımının, birleştirilmiş yapay sinir ağının gözetimli olarak eğitilmesiyle birlikte genel performans değerlerini arttırdığını göstermiştir. Buna ek olarak, yapılan deneyler, öz-gözetimli video renklendirmenin modern ve daha derin yapay sinir ağlarının siyam ağlarda kullanımı ile ilgili olarak alternatif bir çözüm sunduğunu göstermiştir.

Anahtar Kelimeler: obje takibi, siyam mimariler, öz-gözetimli öğrenme

To my late father...

To my daughter to be born...

ACKNOWLEDGMENTS

Throughout the writing of this thesis I have received a great deal of support and assistance.

I would first like to thank my supervisor, Assoc. Prof. Dr. Emre Akbaş, whose insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge my colleagues in HAVELSAN A.Ş. and thank my supervisor for their wonderful collaboration. I would particularly like to single out my supervisor, Mesut Gözütok, I want to thank you for your patient support and for all of the encouragement I was given to further my research.

Finally, I would also like to thank my lovely wife, Yağmur, and I am very grateful to my family and my family-in-law for their endless support. You provided me with a sympathetic ear and you are always there for me. Also, you provided happy distractions to rest my mind outside of my research.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xviii

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Proposed Methods and Models 2

1.3 Contributions and Novelties 4

1.4 The Outline of the Thesis 5

2 RELATED WORK 7

2.1 Legacy Methods 7

2.2 Deep Neural Network Based Methods 8

3 BACKGROUND 13

3.1 Object Tracking by Fully Convolutional Siamese Networks 13

3.1.1 Introduction . . . 13
3.1.2 Fully Convolutional Siamese Networks Based Tracker - SiamFC 14
3.1.3 Network Type . . . 16
3.1.4 Training . . . 16
3.1.5 Explanation of tracking . . . 18
3.1.5.1 Initialization . . . 19
3.1.5.2 Tracking . . . 20
3.2 Object Tracking via A Pretext Task: Video Colorization . . . 22
3.2.1 Introduction . . . 22
3.2.2 Method . . . 25
3.2.3 Network Architecture . . . 28
3.2.4 Casting Video Colorization Problem to Some Downstream Tasks 30
3.2.5 Training . . . 32
4 PROPOSED METHOD . . . 35
4.1 Introduction . . . 35
4.2 Motivation . . . 36
4.3 Method . . . 38
5 EXPERIMENTS AND RESULTS . . . 41
5.1 Introduction . . . 41
5.2 Experiments with SiamFC Implementation . . . 41
5.2.1 OTB Evaluation . . . 41
5.2.2 Experiments . . . 43
5.3 Experiments for the Colorization Framework Implementation . . . 45

5.3.1 Preparing data for training 45
5.3.2 Analysis for number of clusters 49
5.3.3 Checking the implementation 50
5.3.4 Checking for the learning capacity of the network 52
5.3.5 Training by using the overall dataset 52
5.3.6 DAVIS Evaluation 53
5.3.7 Experiments on DAVIS dataset 54
5.4 Integration of Colorization Framework to SiamFC Framework 55
5.4.1 Experiment 1 - Test Proposed Method for Different α Values, ResNet architecture as Colorization Backbone 56
5.4.2 Experiment 2 - Train Proposed Method for Different α Values 57
5.4.3 Experiment 3 - Train Proposed Method by Using Random Initialized Colorization Network Weights 58
5.4.4 Experiment 4 - Test Proposed Method for Different α Values, AlexNet architecture as Colorization Backbone 59
5.4.5 Experiment 5 - Traing Proposed Method for Different α Values, AlexNet architecture as Colorization Backbone 61
5.4.6 Summary for Experiments 62
5.4.7 Attribute Based Performance Analysis 63
5.5 Comparison with other trackers 64
5.5.1 VOT Evaluation 64
5.5.2 Results of comparison 66
6 CONCLUSION 69
REFERENCES 71

LIST OF TABLES

TABLES

Table 3.1 Comparison of Original and Modifed AlexNet NN Architectures . . 16

Table 3.2 Common Object Tracking Datasets in Literature 23

Table 3.3 Neural Network Architecture 29

Table 5.1 Table of 9 attributes to label sequences in OTB Benchmark [1] . . . 42

Table 5.2 Comparison of different algorithms in DAVIS 2017 [2] benchmark (higher Average score means better performance) 55

Table 5.3 Performance comparison for models in Experiment 2 that outperforms the baseline method 58

Table 5.4 Performance comparison for models in Experiment 3 that outperforms the baseline method 60

Table 5.5 Performance comparison for models in Experiment 4 that outperforms the baseline method 61

Table 5.6 Performance comparison for models in Experiment 5 that outperform baseline model for Experiment 5 61

Table 5.7 Calculated Performances for 9 Different Attributes. Attributes are defined in OTB [1] Benchmark. B corresponds to the baseline and 1 - 5 corresponds to the Experiment 1 to Experiment 5. Table 5.1 shows the attributes in detail. 64

Table 5.8 Neural Network Architectures Used in Different Methods 66

Table 5.9 Comparison of trackers, OTB2015 [1] benchmark 67

Table 5.10 Comparison of trackers, VOT2016 benchmark 68

Table 5.11 Comparison of trackers, VOT2018 benchmark 68

LIST OF FIGURES

FIGURES

Figure 1.1 A simplified diagram illustrating the pipeline of our proposed method. 3

Figure 3.1 Neural Network Architecture of SiamFC (image taken from original work [3]) 15

Figure 3.2 An Example for Ground Truth Score Map with $R = 16$, $k = 8$. 17

Figure 3.3 Overall architecture used in SiamFC framework (Determine Target Position is detailed in Section 3.1.5.2) 19

Figure 3.4 An example for exemplar and 3 different search patches that has different scale ratios. Leftmost to rightmost search patches has scale of $\{1.0375^1, 1.0375^0, 1.0375^{-1}\}$ respectively. 20

Figure 3.5 Scaling of displacement vector (a) Displacement in score map space (b) Displacement in search patch space (c) Displacement in original frame space 21

Figure 3.6 Architecture of Method: Tracking Emerges by Colorizing Videos 24

Figure 3.7 Working Principle of Method: Tracking Emerges by Colorizing Videos (taken from original work [4]) 26

Figure 3.8 Similarity matrix holds the similarity score for the spatial locations of reference and target images. 27

Figure 3.9 An example of similarity matrix for 1 reference frame and 1 target frame 28

Figure 3.10 Encoding spatial information vertically and horizontally 30

Figure 3.11 Tracking Segments as a Downstream Task 31

Figure 3.12 Visualization of color labels for k=8 (a) 0^{th} sample taken from video (b) 6^{th} sample taken from video (c) 14^{th} sample taken from video 34

Figure 4.1 Proposed Architecture 37

Figure 5.1 Learning Curve for SiamFC Model 44

Figure 5.2 Precision Curves for Trained SiamFC Model 45

Figure 5.3 Success Curves for Trained SiamFC Model 46

Figure 5.4 Examples of close centroids $k = 16$ 47

Figure 5.5 Centroids used in training $k = 8$ 48

Figure 5.6 Centroids for single sequence training test 49

Figure 5.7 Visualization of learning for a single video (a) 0^{th} step (b) 10^{th} step (c) 30^{th} step (d) 490^{th} step (e) Difference of 490^{th} step to ground truth 50

Figure 5.8 Learning curves for different number of clusters 52

Figure 5.9 Learning curves for two different variations 53

Figure 5.10 Success curves for Experiment 1 and related AUC values 56

Figure 5.11 Success curves for Experiment 2 and related AUC values 57

Figure 5.12 Success curves for Experiment 3 and related AUC values 59

Figure 5.13 Success curves for Experiment 4 and related AUC values 60

Figure 5.14 Success curves for Experiment 5 and related AUC values 62

LIST OF ABBREVIATIONS

ABBREVIATIONS

AUC	Area Under Curve
CVPR	Computer Vision and Pattern Recognition
DAVIS	Densely Annotated Video Segmentation Benchmark
EAO	Expected Average Overlap
FPS	Frames Per Second
GOT-10k	Generic Object Tracking Benchmark 10k Samples
ICCV	International Conference on Computer Vision
IoU	Intersection over Union
KLT	Kanadi-Lucas-Tomasi Feature Tracker
MOSSE	Minimum Output Sum of Squared Error
NMS	Non Maximum Suppression
OPE	One Pass Evaluation
ORB	Oriented FAST and Rotated BRIEF Feature
OTB	Object Tracking Benchmark
ROI	Region of Interest
SiamFC	Fully-Convolutional Siamese Networks for Object Tracking
SiamMask	Fast Online Object Tracking and Segmentation: A Unifying Approach
SiamRPN	High Performance Visual Tracking with Siamese Region Proposal Network
SIFT	Scale-invariant Feature Transform
SRE	Spatial Robustness Evaluation
SURF	Speeded Up Robust Features

TRE	Temporal Robustness Evaluation
VOT	Video Object Tracking Benchmark

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Object tracking is defined as a system which tracks the objects among the consecutive frames in computer vision domain. It is one of the fundamental problems in computer vision and machine learning. An object tracker is named as supervised if it needs to be initialized with a bounding box of an object to be tracked. On the other hand, an object tracker is named as unsupervised if it does not need explicit initialization. Instead, this kind of trackers use a mechanism, eg. an object detector, in order to determine the object to be tracked. The naming convention used here follows the VOT Challenge [5]. The object detection problem is a challenging problem due to a number of factors, such as change in object pose, change in lighting conditions, scene changes, and motion blur.

Object trackers are needed in various domains including, but not limited to, autonomous vehicles, civilian and military surveillance systems, and demographic analysis systems. These use cases will be detailed in following paragraphs.

Autonomous vehicles use some sensors in order to understand the outer world in which these vehicles are moving. Conventional RGB cameras, RADARs and LIDARs are the sensors that are frequently deployed in current autonomous cars. For example the self-driving car developed by Waymo consists of all of the mentioned sensors [6]. An autonomous vehicle's perception subsystem detects and tracks nearby objects using the sensory data in order to provide important data for the subsequent decision and control subsystems. This is why the important datasets in the field of autonomous driving domain such as Waymo Open Dataset [7], NuScenes Dataset [8], Argoverse

Dataset [9] include training and test data as well as benchmarks related to object tracking.

Surveillance systems are used in commercial, law enforcement, and military applications. Placing the video cameras to appropriate places is a cheap operation; however, finding human labor for processing the output footage is hard. Besides, processing the footage and extracting useful information from it are error prone tasks for humans. A surveillance system without any automation becomes an after the fact forensic tool and loses its primary benefit as an active, real-time medium as it is mentioned in the work done by Collins et al.[10]. Because of this fact some companies and startups, such as Dahua [11], Briefcam [12], Shield.ai [13] have been working hard on adopting some modules to surveillance systems. These modules bring not only object tracking but also object detection, human motion analysis, and activity analysis to the surveillance systems [10].

Demographic analysis systems also make use of such object trackers heavily. For example, demographic analysis system developed by Quividi [14] tracks human bodies and generate a heat map of human traffic in a retail store. In addition to this functionality, the specified system also tracks human faces in order to calculate total interest time, and gender, age, and mood of a person.

There exists other examples for domains in which object tracking algorithms are needed. The demand to robust object tracking algorithms will be increased as the automation and security issues find more place in daily life.

1.2 Proposed Methods and Models

In this thesis, we studied combining a siamese network based object tracker with another model that is trained by using self-supervised learning paradigm using video colorization as a pretext task.

Siamese networks learn a similarity metric by calculating cross correlation between the outputs of two heads of a siamese network [15]. An object is tracked among the consecutive frames by using the similarity measure calculated from the two heads of

the siamese architecture. We choose the method *Fully-Convolutional Siamese Networks for Object Tracking* proposed by Bertinetto et al. [3] as the baseline method.

Self-supervised learning paradigm offers a method for automatically extracting the ground truth labels from the data without needing any explicit annotation. Works done by Fernando et al. [16], Misra et al. [17], and Vondrick et al. [4] use this paradigm in which the ground truth labels are extracted from the data itself.

Video colorization is one of the problem areas that both computer vision and machine learning communities try to solve. Vondrick et al. [4] proposed a method that learns how to colorize grayscale videos by using the self-supervision learning paradigm. Conversion of a colored video to grayscale is an easy process and grayscale frames can be used as training data whereas the color information used as ground truth labels. This way the labels required for training is obtained from the raw data itself. No explicit labeling process is required. This is an important advantage offered by self-supervised learning paradigm.

We choose the work titled as *Tracking Emerges by Colorizing Videos* which is proposed by Vondrick et al. [4] to combine with the baseline object tracker. The method aims to solve the colorization problem in a self-supervised manner. An artificial neural network is trained in order to learn a similarity metric between the consecutive frames of a video. Learning such a similarity metric is possible because of the existence of spatio-temporal coherency between the frames of a video. A similarity metric can then be used for propagating the color information from reference frame to target frames. Solving the video colorization problem using self-supervision brings the opportunity to use any video as training data without any data labeling effort.

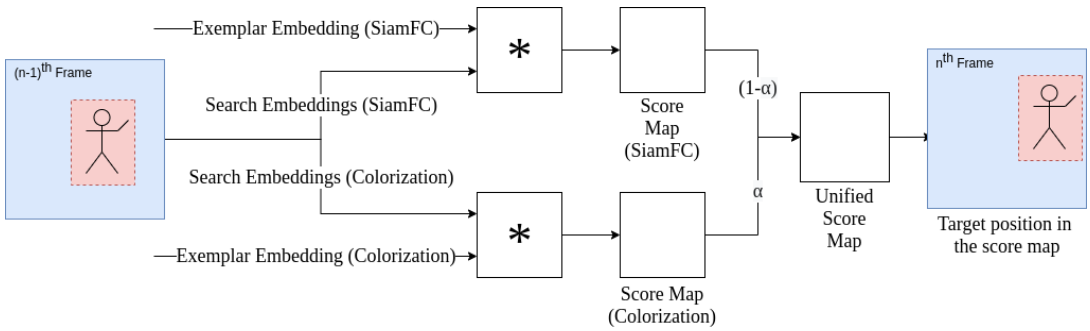


Figure 1.1: A simplified diagram illustrating the pipeline of our proposed method.

Our main motivation for this work is to bring the advantages of self-supervised learning to the object tracking domain. Hence, we combined the specified siamese network based object tracker [3] and the self-supervised learning based video colorization method [4]. This way distinct advantages of both approaches are unified in one combined system. Figure 1.1 shows the combined method from a higher level. Fully convolutional siamese networks have the advantage of similarity metric learning which brings the object agnostic object tracking whereas the self-supervised learning based video colorization method has the advantage of avoiding the need for explicit data labeling which is a labor intensive process especially for object tracking domain.

We systematically experiment with and study different ways of combining two networks. Specifically, we experimented with different α values which determines the weight of the combination. Additionally, we further trained the combined network in two ways: 1) By using the weights that are obtained by self-supervised training, 2) By using the randomly initialized weights. We aimed to see the effect of the knowledge obtained by the self-supervised colorization method with these experiments. Finally, we experimented with different backbones, ResNet-18 [18] and AlexNet [19], in order to gain insight for translational invariance restriction introduced by siamese architectures [15].

Our experiments show that knowledge obtained from colorization framework increases the overall tracking performance when the combined network is further trained in a supervised manner regardless of the backbone architecture used in colorization method. In addition, the negative effect of usage of a deeper network, for example ResNet-18 [18], which does not satisfy the translational invariance requirement needed for siamese architectures [3] is reduced. In other words, self-supervised learning opens a new way for using deeper networks that do not satisfy the translational invariance restriction.

1.3 Contributions and Novelties

Although self-supervised learning is relatively a new paradigm it has been receiving increasing interest. There has been some effort to leverage self-supervised learning

in object tracking problem. For example, Yuan et al. [20] integrated self-supervised learning to method named as SiamMask which is proposed by Wang et al. [21]. Self-supervised learning method alleviates the problems that emerge due to the massive data labeling requirements.

In this thesis, we utilize video colorization as a pretext task, and integrate it to siamese based object tracking network as a downstream task.

The novelty of this work is that it is one of the promising works that focuses on using self-supervised learning method in object tracking problem.

Our contributions in this thesis work can be summarized as follows:

- We combined two similarity metrics for object tracking. One similarity metric is learnt as a result of a supervised learning process whereas the other one is learnt through a self-supervised process using the video colorization task as a pretext task.
- We systematically analyzed different ways for combining two methods and evaluated the performance on benchmarks.
- We offer an alternative way for using modern and deeper networks in siamese architectures by alleviating the strict translational invariance restriction needed by siamese architectures.

1.4 The Outline of the Thesis

In Chapter 2, we provide a detailed literature review on object tracking. Different types of methods that become a milestone in object tracking literature is explained. From legacy methods to modern methods that make use of artificial neural networks are included.

In Chapter 3, we provide a detailed explanation on siamese network based object tracker proposed by Bertinetto et al. [3] and video colorization method proposed by Vondrick et al. [4].

In Chapter 4, we describe our proposed method in detail.

In Chapter 5, we present the experiments performed. Additionally, we provide information about the datasets, benchmarks and performance metrics used in this work.

In Chapter 6, we discuss the results of experiments and conclude the thesis work.

CHAPTER 2

RELATED WORK

Object tracking is a challenging problem as mentioned before. There are many methods in the literature that tries to attack to the problem from different perspectives.

Before digging into the artificial neural network based approaches, we give a short summary of legacy approaches to provide some context. Section 2.1 gives a short summary on legacy methods whereas Section 2.2 summarizes modern methods that mainly based on deep neural networks.

2.1 Legacy Methods

One such approach uses optical flow as a basis. This family of algorithms [22] [23] extract features from the bounding box of the object being tracked and tries to track these features within consecutive frames. Extracted features are generally scale and transformation invariant, hence it is expected that features are extracted from similar regions of an object for consecutive frames although the pose of camera or object is changed. Object tracking is done by matching extracted features. A mathematical descriptor is calculated for extracted features and this descriptor is used for feature matching. An object is tracked by tracking the matching features that are extracted from ROI of an object. SIFT [24], SURF [25], ORB [26] are some examples for feature extracting algorithms that are used in this family object tracking algorithms. A best known method in the literature is named as KLT Feature Tracker [22] [23]. This algorithm is known as a sparse optical flow algorithm that tracks the extracted feature points for a given ROI.

Another approach to object tracking problem uses color information. A well known method in the literature is named as Meanshift algorithm which was first proposed in the work of Fukunaga et al. [27]. This algorithm locates the maxima of a density function in given data. For object tracking, a color histogram calculated from ROI of an object could be the density function and be used for locating the object in consecutive frames. Another well known algorithm is named as Camshift proposed by Bradski [28] and it uses Meanshift algorithm as a basis. In this method, the color histogram for the object being tracked is updated in consecutive frames.

Another approach to object tracking use Kalman filters [29]. A typical Kalman filter mainly consists of two phases named as prediction and update. These two phases run consecutively in each iteration. In prediction phase the filter tries to estimate the position of the object for the next frame. In case of object tracking a motion model is used in order to predict the object's position. After the prediction phase, update phase runs. Filter uses sensory data in order to measure the position of the object in the space. For example, the output of an object detector may be used as a sensor measurement within the filter. Both prediction and update phase report the results within some certainty. Filter outputs a more certain result on the position of the object being tracked, by using the results of both prediction and update phases.

2.2 Deep Neural Network Based Methods

Modern object trackers are studied in two main branches. These two branches use deep neural networks as a primitive building block. The first branch is based on correlation filters. A correlation filter is used in order to locate the position of object being tracked. Online tracking is available by updating the weights of the filters while tracking the object. The ancestor of this kind of methods are first proposed in the work named *Visual Object Tracking using Adaptive Correlation Filters* [30]. In this work a correlation filter named as MOSSE is proposed. This method does not make use of neural networks, however successor methods use neural networks for learning filters online. The second branch makes use of siamese neural networks. The ancestor of this kind of trackers is the work named as *Fully-Convolutional Siamese Networks for Object Tracking* [3]. A siamese network consists of two heads and learns a similarity

metric by cross correlating the outputs from its two heads. The learnt similarity metric is used for tracking the object in consecutive frames.

Correlation filter based methods learns a discriminative filter online. The initialization frame is used in order to learn the filters online. Online learning needs more and more computation power as deeper features are used in correlation filters. Hence, recently proposed methods mainly focus on solving the online learning problem. On the other hand, siamese network based object trackers learn offline. Hence, these kind of object trackers are computationally efficient than the correlation filter based methods.

The work of Bolme et al. [30] is the first attempt that applies correlation filters to object tracking problem. Filters model the appearance of the object to be tracked. Tracking is performed by correlating a filter in the consecutive search regions. The location of the object in the next frame is determined by the location where the maximum output is obtained from correlation filter. However, localizing the object and estimating the scale conducted on the same feature space requires multi-scale feature maps during the tracking process as stated in the work done by Dai et al. [31]. This increases the computational load of correlation filter based methods especially when tracker uses the features obtained by deep neural networks. Both of the works done by Danelljan et al. named as *ECO: Efficient Convolution Operators for Tracking*[32] and Valmadre et al. named as *End-to-end representation learning for correlation filter based tracking* [33] can be shown as examples to variations in which deep features are used.

The work of Bertinetto et al. named as *Fully-Convolutional Siamese Networks for Object Tracking* [3] is the first attempt to use siamese neural networks in object tracking problem. This method uses two heads that use the same neural network as a feature extractor. A similarity metric is learnt by using a score map which is obtained by cross correlating two resultant feature maps from the heads. An object can be tracked among the consecutive frames by using this similarity metric. Siamese networks have superior running time efficiency among other family of algorithms. This situation makes development of real time object trackers possible.

There are other methods that use siamese networks. One of these methods is named as *High Performance Visual Tracking with Siamese Region Proposal Network* [34].

This method, inspired by the SiamFC method, implements object detection in search region. Method extracts features from search and exemplar regions by using the same neural network in the heads of siamese network. A modified version of AlexNet [19] is used as backbone for feature extraction. Modification is performed by removing the paddings in order to satisfy the translational invariance restriction introduced by SiamFC method [3]. These extracted features are fed to a Region Proposal Network, as used in Faster R-CNN [35], in order to generate object proposals in search region. Hence, this method can be classified as a two-staged object detector that detects the object defined in search region. Similar to Faster R-CNN, k anchors are used. Classification subnetwork in RPN classifies each proposal as foreground or background, hence outputs $2k$ channels. Regression subnetwork in RPN regress region proposals by calculating four coordinates (x, y, w, h) with respect to the ground truth anchors, hence outputs $4k$ channels. Binary cross entropy loss is used in classification subnetwork and L1 loss is used in regressor subnetwork. Region proposals are eliminated by composition of three strategies. The first strategy is to discard proposals that are far away from the center of the score map. This strategy is derived from the assumption that an object does not move rapidly between consecutive frames of a video. Second strategy is to rescoring the proposals by using cosine window and scale change penalties. This way rapid motions and rapid scale changes are discarded. The third strategy is to apply NMS in order to select the best proposal. The final proposal represents the object in search region.

The work named as *SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks* [36] is an attempt to improve the performance of the SiamRPN [34] method. Authors attempt to use a deeper network like ResNet-50 [18] as a feature extractor instead of using AlexNet [19]. However, experiments show that using a deeper network does not increase the tracking performance to an expected level. It is specified that the main cause of this result is breaking the translational invariance restriction introduced by the siamese architecture [15] itself. ResNet-50 [18] with its paddings in convolutional and residual layers breaks the translational invariance restriction. Authors hypothesize that such a violation introduces a spatial bias to the tracking system. In order to alleviate this bias, a sampling method in training phase is proposed. This method shifts the ground truth objects in a dataset within a range of

64 pixels from the center when augmenting the training data. By shifting the center of the objects it is shown that the bias of the network to find the target object in the center of search region is removed. Authors indicate that removing such a bias alleviates the negative effect of breaking the translational invariance restriction. Another experiment shows that removing the bias results in a increase in the performance of a siamese tracker implemented with ResNet-50 [18] backbone. EAO is reported as nearly 0.15 in VOT2018 [37] test dataset when there is no shift however EAO increased to nearly 0.35 when the ground truth objects are shifted 64 pixels from center.

A similar method is named as *Fast Online Object Tracking and Segmentation: A Unifying Approach*. This method extends the work of SiamFC [3] and SiamRPN [34]. Three objectives are implemented in a joint multi-task loss function in this work. First one is the similarity metric as introduced in method SiamFC [3]. The second one is the bounding box regression using Region Proposal Networks [35] as in SiamRPN [34]. Bounding box regression is used to estimate the location of the object that is being tracked in consecutive frames. Anchors are used in order to regress the bounding boxes in the given search region. The third one is a class agnostic segmentation that segments pixels as background and foreground. Hence, the overall loss function is implemented by using the following formula: $L_{3B} = \lambda_1 x Lmask + \lambda_2 x L_{score} + \lambda_3 x L_{box}$, where λ_1, λ_2, λ_3 are weights for different losses and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. It is shown that tracking the object by using segmentation masks results in more robust track scores. Authors claim that segmentation masks avoid the problems related to wrong initializations in the first frame. A wrong initialization is caused by an ROI that does not perfectly encapsulate the object that is intended to be tracked. As an example, such a case occurs when a semi-supervised object tracker is initialized by an object detector and used object detector generates bad bounding boxes for an object.

CHAPTER 3

BACKGROUND

3.1 Object Tracking by Fully Convolutional Siamese Networks

3.1.1 Introduction

Siamese Neural Networks [15] are used in order to learn a similarity metric between two input signals. Same non-linear function, a neural network, is used to calculate logits from these signals then a similarity measurement between these logits is performed in order to obtain the loss signal.

These kind of networks are used in many applications. The first definition of it appeared at a signature verification application developed by LeCun et al. [15]. A signature recording device is used to capture signatures. At most 80 bytes features are obtained from the device and recorded on a magnetic strip of a bank card. Similarity between recorded features and features belonging to a signature that is intended to be validated is calculated by using siamese networks in this work.

Another application of mentioned networks appeared in a facial recognition and verification application in the work performed by Taigman et al. [38]. The overall network is trained to learn from Social Face Classification dataset [38] which includes 4.4 million faces belonging to 4030 subjects. After learning face classification task, network weights are used for calculating the embeddings of two face images. Absolute difference of these embeddings fed to a fully connected layer which outputs a single logistic unit that determines whether the given two faces belonging to same person or not.

Another application domain of siamese networks is object tracking. A similarity metric is learnt offline and then this metric is used in order to track the object in successive frames. Siamese networks based trackers does not learn anything online as opposed to correlation filter based trackers which explained in Chapter 2. Hence, siamese networks based trackers can meet real time requirements in object tracking applications. For example, the first and second places in real time evaluation of Video Object Tracking Challenge 2019 [39] are hold by siamese network based trackers as reported in result paper. This result supports the fact of fast execution of specified networks.

The method named as *Fully-Convolutional Siamese Networks for Object Tracking* proposed by Bertinetto et al. [3] is selected as baseline method in this thesis work. This method is selected due to its simplicity and easiness of extension. In the following, we refer to this tracker as "SiamFC".

A detailed information on SiamFC tracker will be given in Section 3.1.2. Concepts like the network architecture, training procedure and tracking with the specified tracker will be detailed.

3.1.2 Fully Convolutional Siamese Networks Based Tracker - SiamFC

SiamFC is, to the best to my knowledge, the first object tracker using siamese networks. We reimplemented the MATLAB based original implementation using the PyTorch framework in the scope of this thesis work and we trained a model.

We used fully convolutional neural networks in this work. A fully convolutional neural network is same with a convolutional neural network except it does not have any dense layers. More formally, a function is fully convolutional if it commutes with translation. Defining a translation operator L_τ which translates the input signal as $(L_\tau x)[u] = x[u - \tau]$, the following formula should be satisfied if function $h()$ is commutative with translation:

$$h(L_{k\tau}x) = L_\tau h(x) \tag{3.1}$$

where k is total network stride.

An exemplar image and a search image is fed to a fully convolutional embedding function in order to obtain image features. The cross-correlation of these features produces a score map in which higher score points to higher similarity.

High level architecture of the original work can be seen in Figure 3.1. Input images denoted with z and x are named as exemplar image and search image respectively. Embedding function is shown with φ. The function φ is implemented using a fully convolutional neural network as specified before. $\star$ denotes the cross-correlation operation and it is implemented as a convolutional operator.

Cross-correlation operation outputs a score map which holds similarity between exemplar and search images by convolving exemplar embeddings on search embeddings. In other words, similarity map between two embeddings are calculated by sliding iteratively the exemplar embeddings over the search embeddings in a sliding window manner. Similarity map holds similarity scores for every spatial location. Location of the highest score points to the spatial location where the maximum similarity is occurred. The center of the search region is updated to this location in the next frame. Object tracking is performed by updating the center of the object to the spatial location where the maximum similarity is obtained in every iteration.

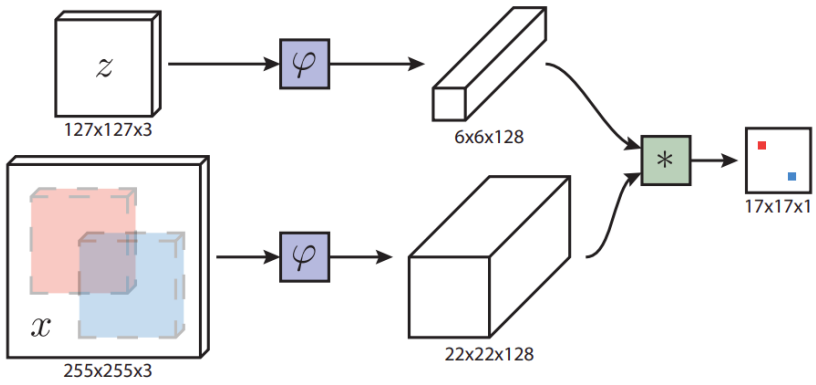


Figure 3.1: Neural Network Architecture of SiamFC (image taken from original work [3])

3.1.3 Network Type

AlexNet [19] is used as neural network with some modifications. Firstly, all paddings are removed from the network in order to provide fully convolutional property as explained in Section 3.1.2. Secondly, the latest max pooling layer is removed. Thirdly, strides for some convolutional layers are changed. The original architecture and the modified one is given in Table 3.1. One can see the specific changes in detail on the table. Input size to the original architecture is $227 \times 227 \times 3$ and the sizes of search image and exemplar image are $255 \times 255 \times 3$ and $127 \times 127 \times 3$ respectively. Fully connected layers in original architecture are not shown in the table for the sake of simplicity.

Table 3.1: Comparison of Original and Modifed AlexNet NN Architectures

Original AlexNet Architecture					**Modified AlexNet Architecture**					
Type	*Kernel Size*	*s*	*p*	*Output*	*Type*	*Kernel Size*	*s*	*p*	*Search Output*	*Exemplar Output*
CONV	11×11	4	0	$55 \times 55 \times 96$	CONV	11×11	2	0	$59 \times 59 \times 96$	$123 \times 123 \times 96$
MPOOL	3×3	2	0	$27 \times 27 \times 96$	MPOOL	3×3	2	0	$29 \times 29 \times 96$	$61 \times 61 \times 96$
CONV	5×5	1	2	$27 \times 27 \times 256$	CONV	5×5	1	0	$25 \times 25 \times 256$	$57 \times 57 \times 256$
MPOOL	3×3	2	0	$13 \times 13 \times 256$	MPOOL	3×3	2	0	$12 \times 12 \times 256$	$28 \times 28 \times 256$
CONV	3×3	1	1	$13 \times 13 \times 384$	CONV	3×3	1	0	$10 \times 10 \times 192$	$26 \times 26 \times 192$
CONV	3×3	1	1	$13 \times 13 \times 384$	CONV	3×3	1	0	$8 \times 8 \times 192$	$24 \times 24 \times 192$
CONV	3×3	1	1	$13 \times 13 \times 384$	CONV	3×3	1	0	$6 \times 6 \times 128$	$22 \times 22 \times 128$
MPOOL	3×3	2	0	$6 \times 6 \times 256$	This layer is removed.					

3.1.4 Training

Method needs annotated video data for training. Exemplar and search images are extracted by using the ground truth bounding box information in dataset. A pair is constructed by extracting an exemplar patch from $(n-1)^{th}$ frame and a search patch from n^{th} frame. Ground truth objects are centered on the patches. Ground-truth score

map belonging to a specific pair is labeled according to the following formula:

$$f(x) = \begin{cases} +1, & \text{if } k|u-c| \leq R \\ -1, & \text{otherwise} \end{cases} \tag{3.2}$$

Here R is defined as radius from object centre and k is defined as total stride of fully convolutional network. A generated ground truth score map by using this formula can be seen in Figure 3.2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
3	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
4	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
5	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
6	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
7	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
8	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
9	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
10	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
11	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
12	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
13	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
14	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
15	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
16	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

Figure 3.2: An Example for Ground Truth Score Map with $R = 16$, $k = 8$

Loss for a single value in score map is simply defined by binary cross entropy loss:

$$\ell(y, v) = log(1 + e^{-yv}) \tag{3.3}$$

where y is the ground truth taking values of {-1, 1} indicating whether the given pairs are positive or negative and v is the single value of the score map. The loss function is defined as mean of the individual losses for a score map:

$$L(y, v) = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} \ell(y[u], v[u]) \tag{3.4}$$

Note that score map is a grid of real valued numbers defined as $v : \mathcal{D} \to \mathbb{R}$. Hence, $y[u]$ and $v[u]$ corresponds to single ground truth and score map values in binary cross entropy formula. Stochastic gradient descent is used as optimizer in order to determine the weights θ that minimizes the loss function L for the neural network used:

$$\underset{\theta}{\operatorname{argmin}} \underset{(z,x,y)}{\mathbb{E}} L(y, f(x, z; \theta)) \tag{3.5}$$

The initial values for θ is initialized with Gaussian noise and scaled by using the Xavier method [40]. Neural network sees 50.000 samples for each epoch and training lasts for 50 epochs in total. Mini batches consists of 8 samples and learning rate starting from 10^{-2} to 10^{-5} decreasing gradually in each epoch.

Positive training samples are extracted from annotated video dataset as exemplar and search patches. Every pair is at least *T* apart from each other. Video frames are preprocessed offline before training. Patches are extracted by using the following constraint:

$$s(w+p)\, s(h+p) \;=\; A \tag{3.6}$$

where A is area constant, p is context margin, (w, h) is the tight bounding box defined in dataset. $A = 127^2$ for exemplar patch and $A = 255^2$ for search patch. Context margin is defined as $p = \frac{w+h}{4}$. Image patches that do not obey the specified width and height are filled by RGB mean color. 2015 edition of ImageNet Large Scale Visual Recognition Challenge - ImageNetVID [41] dataset is used to train the neural network in original work. This dataset contains 4500 train and validation video in total. Authors are used all the 4417 training videos to train the network. Nearly 2M search and exemplar pairs are extracted from these 4417 videos. EAO metric, measured using VOT-15 test dataset [42], improves from 0.168 to 0.274 as increasing the used percentage of the ImageNetVID. This situation points to the fact that a larger dataset could increase the performance. In addition to this, although 2M train samples seems larger, reader should note that these samples are fetched from 4417 videos in total which is very small comparing to amount of data ImageNet has. ImageNet contains 14M images, that is used to train image classification networks. Hence, this fact gives a clue about the importance of amount of data to train a neural network based tracker.

3.1.5 Explanation of tracking

In this section how tracking is performed within the SiamFC framework will be explained in detail. Overall tracking architecture can be seen in Figure 3.3. Tracking is performed in two main steps. The first one is the *Initialization* step which executes only once during the lifetime of the tracker. The second one is the *Tracking* step

which executes iteratively for all of the frames in video.

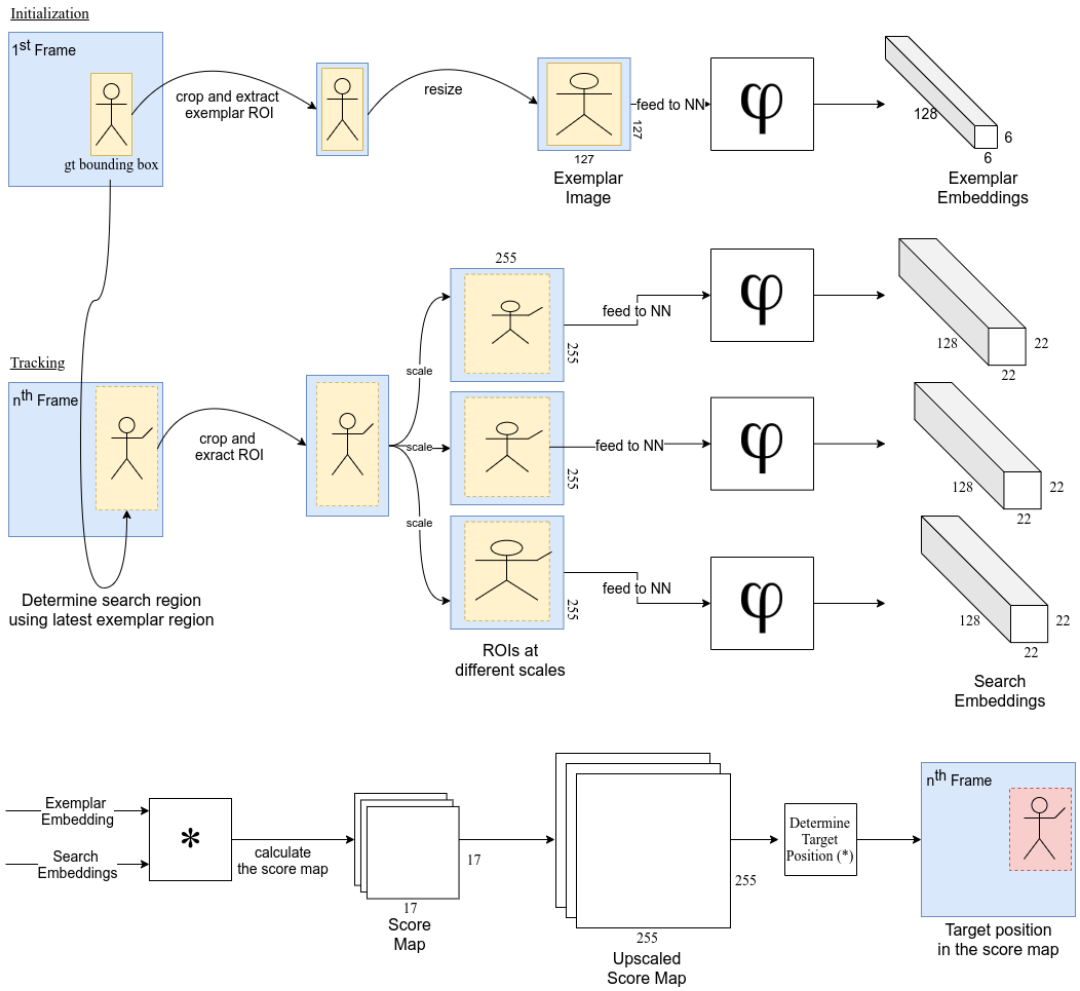


Figure 3.3: Overall architecture used in SiamFC framework (Determine Target Position is detailed in Section 3.1.5.2)

3.1.5.1 Initialization

Tracker is initialized by taking the ROI of the area that is intended to be tracked. In a supervised tracker ROI is fetched from the dataset itself, the ground truth bounding box for the first frame.

Exemplar patch is cropped and extracted from the 1^{st} image. Extraction is performed by obeying the restriction defined as $s(w + p)\ s(h + p) = A$, where $A = 127$ for exemplar patch. This patch is then fed to the neural network defined by function φ.

The output embedding for exemplar patch has dimensions $6 \times 6 \times 128$. Extracted embedding act as an filter for convolution operator $\star$. This embedding is kind of a template for the object being tracked and is updated during the tracking phase in order to handle the changing poses and scales for the object. This phenomenon will be detailed in *Tracking* phase.

3.1.5.2 Tracking

In tracking phase, tracker simply calculates the similarity between exemplar and search patches. The spatial position of the score map that has the highest similarity is chosen as the updated center position of the object that is being tracked in the search patch. First of all, search patch is extracted by using the object's latest center position. The ROI is determined again by obeying the following restriction: $s(w + p)\ s(h + p) = A$, where $A = 255$. Extracted search patch is then scaled by using three different scale factors. Scaling is performed in order to handle changing poses and scales of the object being tracked. Following scale factors are being used: $\{1.0375^{-1}, 1.0375^{0}, 1.0375^{1}\}$. An example for exemplar and search patches can be seen in Figure 3.4. Every scaled search patch is fed to neural network defined by function φ. Three embeddings for three different scales has the same dimensions: $22 \times 22 \times 128$. Cross correlation of exemplar and three search embeddings are

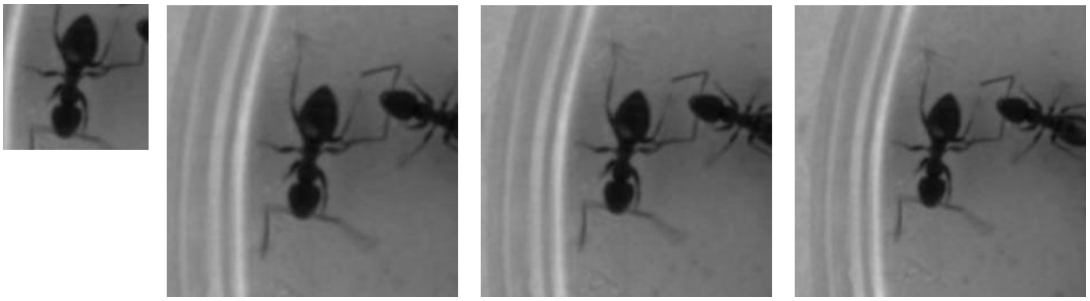


Figure 3.4: An example for exemplar and 3 different search patches that has different scale ratios. Leftmost to rightmost search patches has scale of $\{1.0375^{1}, 1.0375^{0}, 1.0375^{-1}\}$ respectively.

calculated and three different score maps are obtained for specified scales. Every score map has dimensions $17 \times 17 \times 1$. Score maps are then upscaled to dimensions

$255 \times 255 \times 1$ by using bicubic scaling method. Two score maps for scale ratios other than 1 are penalized by a constant equals to 0.9745. Penalization is used because of the fact that object scale does not tend to change rapidly in general. After penalization the best scale ratio is selected by determining the best similarity value among three different score maps for different scale ratios. Scale ratio of the score map which contains the best similarity value is chosen as the best scale ratio. Displacement is calculated after selection of the best scale ratio. Displacement equals to the spatial distance between the most similar region in the score map to the center point of the score map. Displacement vector is scaled multiple times in order to obtain displacement in original frame space since displacement is measured originally in score map space. The overall process is detailed in Figure 3.5. Displacement vector is first

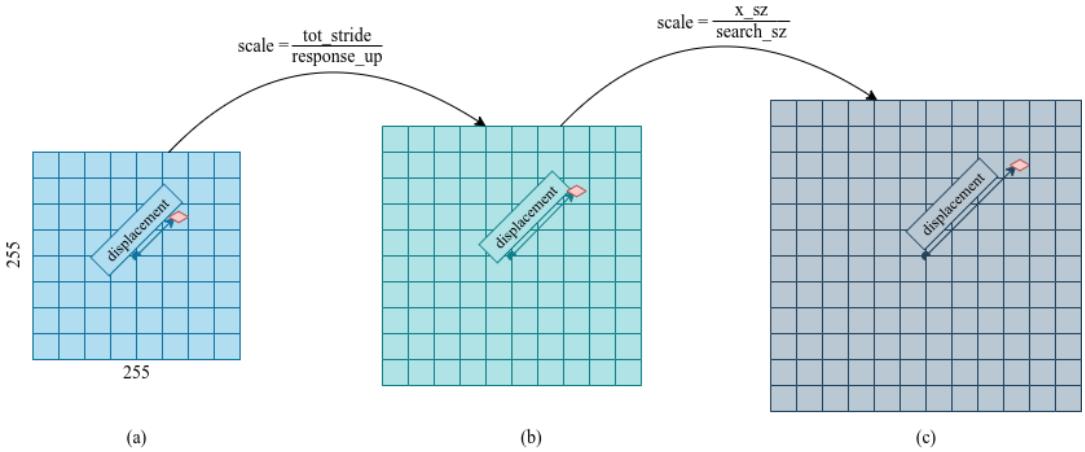


Figure 3.5: Scaling of displacement vector (a) Displacement in score map space (b) Displacement in search patch space (c) Displacement in original frame space

scaled by $\frac{total\_stride}{response\_up}$ in order to transform displacement vector to search patch space. Here *total_stride* equals to total stride of the neural network and *response_up* equals to the scale difference between original and scaled sizes of score maps, $17 \times 17 \times 1$ and $255 \times 255 \times 1$ respectively. After that, displacement vector is scaled by $\frac{x\_sz}{search\_sz}$ in order to transform displacement vector to original frame space. Here *search_sz* equals to search patch size which equals to 255 and *x_sz* is the size of the image patch in original frame space. After scaling the displacement vector, one finally obtains displacement in original frame space.

Object width and height is scaled by using selected best scale. Weighted average of

original size and scaled size is used in order to obtain object size in search image.

$$W = (1 - \alpha)W + \alpha W S^* \tag{3.7}$$

$$H = (1 - \alpha)H + \alpha H S^* \tag{3.8}$$

where W denotes width, H is for height and S^* is the best scale.

Here α is named as scale learning rate and equals to 0.59 in original implementation.

Object being tracked is located in search image at this point. Exemplar embeddings that represents a template for tracking object is updated as a last step. Initialization step in Figure 3.3 is executed again in order to calculate exemplar embedding for object. Updated embedding is calculated by using weighted average of older and newly calculated embedding:

$$E^{'} = (1 - \beta)E + \beta E^* \tag{3.9}$$

where $E^{'}$ is updated embedding, E is the older embedding, E^* is the newly calculated embedding, and β is the weight for averaging also named as template learning rate.

Tracker becomes more robust by updating the template embedding. Any pose, scale and texture changes in object or light changes in scene is captured and saved in exemplar embedding by updating it in every iteration. Template learning rate is selected as 0.01 in original implementation. This value points to a slow update in embedding. Effects of changing values of β is not stated in paper [3].

3.2 Object Tracking via A Pretext Task: Video Colorization

3.2.1 Introduction

Training a neural network that aims to track objects needs labeled video frames in a supervised learning framework. In order to label a video every frame needs to be annotated by hand which takes quite long time and needs extensive amount of human labor. Besides that labeling is an error prone process.

Table 3.2: Common Object Tracking Datasets in Literature

Dataset Name	**Number of Samples**	**Number of Labels**
ImageNet VID Dataset [41]	3862	866870
GOT-10k Dataset [43]	10000	1447200
OTB Dataset [1]	100	61977
VOT Dataset [5]	60	19935

Table 3.2 contains datasets that is used frequently in object tracking literature. Although the largest dataset, named GOT-10k [43], has nearly 1.5M manually annotated frames it only has 10.000 video samples only. On the other side ImageNET dataset [41] which is extensively used in image classification literature has 14M samples.

The huge difference in total sample numbers that different datasets have points to the fact that more accurate object trackers may be trained by increasing the total number of samples. However, due to the difficulties explained before, developing such a dataset is unfeasible. This causes the number of samples in a dataset still sets an upper bound for more successful state-of-the-art object tracking models.

Self-supervised learning is seen as a solution for the problem defined and some different methods have proposed in literature. One example is using frame sequence validation as a pretext task in order to learn a model for object tracking problem. Studies performed by Misra et al. [17] and Fernando et al. [16] are examples to this approach. A model is trained in order to validate whether the given sequence of images are in correct temporal order in method proposed by Misra et al. [17]. This way the model learns to differentiate between small motion differences across frames. It is shown that the specified model is successful in object tracking problem and used as a pretext task. In addition, authors reported that the trained model improves the success rate when used as a pretraining step. A similar pretext task is proposed in work done by Fernando et al. [16]. The model is trained in order to select the video sequence that is incorrectly sequenced among a set of video sequences.

Video colorization is another pretext task that is used in order to learn object tracking problem as a downstream task. A typical video has temporal color coherency so that

a model that is capable to colorize grayscale videos also has implicit knowledge to solve both of the segment tracking and object tracking tasks.

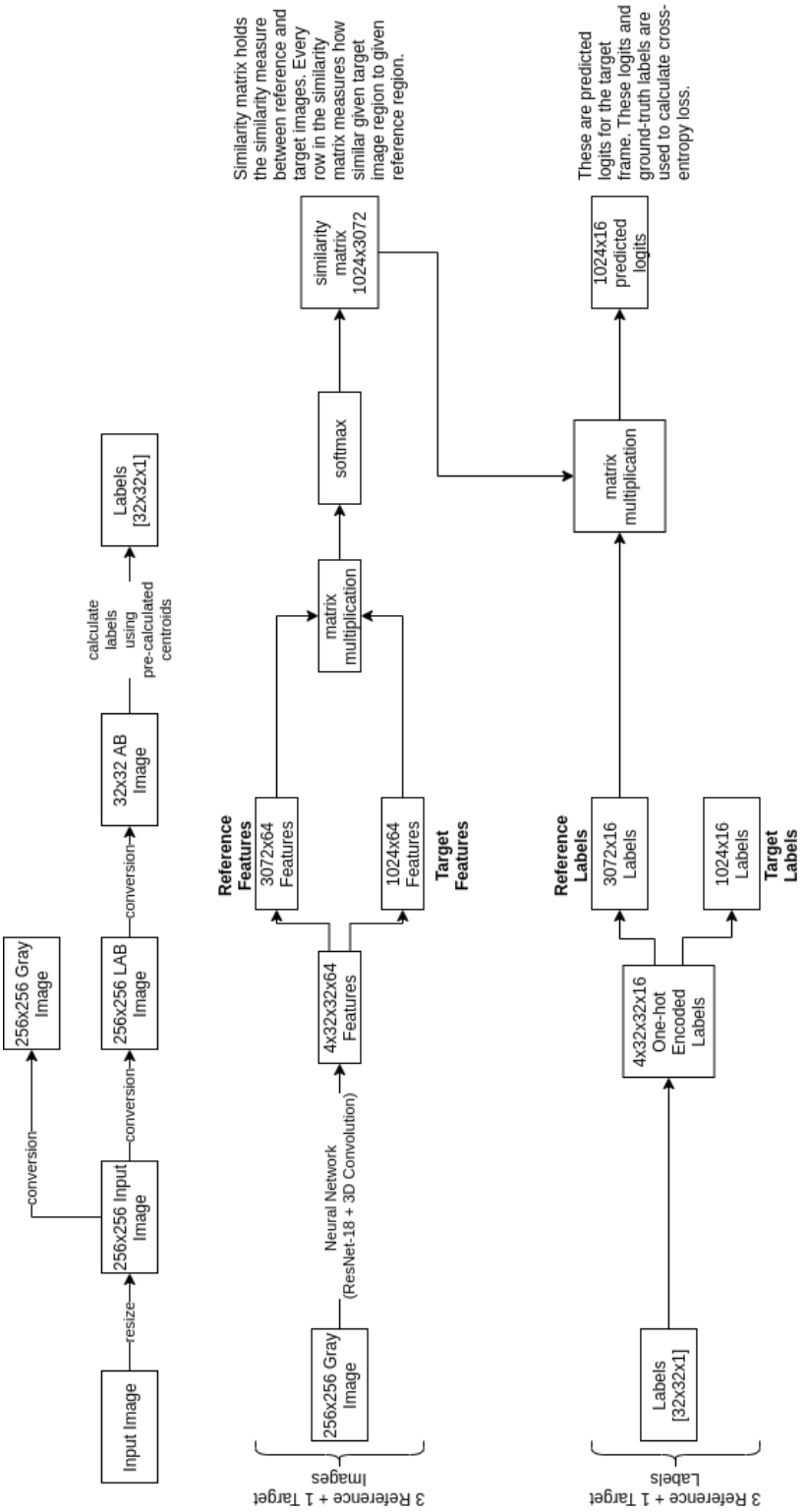


Figure 3.6: Architecture of Method: Tracking Emerges by Colorizing Videos

Methods that expose self-supervised learning framework may break the upper bound threshold introduced by dataset size and new state-of-the-art models in object tracking problem may be proposed. By using huge amount of videos that naturally exists on the internet, new state-of-the-art object trackers can be developed that perform better than methods that use supervised learning framework. Every day, 500 hours of video is uploaded to YouTube by people as of May'19 [44]. A self-supervised learning framework make usage of huge amount of videos possible by avoiding excessive amount of annotation labor.

One example that exposes this phenomenon is worked by Vondrick et al in the work named as *Tracking Emerges by Colorizing Videos* [4]. Video colorization is selected as a pretext task and the work shows that the learnt model achieves acceptable performance in video object segmentation problem as a downstream task. This method will be explained in detail in following section.

3.2.2 Method

The proposed method calculates a similarity matrix between a set of reference frames and a target frame. The matrix contains similarity information between different spatial positions of input images. Using this similarity information, color of reference frame can be copied to target frame. Figure 3.7 and Figure 3.6 explain the overall idea.

Copying colors from reference frame to target frame is defined by following formula:

$$y_j = \sum_i A_{ij} c_i \tag{3.10}$$

where y_j is copied (predicted) color labels for target frame, A_{ij} is the similarity matrix between reference frame and target frame, and c_i is the color labels for reference frame.

Similarity matrix is calculated by using inner product similarity normalized by softmax:

$$A_{ij} = \frac{\exp(f_i^T f_j)}{\sum_k \exp(f_k^T f_j)} \tag{3.11}$$

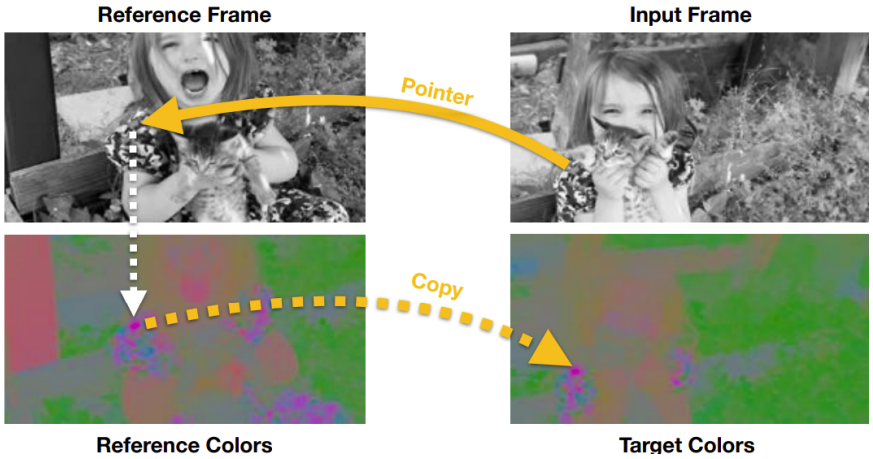


Figure 3.7: Working Principle of Method: Tracking Emerges by Colorizing Videos (taken from original work [4])

where f_i, f_j are embbeddings calculated by a neural network. Embeddings are learnt in a way so that inner product of two embeddings for reference and target frames outputs how much similarity exists for every possible spatial location. Figure 3.8 shows an example to the similarity matrix for a specific case in which there exists three reference frames and one target frame. Every spatial location of the matrix holds the similarity value between a specific reference frame and the target frame.

Figure 3.9 shows a real example of similarity matrix. In this example there exists one reference frame and one target frame. When the given two frames are similar to each other, one expects mostly a diagonal matrix as in this example. On the other hand, when given two examples are not similar to each other, the similarity matrix will not be a diagonal one. A diagonal matrix is expected because of the spatio-temporal coherency in sequence of frames. Spatio-temporal coherency exists between frames due to the fact that both color and scene is not change rapidly among the frames of a video.

Applying the softmax function to inner similarity product enlarges the differences in a specific row of similarity matrix. The highest column in a specific row points to the spatial location where the highest similarity exists. This is the specific spatial location where the color is copied from reference frame to target frame. This way gray scaled target frames are colorized by using the color information from reference frame.

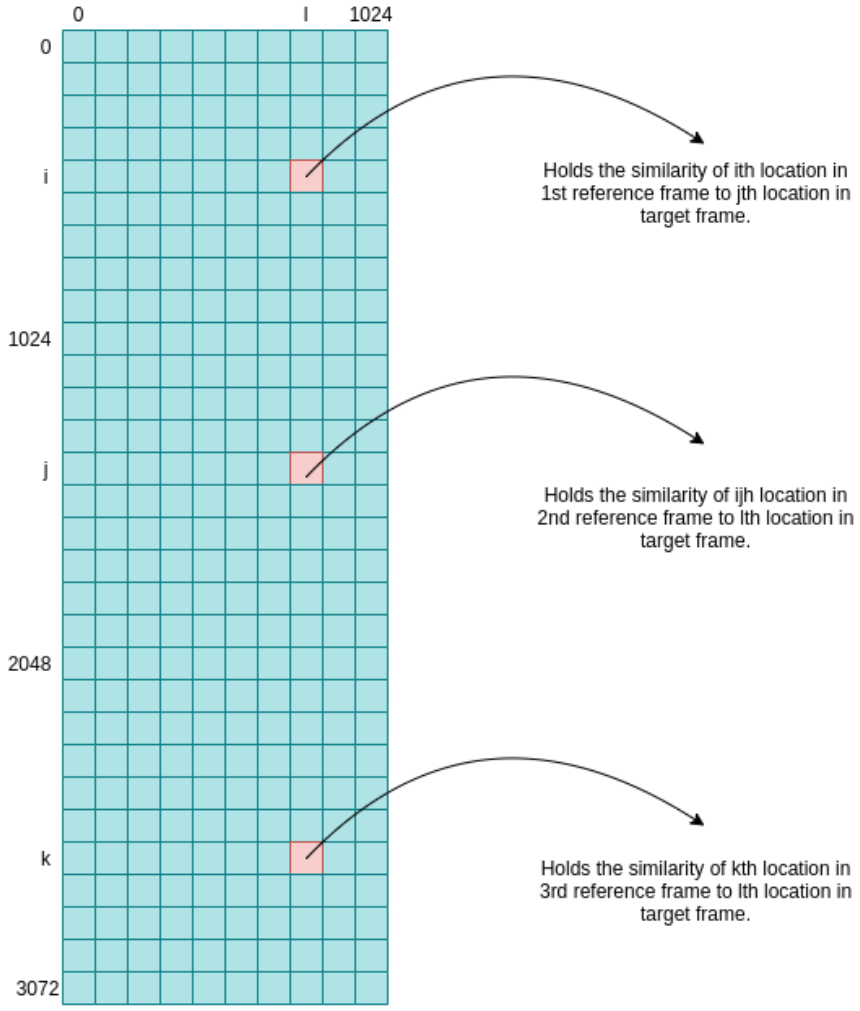


Figure 3.8: Similarity matrix holds the similarity score for the spatial locations of reference and target images.

Similarity matrix plays a key role in this method. Calculation of a robust similarity matrix is possible with existence of robust embeddings. Robust embeddings are calculated by a function φ which is implemented by a neural network.

The parameters θ of the function φ is obtained by the supervisory signal that is generated by multi-class cross entropy loss function between predicted labels and ground truth labels. Loss function will be detailed in Section 3.2.5.

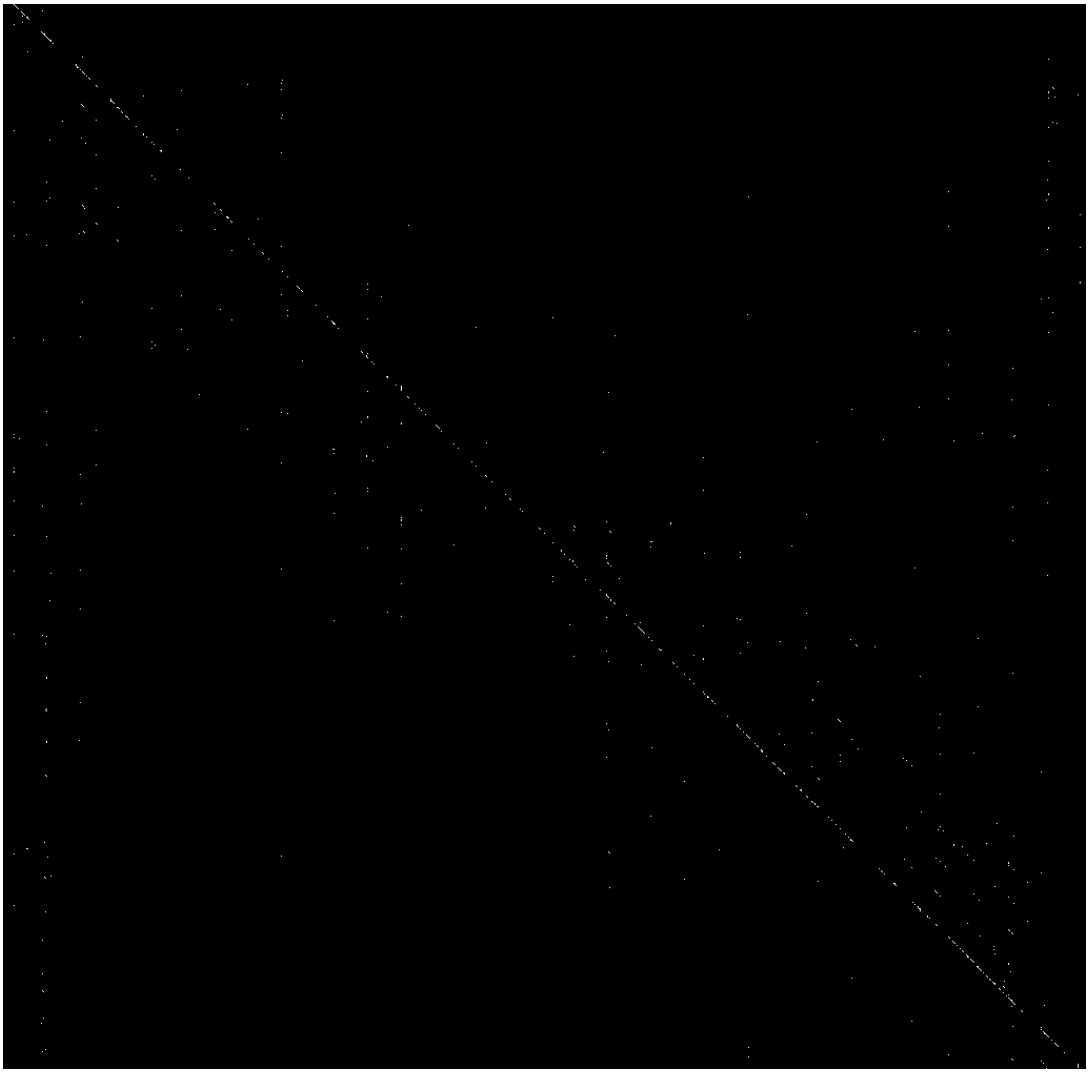


Figure 3.9: An example of similarity matrix for 1 reference frame and 1 target frame

3.2.3 Network Architecture

A modified version of ResNet-18 [18] architecture is followed by a five layer 3D convolutional neural network. Additional features are concatenated between the ResNet-18 and 3D convolutional neural network in order to encode the spatial information. These features are between [-1, 1] and encodes the spatial location in both horizontal and vertical directions. Figure 3.10 shows an example to spatial location encoding features for $[8x8]$ dimensions. The function φ specified in Section 3.2.2 is composition of modified ResNet-18 and 3D convolutional network.

Table 3.3 shows the neural network architecture used. The ResNet-18 architecture is

Table 3.3: Neural Network Architecture

	Filter Type	Filter Size	Number of Filters	Stride	Dilation
ResNet-18 [18]	Convolutional	7×7	64	2	1×1
	MaxPool	3×3	N/A	2	1×1
	Residual	3×3	64	2	1×1
	Residual	3×3	128	1	1×1
	Residual	3×3	256	1	1×1
	Residual	3×3	256	1	1×1
3D Convolutional Network	Convolutional	$1 \times 3 \times 3$	256	1	$1 \times 1 \times 1$
	Convolutional	$3 \times 1 \times 1$	256	1	$1 \times 1 \times 1$
	Convolutional	$1 \times 3 \times 3$	256	1	$1 \times 1 \times 1$
	Convolutional	$3 \times 1 \times 1$	256	1	$1 \times 2 \times 2$
	Convolutional	$1 \times 3 \times 3$	256	1	$1 \times 1 \times 1$
	Convolutional	$3 \times 1 \times 1$	256	1	$1 \times 4 \times 4$
	Convolutional	$1 \times 3 \times 3$	256	1	$1 \times 1 \times 1$
	Convolutional	$3 \times 1 \times 1$	256	1	$1 \times 8 \times 8$
	Convolutional	$1 \times 3 \times 3$	256	1	$1 \times 1 \times 1$
	Convolutional	$3 \times 1 \times 1$	256	1	$1 \times 16 \times 16$
	Convolutional	$1 \times 3 \times 3$	256	1	$1 \times 1 \times 1$
	Convolutional	$3 \times 1 \times 1$	64	1	$1 \times 1 \times 1$

	0	1	2	3	4	5	6	7
0	-1.00000	-1.00000	-1.00000	-1.00000	-1.00000	-1.00000	-1.00000	-1.00000
1	-0.75000	-0.75000	-0.75000	-0.75000	-0.75000	-0.75000	-0.75000	-0.75000
2	-0.50000	-0.50000	-0.50000	-0.50000	-0.50000	-0.50000	-0.50000	-0.50000
3	-0.25000	-0.25000	-0.25000	-0.25000	-0.25000	-0.25000	-0.25000	-0.25000
4	0.25000	0.25000	0.25000	0.25000	0.25000	0.25000	0.25000	0.25000
5	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000
6	0.75000	0.75000	0.75000	0.75000	0.75000	0.75000	0.75000	0.75000
7	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000

	0	1	2	3	4	5	6	7
0	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
1	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
2	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
3	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
4	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
5	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
6	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000
7	-1.00000	-0.75000	-0.50000	-0.25000	0.25000	0.50000	0.75000	1.00000

Figure 3.10: Encoding spatial information vertically and horizontally

modified by removing the global average pooling layer and softmax layer so that it outputs features in dimensions $32 \times 32 \times 256$.

3D convolutional layers are used in order to capture spatio-temporal relations between frames of videos. This part of network causes the embeddings to capture similarities across the frames.

Input frames to the network is in dimensions of $256 \times 256 \times 1$ and the network outputs embeddings in dimensions of $32 \times 32 \times 64$. Reference and target frames are fed to the neural network φ and two embeddings belonging to reference and target frames are calculated. These embeddings are then used to calculate similarity matrix between reference and target frames.

3.2.4 Casting Video Colorization Problem to Some Downstream Tasks

Colorization task is performed by using the similarity matrix. The following formula defines the task mathematically as mentioned in Section 3.2.2:

$$y_j = \sum_i A_{ij} c_i \tag{3.12}$$

Color labels are fetched from the first frame and then this information is propagated

over the video by copying the color information in a video colorization downstream task. Every n^{th} frame is used as reference frame relative to every $(n+1)^{th}$ frame. Then the color label is copied by calculating the similarity matrix between frame pairs n^{th} and $(n+1)^{th}$.

Some other downstream tasks are defined within the work. Feature tracking is one of them. Binary reference labels, c_i, are used in this case. The framework tries to predict target labels whether the current pixel is the given feature point or not. Another downstream task is human body pose tracking. Similar to feature tracking task, human body pose tracking is performed by casting the labels to binary domain. Last example is the segment tracking task. Initial ground truth segments are labeled and prediction includes the information whether a spatial location in target frame belongs to a segment or not.

The segment tracking pretext task, which is the main consideration for this thesis work is evaluated on DAVIS [45] dataset in original work. This dataset contains ground truth segmentations for videos. The trained model is fed by first frame's ground truth segmentation and then this segment mask is propagated to consecutive frames by using the similarity matrix. Figure 3.11 explains how similarity matrix is used in order to track the segments across the images.

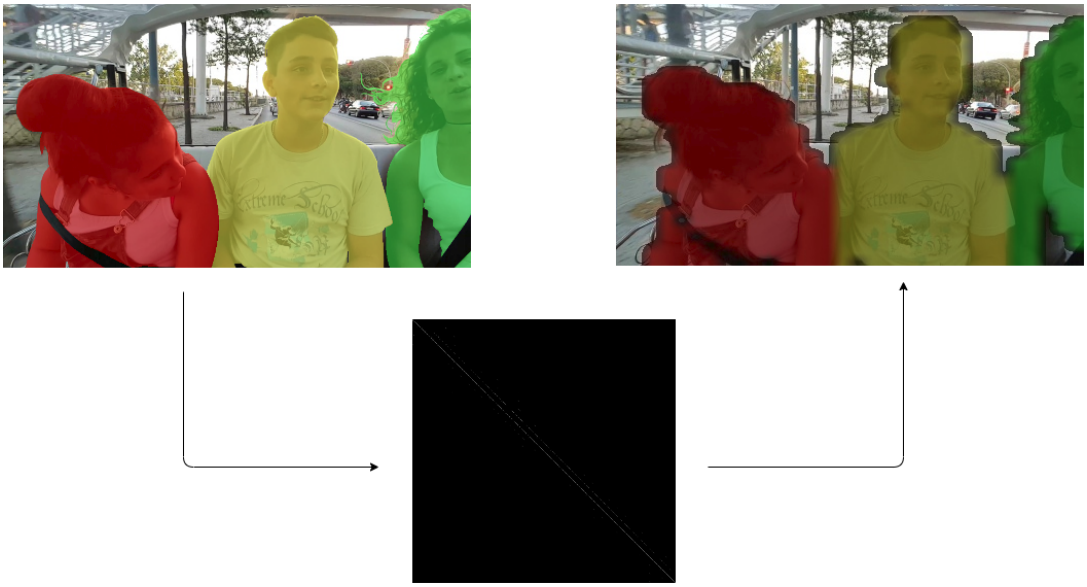


Figure 3.11: Tracking Segments as a Downstream Task

3.2.5 Training

Training is performed by using the unlabeled video data. Kinetics-700 [46] is a high volume dataset that is originally used for human action recognition. Dataset contains of 700 classes of human action and there are nearly 600 videos for each action class. Every video is obtained from YouTube and nearly a 10 second part of each video is used within the dataset. A video has only one label which holds the human action class information. There are approximately $650k$ videos in the dataset. Videos are $25fps$ in average. This means there are approximately $162.5M$ images in the dataset.

Training is performed in a self-supervised manner. Color labels are obtained from the dataset itself. Every pixel is labeled by using the k-means algorithm in the AB color space. 16 clusters are used in the original work. Before obtaining the labels, the specified 10-second-region is extracted from the original video. Extracted frames are first resized to 256 pixels in width by keeping the original aspect ratio and then cropped to obtain 256×256 images. Batch size of 32 is used. $400k$ iterations are performed. For the first $60k$ iterations learning rate is set to 0.001. For the remaining iterations learning rate is set to 0.0001. ADAM optimizer is used during the training. The network is initialized randomly by gaussian noise. All of these preprocess steps and hyperparameters are used in the original work. Differences in this thesis work from the original work is explained later in Chapter 5.

Figure 3.12 shows an example for color labels. A unique color is assigned to each label. Visualization is performed by using these color labels. Each group of figures shows three reference images and one target image. Here 8 clusters are used in k-means clustering. Figure shows not only the color labels but also an evidence for color coherency among the sequence of images in a video.

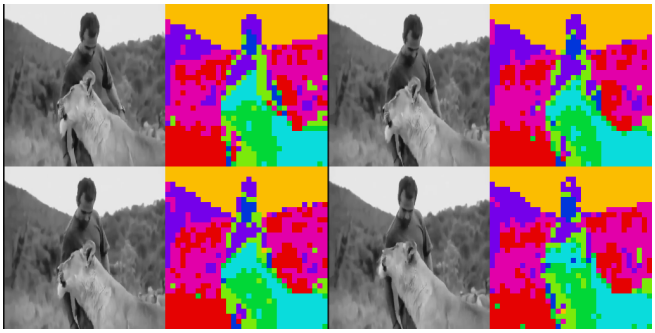
Loss function is calculated by using multi-categorical cross entropy loss function between predicted color labels, y_j and ground truth color labels, c_j:

$$\underset{\theta}{\operatorname{argmin}} \sum_{j} L(y_j, c_j) \tag{3.13}$$

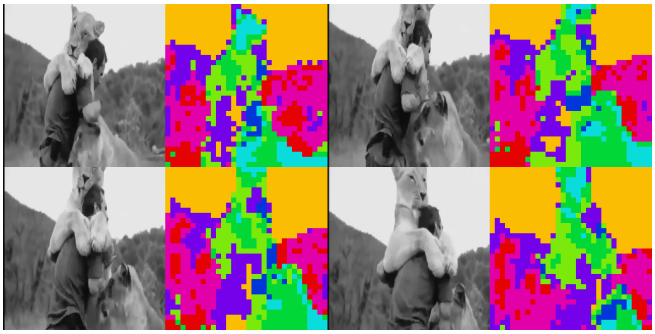
where loss function is defined by:

$$L(y_j, c_j) = -\sum_{c=1}^{N} c_j \log(y_j) \tag{3.14}$$

Ground truth color labels, c_j, is calculated by using k-means clustering algorithm. Original RGB colored images are first converted to LAB color space and then, AB channels are used in order to calculate the cluster centroids. Then, each pixel is labeled by using the euclidean distance to the centroids. y_j is the predicted color label by using the similarity matrix. Multi-class cross entropy loss measures how similar the predicted labels to the ground truth color labels. This loss function supervises the network to learn a similarity metric between reference and target frames by using video colorization as a pretext task.



(a)



(b)

(c)

Figure 3.12: Visualization of color labels for k=8 (a) 0^{th} sample taken from video (b) 6^{th} sample taken from video (c) 14^{th} sample taken from video

CHAPTER 4

PROPOSED METHOD

4.1 Introduction

The aim of this thesis work is to systematically analyze the effect of a model that is trained in the video colorization framework in a self-supervised manner. We combined two methods, one uses siamese networks [15] for object tracking and other one uses a self-supervised learning methodology in order to learn video colorization.

In Section 3.1.4 it was mentioned that the amount of data used for training is important when training the deep neural networks for the purpose of object tracking task. However, object tracking community is lack of massive training datasets because preparing such a dataset is error prone and a costly effort. Using self-supervision when learning for object detection may alleviate the problems related to datasets since the usage of labeled dataset is avoided within this framework. Instead, required data labels are obtained from the dataset itself.

Different approaches to object tracking problem is argued in Chapter 2. The work done by Bertinetto et al. named as Fully Convolutional Siamese Networks [3] is one of them and to the best of my knowledge it is the first proposed method that uses the siamese architecture for object tracking. This method is selected as baseline because of its simplicity and easiness of extension with a self-supervised backbone. However, as it is mentioned in Chapter 2 there are some other variants to SiamFC method which extend the architecture in various ways. These methods can also be extended by a self-supervised backbone. Nevertheless, extension effort is directly proportional to the complexity of siamese architecture used. So that, using SiamFC is a better choice for the scope of this thesis work.

Proposed method integrates the self-supervised trained colorization model as a backbone to the baseline method. Figure 4.1 shows the architecture for proposed method. ω represents the model that is trained within self-supervised colorization framework. Training will be detailed in Section 3.2. φ represents the model that is trained by using the baseline method in a supervised manner. Training is detailed in Section 5.2.2. α represents the averaging weight that is used when unifying the two score maps calculated by two backbones. A weighted unification is implemented in order to perform some experiments which are detailed in Chapter 5. When $\alpha = 0.0$ only the embeddings calculated by the siamfc backbone is used and when $\alpha = 1.0$ only the embeddings calculated by the colorization backbone is used. Two score maps are calculated by using the cross correlation operator as before. The exemplar and search embeddings calculated within siamfc and colorization backbones are the input to the cross correlation operator.

4.2 Motivation

Main motivation of the proposed method is to cope with the enormous training data requirements of supervised learning framework. So we combined a self-supervised colorization method to a siamese tracker to analyze the effects of the self-supervision to the problem. Annotated training data requirement is avoided by using self-supervision and every video published on the internet becomes a training data candidate for an object tracking learning problem. It is expected to train a robust object tracker using unlabeled video and alleviate problems related with labeled data requirement in object tracking. The largest dataset in object tracking literature is found to be Got-10K [43] as specified before. This dataset has nearly 1.5M images; however, it has only 10k different samples. On the other hand, ImageNet [41] dataset which is being used for image classification problem in the literature has nearly about 14M distinct images for 1000 object classes. This comparison motivates us to search ways to make usage of bigger datasets possible in object detection problem. Learning video colorization problem in a self-supervised manner not only avoids data amount restriction but also make usage of every video as a training data possible. Such a dataset serves rich information for learning changes in object orientation, scale, and appearance.

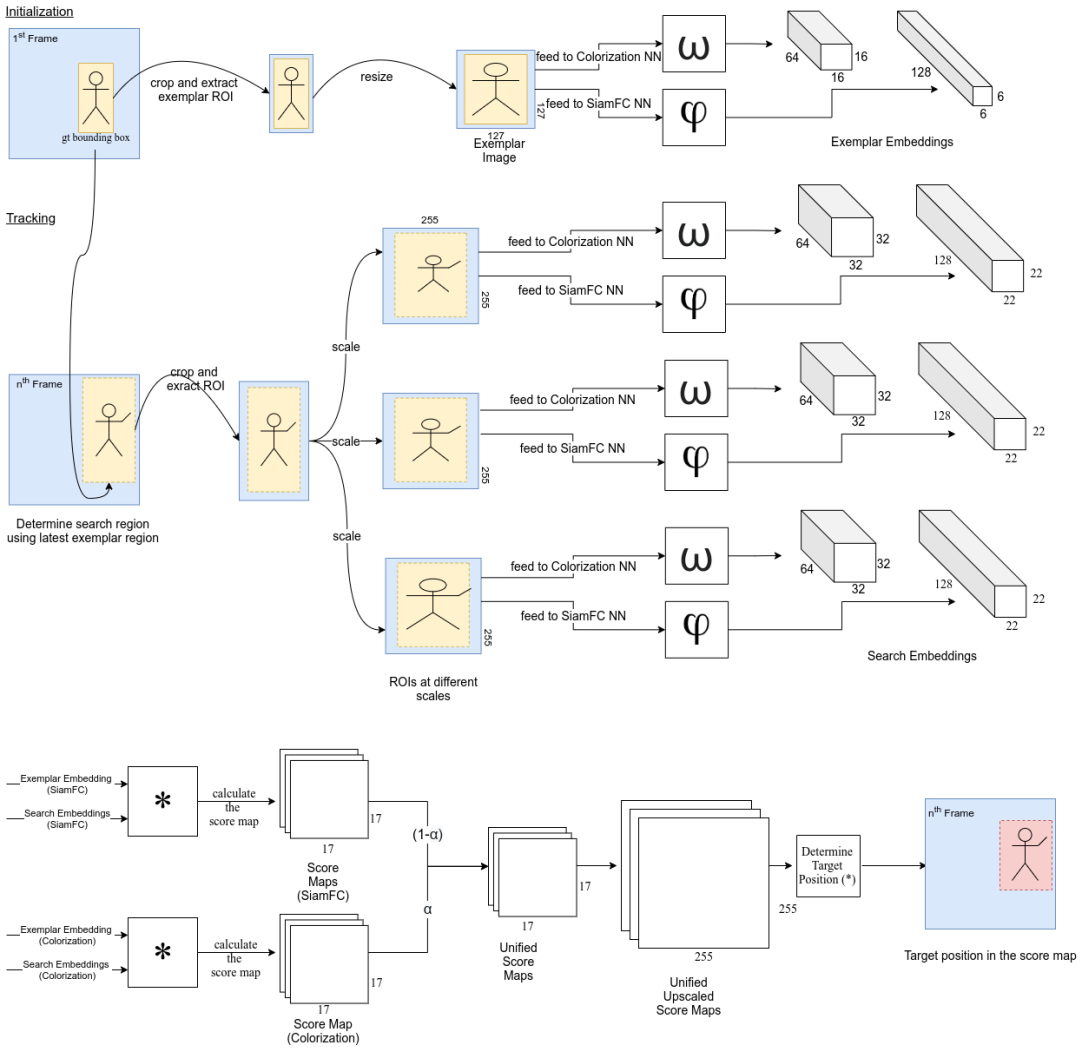


Figure 4.1: Proposed Architecture

Secondly, we believe that an object tracker should be object agnostic. An object tracker system may need to handle cases in which any random object is being requested to be tracked by a user of such system in real life. Nonetheless, many approaches in the literature exploits the objectness of an object that is intended to be tracked. The orientation, scale, and appearance of an object changes rapidly in real life examples. Even changes in scene light and motion blur due to the camera movements affect the appearance of an object without any orientation changes in the object. Trying to solve these real life problems with features that are robust for determining objectness is not a solution. These features are better for object detection, exposing the objectness, but they are not very good at handling changes to an object in a video. Instead, features that are specialized to track objects without any objectness informa-

tion are key to solve this problem. Features shall be strong for determining object orientation, scale, and appearance changes.

Video colorization problem is related with object tracking problem because a candidate solution to video colorization problem needs to learn the color coherency and how the color changes among the frames. This is implicitly similar to learn object movements and scene movements among frames. This way a supervised object tracking may be implemented by using unlabeled training data.

4.3 Method

Two different backbones, the SiamFC and colorization, shown with different functions φ and ω respectively produces two different embeddings and then two score maps are calculated by using these embeddings. These two score maps, then, be unified by using the parameter α.

AlexNet [19] is used with the same modifications made in original work done by Bertinetto et al. [3] in backbone named as φ. Network details can be found in Table 3.1 in *Modified AlexNet Architecture* column. In addition, the same network is also used as it is in original work done by Vondrick et al. [4] in backbone named as ω. Network details can be found in Table 3.3.

Normalization of two score maps are needed because they have different scales in test dataset. Because of this, a normalization function is run before unifying the two score maps. Data normalization is done in a way so that both score maps has zero mean and unit variance. The mathematical formula is as follows:

$$U = (1-\alpha)\textbf{Norm}(S_1, \mu_1, \sigma_1) + \alpha\textbf{Norm}(S_2, \mu_2, \sigma_2) \tag{4.1}$$

where U denotes the unified score map, S_1 and S_2 denotes the score maps calculated from SiamFC and colorization methods respectively, $Norm()$ function performs the data normalization and μ and σ is the mean and variance values calculated for two score maps for a specific dataset respectively and α is the averaging constant as mentioned before. Score map unification will be detailed in Chapter 5. Only SiamFC

network or only Colorization network is used in extreme cases where $\alpha = 0$ and $\alpha = 1$ respectively.

SiamFC network is trained in a supervised manner and the colorization network is trained in self-supervised manner without labeled data. The output embeddings which are used to calculate similarity matrix holds similarity information. SiamFC framework also learns a similarity metric between two input signals as specified in Section 3.1. Since both networks aim to learn a similarity metric between different inputs, colorization network can be used within SiamFC framework.

Some downstream tasks were defined and segment tracking task is one of them as mentioned in Section 3.2. So why is integration to SiamFC needed? Why colorization pretext task alone is not sufficient for tracking problem? Because, SiamFC framework provides a generic object tracking framework. It is able to handle light changes, scene changes, and changes in object, ie. deformations or pose and size changes itself by keeping a template belonging to object in each iteration. Segment tracking with a similarity matrix is not able to solve these kind of problems. These abilities of SiamFC framework makes it reasonable to integrate the colorization framework.

CHAPTER 5

EXPERIMENTS AND RESULTS

5.1 Introduction

We performed some experiments in order to evaluate the proposed method and the trained models within the original works of both Bertinetto et al. [3] and Vondrick et al. [4]. Experiments will be detailed in the following sections.

First of all, we worked for determining a baseline metric. Then we implemented and trained the SiamFC [3] architecture. We compared the performance of our model with the original work[3]. At this point, a baseline metric is obtained and used through this work. Second, we implemented and trained a self-supervised colorization network. Again, we compared our model with the results obtained in original work [4]. Finally, we implemented a combined network as detailed in 4. Various experiments are performed in order to analyze the combined network. Details of these experiments are given in Section 5.4.

5.2 Experiments with SiamFC Implementation

5.2.1 OTB Evaluation

OTB [1] is a benchmark that is first proposed in CVPR 2013. After first proposal, named 2013 version of benchmark, new sequences are added to benchmark and named as 2015 version. 2013 version has 50 sequences and 2015 version has 100 sequences. Some samples are inherited from 2013 version and some are completely new samples in 2015 version. 2015 version has two distinct sequences named as

Table 5.1: Table of 9 attributes to label sequences in OTB Benchmark [1]

Attribute	**Explanation**
Illumination Variation (IV)	The illumination in the target region is significantly changed.
Scale Variation (SV)	The ratio of the bounding boxes of the first frame and the current frame is larger than some threshold scale (ts>1) (ts=2).
Occlusion (OCC)	The target is partially or fully occluded.
Deformation (DEF)	Non-rigid object deformation.
Motion Blur (MB)	The target region is blurred due to the motion of target or camera.
Fast Motion (FM)	The motion of the ground truth is larger than some threshold (tm) pixels (tm=20).
In-plane Rotation (IPR)	The target rotates in the image plane.
Out-of-plane Rotation (OPR)	The target rotates out of the image plane.
Out-of-view (OV)	Some portion of the target leaves the view.
Background Clutters (BC)	The background near the target has the similar color or texture as the target.
Low resolution (LR)	The number of pixels inside the ground-truth bounding box is less than some threshold (tr) pixels (tr =400).

TB50 and TB100, TB100 has additional 50 sequences to TB50 sequence. 2015 version consists of more challenging sequences than 2013 version from the perspective of tracking algorithms.

OTB benchmark has 9 attributes for tagging the sequences. These attributes are explained in detail in Table 5.1.

Evaluation of a tracker in OTB benchmark is performed by plotting precision and success curves and calculating AUC for a given tracker. Precision curve is constructed by calculating the average euclidean distance between center of the object determined by tracker and given in ground truth. This calculation is performed for every frame in a sequence. Precision curve is plotted for some thresholds set as a threshold for euclidean distance. Hence, the x axis holds the threshold values and the y axis holds the percentage of frames in which the estimated center of object is within the threshold. Once the plot is obtained AUC is calculated to get a quantitative result. In addition to precision curve, success curve investigates the IoU between predicted bounding box and ground truth bounding box. Success plot shows the ratio of successful frames

to the total number of frames given in a sequence for a given threshold similar to precision curve.

Three distinct evaluations are performed within the benchmark. The first one is named as OPE. In OPE evaluation, tracker is run once on the test data set. The second one is TRE. In this evaluation, tracker is initialized by using not the first frame but with different frames in sequence. Trackers may behave differently for different initialization conditions so this evaluation evaluates the robustness of tracker to temporal changes. The last one is SRE. In this evaluation, tracker is initialized with bounding boxes that is slightly different from the ground truth. Ground truth bounding boxes are perturbed by shifting and scaling the bounding box. Some trackers may affected by different initialization routines. For example, an object detector is used in order to initialize the tracker. Effects of different initialization schemes is measured by using this evaluation.

A work performed by Kristan et al. [47] states that any evolution that is based on object center measurement is highly brittle. It is also stated that overlap based measures should be preferred. Because of this, we use success curves, which are based on overlap measure, calculated within OTB benchmark for performance comparison between different trackers.

5.2.2 Experiments

We implemented the SiamFC architecture using python scripting language without any changes to the original work done by Vondrick et al. [3]. PyTorch [48] is used to code neural network and training routines.

We used a modified AlexNet [19] architecture. All the paddings are removed in order to obtain translationally invariant convolutional networks. The network architecture was given in Table 3.1 in column *Modified AlexNet Structure*.

We used GOT-10k [43] dataset for training different from the original work. Original work uses ImageNet VID [41] dataset. GOT-10k dataset has 9335 samples in training set. This nearly equals to 1403359 labeled bounding boxes. We chose GOT-10k because it is relatively a newer dataset and got more usage day to day by object tracker

community. A new model is trained with the same hyperparameters and configuration specified in original work. The network is trained for 50 epochs, batch size is 8, learning rate is 10^{-2} and decreased gradually to at least 10^{-5}. Figure 5.1 shows the learning curve for training.

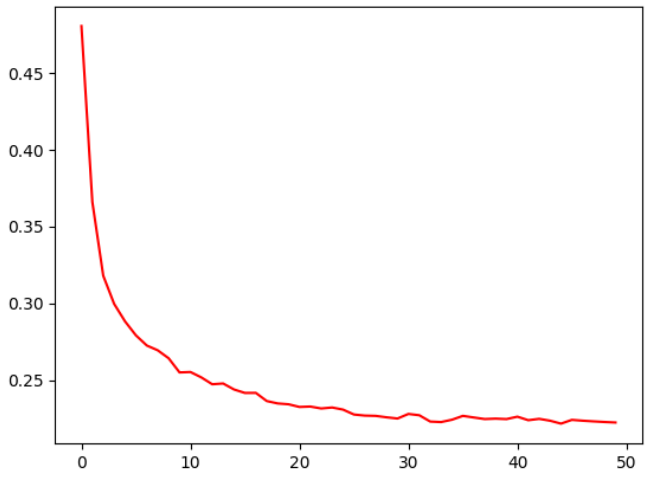
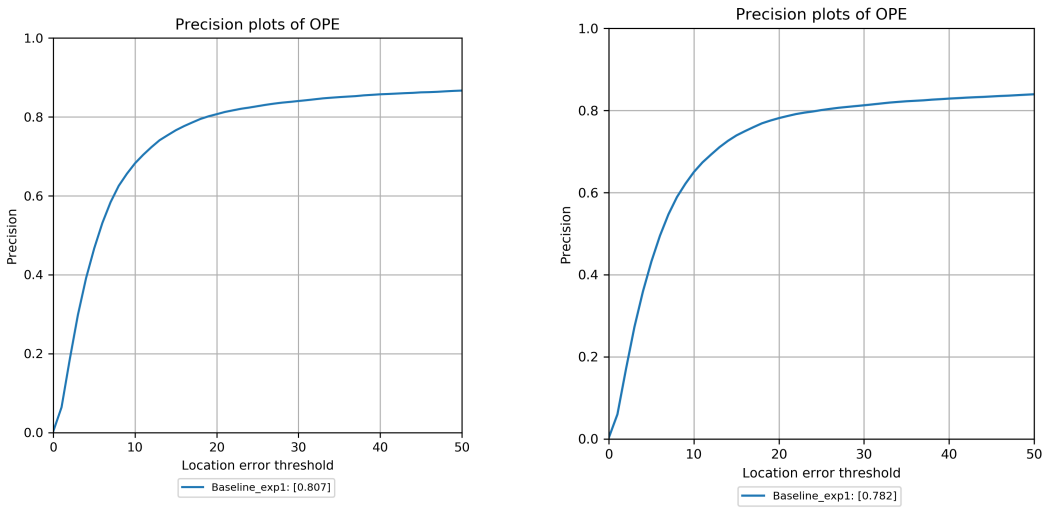


Figure 5.1: Learning Curve for SiamFC Model

We used OTB version 2013 dataset [1] in order to evaluate the model as in original work. Precision and success curves in Figure 5.2 and 5.3 are obtained respectively. Only success curve and related AUC value is reported in original work. Hence, we compared our model with the original work by using only the success curve. Original model get success curve AUC score as 0.612 whereas our model get the same metric as 0.601. SRE and TRE evaluations are not performed because OPE evaluation is enough to make a decision for using the trained model. We decided to use this model as a baseline in the scope of this thesis work since our model shows similar performance with the reported model. The same evaluation is performed on OTB version 2015. Precision and success curves are plotted. AUC is calculated as 0.782 and 0.584 for precision and success curves respectively. Since OTB version 2015 contains more challenging sequences such a drop in performance is normal. We use OTB version 2015 benchmark in experiments since it has more challenging sequences.



(a) OTB version 2013 [1]

(b) OTB version 2015 [1]

Figure 5.2: Precision Curves for Trained SiamFC Model

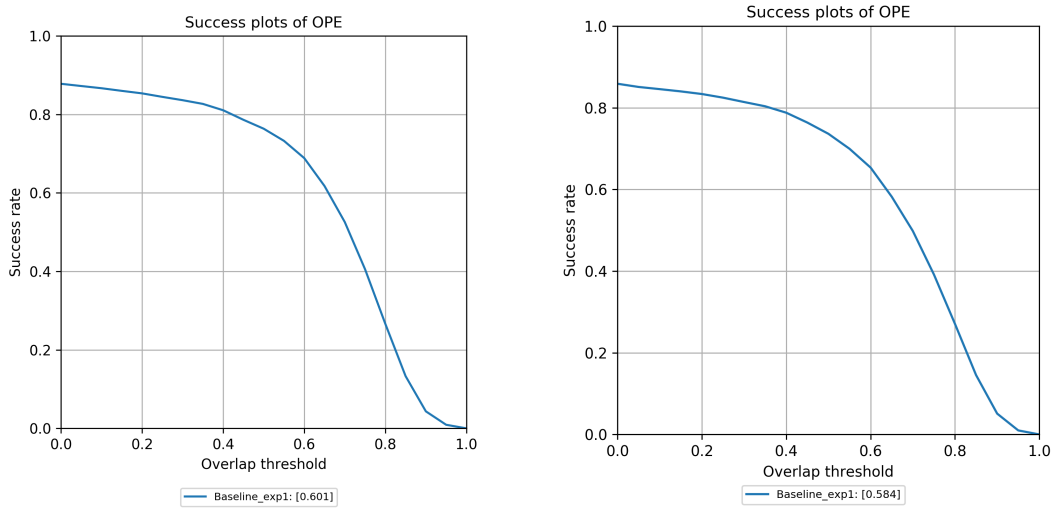
5.3 Experiments for the Colorization Framework Implementation

We implemented and trained a model for video colorization [4] by using python scripting language. PyTorch [48] is used for neural network and loss function implementations.

5.3.1 Preparing data for training

We used Kinetics-700 [46] dataset for self-supervised training. We trained a new model since no open-source pretrained model is found online. The Kinetics-700 [46] is delivered with a JSON file. This file consists of 544886 YouTube video links with additional action class information and a video interval definition consisting of start and end time points. A $10 - seconds$ interval is defined for every video. Every video is clipped according to start and end time points in order to obtain the correct interval in the preprocessing step. Dataset is very huge and only 280859 videos in total can be downloaded and preprocessed in this thesis work.

K-means clustering algorithm is used to calculate the centroids. We used a GPU implementation of k-means clustering algorithm in order to make processing faster.



(a) OTB version 2013 [1]

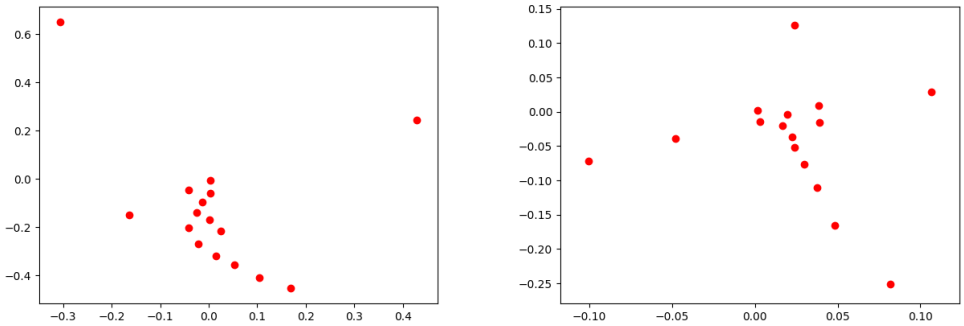
(b) OTB version 2015 [1]

Figure 5.3: Success Curves for Trained SiamFC Model

Every RGB video frame is first converted to LAB colorspace and then A and B channels are used for clustering. Every video is clustered in its own, in other words every video has its centroids and labels accordingly. This method is used because every video has its unique color distribution. Each pixel is labeled by calculating the distance to the nearest centroid. Euclidean distance is used, no limit is set for maximum number of iterations but a threshold for tolerance is set to 10^{-4} in order to stop at the convergence. We set $k = 8$, however $k = 16$ is used in the original work [4]. Effects of number of total clusters will be explained later in this section.

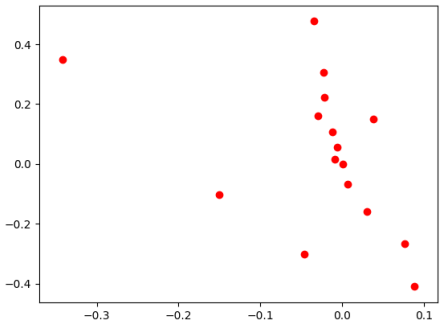
Data is preprocessed before calculating the clusters. Firstly, histogram equalization is applied to frames. This is different from original work. Histogram equalization is performed in order to obtain well separated centroids. Separation of centroids is important because close centroids cause problems in learning phase. This statement will also be detailed later in this section. Secondly, every frame is resized so that width of the image becomes 256 pixels. This operation is performed without changing the aspect ratio. Thirdly, images are center cropped so that every image is 256 pixels in width and height. Mean and variance values for all of the channels of LAB colorspace are calculated for the overall dataset. Mean values are calculated as 0.4388, 0.5171, 0.5242 and standard deviance values are calculated as 0.2242, 0.0312, 0.0416

respectively for L,A,B channels. Finally, normalization is performed on the dataset by using specified mean and standard deviance values.

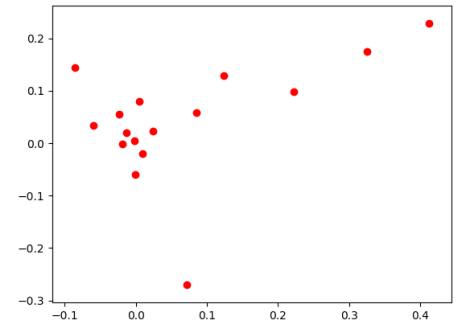


(a) Video sample *-0MLLn0Zg1M*

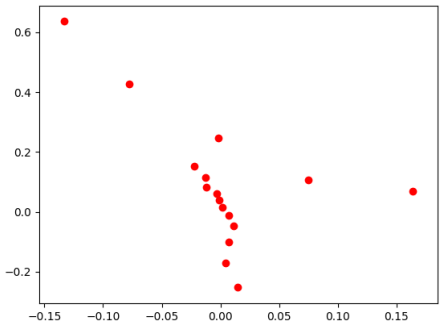
(b) Video sample *-9RiCLmXfWc*



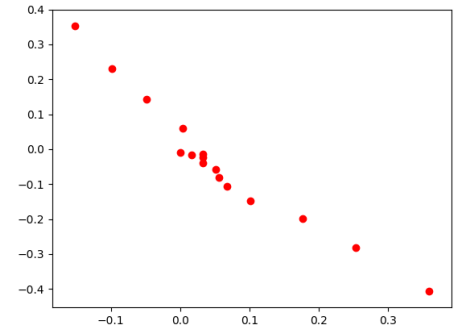
(c) Video sample *-2-KjcZEmrM*



(d) Video sample *-3pjDfx73Aw*

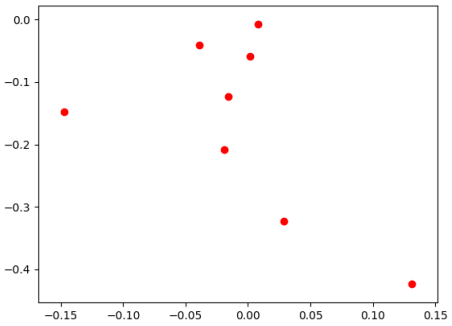


(e) Video sample *-5FM8mMJeYc*

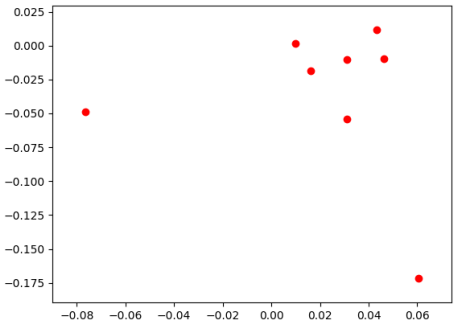


(f) Video sample *-8DvZ48b23E*

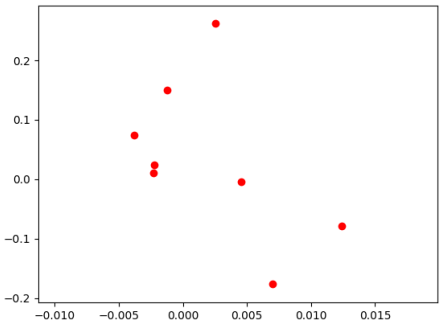
Figure 5.4: Examples of close centroids $k = 16$



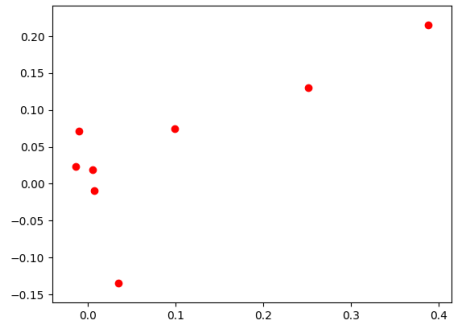
(a) Video sample *-0MLLn0Zg1M*



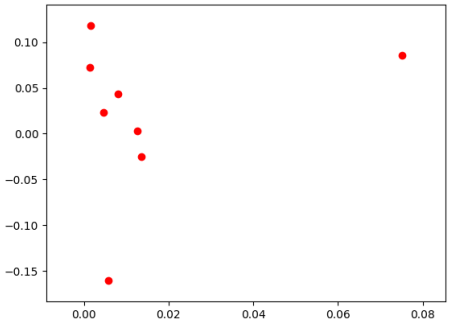
(b) Video sample *-9RiCLmXfWc*



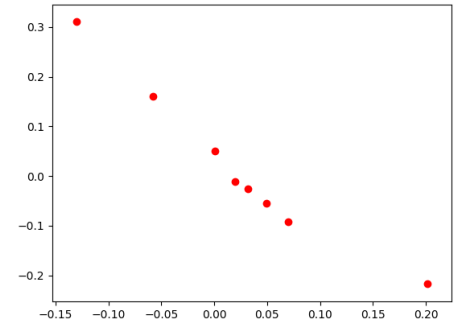
(c) Video sample *-2-KjcZEmrM*



(d) Video sample *-3pjDfx73Aw*



(e) Video sample *-5FM8mMJeYc*



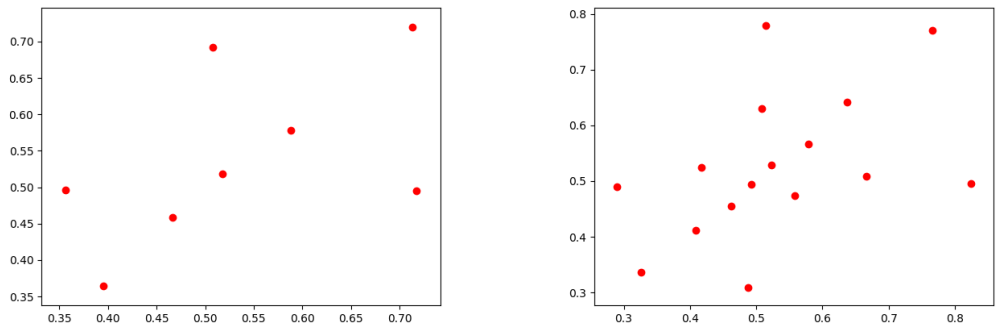
(f) Video sample *-8DvZ48b23E*

Figure 5.5: Centroids used in training $k = 8$

5.3.2 Analysis for number of clusters

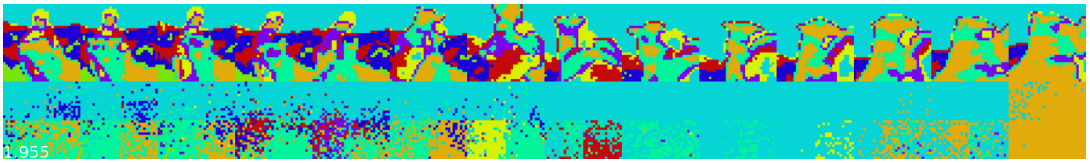
Spatial locations of clusters are very important. Clusters that are close to each other cause wrong signals to be backpropogated over the network. Even the trained model points to the correct spatial location from reference frame to target frame, inconsistencies between the ground truths of reference and target frames cause a wrong supervisory signal to be backpropagated. Calculated ground truths belonging to reference and target frames shall be consistent. We determined close centroids create the inconsistency. A wrong signal is backpropagated in case two matched spatial locations are correct but having different ground truth labels from each other. Figure 5.4 shows some examples to close centroids.

By performing histogram equalization on dataset and input data normalization with specified mean and standard deviation values, clusters are separated to some degree. By setting $k = 8$ centroids are much more separated. Figure 5.5 shows the effect of total cluster numbers and additional preprocessing.

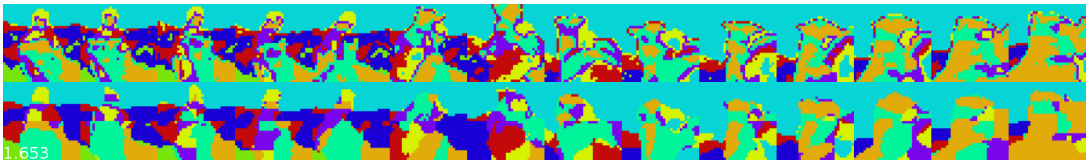


(a) Clusters for sequence *-2gnGuakDzI* $k = 8$ (b) Clusters for sequence *-2gnGuakDzI* $k = 16$

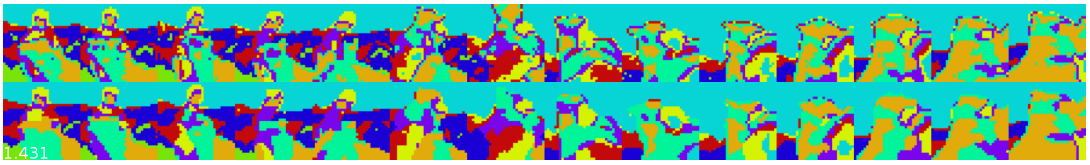
Figure 5.6: Centroids for single sequence training test



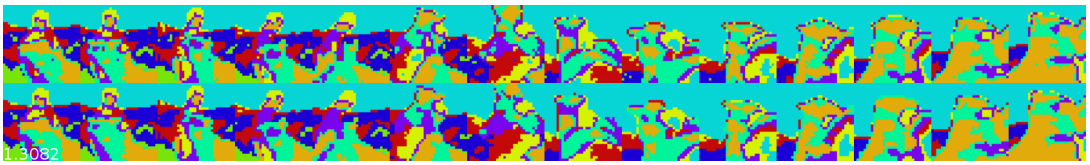
(a)



(b)



(c)



(d)

(e)

Figure 5.7: Visualization of learning for a single video (a) 0^{th} step (b) 10^{th} step (c) 30^{th} step (d) 490^{th} step (e) Difference of 490^{th} step to ground truth

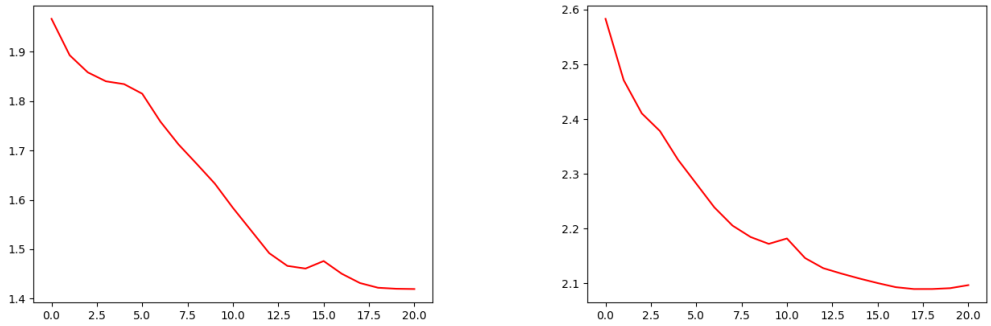
5.3.3 Checking the implementation

We trained the network by using a single video as training data in order to check the network implementation for any possible bugs. We suspected from the implementation of the network because loss value always saturated on a specific value approximately 1.4. Loss value is calculated by using the cross entropy loss as speci-

fied in Section 3.2.5. A typical loss value expected for this experiment is between 0.2 - 0.4 since a classification network trained with cross entropy loss has values near this range. A test is devised that aims to let the network to memorize for the single video input. The test passes if network memorizes and loss value drops rapidly. Such a test is performed on sequence identified with ID *-2gnGuakDzI*. Clusters for this video is shown in Figure 5.6. Note that clusters are separated well and does not have close centroids problem.

We observed that loss function again saturates on a specific value when the test is performed for two settings of clusters, $k = 8$ and $k = 16$. We visualized the learning progress of the network by comparing the predictions of the network against the ground truth in order to investigate the problem. Figure 5.7 shows the learning progress for $k = 8$. The first row of the figure shows the network's prediction when all the layers are initialized with Xavier initialization strategy [40]. The following rows from b to d show the predictions of the network when learning from data. As the learning process continues, network begins to learn semantic shapes and finally it memorizes nearly all the color labels. The last row shows the difference between prediction and the ground truth. According to the calculations performed directly on the image shows that 1103 color labels in total and $103/14 = 78.79$ color labels (there are 14 images in the experiment) per image are predicted wrong. This result convinces us to believe that the network is learning and there is no problem. It is understood from the experiment that relatively big loss value is not a problem because network predicts color labels for $256 \times 256 = 65536$ pixels and seeing such a big loss value seems normal.

The learning process also saturates when $k = 16$ however to a bigger value approximately 2.1. The learning curve shows that the network learns, but using more clusters causes the wrong assignments to be increased supporting the analysis given in Section 5.3.2. The two learning curves belonging to $k = 8$ and $k = 16$ are given in Figure 5.8.



(a) Learning curve when $k = 8$

(b) Learning curve when $k = 16$

Figure 5.8: Learning curves for different number of clusters

5.3.4 Checking for the learning capacity of the network

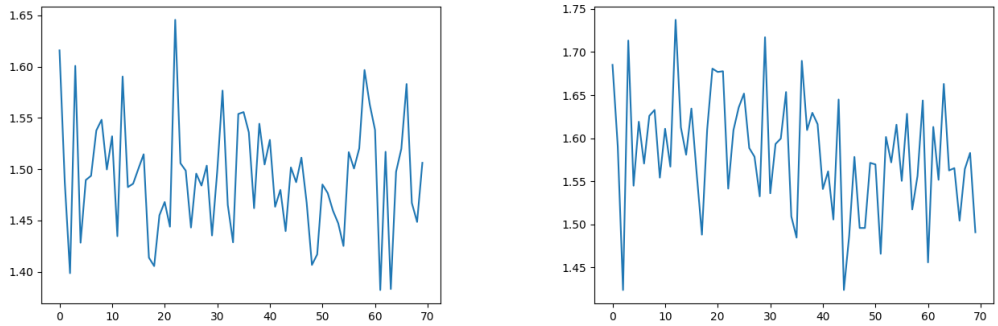
Another test is to check the loss value by changing the learning capability of the network. Learning capability of the network is changed by changing the network used. In addition to the ResNet-18 architecture [18], the architecture used in original work, we trained by using ResNet-34 architecture [18] in order to see the changes. When a bigger network is used, it is expected to see that the loss value decreases in case a smaller network can not generalize training dataset. However, loss value is not decreased but learning is saturated earlier than the smaller network, ResNet-18. Early saturation is normal because ResNet-34 has more learning capacity than ResNet-18. This experiment shows that ResNet-18 architecture is an appropriate one for learning colorization problem.

5.3.5 Training by using the overall dataset

Two different models are trained by using the colorization framework. One is trained by using the ResNet-18 architecture [18] as it is in the original work done by Vondrick et al. [4] and the other one is trained by using AlexNet architecture [19] which is modified as detailed in Section 3.1.3. We tried the AlexNet architecture [19] in order to see the effect of the translational invariance property needed by the siamese architecture, detailed in Section 5.4. Both models are trained for $140k$ samples in

total. Every sample consists of 32 reference and target frames ($8x3$ reference frames and $8x1$ target frames). This means that network saw $4.48M$ reference and target pairs in total. All the training parameters are the same, except the total samples seen in the original work, $400k$ samples are seen in the original work. Learning curve can be seen in Figure 5.9.

Training is done on two NVidia RTX2080 GPUs. Multiple GPUs are used when training. Data parallelism is used, meaning that $16/32$ batches are processed in one GPU and remaining is processed in other GPU. Losses are averaged across the two GPUs and then same signal backpropogated to the networks in two GPUs simultaneously.



(a) ResNet-18 [18] is used as backbone

(b) AlexNet [19] is used as backbone

Figure 5.9: Learning curves for two different variations

5.3.6 DAVIS Evaluation

DAVIS dataset [45] provides 50 high-definition sequences with segmented objects at pixel level accuracy which aims to be a development tool for video object segmentation community. Additionally, this dataset also provides a benchmark in order to be used in comparing state-of-the-art algorithms fairly.

Different algorithms are ranked by using the mean of two performance metrics called region(J) and boundary(F). The region metric measures the number of pixels in the intersection of two masks divided by the pixel size of intersection over the union. The boundary metric measures the accuracy of the boundaries by using the bipartite

matching between the boundary pixels of resulting and ground truth masks. The following formula, which is used to rank the algorithms, defines how to average these two measures:

$$(S) = \frac{1}{2}[m(J,S) + m(F,S)] \tag{5.1}$$

where S is defined as set of sequences exist in dataset. $m()$ is used as a mean function and the mean for a metric in a given sequence is defined as follows:

$$m(M,S) = \frac{1}{|O_s|} \sum_{o \in O_s} \frac{1}{|F_{s(o)}|} \sum_{f \in F_{s(o)}} M(m_o^f, g_o^f) \tag{5.2}$$

where m_o^f and g_o^f is defined as masks of the result and ground truth objects, S is defined as set of sequences in the given dataset, O_s is set of annotated objects in given set. Given an object $o \in O_s$, $s(o) \in S$ is the sequence where the given object appears. F_s is the set of frames in sequence $s \in S$.

5.3.7 Experiments on DAVIS dataset

We implemented a simple segmentation tracker by using the trained model. We initialized the segmentation tracker by ground truth segments and then these segments are tracked by calculating the similarity matrix among the frames. Similarity matrix points from reference frame to target frame and this way moves the segmentation area from reference frame to target frame. Iterating over all frames in a sequence, segments are tracked from first frame to last frame. In order to point from reference frame to target frame correctly, similarity matrix shall be calculated correctly between the frames. Due to the spatio-temporal coherence between consecutive frames, one expects the similarity matrix a diagonal one.

Some experiments are performed in DAVIS dataset. These experiments are not quantitative because pointing from reference to target frames are performed on 32×32 dimensional frames. One needs to upscale these results. Bicubic upscaling works but upscaled segments are not as good as with a trained deconvolutional network. One obtains detailed segments by deconvolving, when upscaling the 32×32 sized segments to 256×256. Because of this, method used for upscaling directly affects the evaluation the model on DAVIS dataset.

Table 5.2: Comparison of different algorithms in DAVIS 2017 [2] benchmark (higher Average score means better performance)

Method	**Backbone**	**J Score**	**F Score**	**Average**
OSVOS [49]	VGG [50]	45.60	52.50	49.00
Original work [4]	ResNet-18[18] + 5 layer 3D Conv	34.60	32.70	33.65
Our implementation - 1	ResNet-18 [18] + 5 layer 3D Conv	29.48	22.93	26.21
Our implementation - 2	AlexNet [19] + 5 layer 3D Conv	26.95	20.90	23.93

Table 5.2 involves the performance metrics evaluated by using the method given in 5.3.6. Here the work done by Caelles et al. [49] is used as baseline, and it is given in the table in order to compare the performances of different algortihms. Reader shall note that OSVOS [49] is a fully supervised method.

There is a performance drop when comparing obtained metrics to the original work. Three factors may cause the specified performance drop. The first one is not all the data from Kinetics-700 dataset [46] can be used in this thesis work since the dataset is very huge and and big storage areas are needed in order to keep data. The second one is models cannot be trained for number of iterations as in original work. Because training over this huge dataset consumes so much time and requires powerful hardware. The third one is related to the upscaling method used. We used bicubic scaling when upscaling the segments from 32×32 to 256×256. This procedure generates segments with rough boundaries. This explains more performance drop obtained for boundary (F) metric comparing to the region (J) metric.

5.4 Integration of Colorization Framework to SiamFC Framework

We performed various experiments by using the proposed combined framework. These experiments are detailed and the results are shared within the following sections.

We use the SiamFC implementation, detailed in Section 3.2, as a baseline for the specified experiments. All the experiments in following sections are done by using OTB 2015 [1] benchmark. Baseline method obtained AUC score as 0.584 for success curve as stated in Section 5.2.2.

5.4.1 Experiment 1 - Test Proposed Method for Different α Values, ResNet architecture as Colorization Backbone

In this experiment, the baseline model and colorization model trained by using modified Resnet-18 [18] architecture is combined. The combined method is tested on OTB 2015 benchmark [1] for different α values, set from 0.0 to 1.0 increased by 0.1 steps.

Figure 5.10 shows the detailed results for the Experiment 1.

Mean and standard deviation values are obtained for score maps that are calculated by baseline method and colorization method. Mean and standard deviation values are calculated as -3.13 and 3.58 respectively for score map that is calculated by baseline method. On the other side, mean and standard deviation values are calculated as 422.22 and 34.37 respectively for score map that is calculated by colorization method. Two score maps are combined after normalization is performed.

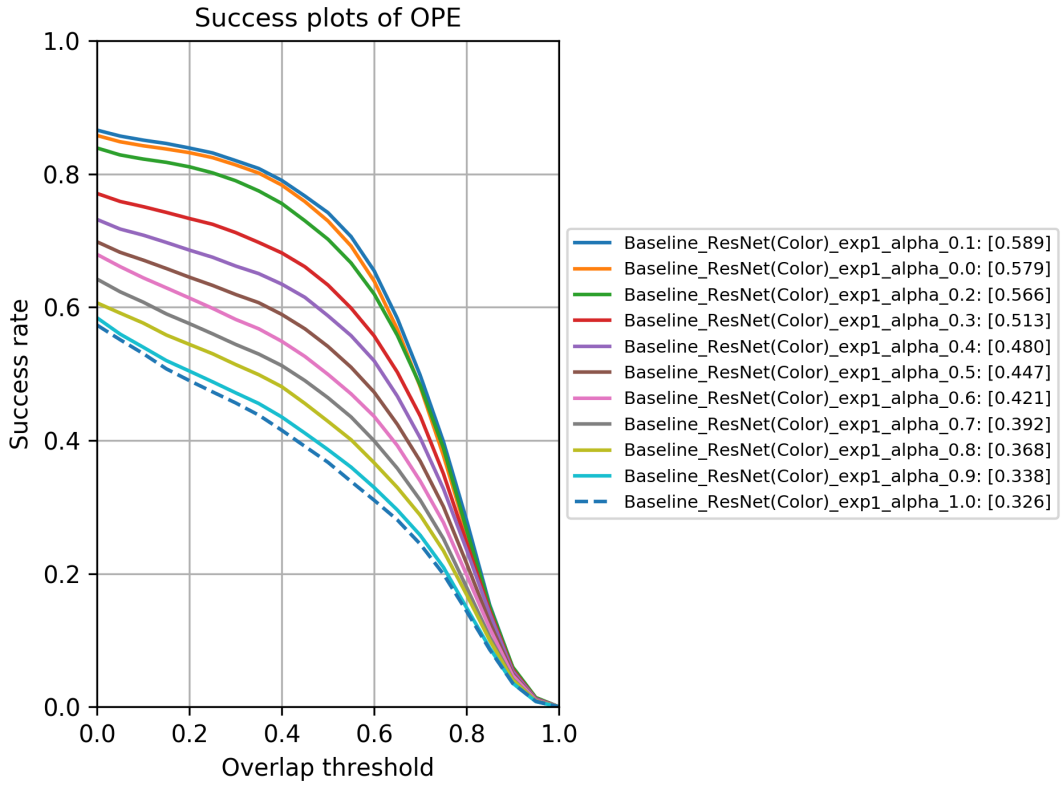


Figure 5.10: Success curves for Experiment 1 and related AUC values

The AUC scores for success curve is obtained as 0.579 and 0.326 for $\alpha = 0.0$ and $\alpha = 1.0$ respectively. The evaluation score generally drops from its best value to its worst value as the α increases.

5.4.2 Experiment 2 - Train Proposed Method for Different α Values

In this experiment, the combined method is trained supervised. GOT-10k [43] dataset is used fro training. α values set 0.0 to 1.0 increased by 0.1 steps. Training is performed for 50 epochs. The purpose of this experiment is to see whether the colorization method increase the performance when working with the baseline method together.

Evaluation is performed by using OTB 2015 [1] benchmark. Success curves are pilot in Figure 5.11.

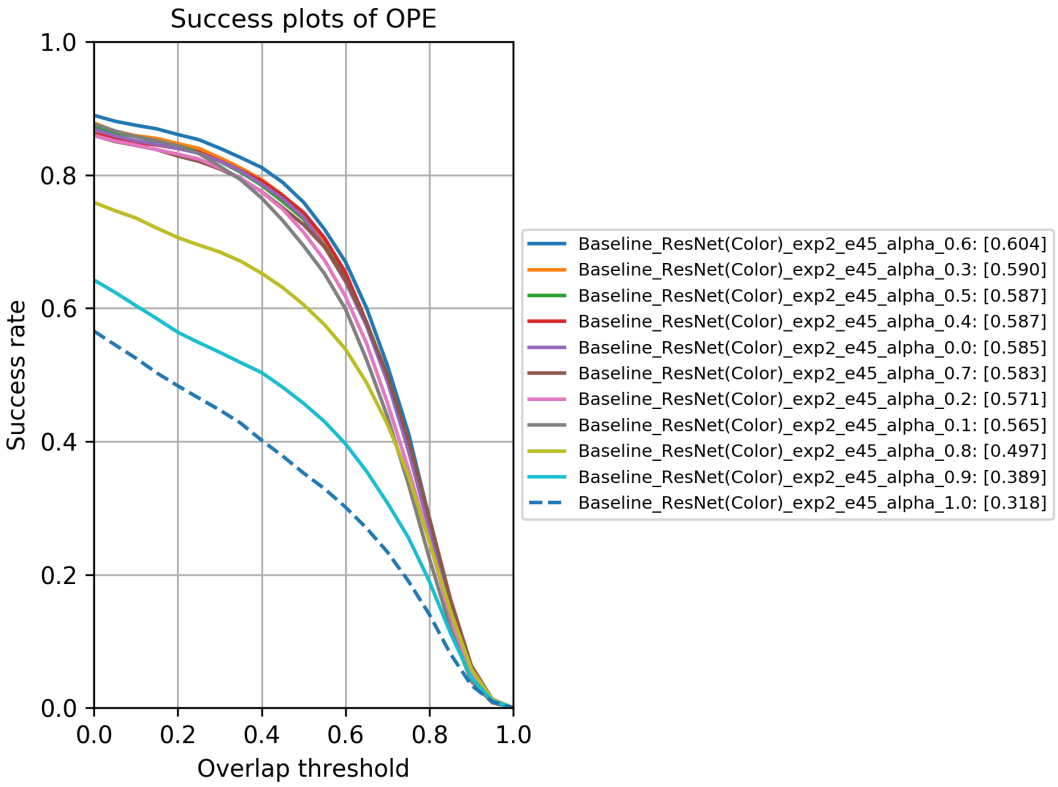


Figure 5.11: Success curves for Experiment 2 and related AUC values

Trained models, for $\alpha = 0.6$, $\alpha = 0.3$, $\alpha = 0.5$, $\alpha = 0.4$ outperforms the baseline

Table 5.3: Performance comparison for models in Experiment 2 that outperforms the baseline method

	Obtained AUC value	**Performance change**
$\alpha = 0.6$	0.604	3.42% improvement
$\alpha = 0.3$	0.590	1.02% improvement
$\alpha = 0.5$	0.587	0.51% improvement
$\alpha = 0.4$	0.585	0.17% improvement

method which has AUC value for success curve as 0.584. Details are given in Table 5.3.

5.4.3 Experiment 3 - Train Proposed Method by Using Random Initialized Colorization Network Weights

In this experiment, the combined method is trained with GOT-10k [43] dataset for different α values starting from 0.0 to 1.0. But this time ResNet-18 [18] architecture used in colorization method is initialized by using the random weights. The purpose of this experiment is to see whether the weights coming from self-supervised learning helps the combined network learn better or not.

The resulting success curves are pilot in Figure 5.12.

Some models outperform the the baseline method which are given in the Table 5.5. Comparisons are given relative to the baseline method which obtained AUC score for success curve as 0.584. Results of two experiments close to each other; however, results obtained by using self-supervised colorization weights are better than random initialized weights.

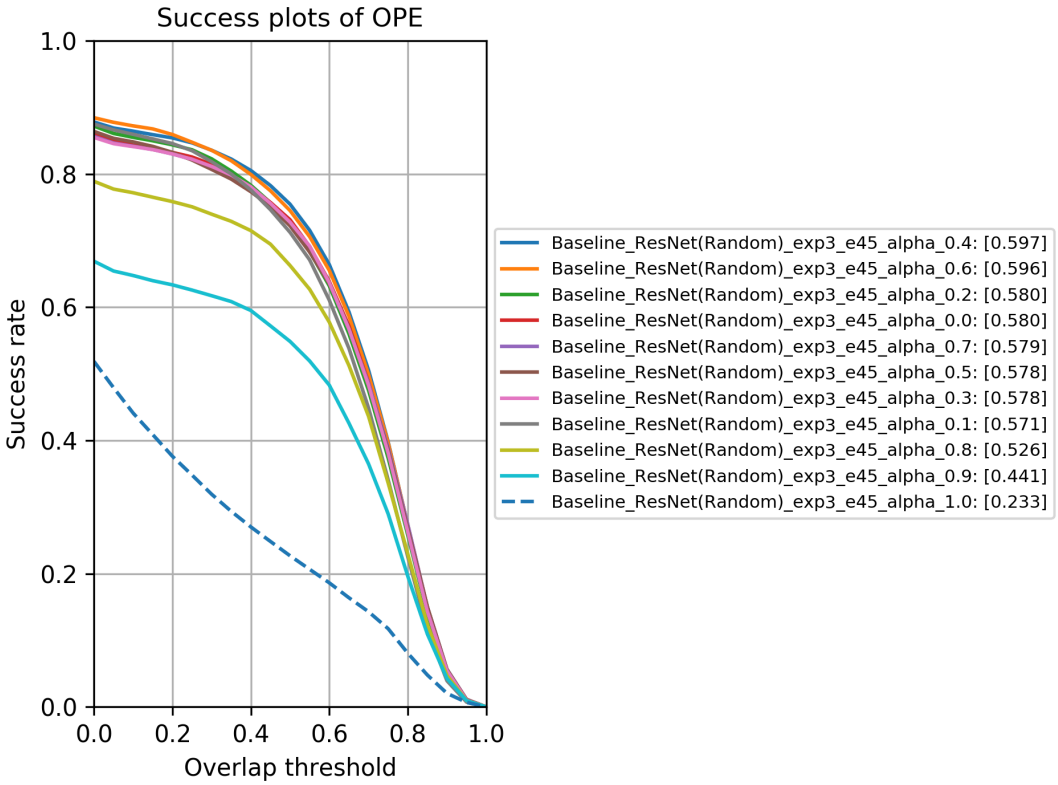


Figure 5.12: Success curves for Experiment 3 and related AUC values

5.4.4 Experiment 4 - Test Proposed Method for Different α Values, AlexNet architecture as Colorization Backbone

In this experiment, the baseline method and colorization model trained by using AlexNet [19] is combined. The combined method is tested on OTB 2015 benchmark [1] for different α values, set from 0.0 to 1.0 increased by 0.1 steps. The purpose of this experiment is to see the performance of the modified AlexNet architecture [19] which does not have any paddings and hence satisfies the translational invariance propoerty required by siamese networks [15].

Mean and standard deviation values are obtained for score maps that are calculated by baseline method and colorization method. Mean and standard deviation values are calculated as -3.13 and 3.58 respectively for score map that is calculated by baseline method. On the other side, mean and standard deviation values are calculated as 1122.44 and 3481.73 respectively for score map that is calculated by colorization method. Two score maps are combined after normalization is performed.

Table 5.4: Performance comparison for models in Experiment 3 that outperforms the baseline method

	Obtained AUC value	**Performance change**
$\alpha = 0.4$	0.597	2.22% improvement
$\alpha = 0.6$	0.596	2.05% improvement

The resulting success curves are plot in Figure 5.13.

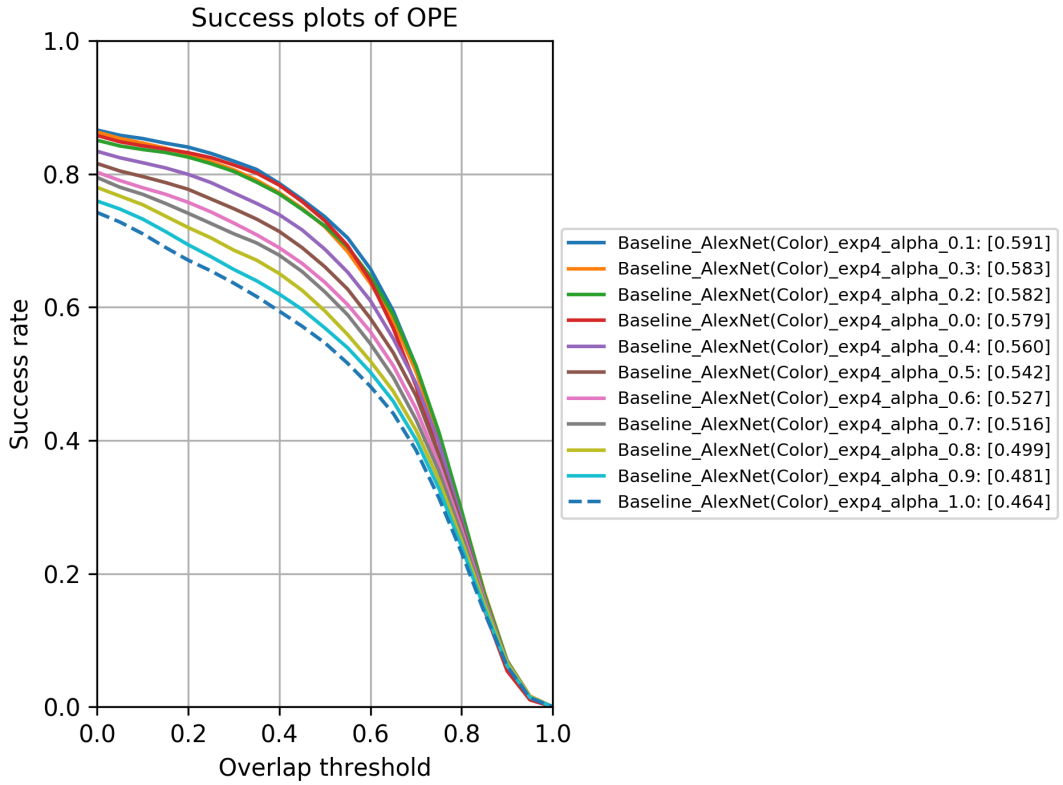


Figure 5.13: Success curves for Experiment 4 and related AUC values

The combined method outperforms the baseline method when $\alpha = 0.1$. Baseline method got AUC score for success curve as 0.584 as it is specified before. Table 5.5 shows the details.

A comparison for performance values at $\alpha = 1.0$ is required in order to see the effect of the network used in colorization method. In Experiment 1 0.326 AUC score is obtained when $\alpha = 1.0$. On the other side, 0.464 AUC score is obtained when $\alpha =$

Table 5.5: Performance comparison for models in Experiment 4 that outperforms the baseline method

	Obtained AUC value	**Performance change**
$\alpha = 0.1$	0.591	%1.19 improvement

1.0. The result of this experiment supports that violence of translational invariance results in a performance drop.

5.4.5 Experiment 5 - Traing Proposed Method for Different α Values, AlexNet architecture as Colorization Backbone

In this experiment, the combination of baseline and colorization framework trained with AlexNet architecture [19] is further trained by using GOT-10k [43] dataset as training dataset. Training is performed for different α values, set from 0.0 to 1.0 increased by 0.1 st

The combined network outperforms the baseline model when $\alpha = 0.6$, $\alpha = 0.4$, $\alpha = 0.7$, $\alpha = 0.5$, and $\alpha = 0.3$. Table 5.6 shows the details.

Table 5.6: Performance comparison for models in Experiment 5 that outperform baseline model for Experiment 5

	Obtained AUC value	**Performance change**
$\alpha = 0.6$	0.610	4.45% improvement
$\alpha = 0.4$	0.600	2.73% improvement
$\alpha = 0.7$	0.597	2.22% improvement
$\alpha = 0.5$	0.588	0.34% improvement
$\alpha = 0.3$	0.586	0.34% improvement

The success curves are plot in Figure 5.14.

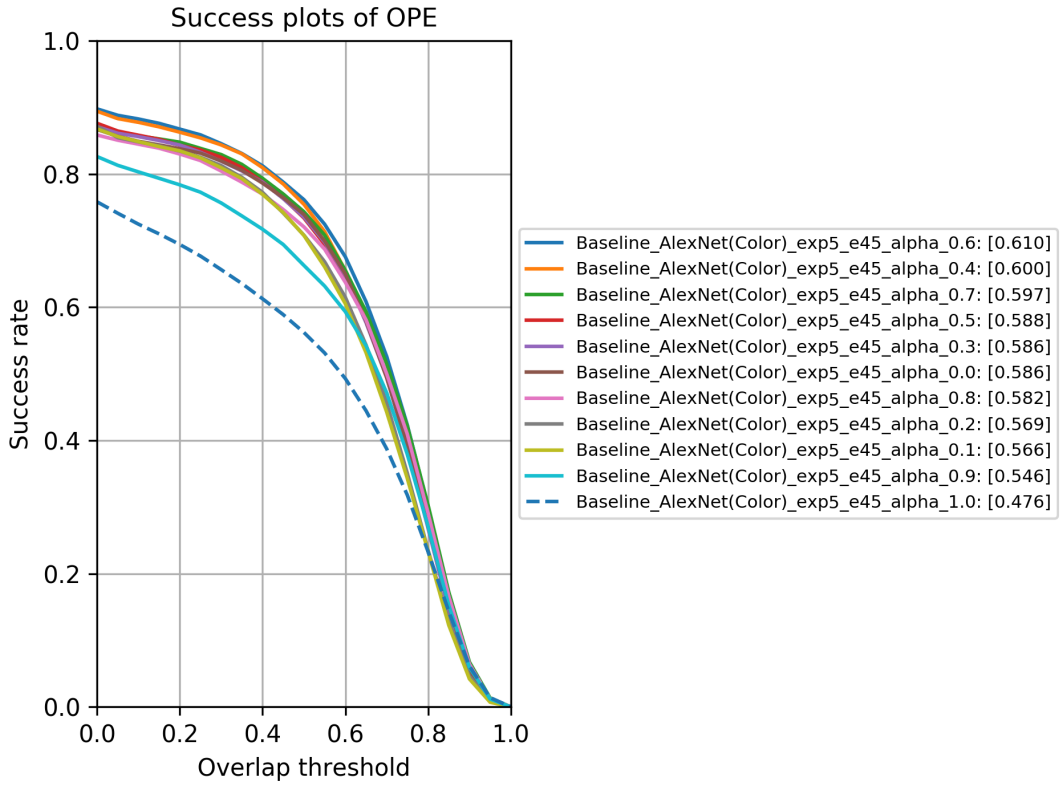


Figure 5.14: Success curves for Experiment 5 and related AUC values

5.4.6 Summary for Experiments

Some implications are obtained from the results of the experiments. We try to summarize these implications in this section.

Firstly, the violation of the translational invariance restriction required by siamese architecture reduces the performance of the system. We obtained such a result by comparing the results of experiment 1 and experiment 4. The obtained AUC values of the success curves are 0.326 and 0.464 in experiment 1 and experiment 4 respectively when $\alpha = 1.0$. The performance drop using ResNet-18 architecture [18] instead of AlexNet [19] is calculated as $\%29.74$. ResNet-18 [18] architecture used in experiment 1 must have paddings because of the residual connections. Nonetheless, paddings break the restriction of translational invariance required for siamese architecture [15]. On the other hand, AlexNet architecture [19] used in experiment 4 has no paddings, so that translation invariance restriction is not broken. Method proposed by Li et al. [36] proposes a data sampling method in training phase which reduces the violation

of translational invariance requirement. ResNet-50 [18] is used in this work. More detailed discussion was given in Section 2.2.

Secondly, a self-supervised framework reduces the effect of violation of translational invariance property. By comparing the AUC scores of success curves for experiments 1, 2, and 3 when $\alpha = 1.0$, it is shown that the effect of violating the translational invariance property is avoided to a some degree. Following AUC values for success curves are obtained 0.326, 0.318, 0.233 for experiment 1, 2, and 3 respectively. The worst result is obtained for experiment 3 in which random weights for ResNet-18 [18] backbone is used. Yet, better results are obtained when weights from self-supervised colorization framework is used for ResNet-18 [18] backbone. This result shows that, using massive data relative to GOT-10k [43] in a self-supervised framework avoids the effect of strict requirement of translational invariance. This fact makes usage of deeper networks possible in siamese architectures [15].

Lastly, usage of massive data relative to GOT-10k [43] in self-supervised framework increases the tracking performance to a some degree in training of the combined network. In all of the experiments 1, 2, 4, and 5 the baseline method is outperformed. It was pointed out in Section 4.2 that the size of labeled dataset for tracking problem are small compared to size of dataset used for classification or detection problems. Self-supervised learning framework makes using massive amount of training data without labels possible.

5.4.7 Attribute Based Performance Analysis

OTB [1] benchmark has 9 attributes that are used for defining a sequence. These attributes are defined in Table 5.1. Multiple attributes may be used for defining a sequence. We calculated the performance of baseline model and the models which performed best in experiments 1 to 5 in order to see the performance changes for different attributes.

Table 5.7 shows the calculated performance values for different attributes.

We obtained close results to the results obtained for OTB [1] benchmark for all attributes except for OV(out-of-view) and LR(low resolution) attributes. All the mod-

Table 5.7: Calculated Performances for 9 Different Attributes. Attributes are defined in OTB [1] Benchmark. B corresponds to the baseline and 1 - 5 corresponds to the Experiment 1 to Experiment 5. Table 5.1 shows the attributes in detail.

	α	**tb100**	**IV**	**SV**	**OCC**	**DEF**	**MB**	**FM**	**IPR**	**OPR**	**OV**	**BC**	**LR**
B	0.0	0.589	0.559	0.555	0.549	0.542	0.593	0.583	0.590	0.575	0.491	0.503	0.523
1	0.1	0.589	0.555	0.553	0.551	0.552	0.578	0.581	0.566	0.564	0.491	0.531	0.466
2	0.6	0.604	0.577	0.576	0.567	0.558	0.605	0.588	0.588	0.584	0.524	0.555	0.569
3	0.4	0.597	0.574	0.571	0.553	0.567	0.608	0.585	0.589	0.580	0.496	0.558	0.540
4	0.1	0.591	0.571	0.557	0.544	0.545	0.587	0.584	0.568	0.558	0.474	0.522	0.482
5	0.6	0.610	0.593	0.588	0.571	0.567	0.622	0.593	0.591	0.589	0.527	0.560	0.606

els from experiments and baseline model have poor performance for these attributes. This result shows that models are not robust for low resolution and out-of-view cases.

Only for the model obtained from Experiment 5, we obtained close results to the results obtained for OTB [1] benchmark. This shows that AlexNet based colorization backbone affects the tracking performance positively for low resolution objects.

5.5 Comparison with other trackers

5.5.1 VOT Evaluation

VOT [5] is another benchmark that is proposed in order to evaluate and benchmark object trackers. The benchmark was first shown on 2013 in a workshop of ICCV2013. The benchmark consists of a dataset and a evaluation method.

The evaluation method is defined by Kristen et al. in the work named as *A Novel Performance Evaluation Methodology for Single-Target Trackers* [5]. Two weakly-correlated evaluation metrics are defined in this work: *Accuracy* and *Robustness*.

Accuracy at a specific time measures the IoU of two bounding boxes, one is calculated by the tracker and the other one is defined by the ground truth. The mathematical definition is as follows:

$$\phi_t = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}$$

where A_t^G is ground truth bounding box at time t and A_t^T is bounding box that is calculated by the tracker.

A tracker is run over each sequence for N_{rep} times. This way a potential variance of a tracker can be handled. Per frame accuracy can be calculated by using the following formula:

$$\Phi_t = \frac{1}{N_{rep}} \sum_{k=1}^{N_{rep}} \phi_t(k)$$

where $\Phi_t(k)$ denotes the per frame accuracy of tracker at time t averaged over repetitions.

A re-initialization is triggered whenever the IoU drops to zero. A tracker may again failed immediately after the re-initialization. This is because IoU drops to zero because of an occlusion and it will likely be zero again for the consecutive frame. In order to prevent this bias in performance evaluation $N_{skip} = 5$ is used in order to skip N_{skip} much frames after the re-initialization. After the re-initialization a tracker shows superior performances. In other words, IoU is biased towards to higher values for several frames after the re-initialization process. These frames are labeled as invalid in the performance evaluation in order to avoid the effect of this high bias. The number of frames that will be labeled as invalid is determined by the parameter N_{burnin} and set to 10.

Accuracy metric is defined after all:

$$\rho_A = \frac{1}{N_{valid}} \sum_{j=1}^{N_{valid}} \Phi_j$$

Robustness measures the number of times where the tracker is re-initialized. The re-initialization procedure was given in the beginning of this section. The robustness is defined as follows:

$$\rho_R = \frac{1}{N_{rep}} \sum_{k=1}^{N_{rep}} F(k)$$

where $F(k)$ is the number of times the tracker is failed, ie. IoU drops to zero.

The overall performance of a tracker is then determined by calculating the average of ρ_A and ρ_R. Hence, a tracker becomes a high rank tracker with increasing values of average of ρ_A and ρ_R.

Table 5.8: Neural Network Architectures Used in Different Methods

Method	**Backbone**
SiamFC [3]	AlexNet [19]
SiamMask [21]	ResNet-50 [18]
SiamRPN [34]	AlexNet [19]
SiamRPN++ [36]	ResNet-50 [18]
CycleSiam [20]	ResNet-50 [18]
CycleSiam+ [20]	ResNet-50 [18]
Combined Method (Ours)	AlexNet[19] + ResNet-18 [18] & 5 Layer 3DConv
Combined Method (Ours)	AlexNet[19] + Modified AlexNet & 5 Layer 3DConv

5.5.2 Results of comparison

In this section, performance comparison of self-supervised and trained trackers with some other siamese network based and supervised trackers is given.

Overlap measurement based performance metrics are evaluated and compared. This decision is given due to the fact that overlap based measurements are more robust than center distance based measurements as it is stated in Section 5.2.1.

Comparing two performance metrics belonging to a supervised and self-supervised method is not fair; however, such a comparison is important in order to see whether a self-supervised method catches the performance of a supervised method or not.

SiamRPN [34], SiamRPN++ [36], SiamMask [21] are selected as state of the art examples of trackers that inherit most of their base features from the SiamFC [3] method. In addition to these supervised methods self-supervised methods CycleSiam and CycleSiam+ [20] are included as self-supervised methods. Table 5.8 shows the architectures used in different methods. The backbone architectures are considered in order to make a fair comparison between different methods.

Table 5.9 gives the OTB2015 [1] benchmark results for different trackers. The best performed trackers from each experiment is added to the table. Table shows that supervised methods achieve better results; but, there is not a huge performance between

Table 5.9: Comparison of trackers, OTB2015 [1] benchmark

Method	**Training Dataset**	**Success Curve AUC**
SiamFC [3]	ILSVRC15 [41]	0.584
SiamRPN [34]	ILSVRC15 [41], YouTube BB [51]	0.636
SiamRPN++ [36]	ILSVRC15 [41], COCO [52], YouTube BB [51]	0.696
Experiment - 1 ($\alpha = 0.1$)	Kinetics-700 [46]	0.589
Experiment - 2 ($\alpha = 0.6$)	Kinetics-700 [46], GOT-10k [43]	0.604
Experiment - 3 ($\alpha = 0.4$)	GOT-10k [43]	0.597
Experiment - 4 ($\alpha = 0.1$)	Kinetics-700 [46]	0.591
Experiment - 5 ($\alpha = 0.6$)	Kinetics-700 [46], GOT-10k [43]	0.61

supervised and self-supervised ones.

Table 5.10 gives the VOT2016 benchmark results for different trackers. The best performed trackers from each experiment is added to the table. In this comparison, two methods which are proposed lately in the work performed by Yuan et al. [20] are included in the performance comparison. These methods are important because a siamese architecture is trained in self-supervised manner. It is shown that, CycleSiam [20] shows a performance that is too close to a supervised alternative SiamMask [21] which is one of the best performing supervised trackers in the literature.

Table 5.11 gives the VOT2018 [37] benchmark results for different trackers. The best performed trackers for our combined method from each experiment is added to the table. For experiments in which combined network is not trained further, results for $\alpha = 0.0$ and $\alpha = 1.0$ is shared. Close results to SiamRPN [34] method is obtained. On the other hand, a performance gap is shown to SiamMask [21].

Table 5.10: Comparison of trackers, VOT2016 benchmark

Method	**Training Dataset**	**Accuracy**
SiamFC [3]	ILSVRC15 [41]	0.535
SiamRPN [34]	ILSVRC15 [41], YouTube BB [51]	0.560
SiamMask [21]	ILSVRC15[41], COCO [52], YouTube-VOS[53]	0.609
CycleSiam [20]	ILSVRC15 [41]	0.603
CycleSiam+ [20]	ILSVRC15 [41]	0.601
Experiment - 1 ($\alpha = 0.0$)	Kinetics-700 [46]	0.504
Experiment - 1 ($\alpha = 1.0$)	Kinetics-700 [46]	0.471
Experiment - 2 ($\alpha = 0.6$)	Kinetics-700 [46], GOT-10k [43]	0.543
Experiment - 3 ($\alpha = 0.7$)	GOT-10k [43]	0.539
Experiment - 4 ($\alpha = 0.0$)	Kinetics-700 [46]	0.502
Experiment - 4 ($\alpha = 1.0$)	Kinetics-700 [46]	0.510
Experiment - 5 ($\alpha = 0.7$)	Kinetics-700 [46], GOT-10k [43]	0.541

Table 5.11: Comparison of trackers, VOT2018 benchmark

Method	**Training Dataset**	**Accuracy**
SiamFC [3]	ILSVRC15 [41]	0.498
SiamRPN [34]	ILSVRC15 [41], YouTube BB [51]	0.49
SiamMask [21]	ILSVRC15[41], COCO [52], YouTube-VOS[53]	0.609
CycleSiam [20]	ILSVRC15 [41]	0.562
CycleSiam+ [20]	ILSVRC15 [41]	0.549
Experiment - 1 ($\alpha = 0.0$)	Kinetics-700 [46]	0.4913
Experiment - 1 ($\alpha = 1.0$)	Kinetics-700 [46]	0.4264
Experiment - 2 ($\alpha = 0.2$)	Kinetics-700 [46], GOT-10k [43]	0.5029
Experiment - 3 ($\alpha = 0.6$)	GOT-10k [43]	0.5106
Experiment - 4 ($\alpha = 0.0$)	Kinetics-700 [46]	0.4906
Experiment - 4 ($\alpha = 1.0$)	Kinetics-700 [46]	0.4667
Experiment - 5 ($\alpha = 0.3$)	Kinetics-700 [46], GOT-10k [43]	0.5062

CHAPTER 6

CONCLUSION

In this chapter we conclude this thesis work and make a discussion on the results.

The strict translational invariance restriction needed by siamese networks [15] blocks the usage of deeper networks within siamese networks. This statement is validated within the experiments performed in this work. The same statement is also stated in proposed method SiamRPN++ [36]. ResNet-18 [18] based colorization model performs worse than AlexNet [19] based colorization model. Nevertheless, our experiments show that, self-supervised learning provides a way for coping with the strict translational invariance violation and make usage of deeper networks possible in siamese networks [15].

Self-supervision makes usage of massive data relative to labeled datasets for object tracking problem. Experiments show that the knowledge obtained by colorization framework by seeing massive amount of data boosts the performance when the combined network trained further in supervised manner. As we specified before labeled datasets for object tracking is really small comparing to the ImageNet [41] dataset that is used for image classification domain.

The combined method in the scope of this thesis work seems promising; yet, it is believed that additional work should be performed in order to increase the tracking performance. First of all, the colorization network is not performing well, as shown in Section 5.3.7. So, further work in order to increase the performance of the colorization network needs to be performed. Different loss functions for colorization task are discussed; however, applying and evaluating these loss functions are not in the scope of this thesis work. In addition, other self-supervised methods which are discussed in

Section 3.2 can be tried for integrating to the siamese architecture.

Second, methods like SiamRPN [34], SiamRPN++ [36], and SiamMask [21] made extensions to SiamFC [3]. It would be nice if colorization model were integrated to such methods. SiamFC [3] is a relatively older method and there are too much proposed work that tries to extend SiamFC [3] in different ways in the literature. For example, CycleSiam and CycleSiam+ [20] integrates self-supervision to SiamMask [21] as base method. SiamMask [21] uses a joint loss function which acts as a supervisory signal for multiple tasks, so increasing the overall tracking performance. However, we see that process required for integrating the colorization framework to SiamMask [21] is more complex than for SiamFC [3]. This work will be considered as a future work.

Self-supervision methodology have some disadvantages that needs to be pointed out. Self-supervision makes usage of massive data possible, however pre-processing, keeping, maintaining massive data requires a lot of engineering. Training takes a lot of time and needs high-performance hardware. This situation avoids to perform hyper-parameter tuning or at least such a work takes too much time. Besides these disadvantages; however, self-supervised methodology have much more advantages that were pointed out several times throughout this work.

REFERENCES

[1] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[2] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbelaez, A. Sorkine-Hornung, and L. V. Gool, "The 2017 DAVIS challenge on video object segmentation," *CoRR*, vol. abs/1704.00675, 2017.

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Computer Vision – ECCV 2016 Workshops* (G. Hua and H. Jégou, eds.), (Cham), pp. 850–865, Springer International Publishing, 2016.

[4] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, "Tracking emerges by colorizing videos," *CoRR*, vol. abs/1806.09594, 2018.

[5] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2137–2155, Nov 2016.

[6] "Waypoint - the official waymo blog: Introducing the 5th-generation waymo driver: Informed by experience, designed for scale, engineered to tackle more environments," Mar 2020.

[7] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, S. Zhao, S. Cheng, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2020.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan,

Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2020.

[9] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," 2019.

[10] R. T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 745–746, 2000.

[11] "Dahua technologies products." `https://us.dahuasecurity.com/?page_id=14784`, visited on Jan 23, 2021.

[12] "Briefcam solutions uses deep learning and ai technology." `https://www.briefcam.com/technology/overview/`, visited on Jan 23, 2021, Sep 2020.

[13] "Ai pilots to transform your unmanned systems into intelligent teammates." `https://www.shield.ai/products`, visited on Jan 23, 2021.

[14] "Quividi amp - quividi - the leading audience measurement platform." `https://quividi.com/audience-measurement-platform/`, visited on Jan 23, 2021, Jun 2020.

[15] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, (San Francisco, CA, USA), p. 737–744, Morgan Kaufmann Publishers Inc., 1993.

[16] B. Fernando, H. Bilen, E. Gavves, and S. Gould, "Self-supervised video representation learning with odd-one-out networks," *CoRR*, vol. abs/1611.06646, 2016.

[17] I. Misra, C. L. Zitnick, and M. Hebert, "Unsupervised learning using sequential verification for action recognition," *CoRR*, vol. abs/1603.08561, 2016.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[20] W. Yuan, M. Y. Wang, and Q. Chen, "Self-supervised object tracking and segmentation with cycle-consistent siamese networks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020.

[21] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," *CoRR*, vol. abs/1812.05050, 2018.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, (San Francisco, CA, USA), p. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[23] C. Tomasi and T. Kanade, "Detection and tracking of point features," tech. rep., International Journal of Computer Vision, 1991.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[25] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, (USA), p. 2564–2571, IEEE Computer Society, 2011.

[27] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[28] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," 1998.

[29] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[30] D. Bolme, J. Beveridge, B. Draper, and Y. Lui, "Visual object tracking using adaptive correlation filters," pp. 2544–2550, 06 2010.

[31] K. Dai, D. Wang, H. Lu, C. Sun, and J. Li, "Visual tracking via adaptive spatially-regularized correlation filters," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[32] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: efficient convolution operators for tracking," *CoRR*, vol. abs/1611.09224, 2016.

[33] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "End-to-end representation learning for correlation filter based tracking," *CoRR*, vol. abs/1704.06036, 2017.

[34] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[36] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4277–4286, 2019.

[37] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin Zajc, T. Vojir, G. Bhat, A. Lukežič, A. Eldesokey, G. Fernandez Dominguez,

A. Garcia-Martin, Iglesias-Arias, A. Alatan, A. Gonzalez-Garcia, A. Petrosino, A. Memarmoghadam, A. Vedaldi, and A. Muhič, *The Sixth Visual Object Tracking VOT2018 Challenge Results*, pp. 3–53. 01 2019.

[38] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[39] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kamarainen, L. C. Zajc, O. Drbohlav, A. Lukezic, A. Berg, A. Eldesokey, J. Kapyla, G. Fernandez, A. Gonzalez-Garcia, A. Memarmoghadam, A. Lu, A. He, A. Varfolomieiev, A. Chan, A. S. Tripathi, A. Smeulders, B. S. Pedasingu, B. X. Chen, B. Zhang, B. Wu, B. Li, B. He, B. Yan, B. Bai, B. Li, B. Li, B. H. Kim, and B. H. Ki, "The seventh visual object tracking vot2019 challenge results," in *IEEE International Conference on Computer Vision Workshops*, 2019.

[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[42] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. ˇ Cehovin, G. F. andez, T. V. ˇ r, G. H. ager, G. Nebehay, R. Pflugfelder, A. Gupta, A. Bibi, A. L. zič, A. Garcia-Martin, A. Petrosino, A. Saffari, A. Montero, A. Varfolomieiev, A. Baskurt, B. Zhao, B. Ghanem, B. Martinez, B. Lee, B. Han, C. Wang, C. Garcia, C. Zhang, C. Schmid, D. Tao, D. Kim, D. Huang, D. Prokhorov, D. Du, D.-Y. Yeung, E. Ribeiro, F. Khan, F. Porikli, F. Bunyak, G. Zhu, G. Seetharaman, H. Kieritz, H. Yau, H. Li, H. Qi, H. Bischof, H. Possegger, H. Lee, H. Nam, I. Bogun, J.-C. Jeong, J.-I. Cho, J.-Y. Lee, J. Zhu, J. Shi, J. Li, J. Jia, J. Feng, J. Gao, J. Choi, J.-W. Kim, J. Lang, J. Martinez, J. Choi, J. Xing, K. Xue, K. Palaniappan, K. Lebeda, K. Alahari, K. Gao, K. Yun, K. Wong, L. Luo,

L. Ma, L. Ke, L. Wen, L. Bertinetto, M. Pootschi, M. Maresca, M. Danelljan, M. Wen, M. Zhang, M. Arens, M. Valstar, M. Tang, M.-C. Chang, M. Khan, N. Fan, N. Wang, O. M. s´ ık, P. Torr, Q. Wang, R. Martin-Nieto, R. Pelapur, R. Bowden, R. Laganière, S. Moujtahid, S. Hare, S. Hadfield, S. Lyu, S. Li, S.-C. Zhu, S. Becker, S. Duffner, S. Hicks, S. Golodetz, S. Choi, T. Wu, T. Mauthner, T. Pridmore, W. Hu, W. Hübner, X. Wang, X. Li, X. Shi, X. Zhao, X. Mei, Y. Shizeng, Y. Hua, Y. Li, Y. Lu, Y. Li, Z. Chen, Z. Huang, Z. Chen, Z. Zhang, Z. He, and Z. Hong, "The visual object tracking vot2015 challenge results," in *ICCV workshop on Visual Object Tracking Challenge*, pp. 564 – 586, December 2015.

[43] L. Huang, X. Zhao, and K. Huang, "Got-10k: A large high-diversity benchmark for generic object tracking in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2019.

[44] J. Clement, "Youtube: hours of video uploaded every minute 2019," Aug 2019.

[45] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Computer Vision and Pattern Recognition*, 2016.

[46] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *CoRR*, vol. abs/1907.06987, 2019.

[47] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojír, G. Häger, A. Lukežič, G. Fernández, A. Gupta, A. Petrosino, A. Memarmoghadam, A. Garcia-Martin, A. S. Montero, A. Vedaldi, A. Robinson, A. J. Ma, A. Varfolomieiev, A. Alatan, A. Erdem, B. Ghanem, B. Liu, B. Han, B. Martinez, C.-M. Chang, C. Xu, C. Sun, D. Kim, D. Chen, D. Du, D. Mishra, D.-Y. Yeung, E. Gundogdu, E. Erdem, F. Khan, F. Porikli, F. Zhao, F. Bunyak, F. Battistone, G. Zhu, G. Roffo, G. R. K. S. Subrahmanyam, G. Bastos, G. Seetharaman, H. Medeiros, H. Li, H. Qi, H. Bischof, H. Possegger, H. Lu, H. Lee, H. Nam, H. J. Chang, I. Drummond, J. Valmadre, J. chan Jeong, J. il Cho, J.-Y. Lee, J. Zhu, J. Feng, J. Gao, J. Y. Choi, J. Xiao, J.-W. Kim, J. Jeong, J. F. Henriques, J. Lang, J. Choi, J. M. Martinez, J. Xing, J. Gao, K. Palaniappan, K. Lebeda, K. Gao, K. Mikolajczyk, L. Qin, L. Wang,

L. Wen, L. Bertinetto, M. K. Rapuru, M. Poostchi, M. Maresca, M. Danelljan, M. Mueller, M. Zhang, M. Arens, M. Valstar, M. Tang, M. Baek, M. H. Khan, N. Wang, N. Fan, N. Al-Shakarji, O. Miksik, O. Akin, P. Moallem, P. Senna, P. H. S. Torr, P. C. Yuen, Q. Huang, R. Martin-Nieto, R. Pelapur, R. Bowden, R. Laganière, R. Stolkin, R. Walsh, S. B. Krah, S. Li, S. Zhang, S. Yao, S. Hadfield, S. Melzi, S. Lyu, S. Li, S. Becker, S. Golodetz, S. Kakanuru, S. Choi, T. Hu, T. Mauthner, T. Zhang, T. Pridmore, V. Santopietro, W. Hu, W. Li, W. Hübner, X. Lan, X. Wang, X. Li, Y. Li, Y. Demiris, Y. Wang, Y. Qi, Z. Yuan, Z. Cai, Z. Xu, Z. He, and Z. Chi, "The visual object tracking vot2016 challenge results," in *Proceedings, European Conference on Computer Vision (ECCV) workshops*, pp. 777–823, 8Oct. 2016.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[49] S. Caelles, K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool, "One-shot video object segmentation," *CoRR*, vol. abs/1611.05198, 2016.

[50] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[51] E. Real, J. Shlens, S. Mazzocchi, X. Pan, and V. Vanhoucke, "Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video," *CoRR*, vol. abs/1702.00824, 2017.

[52] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[53] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. S. Huang,

"Youtube-vos: A large-scale video object segmentation benchmark," *CoRR*, vol. abs/1809.03327, 2018.