MOVING OBJECT DETECTION WITH SUPERVISED LEARNING METHODS

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

BY

AYBORA KÖKSAL

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

SEPTEMBER 2021

Approval of the thesis:

MOVING OBJECT DETECTION WITH SUPERVISED LEARNING METHODS

submitted by **AYBORA KÖKSAL** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

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

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

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

Dr. Kutalmış Gökalp İnce
Co-supervisor, **Center for Image Analysis, OGAM, METU** _____

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

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

Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU _____

Prof. Dr. Alptekin Temizel
Informatics Institute, METU _____

Prof. Dr. Hakan Çevikalp
Electrical and Electronics Engineering, Osmangazi University _____

Date: 7.9.2021

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: Aybora Köksal

Signature :

ABSTRACT

MOVING OBJECT DETECTION WITH SUPERVISED LEARNING METHODS

Köksal, Aybora
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. A. Aydın Alatan
Co-Supervisor: Dr. Kutalmış Gökalp İnce

September 2021, 95 pages

In this thesis, single target object detection problem is examined. Object detection is a problem that aims defining all of the objects of interest with their pre-defined classes in an image, or in a series of images. The main objective of this thesis is to exploit spatio-temporal information for performance enhancement during moving object detection. To this extent, modern object detection algorithms which are based on CNN architectures are analyzed. Based on this analysis, state-of-the-art techniques which are focused on utilization of temporal information on object detection are studied and some new methods are proposed.

Apart from the aforementioned analysis, some additional studies are also covered. The state-of-the-art object detection algorithms are based on the deep neural networks which are trained via supervised learning. Since these methods need lots of annotated data which also requires a lot of human labor, automatic and semi-automatic annotation methods are employed to overcome this problem. However, automated annotation methods sometimes result in erroneous annotations and effects of these type of errors are examined. A novel method is proposed to correct some type of such anno-

tation errors. This effort is extended with another preceding work, which suggests an alternative method for semi-automatic bounding box annotation for object detection with a significant reduction on annotation effort.

Keywords: object detection, video object detection, tracking with detection, annotation errors, annotation correction, semi-automatic annotation

ÖZ

DENETİMLİ ÖĞRENME METOTLARIYLA HAREKETLİ NESNE TESPİTİ

Köksal, Aybora
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan
Ortak Tez Yöneticisi: Dr. Kutalmış Gökalp İnce

Eylül 2021 , 95 sayfa

Bu tezde, tek hedefli nesne tespit konusu incelenmektedir. Nesne tespit bir görüntü veya görüntü grubundaki her bir nesneyi ön tanımlı sınıfıyla beraber tanımlamayı hedefleyen bir problemdir. Bu tezin ana hedefi uzam-zamansal bilgiyi kullanarak hareketli nesne tespit çalışmalarında performans artırımı elde etmektir. Bu amaçla, evrişimsel sinir ağları mimarisini baz alan modern nesne tespit algoritmaları incelenmiştir. Bu inceleme dahilinde, nesne tespit için zamansal bilgiyi kullanan güncel literatür araştırılmış, bazı yeni yöntemler önerilmiştir.

Üstte belirtilen incelemelerin yanında, çalışmada ayrıca bazı ek araştırmalar da yapılmıştır. Güncel nesne tespit algoritmaları denetimli öğrenme yöntemlerini içeren derin sinir ağlarını temel almaktadır. Bu metotlar yüksek seviyede insan emeği isteyen büyük miktarda işaretli veriye ihtiyaç duyduğundan bunu aşmak adına otomatik ve yarı-otomatik işaretleme yöntemleri ortaya çıkmıştır. Bunun yanında, otomatik işaretleme yöntemleri bazen hatalı işaretlemelere sebep olabilmektedir ve bu tarz işaretlemelerin etkisi araştırılmıştır. Bazı işaretleme hatalarını düzeltmek için yeni bir yöntem önerilmiştir. Bu çalışma başka bir araştırmayla bir aşama daha ilerletilmiş ve yarı otomatik

nesne işaretleme için işaretleme zahmetinde önemli ölçüde düşüş sağlayan alternatif bir yöntem önerilmiştir.

Anahtar Kelimeler: nesne tespiti, video nesne tespiti, tespitle takip, işaretleme hataları, işaretleme düzeltme, yarı otomatik işaretleme

To Mom and Dad,

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitute to my supervisor Prof. Dr. A. Aydın Alatan. His support from day one encouraged me to create this thesis. He broadened my vision not only on research but also teaching and other life aspects. I also would like to thank him for great lab environment (both physically and virtually) and immediate solutions for any of my problems.

Secondly, I would like to thank my co-advisor Dr. Kutalmış İnce for his support on my project and thesis work. With his ability to find quick and creative solutions even at the hardest times, I was able to cope with the problems that come up during the work.

This work is funded by ASELSAN Inc. I also would like to thank Dr. Yoldaş Ataseven and Burak Künkçü from ASELSAN SST for their help and support and lab equipments they have provided during our project.

This work is funded by TÜBİTAK within the scope of 2210/E scholarship.

I also would like to thank my friends from METU Center for Image Analysis (OGAM) for their help, support and friendship.

Finally, I am grateful to my family, Derya and Aydeniz, for their unconditional support and love throughout my life. They always take care of me and give guidance whenever I need. I dedicate my thesis to them.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional
ANN	Artificial Neural Networks
AP	Average Precision
CNN	Convolutional Neural Networks
FP	False Positive
IoU	Intersection over Union
LSTM	Long Short Term Memory
mAP	Mean Average Precision
SVM	Support Vector Machine
TP	True Positive

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Visual object detection is to predict the bounding box and the predefined object class information for each object in a given image. Due to the advancements introduced by Deep Learning paradigm and its most popular tool Convolutional Neural Networks (CNN) [1], the state-of-the-art object detection algorithms are mostly based on CNNs.

Most of the state-of-the-art object detection algorithms are designed to work on still images [2, 3, 4, 5, 6]. In other words, this class of algorithms do not utilize temporal data, such as temporal changes or correlation of the features of the adjacent frames. There are some visual object trackers [7, 8, 9] which consider such temporal information, but they are incompetent in terms of capabilities of object detectors, such as finding object class and bounding box without any prior.

Learning-based object detection algorithms are trained via supervised learning. Therefore, they require lots of annotated (i.e. labelled) data. In a normal setting, the annotation process requires a lot of human labor. In order to overcome this problem, automatic and semi-automatic annotation methods are employed. However, automated annotation methods are generally based on trackers, which might drift quite easily. This undesired drift causes erroneous annotations. One of the studies covered in the thesis examines the effects of such errors. Then, a semi-automatic method is proposed to correct specific type of annotation errors.

Another study is also achieved to complement the previous work by suggesting an alternative approach for semi-automatic annotation. This study proposes a solution

for a single target tracking scenario. The proposed algorithm creates tracklets which are the groups of visually similar detected objects in time. The proposed user interface helps human annotator to approve the tracklets for the correct annotation, which decreases annotation workload.

Modern object detection algorithms perform well on the public datasets, such as Imagenet [10], MS-COCO [11]. On the other hand, in more sophisticated scenarios, such as small objects, dim targets or objects in a cluttered background, an image object detector might perform poorly. Similarly, using a tracker cannot help in such cases, since a detector is required to start a track. Such objects are hard to examine even by a human in a single frame. An architecture which uses multiple frames for object detection might help to solve this problem.

In literature, there are some algorithms which aim to use the information of adjacent frames. Some of them uses the temporal data after the detection [12, 13, 14, 15], while some others use temporal information within the algorithm, for feature extraction [16, 17, 18, 19]. These are mostly based on former algorithms, which are not implemented with one shot detector, but their ideas can be applied to these detectors.

As the main aim of this thesis, novel algorithms that exploit temporal information for object detection are proposed. Initially, the usage of the temporal data is achieved as a post processing algorithm at the decision level. Then, the network architecture of some object detectors are modified in order to achieve spatio-temporal feature extraction.

1.2 Scope of The Thesis

This thesis has four main contributions. As an intial step, the state-of-the-art object detectors are compared on a fixed dataset that contains small targets to asses the merits of the related literature. As a second contribution, a recent object detector is used to examine the effect of annotation errors and a simple method is proposed to fix the annotation errors. Next, an object detector and a tracker-based method is proposed to semi-automatically annotate videos with single object. Finally, a number of algorithms are proposed to exploit spatio-temporal information of videos for moving

object detection.

1.3 Outline

In Chapter 2, up to date object detection literature is investigated and state-of-the-art detectors are trained with a dataset for comparison.

In Chapter 3, effect of annotation errors on object detection are investigated with some experiments and a method is proposed to correct one type of annotation errors.

In Chapter 4, a new method is proposed to semi-automatic video annotation for single object. The algorithm is also compared with the ones on literature.

In Chapter 5, literature of the algorithms which are responsible for moving object detection at decision level are investigated and some methods are proposed.

In Chapter 6, literature of the algorithms which are responsible for moving object detection at feature level are investigated and some methods are proposed.

CHAPTER 2

OBJECT DETECTION FROM STILL IMAGES

2.1 Introduction

Visual object detection techniques has been developed quite rapidly throughout the last decade. After CNNs become popular, conventional feature extraction algorithms are replaced with neural network-based convolutional backbones. Krizhevsky et. al. [1] initially showed the importance of a CNN architecture for feature extraction in image classification problem. After the introduction of VGG [20] and ResNet [21] architectures, the importance of the neural networks for feature extraction has increased even further.

CNN-based solutions that are commonly employed in object detection problem can be roughly divided into two classes, as two-state and single-stage approaches.

A two-stage object detector is composed of two sub stages. Firstly, for each image, a proposal is generated with the methods introduced in Section 2.2.1. Then, each proposal is processed by CNN layers for feature extraction. Finally, some fully connected layers or CNN layers used for bounding box regression and classification.

On the other hand, a single-stage (one-shot) detector, extracts features over a regular low resolution grid for the whole image, and performs detection, classification and bounding box bounding regression for each cell of the grid. After this step, some post processing operations, such as non-maxima suppression for the elimination of duplicated parts, are usually performed. These methods process each frame independently and since there is no object proposal generation step, a one-shot detector techniques usually process images much faster compared to a two-stage object detector.

In the rest of this chapter, deep learning-based image object detectors are examined. Firstly, they are compared in terms of their backbone architectures . Then, the state-of-the-art detectors are compared against each other in terms of accuracy and execution time.

2.2 Preliminaries

In object detection, or computer vision in general, raw images are neither discriminative nor computationally inefficient to process; hence, it is important to pre-process them to extract the information contained. This pre-processing step is called as feature extraction. A feature should be able to differentiate images with different properties for a given task. The features can either be high-level, such as height, brightness, corners etc. or complex, such as N-dimensional vectors, which cannot be understood by human. The feature extraction process is historically hand-crafted; i.e. they are generated by different algorithms that are carefully designed by computer vision experts. Recently, learning-based feature extraction methods are emerged. Such type of features are generated by artificial neural networks to minimize a loss function defined for a specific task. Therefore, such features rely on the size and variety of the dataset.

In the rest of this section, some classical and learning-based approaches will be discussed. Conventional approaches are mostly considered for proposal generation. Meanwhile, the learning-based approaches can be used for feature extraction, proposal generation or regression. The methodology to utilize these algorithms in object detection will be discussed in the next section.

2.2.1 Classical Object Proposal Generation Approaches

Selective Search algorithm [22] uses a superpixel generation method to create object proposals. A selection is achieved by graph-based segmentation which is introduced by Felzenszwalb and Huttenlocher [23]. By checking the similarity score with neighbour regions, similar neighbours are merged into the larger area. This process continues on until there is no distinct area left with high similarity score.

EdgeBoxes [24] is an object proposal generation approach which suggests that number of contours in a bounding box is related with the probability of the object existence in that box. The algorithm simply proposes an objectness score which is proportional to the number of edges in the box.

BING (Binarized Normed Gradients for Objectness Estimation) is a fast, handcrafted proposal generation method [25]. The main idea for BING is shown in Figure 2.1. BING extracts normed gradient map for different scales of the image and uses this information as the normed gradient features.

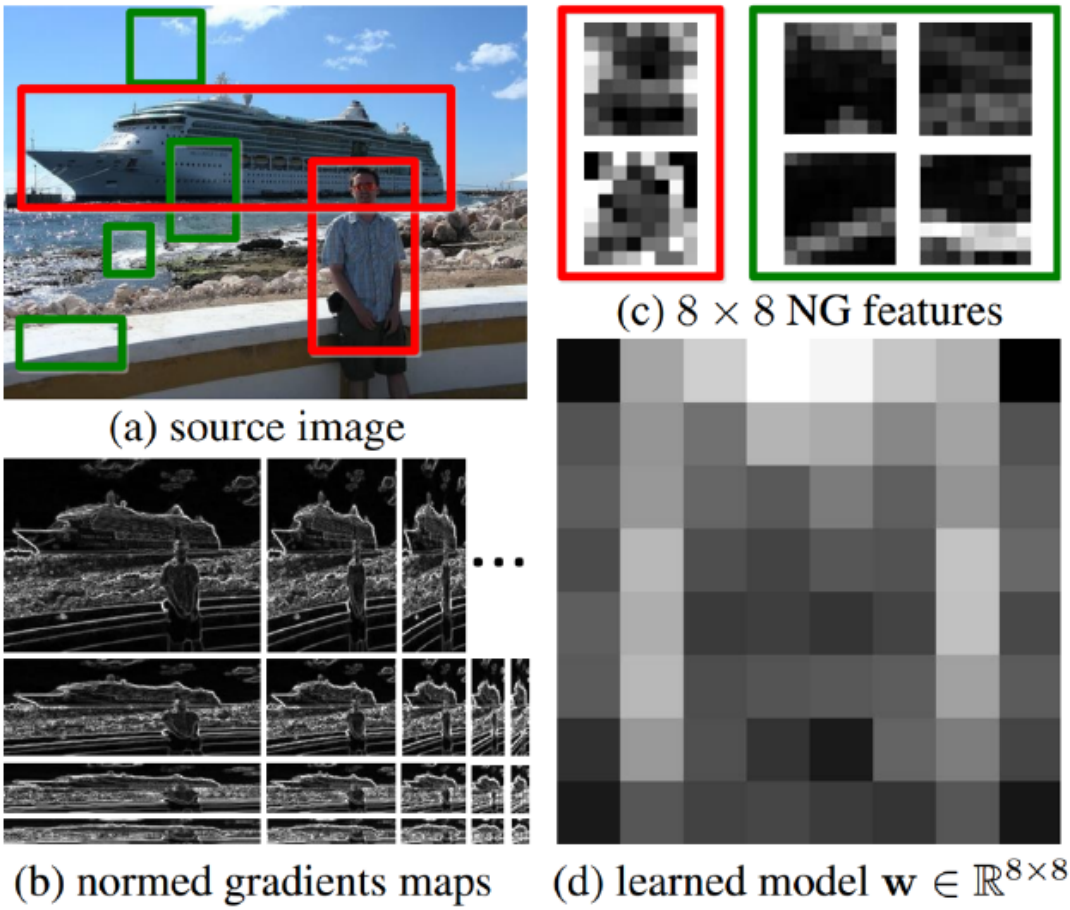


Figure 2.1: BING Algorithm. (a) Red boxes present objects while green ones present non-objects. (b) Image are resized with different scales and ratios, normed gradient maps are extracted for each scale. (c) normed gradient features for boxes given in (a). (d) linear model weights for normed gradient features. Taken from [25].

Multi-scale Combinatorial Grouping (MCG) [26] is another segmentation based ob-

ject proposal generation approach. It is a bottom-up hierarchical image segmentation technique which exploits multi-scale features. This multi-scale segmentation is obtained from newly proposed fast normalized cuts algorithm. Finally, a combinatorial grouping strategy is introduced to merge multi-scale features into one meaningful object proposal.

As it can be observed from the examined conventinonal methods, the objectness measure within a region is usually obtained in terms of a homogeneity metric, such as edges, gradients or intensity coherence.

2.2.2 Learning-based Object Proposal Generation

In this part, fundamental part of the learning based approaches will be considered. Object detection algorithms are significantly changed and advanced after the effect of convolutional layers on images are discovered. 2D and 3D convolutional layers can be used considering the dimension of the input. Later, ResNet architectures are introduced. Recently, various kinds of recurrent layer networks are used to exploit temporal information.

2.2.2.1 Basics of 2D CNN Architectures

Although convolutional neural networks (CNN) are introduced at late 90s by LeNet [27], they gain popularity at the beginning of 2010s, after the GPUs are started using for mathematical operations. One of the typical advantage of a convolutional network is that CNNs take an input with multiple dimensions and different sizes. Meanwhile, a fully connected network can only have a 1D input with a fixed size.

Another advantage of CNN is the reduction of mathematical operations. For example, for an edge detection operation on $320x280$ input, fully connected network need $320x280x319x280 \approx 8$ Billion operations. Meanwhile, convolutional network lets the same calculations to be done with $319x280x3 = 267960$ operations [28].

With the efficiency of CNNs, people are started to use them as a visual cortex for feature extraction. Sample convolutional operation can be seen in Figure 2.2.

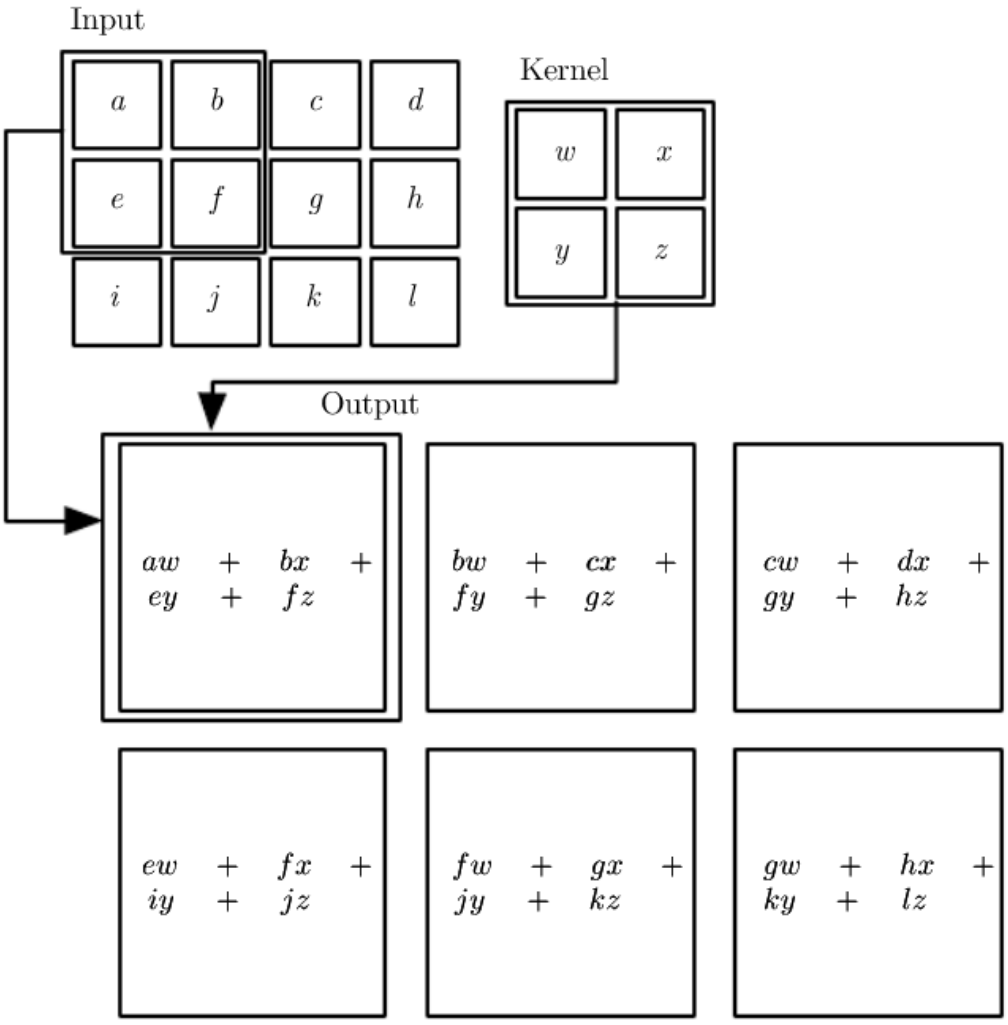


Figure 2.2: Sample 2D Convolution operation. Taken from [28].

Convolution operations can even get faster with some post operations. Stride and pooling can be used for dimension reduction. Stride lets the convolution operation skip n pixel(s) while sliding the kernel (2D convolutional block). Pooling on the other hand, works separately after the convolution is done. The most popular pooling types are max pooling and average pooling. Max pooling lets the network take the maximum of the output of m pixel(s). Meanwhile average pooling lets the network take the average of these m pixel(s).

2.2.2.2 3D CNN Architectures

The idea of 3D Convolution operations are same with their 2D counterparts. The main factor is the difference in the input type. In computer vision, 3D convolutional networks are generally used either for spatial feature extraction of 3D images, or spatio-temporal feature extraction of 2D images [29].

3D convolution operation brings much higher computational cost for the network. In order to deal with such a computational load, Pseudo 3D Networks are proposed [30]. The network uses 1x3x3 kernels for spatial features and 3x1x1 kernels for temporal features instead of using 3x3x3 kernels for spatio-temporal features, which reduces the cost.

2.2.2.3 ResNet

As the improvements of CNNs becomes more evident in computer vision, their architectures become deeper than before. However, deeper neural networks are not easy to train due to some issues such as gradient vanishing. He et. al. [31] find a solution to that problem by connecting input of a convolutional block to its output. This well-known network is denoted as Residual Network (ResNet), whose sample residual block is given in Figure 2.3.

2.2.2.4 Fundamentals of Recurrent Network Architectures

On contrary to a CNN architecture, which is used for regular 2D lattice inputs, such as images, recurrent layers are designed for processing sequential, mostly temporal, data. By parameter sharing, a model can be generalized over the different types of input. Using separate parameters may cause the failed generalization.

In order to use spatio-temporal information, machine learning and vision literature considers using LSTM alongside with 3D CNN.

LSTM: In 1997, Hochreiter et. al. [32] introduced Long Short Term Memory (LSTM) to cope with time lags on long time which can be seen standard recurrent

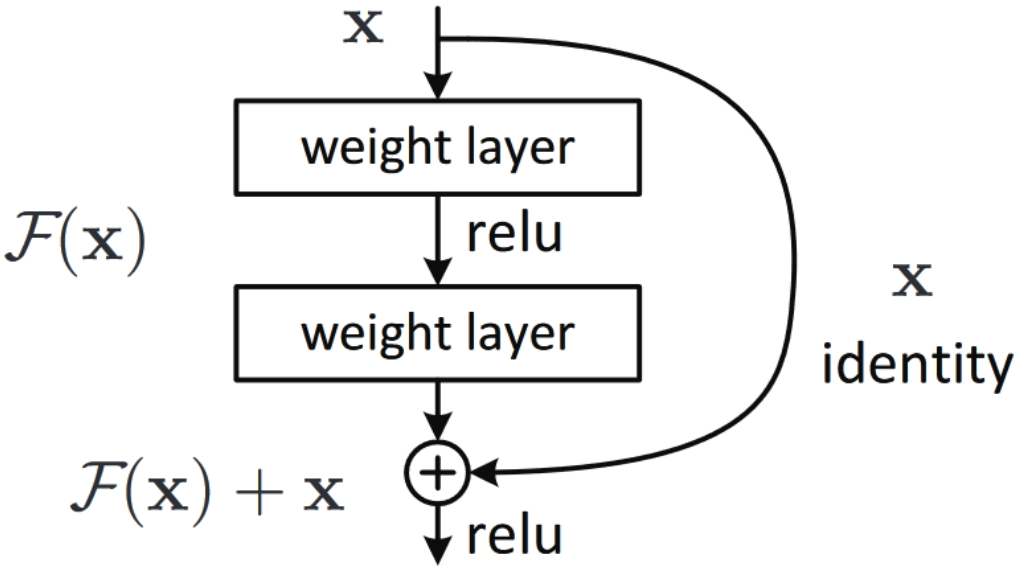


Figure 2.3: A residual learning block. Taken from [31].

layers. LSTM structure is presented in Figure 2.4. Currently, LSTM is the most common structure used for extracting spatio-temporal features in computer vision.

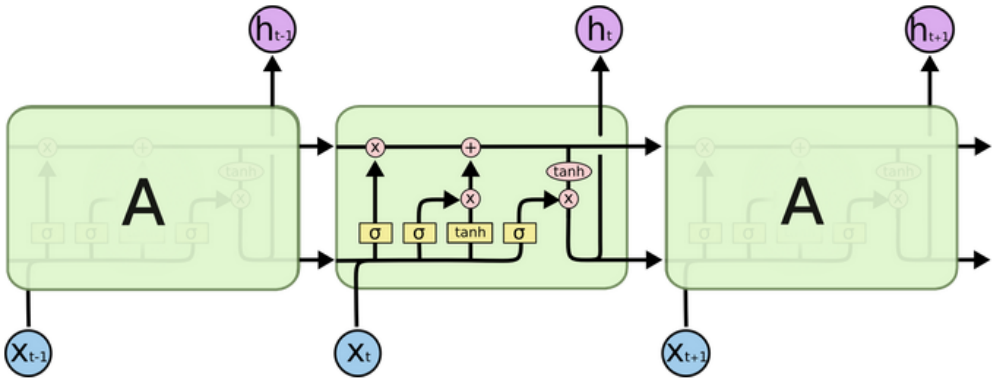


Figure 2.4: LSTM block. Taken from [33].

Using forget gate layer which uses sigmoid activation shown in Figure 2.5 (a), a cell state can be used completely, partially or it can be omitted.

The next stage, which is indicated in Figure 2.5 (b), is responsible for the decision of information stored in the cell state. This part is consisted of two parallel steps which will be combined. First step is a sigmoid activated input gate layer. This structure defines the values which will be updated. The second one is a layer with

tanh activation, which resolves the additions to the state, according to the new input. The third part which is visualized in Figure 2.5 (c), controls the update of the cell state, using the values found in first two parts. The last part which is shown in Figure 2.5 (d) is responsible for the decision of the output. The output value is a filtered version of the cell state. Initially, an output layer with a sigmoid activation is used for deciding which parts of the cell state is used. Then, the cell state is activated with a tanh function, so the parts which are useful are used for the output.

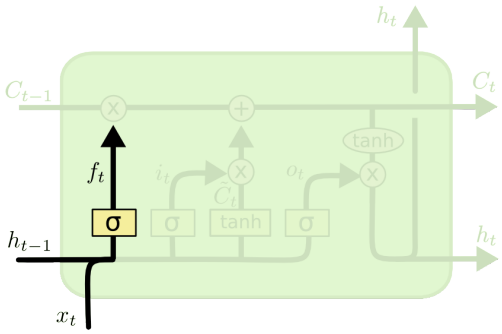
GRU: Cho et. al. [34] introduced Gated Recurrent Unit (GRU) as an LSTM variant. GRU has a single update gate instead of two different forget and input gates. Similarly, hidden state and cell state are also combined. Simplifications like these make the algorithm faster and easier to train.

Convolutional LSTM: Feature extraction capabilities of CNN and sequential information use of LSTM are merged into one layer in this approach [35]. For convolutional LSTMs, spatio-temporal information can be extracted better than fully connected vanilla LSTM with this approach.

In Convolutional LSTM, input should simply be a combined version of convolutional layers and LSTM layers. For a 2D convolutional layer, an input is a 4D structure with `(batch, channel, rows, cols)` to get useful information in 2D. For an LSTM layer, an input is a 3D structure with `(batch, time_step, features)` to exploit temporal data. As a result, a convolutional LSTM input should have a 5D structrure with `(batch, time_step, channel, rows, cols)` to extract spatio-temporal information.

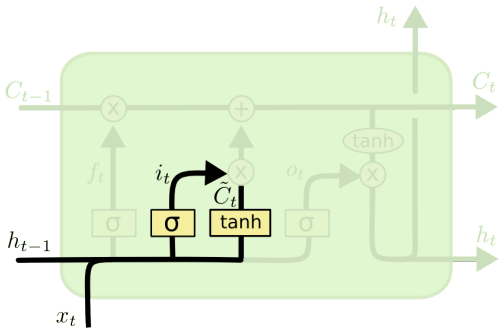
2.3 Related Work on Object Detection

As it is mentioned before, learning-based object detection techniques are broadly classified in two classes, as two-stage and single-stage methods. The following subsections examines the related literature for these classes.



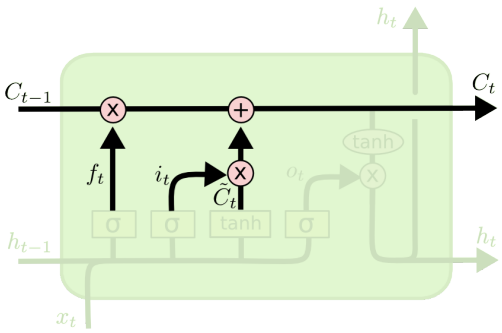
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

(a)



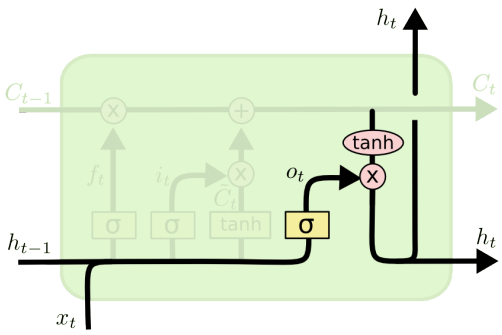
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

(b)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(c)



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

(d)

Figure 2.5: Detailed LSTM walkthrough. Taken from [33].

2.3.1 Two-stage Object Detection

After witnessing the success of CNN on image classification with AlexNet [1], R-CNN [36] is introduced. R-CNN is an hybrid method employing both classical techniques and learning-based methods. The method employs Selective Search [22] for proposal generation. For each region proposal feature extraction is performed via CNN and these feature vectors are classified with SVM, whereas the bounding box regression is performed by a fully connected neural network.

Shortly after R-CNN, Girshick et. al. proposed an improved version for it, namely Fast R-CNN [37]. This algorithm introduced RoI Pooling layers to reduce the computational cost. RoI Pooling basically pools feature proposals to the fixed sized shapes. Hence, for each image, feature extraction is processed once, instead of doing it for each proposal. They also introduced a deep learning based method for classification, and a regression network for finding bounding boxes, which is integrated with the feature extraction network.

Later, Faster R-CNN [38] is introduced by Ren et. al., whose main difference from Fast R-CNN is the proposal generation step, which is also performed by CNN layers (Region Proposal Network - RPN) instead of additional proposal generators. For the rest, Fast R-CNN is used and the two structures are merged into one network which makes the architecture end to end trainable for the first time. RPN structure and sample detections are provided in Figure 2.6.

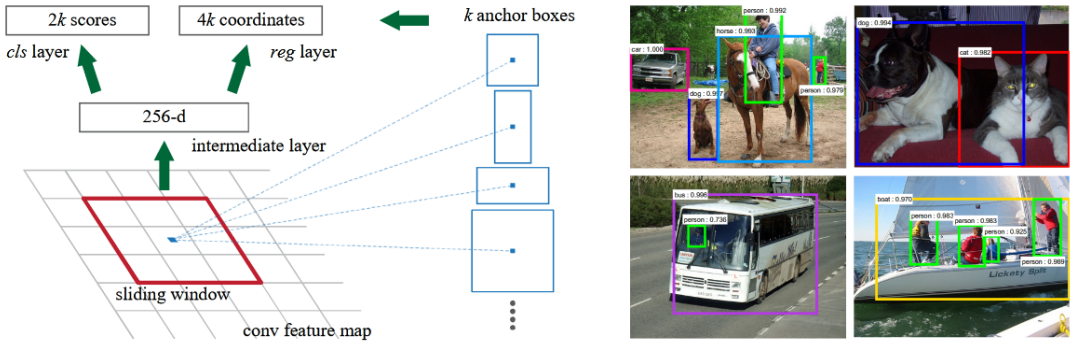


Figure 2.6: RPN structure and detection examples. Taken from [38].

Dai et. al. introduced Region-based Fully Convolutional Networks (R-FCN) [39] as

another two stage object detector. As the feature extraction backbone, ResNet-101 [21] is preferred. At the final layer of the network, position-sensitive RoI pooling layer is employed. This layer uses selective pooling, which introduces voting over divided RoI subregions. Calculating scores of these subregions, instead of the whole image, reduces the computational cost significantly.

He et al. [40] implemented Mask R-CNN as a modified version of Faster R-CNN to create a framework for object instance segmentation. In Mask R-CNN method, ResNet-FPN [41] (feature pyramid network) is utilized after the feature extraction backbone to connect deeper layers with the previous ones. Such an idea increases algorithm accuracy while reducing the computation time. FPN architecture is presented in Figure 2.7 (d).

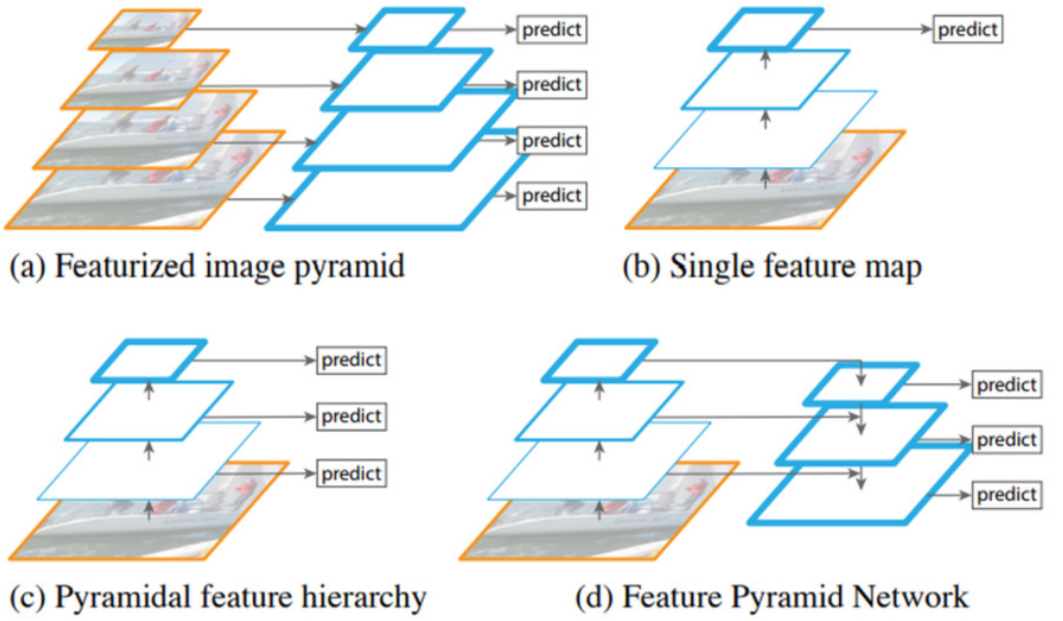


Figure 2.7: Feature pyramid alternatives. (a) Most straight forward solution since every size has its own network. Therefore, the network is very slow. Used in [42]. (b) One prediction at the end of the network. Gradients might vanish for small objects. Used in [43, 37, 38]. (c) Different predictions for different layers but previous layer prediction cannot use deeper layer information. Used in [4] (d) Feature pyramid network. Taken from [41] where this method is also proposed.

Before examining the other object detection class, it should be noted that two-stages

detectors follow the footsteps of conventional techniques by creating region proposals. However, for efficiency, one might argue that these end-to-end procedures could also be achieved in a single stage.

2.3.2 Single-stage Object Detection

As a small breakthrough for object detection literature, Redmon et al. [2] proposed the first one-stage object detector, YOLO (You Only Look Once) algorithm which works as a real-time application. Unlike the previous algorithms, such as Fast R-CNN, which does selective search for thousands of region proposals, YOLO simply predicts less than 100 bounding boxes for each image and this is the main reason for its real time performance. The algorithm simply divides the image into cells and each cell is responsible for the detection of the object, if the center of the object falls into that cell. That process is demonstrated in Figure 2.8. It should be remembered that such an approach tries to regress 2D coordinates of object locations from input images and such a procedure is not trivial in any aspect.

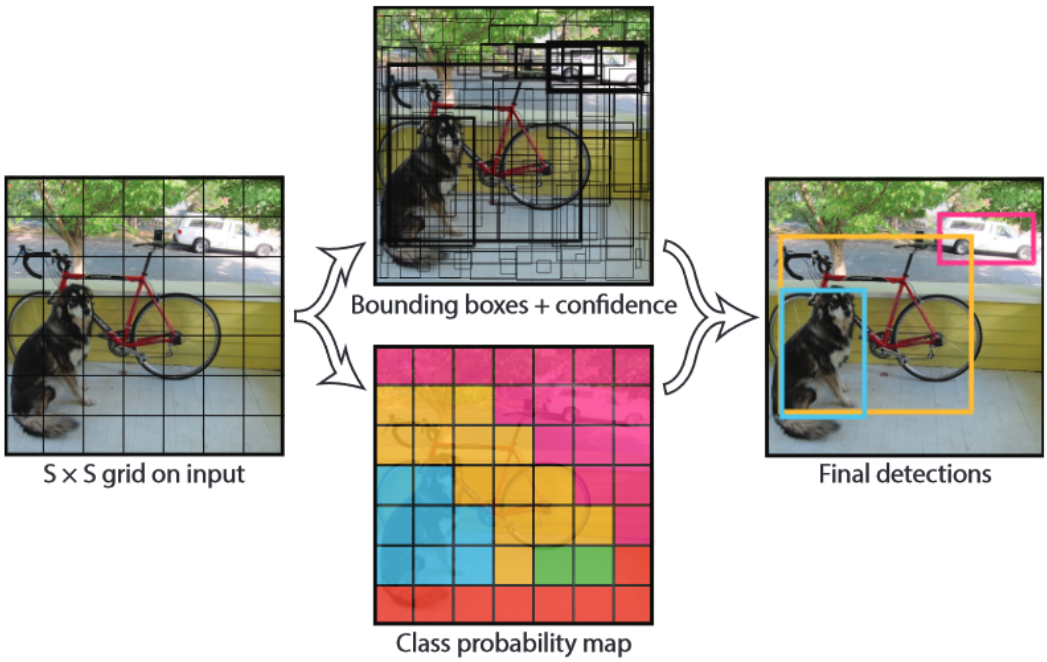


Figure 2.8: Each cell is responsible for the objects fall into that cell [2].

Shortly after YOLO algorithm is introduced, SSD (single shot detector) [4] is pro-

posed by Liu et al., works as another multiple class single-stage object detector, which regresses class confidences and bounding boxes from a fixed set of bounding boxes of different sizes and scales. SSD combines ideas from RPN of Faster R-CNN and YOLO; moreover, it also adds multiscale convolutional layers for feature extraction to increase detection speed while preserving accuracy.

Redmon et al. [44] later improved their work YOLO by an improved version, YOLOv2 which utilizes a completely new feature extractor backbone, namely Darknet19, since it consists 19 convolutional layers. In YOLOv2, fully connected layers are removed and only convolutional layers are used to predict bounding boxes.

In 2018, RetinaNet [5] is proposed by Lin et al. as another prominent one-stage object detector. The main novelty of RetinaNet algorithm is a new loss function, namely focal loss, in order to create a robustness against the class imbalance. The examples which give smaller loss values are considered as "easy samples" and the ones with higher loss values are assumed to be "hard samples". The focal loss metric decreases the weights of such easy samples and increases the weights of hard samples during training. Before the introduction of focal loss, the cross entropy (CE) function is used over the confidence scores as below

$$CE(p) = -\log(p) \tag{2.1}$$

Focal loss is introduced as a function by the following relation

$$FL(p) = -\alpha(1-p)^{\gamma}\log(p) \tag{2.2}$$

where α is a correction factor and γ is a focusing parameter. Parameter γ values larger than zero value increases the importance of the hard sample losses relative to the easy ones. Change of the loss values with respect to different focusing parameter levels can be observed in Figure 2.9.

The rest of the algorithms, which are considered as the state-of-the-art and utilized in this thesis, will be examined in detail in the subsequent sections.

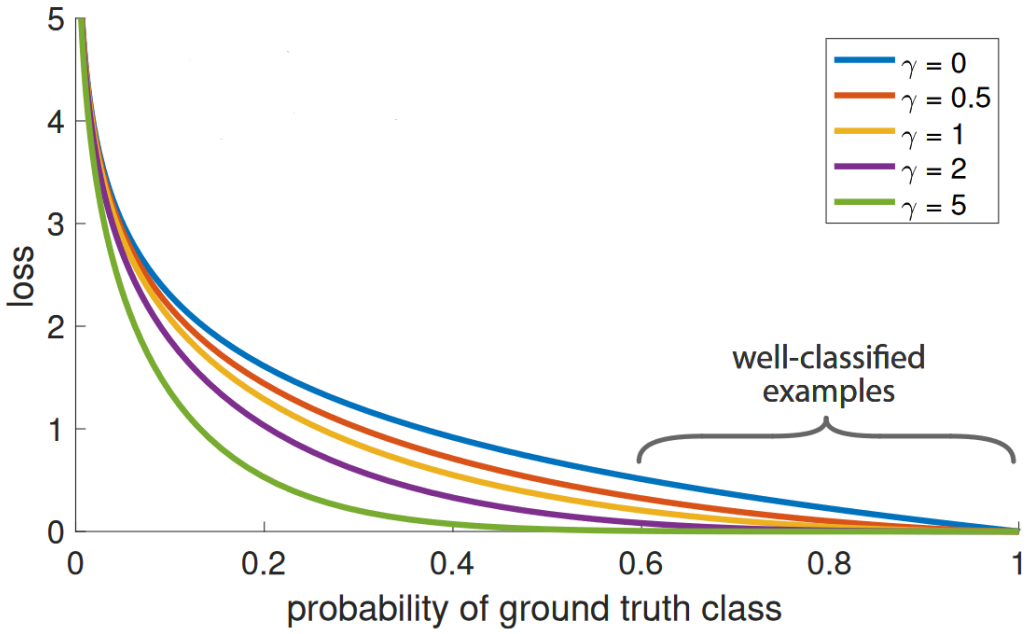


Figure 2.9: Model performance in terms of loss values with different focusing parameter values while α=1 [5].

2.3.2.1 YOLOv3

Redmon et al. [3] further improved YOLO baseline algorithm into a new version, namely YOLOv3. YOLOv3 enables multi-class detection by using logistic loss function instead of softmax layer, since there could be possible cases for which a cell contains more than one class. Detailed performance comparison is presented on Figure 2.10.

According to Figure 2.10, YOLOv3 outperforms SSD and R-FCN in terms of detection performance and Faster R-CNN and RetinaNet in terms of detection time. There are a few important factors which makes YOLOv3 better than the predecessor algorithms in terms of detection performance and/or detection time.

Backbone: Darknet backbone is advanced to Darknet53, with 53 convolutional layers, with batch normalization and Leaky-ReLU activation after each of these layers. Feature extractor backbone also consists of number of Residual blocks. Detailed architecture of YOLOv3 can be examined in Figure 2.11.

Feature Pyramids: Similar to the feature pyramid network (FPN) used in many

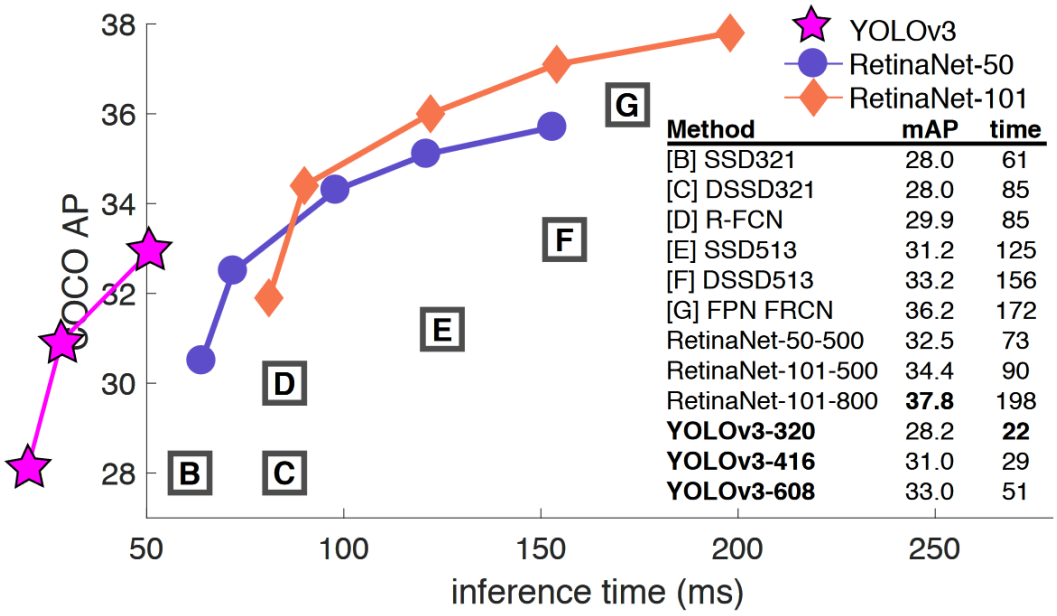


Figure 2.10: Comparison of YOLOv3 and the other state-of-the-art algorithms when the paper is introduced. Taken from [3].

other algorithms, which is illustrated in Figure 2.7 (d), YOLOv3 performs detection at three different scales for various sized objects. By using FPN architecture, each of these levels are connected to the other by a top-down approach. With such a connection, shallower layers are able to exploit the semantic information extracted in deeper layers.

Loss function: YOLOv3, calculates the objectness score with logistic regression for each bounding box. If a bounding box prior coincides a ground-truth object with an IoU larger than any other bounding box prior, then the logistic regression output should be 1. If a bounding box prior coincides, but it is not the best, then the prediction is ignored.

2.3.2.2 EfficientDet

In 2020, Tan et. al. [6] introduced EfficientDet as a state-of-the-art single shot object detector which outperforms its predecessors. According to the authors, EfficientDet achieves up to 55.1% AP score with its most complex network on COCO dataset [11]. Important differences of the algorithm with the previous ones are discussed below.

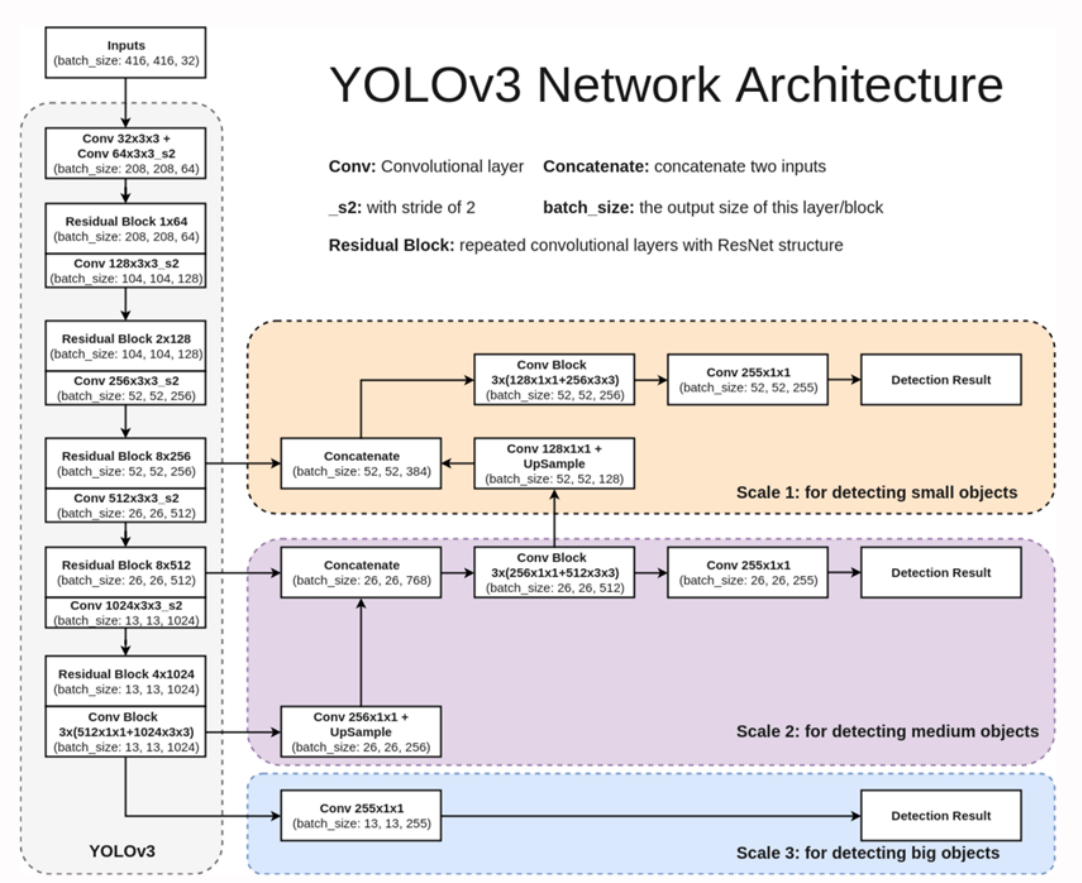


Figure 2.11: YOLOv3 architecture. Retrieved from [45].

Feature Pyramids: As discussed in YOLOv3 section, FPN connects multiple layers with a top-down path to exploit the features extracted in larger scales in smaller ones. In EfficientDet, this idea is improved one step further by the demonstration of a new structure, BiFPN. This method advances FPN with a bi-directional top-down & bottom-up approach without too much additional complexity. Comparison of the recent feature pyramid architectures is given in Figure 2.15.

Model Scaling: EfficientDet algorithm introduces a model scaling method which scales backbone network, feature extraction network and regression network at the same time (width, height, number of layers and/or resolution of these networks). All of these scaling are defined by a parameter ϕ. There are 8 different EfficientDet models that are proposed, from D0 to D7, which have $\phi = 0$ and $\phi = 7$, respectively.

The architecture of EfficientDet is presented in Figure 2.12.

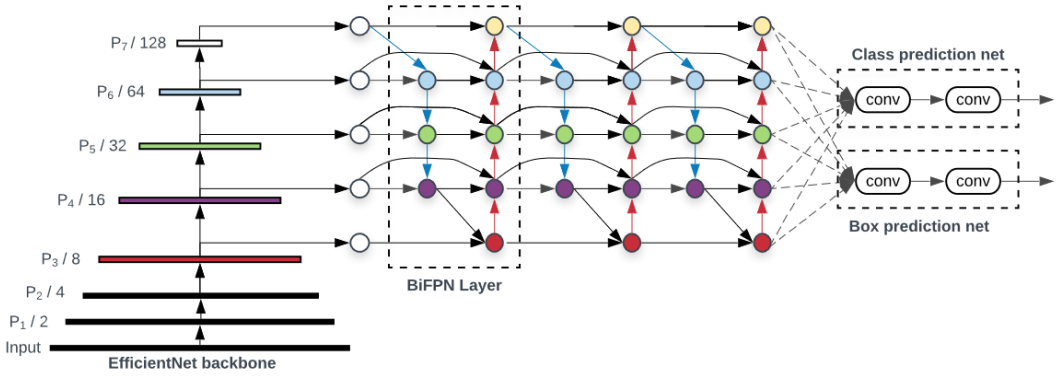


Figure 2.12: Architecture of EfficientDet [6].

2.3.2.3 YOLOv4

Bochkovskiy et al. [46] introduced YOLOv4 which is independent of the inventors of the previous YOLO algorithms. This new structure is also implemented on Darknet, the same framework used for its predecessors. According to the paper, AP score is improved by 10%, whereas FPS is also improved by 12% on MS COCO dataset [11]. Detailed performance comparison is presented on Figure 2.13.

According to Figure 2.13, YOLOv4 is better than YOLOv3 and EfficientDet for an equaivalent inference time. There are a number of important factors which introduces that 10% performance improvement against YOLOv3.

Backbone: YOLOv4 uses CSP-Darknet-53, which is a variant of Cross Stage Partial Network (CSPNet) [47] introduced in Darknet framework. The proposed structure simply connects feature maps by partially concatenating the top and the bottom layers of the network. This approach decreases computation cost by 20 % with at least the same accuracy on ImageNet dataset [10]. Structure of CSPNet is given in Figure 2.14.

SPP: YOLOv4 uses Spatial Pyramid Pooling (SPP) [48] in its backbone. SPP is another idea for handling different sized objects. Originally, the method is used obtain fixed size output for different sized inputs to feed the output to a fully connected layer. On the other hand, in [49], SPP is modified to make feature extraction more powerful. SPP block takes input feature maps and feeds into three parallel max pooling layers

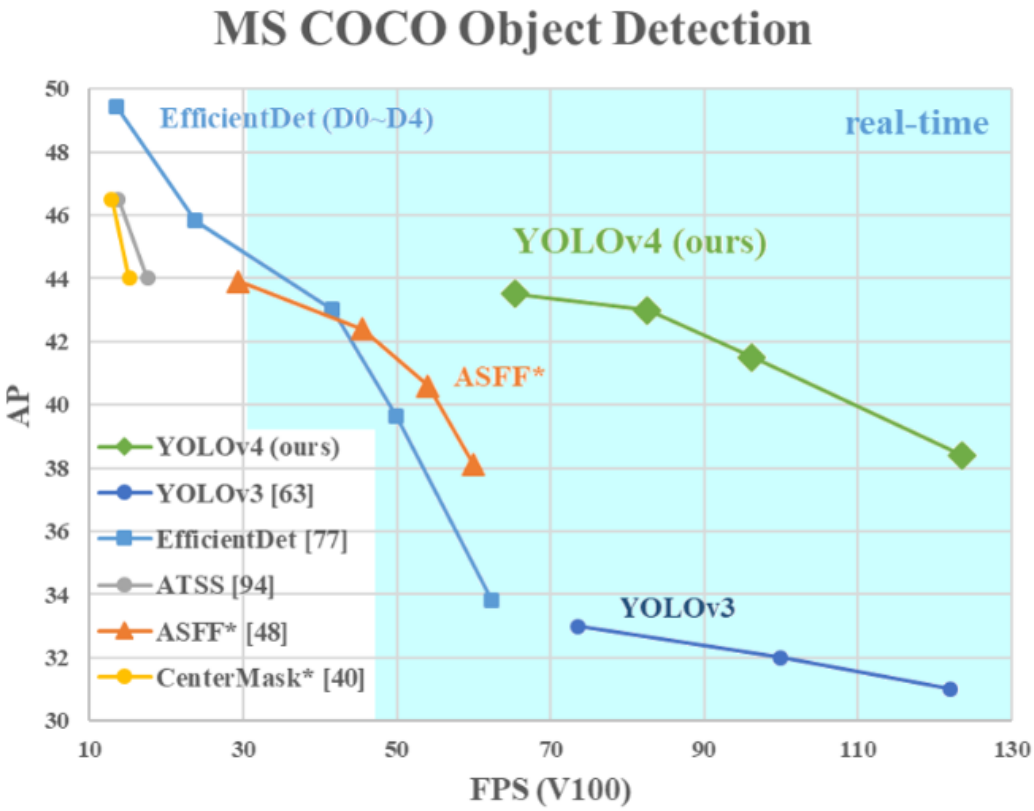


Figure 2.13: Comparison of YOLOv4 and the other state-of-the-art algorithms when the paper is introduced. Taken from [46].

with different scales and strides. Then these three layers are concatenated to have multi-scaled input features.

Activation: YOLOv4 uses Mish activation [50] on some convolutional layers in feature extraction backbone along with Leaky-ReLU activation which is already used in YOLOv3. Mish is a continuously differentiable activation function which can be defined by $f(x) = x \tanh(softplus(x))$. It outperforms Leaky-ReLU in MS-COCO dataset [11] by 2.1% on AP with CSP-Darknet-53 backbone.

Feature Pyramids: YOLOv4 adds a Path Aggregation Network (PANet) [51] to the generic Feature Pyramid Network which is introduced in order to use deeper features for the prior outputs. PANet simply introduces a bottom-up path which increases the communication between lower and higher layers. Comparison of the recent feature pyramid architectures is given in Figure 2.15.

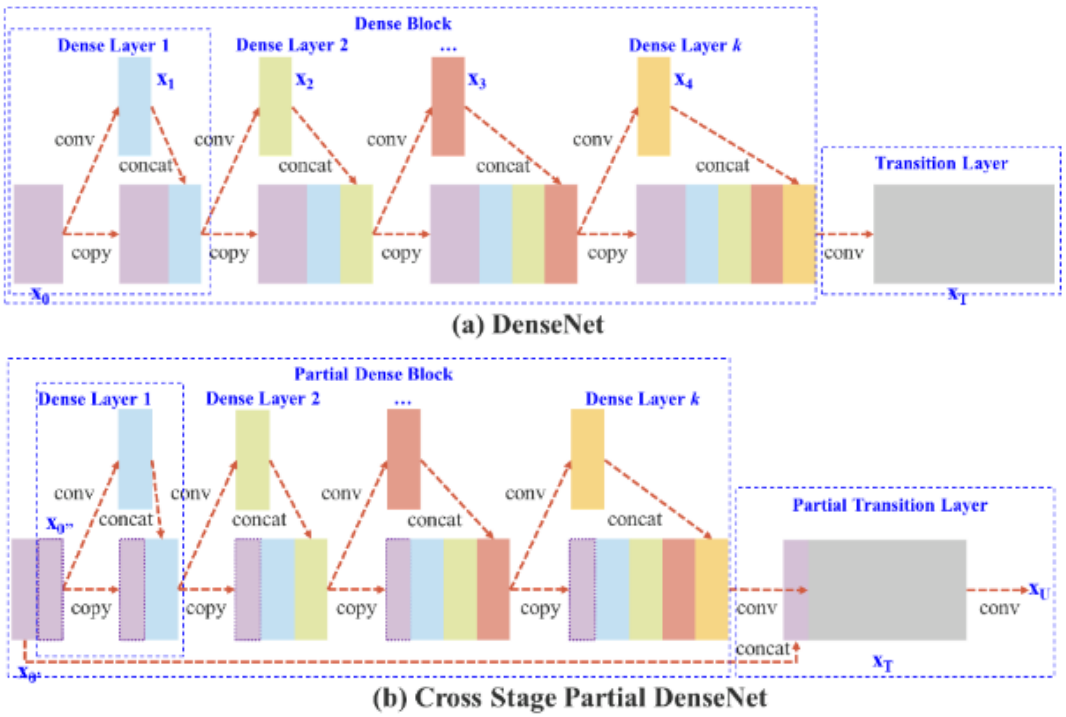


Figure 2.14: Comparison of (a) DenseNet (b) CSP-DenseNet. On CSP-DenseNet, first layer of the Dense block is divided into two parts. Whereas the first part goes through the dense block, the other part is directly concatenated with that output of the dense block. Taken from [47].

Data Augmentation: YOLOv4 is trained by a different fashion. Instead of using the same images at every epoch, it crops four different images and merges into a one network-sized image, so every time network runs across a new input. Moreover, in every image, batch normalization parameters can be trained by four different scenes. Hence, the demand for larger mini-batch size is decreased. This process is called mosaic and some sample images are given in Figure 2.16.

2.3.2.4 YOLOv5

Another YOLO implementation, YOLOv5 [53] is introduced just after the release of YOLOv4 by a different group. On the contrary to its predecessors, this structure is implemented on PyTorch, on top of a YOLOv3 PyTorch implementation [54] which is created by the same group.

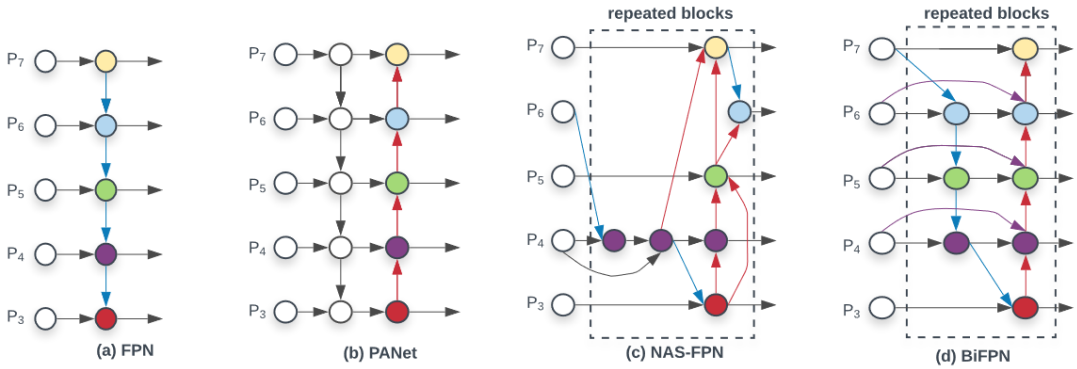


Figure 2.15: (a) FPN [41] suggests a top-down connection to use multi scale features. (b) PANet [51] advances this approach with an additional bottom-up connection. (c) NAS-FPN [52] uses a neural network architecture to connect related layers. (d) BiFPN [6] introduces a weighted bi-directional connection between layers. Taken from [6].

According to their repository, YOLOv5 outperforms another state-of-the-art algorithm EfficientDet around 10% AP in MS-COCO [11] dataset in the similar depth networks which gives a close level of FPS. Detailed comparison of these algorithms is given in Figure 2.17.

Since both algorithms are introduced quite recently, there is no documented comparison between YOLOv4 and YOLOv5. In general, in the datasets similar to ImageNet [10] or MS-COCO [11], their performances are quite close to each other. For small datasets, on the other hand, performances of these algorithms depend on the data.

YOLOv5 does not have any paper or documentation which explains the algorithm yet since it is open source, comparison with YOLOv3 and YOLOv4 is available. Architecture of YOLOv5 is given in Figure 2.18.

Backbone: YOLOv5 uses CSPNet like YOLOv4. This backbone is already discussed in Chapter 2.3.2.3.

SPP: YOLOv5 uses SPP as in YOLOv4. The method is already explained in Chapter 2.3.2.3.

Activation: YOLOv5 uses Leaky-ReLU function after all convolutional layers like

Figure 2.16: Mosaic data augmentation. Taken from [46].

YOLOv3, unlike YOLOv4, which has Mish activation on some layers. Performance of the Mish activation depends on the dataset.

Feature Pyramids: YOLOv5 adds a Path Aggregation Network (PANet) to the generic Feature Pyramid Network, just like YOLOv4. Details of PANet is already discussed in Chapter 2.3.2.3.

Data Augmentation: Mosaic augmentation, which is discussed in Chapter 2.3.2.3, is introduced by Jocher et. al. [54], who also implemented YOLOv5. Therefore, this method is used in both YOLOv4 and YOLOv5.

Focus (also called by DepthToSpace [56]) : Focus is a basic approach that aims to make the algorithm faster by reducing input resolution and cost of the convolution operation. This method simply decreases the width and height of a tensor, while increases the number of the channels.

As it can be realized from the aforementioned comparison, YOLOv4 and YOLOv5 have a similar structure. Architecture in Figure 2.18 is also applicable to YOLOv4 with small implementation differences in boxes. Therefore it is expected them to have close performances in a large enough diversified dataset. On the other hand,

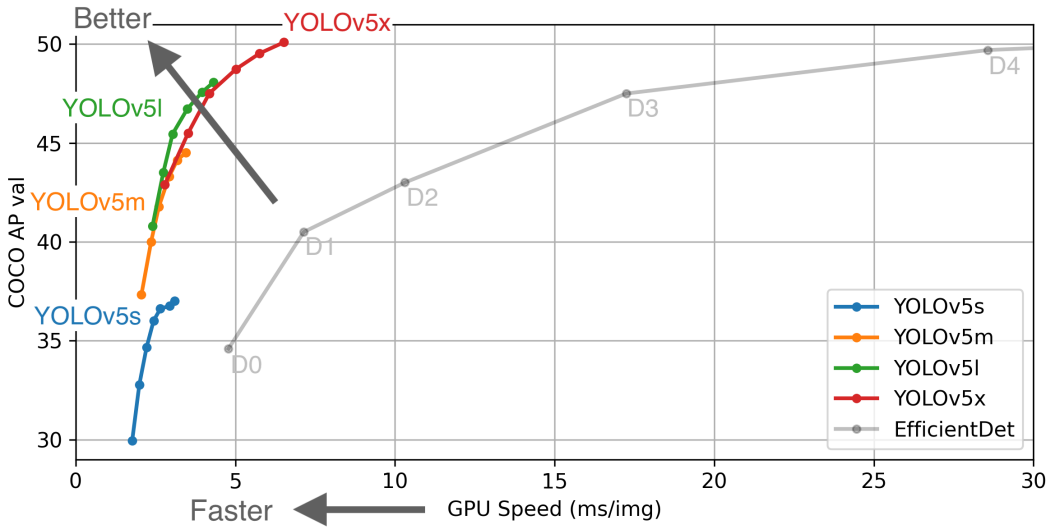


Figure 2.17: Comparison of YOLOv5 and EfficientDet with different network sizes. Taken from [53].

how they work on a small dataset which contains small targets can only be revealed with experimental results.

2.4 Experiments

Some controlled experiments are performed in order to compare state-of-the-art object detectors and decide the one which is working best in the related dataset. Reference object detectors are selected to compare with each other and the successful ones are used in the latter stages of the study. First of all, only one-shot object detectors are considered for the work, since they can process the frames in real-time or near real-time.

The results given in [57] shows that at that time, the only algorithm that compete with YOLOv3 is RetinaNet, which works 4 times slower than YOLOv3. On the other hand, according to [46] and [53], YOLOv4 and YOLOv5 are introduced as better than YOLOv3 and EfficientDet, another state-of-the-art object detector. YOLOv4 and YOLOv5 is not compared with each other for detection performance; therefore, they should be experimented separately. Hence, YOLO variations are considered for the experiments as reference object detector.

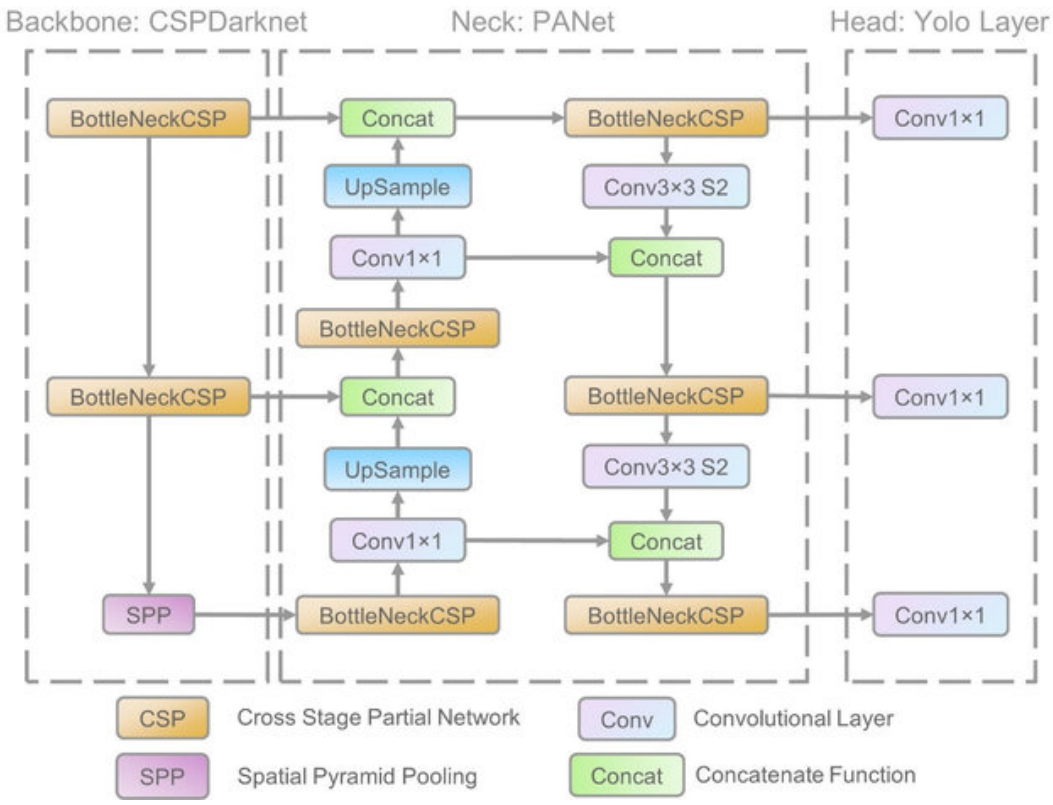


Figure 2.18: YOLOv5 architecture. Features are extracted via CSPNet, then PANet is used for the communication of the features. Then YOLO Layers regress the output for the detection. Taken from [55].

2.4.1 Experimental Setup

2.4.1.1 Dataset

Throughout the thesis, the performance of the detectors on Unmanned Aerial Vehicles (UAV - Drones) detection problem are compared. For this purpose, in this chapter, test-dev folder of ICCV 2021 - 2nd Anti-UAV Challenge Workshop dataset [58] is used. ICCV 2021 - 2nd Anti-UAV Workshop dataset is available at [58]. Some samples from the dataset are presented in Figure 2.19.

Dataset consists of 140 thermal infrared videos including UAVs of various sizes. There are different types of occurrences of drones, such as in front of a building, clear sky, cloudy sky, mountain, sea etc. This type of variety makes it possible to compare the object detection algorithms in different cases.

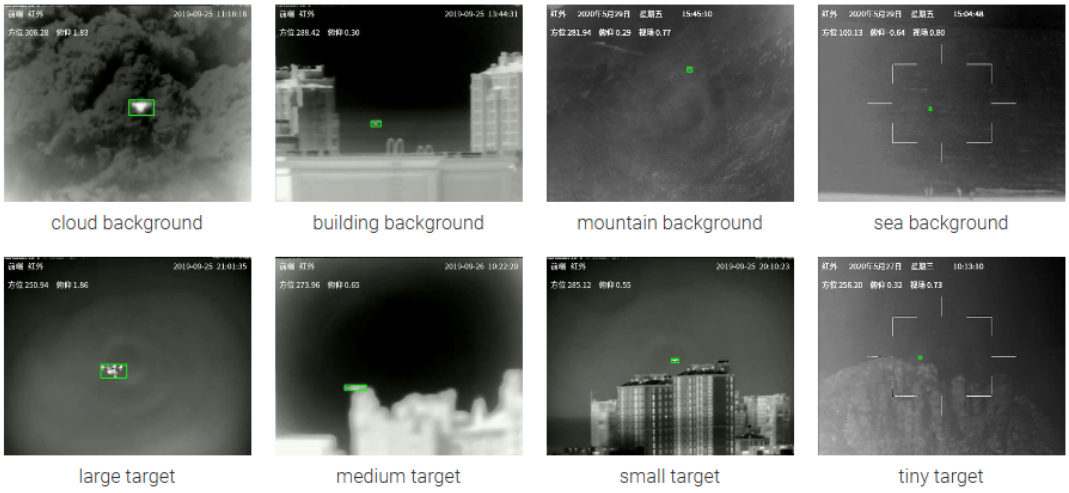


Figure 2.19: Variety of occurences of UAVs in Anti-UAV dataset [58].

In this work, randomly selected 30% of the videos are allocated as validation set and the rest is used as training set.

2.4.1.2 Object Detection Performance Metrics

For the comparison of the algorithms, *Hit Rate* and *False Alarm Rate* metrics are presented. These metrics are calculated by using *IoU, True Positives, False Positives* which are defined as follows and exemplified in Figure 2.20:

$$IoU = \frac{Area\ of\ overlap\ between\ detection\ and\ ground\ truth}{Area\ of\ union\ of\ detection\ and\ ground\ truth} \tag{2.3}$$

$$True\ Positive : \text{Detections whose } IoU > 0.5 \text{ with ground truth.} \tag{2.4}$$

$$False\ Positive : \text{Detections whose } IoU = 0 \text{ with ground truth.} \tag{2.5}$$

$$Hit\ Rate = \frac{Number\ of\ true\ positives}{Number\ of\ samples\ with\ an\ object\ present} \tag{2.6}$$

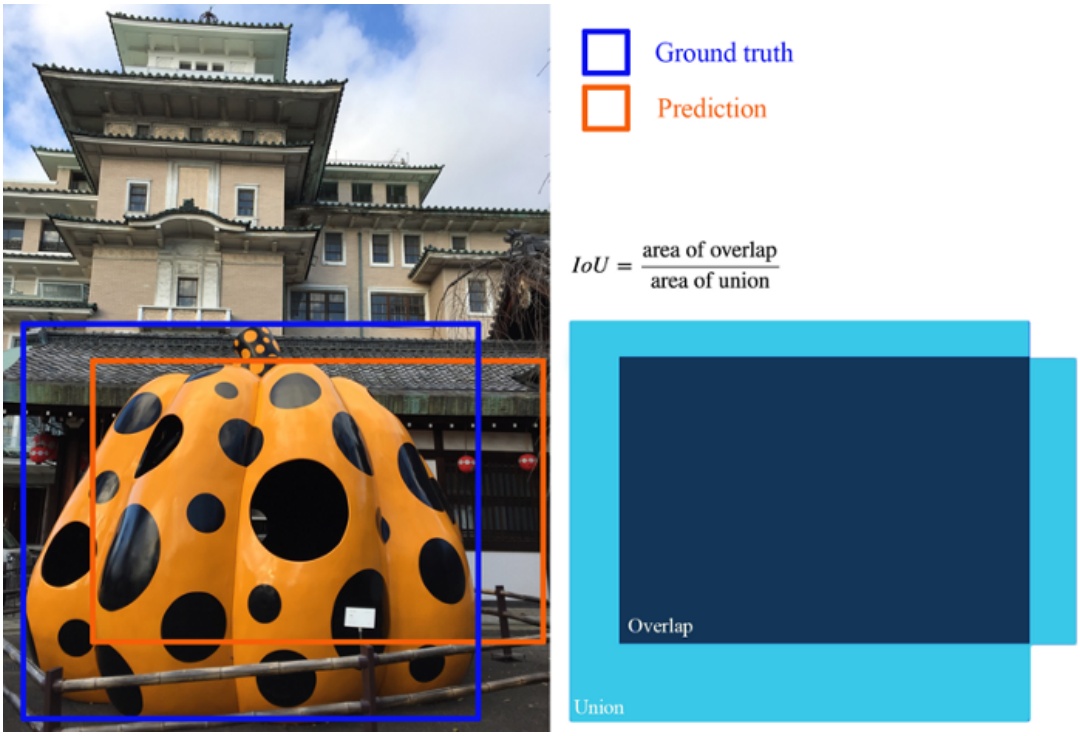


Figure 2.20: IoU Definition. Taken from [59].

$$\textit{False Alarm Rate} = \frac{\textit{Number of false positives}}{\textit{Length of dataset in terms of minutes}} \tag{2.7}$$

2.4.2 Experimental Results

When the objectness threshold is fixed to 0.5, performance comparison on validation set in terms of *Hit Rate* and *False Alarm Rate* for YOLOv3, YOLOv4 and YOLOv5 are presented in Table 2.1.

Table 2.1: Performance comparison of YOLOv3, YOLOv4 and YOLOv5 with a fixed threshold (0.5) in terms of Hit Rate and False Alarm Rate

	HR	FA
YOLOv3	80.90	14.69
YOLOv4	83.03	54.46
YOLOv5	83.34	30.99

According to Table 2.1, YOLOv5 performs better than YOLOv4 on this validation set, having a higher hit rate and a lower false alarm rate. On the other hand, the performance of YOLOv3 cannot be compared with others, since it tends to generate less alarms which result in both lower hit rate and lower false alarm rate.

In order to have a fair comparison, the hit rates are presented for three different false alarm rates, which are achieved by these algorithms when the objectness threshold is set to 0.5. To achieve these pre-defined false alarm rates, objectness threshold is adjusted for each algorithm independently. The resulting hit rates and and corresponding objectness thresholds are presented in Table 2.2.

Table 2.2: Performance comparison of YOLOv3, YOLOv4 and YOLOv5 with fixed FAs in terms of Hit Rate. Objectness threshold levels are also given for reproducibility.

	FA = 14.69		FA = 54.46		FA = 30.99	
	HR	THRS	HR	THRS	HR	THRS
YOLOv3	80.90	0.50	82.48	0.11	82.06	0.22
YOLOv4	78.87	0.73	83.03	0.50	81.44	0.61
YOLOv5	**81.75**	**0.59**	**84.44**	**0.41**	**83.34**	**0.50**

According to Table 2.2, YOLOv5 outperforms the others regardless of the fixed false alarm rate. Comparison of YOLOv3 and YOLOv4, on the other hand, is more complicated. In higher objectness threshold values, i.e. lower false alarm rates, the results are in favour of YOLOv3. However, as the threshold gets smaller, YOLOv4 starts to decrease the gap and at the reference false alarm value 54.46, one can argue that YOLOv4 is better than YOLOv3.

2.5 Conclusion

According to the results, YOLOv3, YOLOv4 and YOLOv5 have similar performances in UAV detection problem, within a 3% range. On the other hand, there are some structural improvements in the more recent versions of YOLO, as explained in the

chapter and such performance improvements are difficult to gain in image object detectors.

Although YOLOv4 and YOLOv5 are introduced in a small amount of time, have similar structures and close performances on public and big datasets, they still have some performance gap due to small implementation changes and architectural differences.

In the rest of this thesis, YOLOv3 and YOLOv5 algorithms are used as the baseline object detectors, since YOLOv3 is the most popular generic object detector and YOLOv5 is found out to outperform YOLOv3 in the selected dataset during simulations.

CHAPTER 3

ANALYSIS AND CORRECTION OF OBJECT ANNOTATION ERRORS

3.1 Introduction

A typical one-shot detector tries to find all the defined class objects with their bounding box information and objectness scores. After this step, some post processing stage might be employed, such as non-maxima suppression, to eliminate duplicated results. Since these methods are designed to perform on each image independently, they can be employed for tracking problems as well without any drift problem. As a result of recent developments in GPU technology and efficiency enhancements of one-shot detectors, there are alternative methods working in near real-time [44, 5] that made them to be employed in real-time tracking problems.

Similar to the other CNN-based object detectors, YOLOv3 also requires a large amount of labeled data during its training process which requires significant amount of labor. In order to save human labor, especially for video annotations, labeling a small number of video frames by hand and interpolating the intermediate video frames via automatic tracking can be an acceptable idea. Unfortunately, as any automatic tracking algorithm is not perfect, there might be a discrepancy between the real data and interpolated data which results in annotation errors; yet their effect on object detection performance is not studied in detail.

This chapter has three objectives: The first objective is to reveal the performance of a state-of-the-art detector for small objects which can serve as a baseline for detection-based tracking methods. The second objective is to investigate the performance of a trained visual object detector in the presence of training annotation errors. The final

This chapter has been presented in [60].

objective of this chapter is to come up with a semi-automatic method to correct such annotation errors.

Firstly, YOLOv3 algorithm is trained by using CVPR-2020 Anti-UAV Challenge dataset [58] after fine-tuning the existing weights of the algorithm to detect drone classes. YOLOv3 is trained for different number of drone classes and different number of epochs with different amount of data to figure out the most efficient way of training in terms of training time and performance. Next, the tracking accuracy of YOLOv3 is analyzed by considering the original annotations.

After selecting the best way for training, some additional annotation errors are applied to the dataset in order to create separate new datasets each of which consists different type of synthetically generated errors, even some combined ones. Then, YOLOv3 is trained with each of these new erroneously annotated datasets and the results are compared in terms of precision, recall and tracking accuracy.

Since some erroneous annotations are observed in CVPR-2020 Anti-UAV Challenge dataset, a novel semi-automatic approach is also proposed to correct erroneous annotations to improve the labeling accuracy of this valuable dataset. Moreover, the accuracy between corrected and original labels are calculated in terms of mean and standard deviation.

The rest of this chapter is organized as follows: Firstly, related work on deep learning-based object detectors and training with noisy data are presented. Section 3.3, 3.4 and 3.5 are dedicated to training of YOLOv3 with Anti-UAV Challenge dataset and semi-automatic annotation correction. In Section 3.6, the results for original, noisy and corrected datasets are compared. Conclusions of the experimental evidence are presented in the last section.

3.2 Related Work

As stated before in this thesis, deep learning-based object detectors are mostly classified into two classes in the literature: Two-stage (region proposal based) and one-stage detectors [57, 61]. On the other hand, there are also few-shot learning algo-

rithms which are mostly used for object counting or segmentation [62, 63]. In order to compete with trackers, in this study, only one-stage detectors are considered that work in real-time or near-real-time. According to the results in [57], on MS COCO dataset [11] YOLOv3 achieves 57.9% mAP, while RetinaNet has 61.1% mAP; meanwhile YOLOv3 operates nearly 4 times faster than two-stage RetinaNet algorithm. Moreover, YOLOv3 is a better alternative for small objects (e.g. with drones), since it uses multi-scale detection. As it provides nearly real time object detection ability with high potential performance on small object detection, YOLOv3 is selected for the erroneous annotation experiments.

3.2.1 Training with Erroneous Annotations for Object Detection

Labeling errors in the training data is already examined within the object detection literature. Frenay et al. [64] defined annotation errors as an independent stochastic process which may or may not be introduced intentionally. The authors have performed a detailed survey that includes learning in presence of labeling noises, such as some probabilistic models which are Bayes-optimal classifiers [65]. Moreover, they included some semi/weakly supervised methods [66] that prevent mislabelled instances from affecting detection performance considerably. Moreover, the authors examined some noise-cleansing algorithms, such as detection of mislabelled instances by using class confidence metrics [67].

Rolnick et al. [68] argues that introducing label noise into a training set reduces the performance of CNNs, although it is not as remarkable as the multi-layer perceptron networks. In addition to this argument, the authors also state that deeper networks, such as ResNet [21], are less affected from such noises. Moreover, the authors conclude that to attain the same accuracy level, the training set with higher rate of noisy labels need to be larger.

Noisy labels can also be a problem for weakly supervised object segmentation tasks. Lu et al. [69] introduces a superpixel noise reduction algorithm, which is based on a sparse learning model. Next, with this cleaned labels, an iterative superpixel label prediction/appearance model is created. Using this method, the authors increased total per-pixel accuracy by 5 to 15% in comparison to the best other method [70].

In one of the most relevant recent efforts [71], the authors trained their SSD-based framework with KITTI dataset [72] and artificial annotation errors, which are additional boxes, missing boxes and shifted boxes. A typical visual of a sample annotation error on KITTI dataset is presented in Figure 3.1. The performance of SSD with and without annotation errors are also reported as shown in Table 3.1. According to the results, additional boxes decreases performance, however this decrease is not related with the noise probability. Missing and shifted boxes on the other hand, decreases precision further by increasing noise probability, with similar rates. Upon all of the noise types, combined labeling noise affects the network most, as expected.

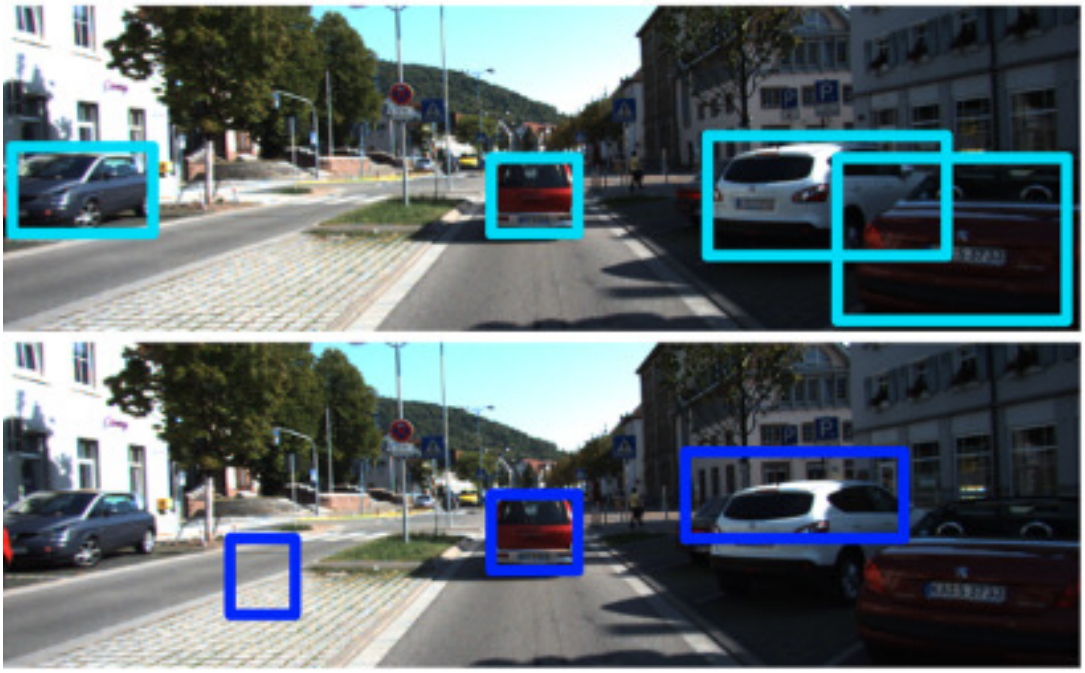


Figure 3.1: Examples of noisy labels on KITTI dataset. First image shows real ground truth labeling while the latter one shows some noises [71].

3.3 Performance Metrics

Since Anti-UAV Challenge dataset is aimed for visual tracking problem, the performance metric for this challenge is announced as the average intersection over union based on the assumption that there is at most one output object on each frame. However, in this chapter, a detection algorithm is studied; therefore, additional perfor-

Table 3.1: Performance on KITTI dataset in terms of average precision with different types of noises with varying levels. [71].

	Noise Probability		
Noise Type	0.0	0.25	0.5
No Noise	0.629	-	-
Additional boxes	-	0.560	0.587
Missing Boxes	-	0.593	0.518
Shifted Boxes	-	0.577	0.502
Combined	-	0.457	0.317

mance metrics are also required. Hence, for this purpose *hit rate* and *number of false alarms* are evaluated as the additional performance metrics. If a detection output has IoU larger than 0.5 for the annotated object, then this result is counted as a *hit* (Pascal criteria). In case of zero IoU, the decision is counted as *false alarm*. Finally, for non-zero IoU smaller than 0.5, no additional penalty is applied as a false alarm, since the annotated object is *missed* and penalty is already included in the hit rate. No detection output for no annotation, i.e. true reject, is not counted. For the rest of the chapter, comparison results are presented in terms of false alarms per minute and hit rate, in addition to the tracking accuracy metric given in Anti-UAV Challenge, TA which is defined as:

$$TA = \frac{1}{T}\sum_{t=1}^{T} IoU_t * v_t * p_t + (1 - p_t)(1 - v_t) \tag{3.1}$$

where T is number frames, IoU_t is intersection over union, $v_t \in \{0, 1\}$ is visibility flag, and $p_t \in \{0, 1\}$ is prediction flag at frame t.

Since the object detector YOLOv3 might generate more than one detection result on a single frame, tracking accuracy metric in Eq. 3.1 cannot penalize additional false alarms. Therefore, this metric is also slightly modified so that the false alarms reduce the accuracy. Modified tracking accuracy is defined as follows to penalize additional

false alarms:

$$MTA = \frac{\sum_{t=1}^{T} IoU_t * v_t * p_t + (1 - p_t) * (1 - v_t)}{\sum_{t=1}^{T} max(v_t, p_t) + (1 - p_t) * (1 - v_t)} \tag{3.2}$$

This modified tracking accuracy is equal to original tracking accuracy as long as the number of detection per frame is limited to one, but each additional false detection reduces the tracking accuracy.

3.4 Training YOLOv3 with Anti-UAV Dataset

Modified Network: YOLOv3 network is pretrained to detect 80 different classes, while the input image is divided into grids on three different scales. For each grid cell in each scale, YOLOv3 generates a vector containing the objectness score, class probabilities and bounding box for three alternative anchor boxes. Therefore, for each cell the length of the output vector is 3x(1+80+4)=255. For drone detection, YOLOv3 have been trained only for one- and three-class alternatives resulting in output vectors of length 3x(1+1+4)=18 and 3x(1+3+4)=24, respectively. For the one-class case, the network is trained only with thermal images to detect drones. For three-class case, the network is trained with RGB day, RGB night and thermal images which correspond to three different drone classes. The performance of these two alternatives are compared to understand whether there is a significant difference between one class and three class cases or not.

Dataset: For the thermal image dataset, "test-dev" part of Anti-UAV Challenge dataset is used. RGB videos are also included for three-class scenario to examine whether the including them increases the accuracy or not. The video sequences are divided randomly as training and validation set with the ratio of 70% and 30%, respectively.

Training: During the training, different dataset sizes and different epoch numbers tested for one- and three-class alternatives. Since for each annotation error, the network should be trained with the simulated erroneous annotations, precision/training time efficiency is considered for comparison. The results of 25, 50, 100th epochs with full dataset, half dataset which is obtained by getting one frame and skipping the next

one, and one quarter dataset which is obtained by getting one frame and skipping the next three, are also compared.

As tabulated in Table 3.2, the trained network produces quite similar results for one-class and three-class cases for the full dataset, whereas training time is doubled for the three-class scenario. Hence for the rest of the chapter, one-class scenario is studied. Moreover, at 100th epoch, false alarms are increased due to some memorizing or overfitting. Therefore, for the rest of the chapter, only 25th and 50th epochs are compared. As presented in Table 3.3, the most efficient performance is on 50th epoch for half dataset. Since data from the adjacent video frames are quite redundant, removing half of the dataset does not decrease performance of the network. On the other hand, using only quarter of the dataset decreases the performance. However, it is difficult to deduce whether this result is due to either losing data variety or number of samples in the set. To sum up, for the rest of the chapter, the network is trained for one-class only with the half of the thermal images for 50 epochs.

Table 3.2: Performance comparison of YOLOv3 on Thermal Test Set when trained only with thermal data (one-class) vs Thermal+RGB data (three-classes) in terms of Hit Rate (%), False Alarm (per minute) and Training Time (hours)

# Epoch	25			50			100		
	HR	FA	TT	HR	FA	TT	HR	FA	TT
Thermal	97.5	2.4	17	97.1	2.2	34	97.3	3.5	68
Thermal+RGB	96.9	2.3	34	97.4	1.7	68	97.9	4.3	136

3.5 Annotation Errors in Anti-UAV Dataset

In order to assess the behavior of YOLOv3 on Anti-UAV Challenge dataset better, the outputs of the algorithm have been carefully inspected, especially the frames on which the algorithm fails, i.e. frames with low IoU, miss or false alarm. After this inspection, it can be easily noticed that there are significant amount of gross human annotation errors, some of which are presented in Figure 3.2.

Table 3.3: Performance comparison of YOLOv3 on Thermal Test Set for different number of epochs and different dataset sizes in terms of Hit Rate (%), False Alarm (per minute) and Training Time (hours)

# Epoch	25			50		
	HR	FA	TT	HR	FA	TT
Full dataset	97.5	2.4	17	97.1	2.2	34
1/2 dataset	95.7	2.4	8.5	97.5	2.4	17
1/4 dataset	93.9	2.7	4.3	95.1	2.1	8.5

Since the dataset is composed of consecutive video frames, and only some of them have significant annotation errors, most of the time, it might be possible to recover those annotation errors by using temporal data and classical methods. Conventional template matching methods, such as cross correlation or phase correlation are quite effective with a high pointing accuracy for the short time periods, i.e. only a few frames. Even if the recent learning-based methods outperform such fundamental methods, in general, it should be reminded that template matching methods have high pointing accuracy performance as long as the pose changes and changes in background are not significant. As Anti-UAV Challenge dataset contains 30fps videos, the pose changes between consecutive frames can be ignored, and the changes in background could be eliminated manually. Moreover, even if the annotations are erroneous, as long as the annotation error is small with respect to the object size, those shifts do not affect template matching methods as the most of the template is still covered by the object of interest.

In order to find the position of an object box (defined on frame k) at frame $k+1$, the neighborhood of annotated object center on frame $k+1$ is searched with cross correlation. Let u_{k+1} be displacement between annotated object center on frame $k+1$ and the matching point of the template defined on frame k. This difference should have three components: annotation error on frame k, w_k; annotation error on frame $k+1$, w_{k+1}; and the error of the matching algorithm v_{k+1}. For the first frame, there are two unknowns (annotation errors in x and y axes) and each new frame introduces four new unknowns (annotation and matching errors on x and y axes), resulting in a underde-

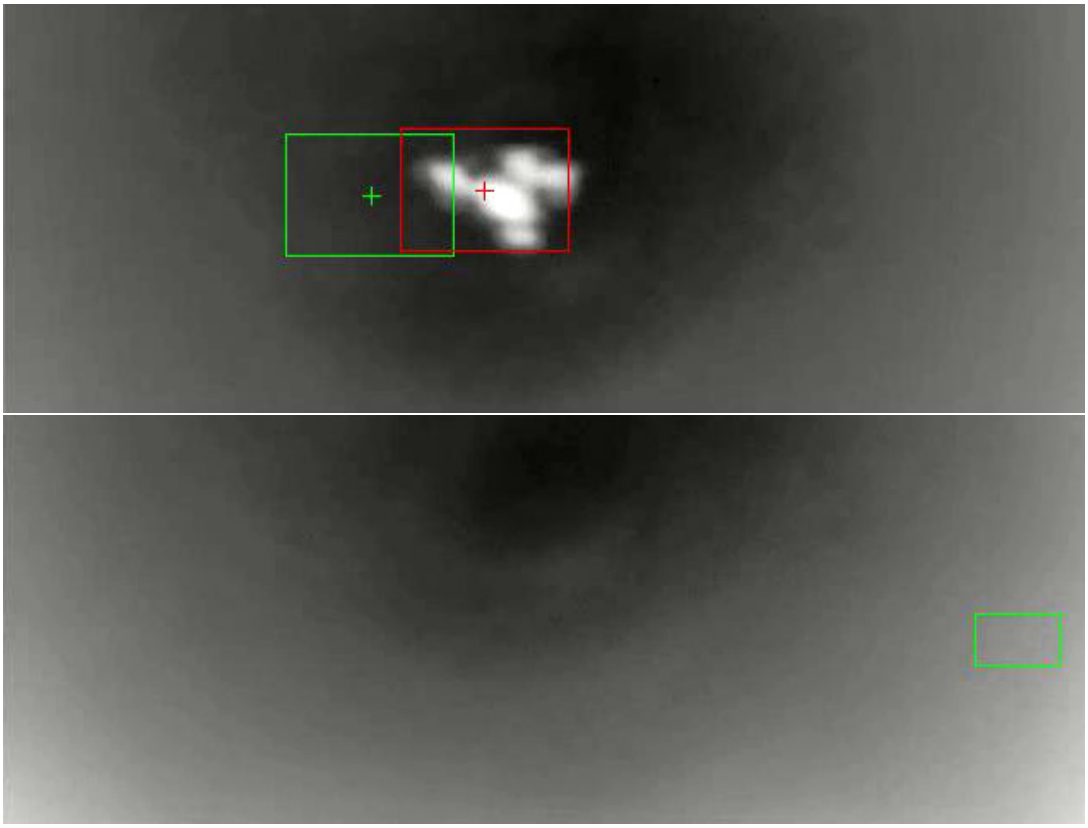


Figure 3.2: Some annotation errors in Anti-UAV dataset. Original annotations are demontrated with green bounding boxes and their corrected versions are shown in red. (a) Taken from 213th frame of IR_20190925_130434_1_4, meanwhile (b) is taken from 620th frame of IR_20190925_130434_1_9.

termined linear system. During the initial attempts, it is observed that minimum-norm solution of such an underdetermined system tends to assign most of the displacements between consecutive frames to matching errors. If the search range is large enough and pose change is not significant, the error of matching algorithm is usually small but it can cause some drift. In order to avoid the drift, the displacements are accumulated, a line is fitted to this cumulative displacement, and the resulting trend is removed from the cumulative.

During the experiments, it is observed that the search range is not large enough for some frames, but increasing search range might result in additional errors; therefore, the annotation correction is performed in two steps for the same search range (20 pixels). After this automatic correction, visual results of original annotations

and automatically corrected annotations compared by a human operator, and better performing one is selected manually. Human operators preferred to use automatically corrected annotations for 66 videos over 100 thermal videos in dataset. For those 66 videos, the first and the second order statistics of difference between original annotations and corrected annotations on x- and y-axes are presented in Table 3.4. When the corrected annotations are investigated, the annotation errors are mostly due to box shifts which are explained in Section 3.6. Therefore, the numerical values in Table 3.4 mainly correspond to parameters of shifted boxes. Corrected annotations and correction algorithm for thermal images of AntiUAV dataset are avaliable at github.com/aybora/CVPR2020-Anti-UAV-OGAM-Correction/

Table 3.4: Mean and standard deviations of difference and normalized difference with respect to width and height of bounding boxes between the center values of given and corrected annotations of 66 videos.

	μ_x	σ_x	μ_y	σ_y
Diff.	0.0970	2.729	0.0102	1.720
Norm. Diff	0.0022	0.0559	0.0015	0.0579

3.6 Experiments

For data annotation, researchers generally either label the objects one-by-one for each image, or they make the labeling between some period of frames (e.g. labeling each 10th frame) and interpolate the bounding box values between the labeled frames by using a reliable tracker, especially for video annotation. Therefore, annotation error sources can be classified into two types: human-based and tracker-based faults. In the next part, both kinds of error sources are examined and their simulation results are presented. At the end, performance of YOLOv3 with such simulated annotation errors is compared with error-free (original annotations) and corrected annotations.

3.6.1 Examining Effects of Various Annotation Errors

Additional boxes: This type of error includes an extra box which does not contain any target. An additional box due to human fault should have a similar appearance with true objects and temporal consistency as a human tends to repeat the fault in consecutive frames. However, additional boxes due to tracker faults is due to either lack of object is visible/invisible decision mechanism, which generates random results without any temporal consistency or an erroneous decision of tracking algorithm which results in additional boxes having a similar appearance to true objects with temporal consistency. Therefore, in this study two types of additional boxes are generated: a) additional boxes at random positions without temporal consistency b) additional boxes initiated on one frame and tracked through consecutive frames to achieve temporal consistency.

In order to insert $P\%$ additional boxes without temporal consistency, $P\%$ for the frames selected randomly and a box having a random position and random size is added. The position of the box is sampled from uniform distribution which covers the whole image, where as the size of the box is selected from a Gaussian distribution, whose mean and variance is set to mean and variance of object size in whole dataset.

Temporally consistent additional boxes should also have a similar appearance to true objects. To insert $P\%$ temporally consistent additional boxes, for every 100 frames, candidate additional boxes are picked at random positions for the first (100 - P) frames. Then, for simulating the visual similarity to true targets, candidate additional box with the highest intensity variance is selected as the true objects have a different appearance from background which results in high intensity variance within the bounding box. In order to simulate temporal consistency, selected additional box on the seed frame is tracked for P frames with correlation tracker.

Missing boxes: A missing box error is simply due to the unavailability of the annotation of a true object. Completely random missing boxes are not expected, either due to human or tracker fault. Labeling operators usually miss the objects due clutter or occlusion which is temporally consistent in general. Trackers have a similar behaviour, when they miss the target on one frame, they tend to miss the object in

consecutive frames. To generate missing boxes with $P\%$, for every 100 frames, labeling of first (100 - P) frames is left as it is and the annotations are removed for the next P frames to achieve temporal consistency. To examine whether this temporal consistency has a significant effect or not, temporally independent missing boxes are also simulated by selecting $P\%$ of the frames independently for each video.

Shifted boxes: A shifted box error is a slightly translated version of the true object box. As human eye cannot detect the object box very precisely in pixel or subpixel level, annotated boxes might be shifted by a few pixels. Trackers have a similar behaviour; even if they mark the true target, resulting bounding box might be shifted by a few pixels. Human errors can be assumed to have a zero mean Gaussian distribution. Tracker errors might be biased due to the drift behaviour of the tracker; however, this effect is neglected in this work. The shifted boxes are generated by adding zero mean Gaussian noise with the specified variance to original boxes without changing the size of the box.

Sample visuals for different types of annotation errors are presented in Figure 3.3.

3.6.2 Performance of YOLOv3 with Simulated Annotation Errors

For all of the experiments presented in this section, the same training and validation sets are utilized. The simulated annotation errors are only applied to the training sets, and YOLOv3 is trained with erroneous annotations for each experiment independently from scratch. For the corrected annotation experiments, the network is trained only with corrected annotations, whereas the results are evaluated both with original and corrected annotations of validation set.

Effect of additional boxes: In the first experiment, 25% additional boxes without temporal consistency are added to the training set. As shown in *Additional Boxes (25%)* column of Table 3.5, when objectness threshold is fixed (0.5), hit rate is slightly increased with respect to training with original annotations, which slightly increases tracking accuracy and modified tracking accuracy as expected, since the number of misses decreases. However, the number of false alarms are increased from 2.4 FA/min to 9.7 FA/min. This result is probably due to a general increase trend in objectness

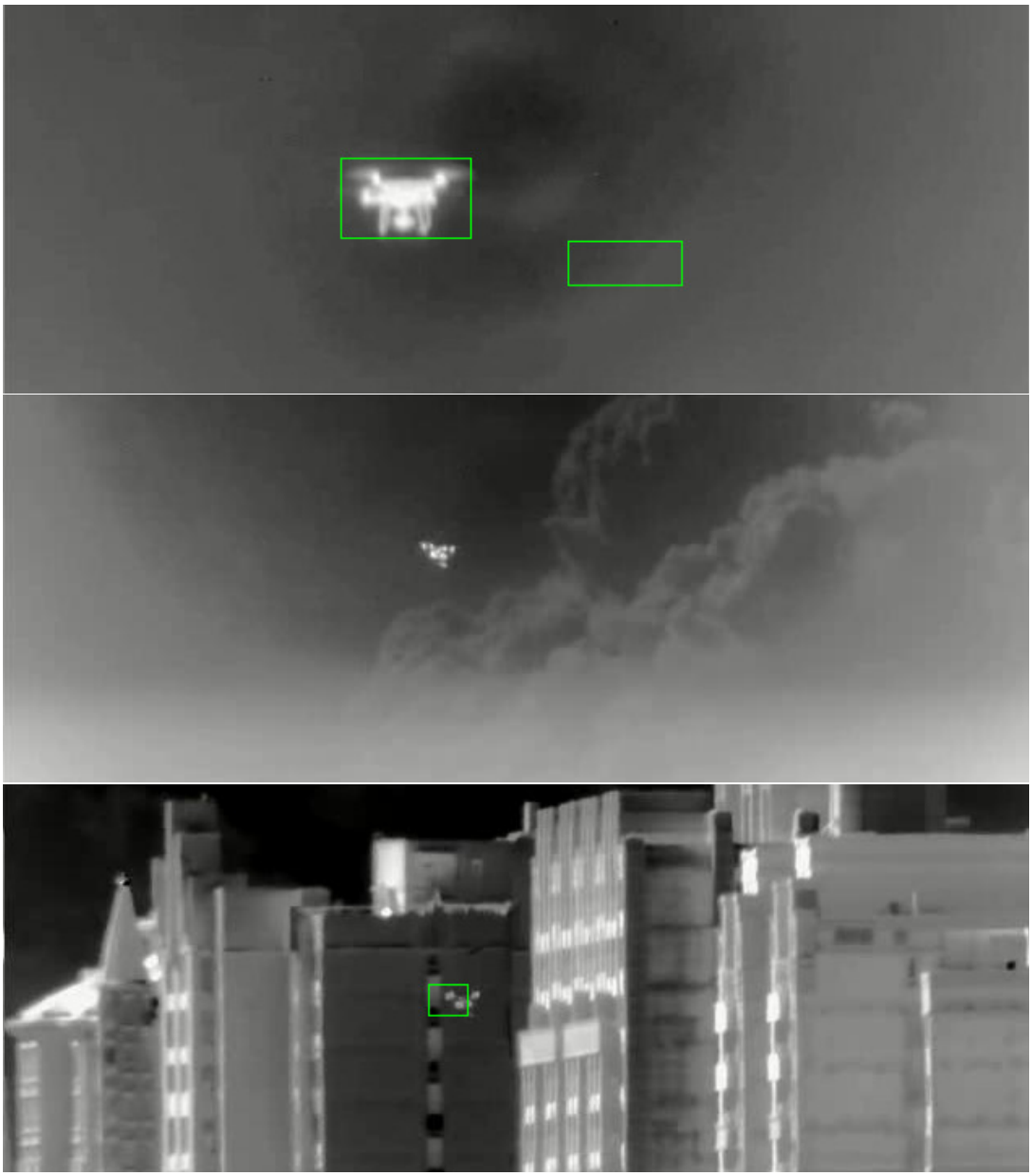


Figure 3.3: Visuals of simulated annotation errors: (a) additional box (b) missing boxes, (c) shifted box.

scores. When objectness threshold is increased to fix the number of false alarms (2.4 FA/min for original annotations), hit rate is dropped by 3.4%, tracking accuracy and modified tracking accuracy are dropped by 2.5% with respect to the original annotations as shown in *Additional Boxes (25%)* column of Table 3.6. It can be concluded that adding completely random boxes of rate 25% is not sufficient to create some pattern that causes the network to learn false positives, it rather forces the network to generate higher objectness scores, which is also supported by the selected objectness threshold of 0.72 to get the same number of false alarms.

When 50% additional boxes without temporal consistency are added to the training set, again the false alarm rate increases significantly as shown in *Additional Boxes (50%)* column of Table 3.5. However, in this case, hit rate and tracking accuracy are decreased. Apart from forcing the network to increase the objectness scores, such a large number of additional boxes seems to detoriate the generalization capacity of the network. To fix the false alarm rate (2.4 FA/min for original annotations) objectness threshold should be increased to 0.68 as shown in Table 3.6. In this case, hit rate is dropped by 5.3%, tracking accuracy and modified tracking accuracy are dropped by 6.6%.

The results for 25% temporally consistent additional boxes of is shown in *Tmp. Cons. Add. Box. (25%)* columns of Tables 3.5 and 3.6. When compared to *Additional Boxes (25%)* column of Table 3.6, the performance is better for temporally consistent additional boxes and objectness threshold to fix the number of false alarms is closer to original threshold. These results indicate that the network finds it easier to reject these consistent additional boxes which is not expected. It can be concluded that the proposed temporally consistent additional box generation method does not work as expected and failed to generate generalizeable additional boxes.

Effect of missing boxes: When missing boxes of %25 without temporal consistency is introduced as the annotation error, hit rate and tracking accuracy decrease as well as the false alarm rate as shown in *Missing Boxes (25%)* column of Table 3.5. When objectness threshold is set to fix the number of false alarms, hit rate is decreased by only 0.3% and tracking accuracy is decreased only by 0.5% as shown in *Missing Boxes (25%)* column of Table 3.6. For the missing boxes without any temporal con-

sistency, the network is still able to generalize the appearance of the object; however, objectness scores tend to decrease.

When missing boxes with temporal consistency is introduced as the annotation error, the detection performance decreases significantly as shown in *Tmp.Cons.Mss.Box. (25%)* and *Tmp.Cons.Mss.Box. (50%)* columns of Tables 3.5 and 3.6. Even for the same rate of missing boxes (25%) performance is degraded significantly. When missing boxes without temporal consistency is applied, the only effect is introducing false negative samples to the training set; however, when missing boxes have temporal consistency, apart from false negatives certain poses of the object are excluded from training set. It can be concluded that, if one has to make a decision between temporally consistent false positives and temporally consistent false negatives in training set; it is better to choose temporally consistent false positives.

Effect of shifted boxes: For shifted boxes two different alternatives are evaluated: standard deviation of Gaussian noise is set to a fixed value (1.5 pixels) to simulate tracker errors and 10% of object size to simulate human faults. As the average size of the objects in Anti-UAV Challenge dataset is 50 pixels in width, the second one corresponds to a standard deviation of 5 pixels. As shown in *Shifted Boxes ($\sigma = 1.5$)* and *Shifted Boxes ($\sigma = 10\%$)* columns of Tables 3.5 and 3.6, shifted boxes decrease the performance significantly. Shifted boxes result in lower objectness scores in general. When the objectness threshold is set to generate 2.4 FA/min, for the noise of 1.5pixels standard deviation the detection outputs has 3.6% lower tracking accuracy and the hit rate is decreased by 5.1%. It should be remembered that hit rate is a thresholded version of pointing accuracy, i.e. low pointing accuracy causes a decrease in IoU and detection result is recorded as a miss due to low IoU.

Effect of combined errors: Finally, 25% temporally consistent missing boxes, 25% additional boxes without temporal consistency and shifted bounding boxes with $\sigma = 10\%$ cases are combined to simulate an extreme annotation error scenario. The results can be seen in *Combined* columns of Tables 3.5 and 3.6. As expected, the performance of YOLOv3 is significantly degraded for such an extreme scenario.

Effect of annotation correction: Up to this point, it is assumed that the published annotations of Anti-UAV Challenge dataset is error-free; however, as stated in Section

3.5, there are significant annotation errors within the dataset. The proposed annotation correction method is applied to whole dataset and for 66 of 100 videos, corrected annotations are preferred by human operators.

When the network is trained with these corrected annotations and the results are evaluated with the original annotations, the performance is increased as shown *Corrected Training* columns in Tables 3.5 and 3.6. For the fixed objectness threshold, false alarm rate is slightly increased as well as the hit rate and the tracking accuracy. To make a fair comparison, objectness score is set to generate same number of false alarms with original annotations case. As in Shifted Boxes ($\sigma = 1.5$) case, objectness threshold is obtained quite close to 0.5, however, this threshold update has no effect on the other metrics. Even evaluated with the original annotations, training with corrected annotations increase the hit rate and the tracking accuracy. This result supports the argument that the corrected annotations are better. Therefore, as a final experiment, performance of the corrected training set is evaluated with the corrected validation set, whose results support the conclusion about annotation errors in dataset. As shown *Corrected Training+Val* column of Table 3.6, when the corrected training set is evaluated with corrected validation set the highest performance is achieved.

Finally, average IoU between corrected and original annotations are compared using Tracking Accuracy metric. TA is found 86.4% in 66 corrected videos. It can be deduced that a perfect tracking algorithm which always gives correct results cannot have a tracking accuracy higher than 86.4% on CVPR-2020 Anti-UAV Challenge test-dev dataset if the performance is measured with the original annotations.

3.7 Conclusion

In this chapter, the performance of a state-of-the-art object detector, YOLOv3, is evaluated for UAV detection problem which can be typically used as a baseline for detection-based tracking methods. For this purpose, YOLOv3 network is trained with Anti-UAV Challenge dataset to detect UAVs, and based on these experimental results, it performs quite well. While the detection performance is yielding relatively high hit rates and small false alarms, the tracking performance is not as good as the detection

Table 3.5: Performance comparison of YOLOv3 on thermal images in terms of False Alarms (FA / minute), Hit Rate (HR %), Tracking Accuracy (TA %) and Modified Tracking Accuracy (MA %) when different noise types are applied with different probabilities and objectness threshold is fixed to 0.5

	FA	HR	TA	MTA
Original Annotations	2.4	97.5	73.6	73.5
Corrected Training	3.0	98.0	74.8	74.7
Corrected Training+Val	2.9	98.8	76.3	76.2
Additional Boxes (25%)	9.7	97.8	74.9	74.3
Additional Boxes (50%)	18.8	95.6	69.4	68.6
Tmp.Cons.Add.Box.(25%)	5.6	96.5	72.7	72.5
Missing Boxes (25%)	0.3	94.1	71.3	71.3
Tmp.Cons.Mss.Box.(25%)	1.0	83.2	62.5	62.4
Tmp.Cons.Mss.Box.(50%)	0.9	34.7	27.2	27.2
Shifted Boxes ($\sigma = 1.5$)	2.2	90.8	68.8	68.8
Shifted Boxes ($\sigma = 10\%$)	1.1	29.9	23.3	23.3
Combined	2.3	71.2	54.2	54.2

performance in terms of tracking accuracy or IoU. It should be noted that the tracking performance of YOLOv3 detector can be improved by utilizing the temporal information, even by employing some classical tracking techniques, such as a conventional Kalman filter that takes measurements from YOLOv3 detector.

The performance of YOLOv3 is also tested on Anti-UAV Challenge dataset for different deliberate erroneous annotations, which is a typical problem in practice. The results are compared against a former effort [71] which was performed on KITTI dataset. Since small targets are already are quite difficult to detect, it is observed that the annotation errors degrade the performance much severely than that of KITTI dataset, especially for the missing boxes scenario. When 50% missing boxes error is introduced, the performance on KITTI dataset drops from 62.9% to 51.8%; meanwhile on Anti-UAV dataset, the performance reduces from 97.5% to 56.0%. Moreover, the changes in objectness scores are quite noticeable when those annotation

Table 3.6: Performance comparison of YOLOv3 on thermal images; Objectness Threshold (TH), Hit Rate (HR %), Tracking Accuracy (TA %) and Modified Tracking Accuracy (MTA %) when different noise types are applied with different parameters for the False Alarm Rate of 2.4 FA/minute

	TH	HR	TA	MTA
Original Annotations	0.50	97.5	73.6	73.5
Corrected Training	0.55	98.0	74.8	74.7
Corrected Training+Val	0.55	98.8	76.3	76.2
Additional Boxes (25%)	0.72	94.1	72.1	72.0
Additional Boxes (50%)	0.68	92.2	67.0	66.9
Tmp.Cons.Add.Box.(25%)	0.58	95.6	72.0	72.0
Missing Boxes (25%)	0.40	97.2	73.2	73.1
Tmp.Cons.Mss.Box.(25%)	0.38	90.8	67.9	67.8
Tmp.Cons.Mss.Box.(50%)	0.30	56.0	43.2	43.1
Shifted Boxes ($\sigma = 1.5$)	0.48	92.4	70.0	69.9
Shifted Boxes ($\sigma = 10\%$)	0.30	87.8	64.8	64.7
Combined	0.49	72.8	55.4	55.3

errors exist. Additional boxes increase the objectness score, while the missing boxes decrease it. Therefore, for a fair comparison, one of the metrics should be fixed and the other ones should be compared.

There are some annotation errors in Anti-UAV Challenge dataset that are observed during the experiments. In order to correct such erroneous annotations, a simple correlation tracker is employed and the provided annotations are updated in such a way that an annotated object bounding box in one frame is searched in the next frame and the center of the annotated object of the next frame is updated with the location giving the highest correlation score. Then, for each video, corrected annotations and the original ones are compared by human operators to select the annotation for that video. After such a correction mechanism, human operators prefer the corrected annotations for 66 videos out of 100 sequences.

Finally, it is observed that the corrected annotations increase both the detection and the tracking performance in terms of hit rate, false alarm rate and tracking accuracy. While the tracking accuracy is calculated 73.6% in original annotations, it increases up to 74.8%, in case of the corrected training set and original validation set being employed. Such a result reveals the success and necessity of the proposed annotation correction method. However, as the validation set also contains erroneous annotations and employed in the performance measurements, the increase in performance is limited. When the corrected training and validation sets are also employed, the tracking accuracy increases to 76.3%. Therefore, to achieve fair results, the annotations of any object detection or tracking set should also be analyzed and corrected, if necessary.

CHAPTER 4

SEMI-AUTOMATIC ANNOTATION FOR SUPERVISED LEARNING

4.1 Introduction

Object detection and tracking algorithms are advanced rapidly during the last decade, especially after realizing the efficiency of Convolutional Neural Networks (CNN) on feature extraction. However, such detection algorithms [36, 37, 2, 5, 3] require a significant amount of annotated data for training which is one of the most important challenges in supervised learning. Similar to object detectors, learning-based trackers [7, 8, 9] also require annotated data. Besides the learning-based methods, annotated data is also necessary to evaluate the performance of tracking and detection algorithms. For all these applications, researchers need human annotators to specify the positions of the objects in each frame.

Object location is defined by a bounding box; hence, each annotation requires two mouse clicks, onto the top-left and bottom-right of the object. For some large public datasets, it is preferred to use crowd-sourcing methods [74] to overcome this enormous effort; however, crowd-sourcing is also a costly method. Moreover, for problem-specific and/or confidential datasets, such as drone detection and tracking on IR videos, crowd-sourcing might not be applicable. For such cases, the researchers would need to annotate their data frame by frame, which requires lots of labor. As a result, automatic and semi-automatic annotation methods are considered as the only solution to this problem and receive significant attention.

For video object detection and tracking cases, object trackers might help to reduce the annotation workload. For example, a popular annotation tool EVA [75] employs pop-

This chapter has been submitted for publication [73].

ular KCF tracker to extrapolate initial bounding boxes in time. However, template matching-based trackers are suffering from the drift problem, which might lead to erroneous annotations. From previous chapter, it is known that the published ground-truth annotations prone to error. For example, Anti-UAV dataset has significant errors which introduces an upper bound of 86.4% for tracking accuracy [60]. Besides using trackers, the research efforts are focused on two different approaches to decrease the annotation workload. The first approach is based on weakly supervised learning. In this approach, researchers give only the image and object class in that image as an input and the network has to find the corresponding bounding boxes [76, 77]. On the other hand, the latter group of techniques exploits active learning. In this approach, the computer actively asks humans to annotate some selected bounding boxes during training [78, 79]. Although these algorithms lead to a decrease in human work, they still require a substantial amount of time to annotate all frames. Recently, some research rfforts are also focused on incremental learning to overcome this problem [80, 81, 82]. In such methods, at each frame, the human annotator only checks whether the inferred bounding boxes are correct or not to decrease the workload.

In this chapter, a novel and yet simple method is proposed to annotate the bounding boxes in a video for single object tracking scenario. By exploiting the temporal information in a video, tracklets, i.e. short tracks of a single object, are automatically formed, which are consecutive and visually similar detection sets of the same object. The proposed user interface displays some visual samples for user verification, especially the challenging ones, for each tracklet at once. Therefore, the human annotator only needs to check the tracklets instead of each detection frame, which decreases workload even further. The general flow of the proposed method is presented in Figure 4.1. It should be noted that the proposed method is an iterative process, during which the object detector is re-trained on each iteration.

4.2 Related Work

As more supervised algorithms are employed in object detection, researchers focused more on decreasing the workload of the annotation process. Papadopoulos et al. [82] proposed an iterative bounding box annotation method with human verification. For

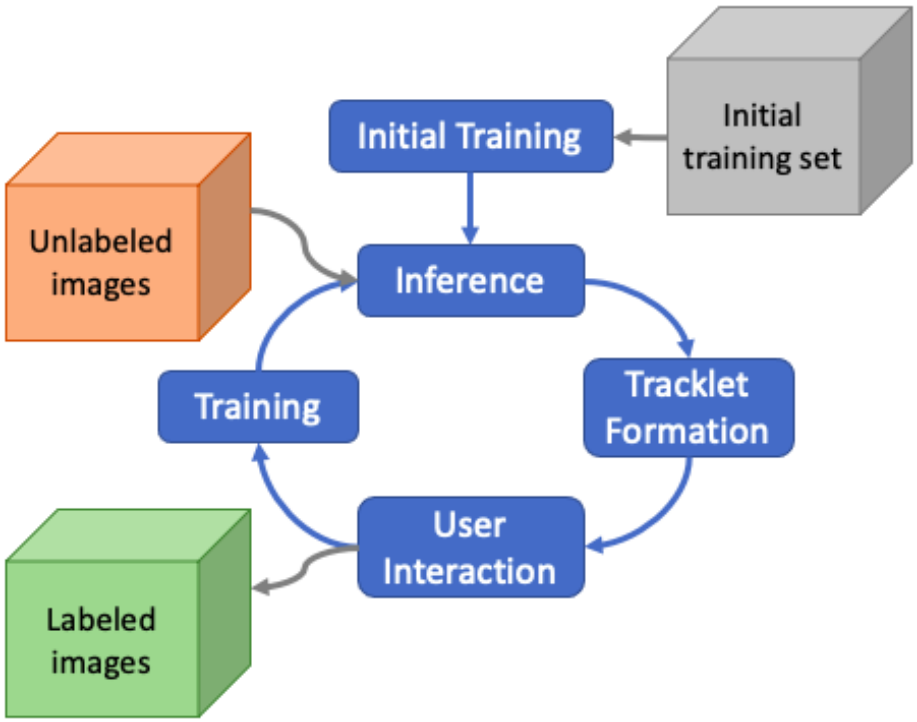


Figure 4.1: Proposed annotation scheme

finding initial bounding boxes an improved multiple instance learning method is proposed. Then, an object detector is trained iteratively, and inferred bounding boxes on each frame are evaluated by a human operator. Even though this approach reduces the annotation workload, the method does not exploit temporal information. Therefore an operator have to evaluate every output of the object detector.

Adhikari et al. also worked on iterative bounding box annotation via two studies [80, 81]. In this approach an off-the-shelf object detector is trained with a small training set which is annotated manually for initial training. Then, the rest of the study follows the iterative approach [82], i.e., remainder of the training set is inferred with this detector, verified by human operators and they are used for next training iteration along with previously annotated data. The set is processed batch by batch and at each iteration a batch is completely annotated. Despite Adhikari et al. [80] focuses on video annotation, they did not leverage temporal information.

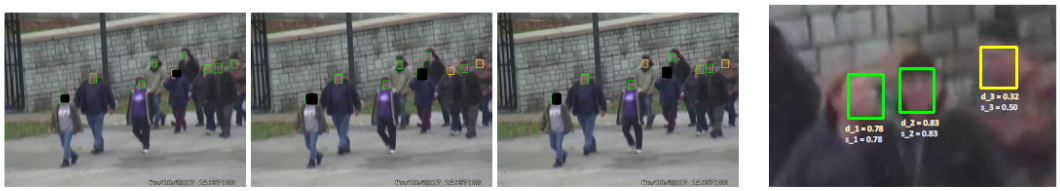
Besides iterative learning approaches, there are also active learning methods that directly need human labeling only for specific frames which are selected automatically depending on the learning performance [83, 84, 85]. Even if these algorithms de-

crease the workload significantly, the operators still have to annotate lots of bounding boxes.

Although the methods mentioned above are quite efficient in reducing annotation workload, the workload can be decreased further for videos by exploiting temporal information. The simplest way to exploit temporal data is to apply tracking as in the well-known annotation tool EVA [75]; however, tracks might drift and this would lead to erroneous annotations as shown in [60].

There are not many fully automatic data annotation methods exploiting temporal data. However, hard example mining techniques in metric learning literature are quite relevant. Jin et al. [86] proposed a hard example mining method by exploiting temporal data. The algorithm matches consecutive detections and classifies temporally consistent detections as *pseudo positive*, while inconsistent ones as *hard negative*. The frames on which detector fails to detect the object but tracker (template matching) succeeds are classified as *hard positive*. The training is performed iteratively by weighting all the hard examples. Although this approach is proposed for hard example mining problem, idea of finding temporally consistent detections via matching consecutive alarms can be employed for bounding box annotation as well.

RoyChowdhury et al. [87] suggested an object detector and tracker combined framework for bounding box annotation, which is also employed to select the hard examples. A baseline object detector, Faster R-CNN [38], is used to generate the pseudo-labels. Next, by exploiting temporal information by using a popular tracker, MD-Net [7], pseudo-labels are refined by forcing them be temporally consistent. For training, weighing the labels inferred from detector (soft-examples) or coming from tracker (hard-examples) also improves the performance. The idea of this approach is illustrated in Figure 4.2. Although this approach is the most related one to the proposed work in this chapter in terms of leveraging temporal data for bounding box annotation, incremental learning may not be possible in this approach, as there is no operator control to eliminate false alarms, which may cause the neural networks to diverge during training.



(a) (b)

Figure 4.2: Simple example for the idea of RoyChowdhury et al. [87]. (a) Pseudo-labels can be seen in either green boxes which represent labels from detector or yellow boxes which are from tracker. (b) d_i: detection confidence scores, s_i: soft scores.

4.3 Proposed Method

The proposed method is an iterative semi-supervised approach for bounding box annotation for single object tracking scenario in a video. At the first step, the selected object detector is trained by a small but fully human annotated set. The training step is followed by the inference step as shown in Figure 4.1, and the detection results are merged to form tracklets. Then, each tracklet is evaluated by an operator, and confirmed tracklets are added to the training set to complete the iteration. The next iteration takes place with this enlarged training set. The procedure is simply selecting the true-positive alarms (detection outputs) in a semi-automatic way by enforcing temporal consistency and getting some human supervision. The flowchart of the proposed method is presented in Figure 4.1 and each step is further explained in the following subsections.

Tracklet Formation: The proposed method employs a tracking-by-detection approach to merge the generated alarms to form tracklets. Multiple hypothesis tracking (MHT) [13] is a popular powerful tool for such problems. The original MHT [13] utilizes only a trajectory model, while a visual model can be incorporated to reduce ambiguity [14]. Multiple hypotheses are utilized to form tracklets. A constant velocity Kalman Filter is employed as the trajectory model (T) as proposed in [14]. The visual model (V) is defined as the raw pixel measurements, and the visual model is completely updated on each measurement update. As opposed to the trajectory model, the visual model is much less ambiguous; therefore, each hypothesis (h) is allowed to take a single measurement (d), which has the highest correlation score, $C(d, h)$, with

the visual model (template), if $C(d, h)$ is higher than the selected threshold or the selected measurement lies in the Kalman Filter's gate; i.e. within uncertainty range of measurement distribution. Each hypothesis is also allowed to continue with no observation update only, if the matching alarm is out of the gate of the Kalman Filter or the correlation score is low. Allowed measurements for different cases are summarized in the Table 4.1 as the primary and alternative ones respectively. For the cases in which more than one measurement is allowed, the original hypothesis continues with the primary measurement and a new hypothesis is generated for the alternative measurement.

Table 4.1: Allowed measurements for different cases

	$C < 0.5$	$0.5 \leq C < 0.8$	$C \geq 0.8$
in gate	$\varnothing$, d	d	d
not in gate	$\varnothing$	$\varnothing$, d	d

Low and moderate visual similarity scores ($C < 0.8$) indicate either fail to detect the object or appearance changes. In case of failure of detection and low visual similarity ($C < 0.5$), a small gate is needed to avoid from generating an unnecessary hypothesis. In case of appearance changes and low visual similarity ($C < 0.5$), gate must be large enough to cover the detection and prevent tracklet to be broken. In moderate visual similarity and failure of detection case, a small gate is required to avoid mismatches. In moderate visual similarity and appearance change case, a small gate help to avoid to generate an unnecessary hypothesis. In short, uncertainty in the position should be reduced as much as possible while keeping the true detections in the gate. To make the constant velocity model fit better to measurement dynamics so to reduce uncertainty, camera movement is taken into account by using a Kalman Filter with an input vector:

$$\begin{aligned} x_{k+1} &= F\, x_k + B\, u_k + w_k \\ y_k &= H\, x_k + v_k \end{aligned} \tag{4.1}$$

where state vector x_k is defined as the position and velocity of the object, u_k is the displacement between frames k and $k+1$, w_k is the process noise, y_k is the observation, and v_k is the measurement noise. Two separate Kalman Filters are used for horizontal and vertical axes. For both axes, following state transition matrix (F), input matrix

(B), measurement matrix (H), process noise covariance (Θ) and measurement noise covariance (R) are used:

$$F = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \Theta = \begin{bmatrix} t^3/3 & t^2/2 \\ t^2/2 & t \end{bmatrix} \sigma_w^2 \qquad (4.2)$$
$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}, R = \sigma_v^2$$

where t is the sampling time between two frames. Velocity per frame is estimated by using $t = 1$, and $\sigma_w^2 = 0.005$ and $\sigma_v^2 = 2.25$ are selected as design parameters through experimentation.

It is assumed that the motion between two frames is translational which corresponds to pan and tilt of the camera. In order to find the camera movement, phase correlation is applied.

The characteristics of the detector is also important for reducing annotation workload. Failing to detect the object would result in break of the tracklet. For avoiding false-negatives, quite low confidence threshold (θ_L) is applied on detector's objectness score (P_d) for generating alarms. However, to reduce the number of hypotheses, a new hypothesis is initiated only for the detections with high confidence scores (θ_H).

For keeping the number of hypotheses under control, a strategy is also required to delete and merge the hypotheses. The significance of a hypothesis is indicated with its average objectness score over the last few frames. The hypotheses, whose average objectness score (P_{avg}) is below a certain threshold, (θ_{avg}) are deleted. When a tracklet is updated with no measurement, the objectness score at that frame is set to zero. Setting objectness score for no measurement update to zero helps to delete hypotheses which do not get measurement update for a few frames as their average objectness score decrease on each no measurement update.

Since the visual model determines the measurement that will be used for the update, two different hypotheses with the same visual model will be updated with the same measurements and will converge to identical tracklets at the end. Therefore, whenever two different hypotheses take the same measurement, the hypothesis with the highest average objectness score is kept and the rest is deleted. The algorithmic flow of the tracklet formation approach is presented in Algorithm 1. It should be noted that the

performance of the employed tracklet formation method mainly affects the workload of human operator, not the annotation accuracy.

Algorithm 1 Tracklet Formation

1. Select the alarms d with $P_d > \theta_L$
2. Perform measurement update:

for each hypothesis h **do**

Select the matching alarm, $d^* = \arg\max_d C(d, h)$

Update V_h and T_h with the primary measurement

Generate a new hypothesis with alternative measurement, if any

end for

3. Generate a new hypothesis for each alarm with $P_d > \theta_H$ and not assigned to a hypothesis
4. Merge the hypotheses matched with the same alarm
5. Remove hypotheses with $P_{avg} < \theta_{avg}$

User Interaction: Once the tracklets are formed, human operator is asked to evaluate tracklets. A sample tracklet evaluation screen is shown in Fig. 4.3. For each tracklet, N frame samples which have equal temporal spacing along the tracklet are selected. In between those samples, instances having (1) lowest objectness score, (2) lowest correlation score, (3) lowest average objectness score, (4) highest distance to estimated position, and (5) highest temporal distance to other displayed instances are presented to the operator for evaluation. The operator is asked to accept or reject each sample in temporal order. Once a sample is accepted, other samples up to the next rejection are accepted and vice versa. With this approach, for the best case, each tracklet can be evaluated with two clicks (accept or reject) on the first and last samples.

Incremental Learning: As the iterations take place, only the frames which contains a measurement from a user confirmed tracklets are added to the set of labeled frames and the detector is trained with this enlarged set. When combined with the proposed user interaction strategy, this approach introduces a limitation for annotating frames containing multiple objects. As user interaction step is not designed to handle missing detections and any frame containing a confirmed object is added to the training set, if there are multiple objects in a frame, objects which are failed to detect will not be

Figure 4.3: Sample tracklet evaluation screen for $N = 7$

annotated. However, as the focus of this chapter is on bounding box annotation for single object tracking scenario, the solution of this limitation is left as a future work.

Incremental Annotation: As the iterations take place, there is no need to re-evaluate the tracklets overlapping with previously accepted/rejected ones. Overlapping instances of those tracklets with the previously accepted/rejected ones are ignored during operator evaluation. As a special case, for single object detection (or tracking) if a frame is already annotated, there is no need to re-prompt that frame for operator evaluation. Since the experiments are performed on AUTH `uav_detection` subset [88, 89], which is a single object detection case, on each iteration only the frames without annotation are evaluated.

4.4 Experiments

Throughout the experiments, YOLOv3 [3] is used as the baseline detector. To demonstrate the effectiveness of the proposed method, the `uav_detection` subset is re-annotated and the `uav_detection_2` subset is annotated, which are the subsets of AUTH Multidrone set [88, 89]. Moreover, the operator workload is also compared against alternative methods.

The results are given tables in terms of recall percent (Rec), number of False Alarms Before Click (FA-B), number of False Alarms After Click (FA-A), number of Clicks, number of Annotated Frames (Ann #), percentage of Annotated Frames over All Frames with a Drone (Ann %). (Rec, FA-B, FA-A are presented if the ground-truth is available).

Enlarging the training set is a quite common necessity for the supervised methods. As the learning-based systems are employed more often, extra data is obtained each day which can be utilized for training to enhance the performance. On the other hand, for a fixed set one might prefer to annotate a part of the available data and annotate the rest by the help of the trained detector. To simulate these two cases, two videos (videos DSC_6303 and DSC_6304) having a total number of 4803 frames from AUTH `uav_detection` subset are selected as the initial training set, and the rest of the subset is re-annotated with the proposed method. User clicks are simulated using the original annotations such that if the intersection over union of the proposed and the original annotation is larger than θ_{IoU} (†) the user is assumed click accept and reject otherwise. As shown in Table 4.2, in 7 iterations, 35960 frames are annotated with only 2712 clicks. As each bounding box is defined by two clicks, the annotation effort for the proposed method corresponds to 3.7% of unaided case. In order to annotate the whole set with this approach, 4803 initial annotations, 2712 tracklet evaluation clicks, and 1005 manual annotations for the remaining boxes are required, which corresponds to 82.84% workload reduction.

Initiating training set might also be performed by annotating some sample frames from each video manually rather than completely annotating some videos manually and leaving the rest. To simulate that case, training set is initiated by selecting the original annotations of 1 frame for every 100 frames, by uniform temporal sampling. Then, rest of the training set (99%) is re-annotated by the proposed method. As shown in Table 4.3, in 5 iterations 40469 frames can be annotated with only 1518 clicks which corresponds to an annotation effort of 1.9% of unaided case. Accounting initial annotations (421 boxes) and manual annotation of remaining frames after iterations, workload reduction is 94.14% for this case. When compared to the first experiment which has 4803 annotations initially, selecting frames from a temporally uniform distribution for initial annotation, helps to converge faster with much less effort.

FA-A columns of Tables 4.2 and 4.3 are not false alarms for real; they are either incorrect ground-truth, or some blurry frames which the operator decided not to annotate. Typical visual examples are given in Figure 4.4. Such errors are known as missing and shifted box errors which reduce the performance of the detector and would mislead the performance measurement [60].

Table 4.2: Results of enlarging training set experiment

Iter	Rec	FA-B	FA-A	Click	Ann #	Ann %
1	66.81	1809	6	757	26162	70.77
2	82.11	573	6	737	31694	85.74
3	85.93	437	6	533	33074	89.47
4	88.74	139	6	410	34136	92.34
5	96.23	56	6	99	35666	96.48
6	96.92	53	6	90	35899	97.11
7	97.09	103	6	86	35960	97.28
Total Clicks: 2712			Annotated: 35960 (97.28 %)			

Table 4.3: Results of initiating training set experiment

Iter	Rec	FA-B	FA-A	Click	Ann #	Ann %
1	88.76	1795	47	671	37823	90.58
2	92.40	798	47	509	39345	94.22
3	96.03	178	47	153	40235	96.35
4	96.47	114	47	93	40412	96.78
5	96.61	277	47	92	40469	96.91
Total Clicks: 1518			Annotated: 40469 (96.91 %)			

It should be noted that the last 3 rows of Tables 4.2 and 4.3 are evaluated with $\theta_{IoU} = 0.2$, while the rest with $\theta_{IoU} = 0.5$ for simulating user action, as some original annotations are slightly shifted (shifted box error).

Annotating AUTH UAV_Detection_2 Subset: Through the first two experiments,

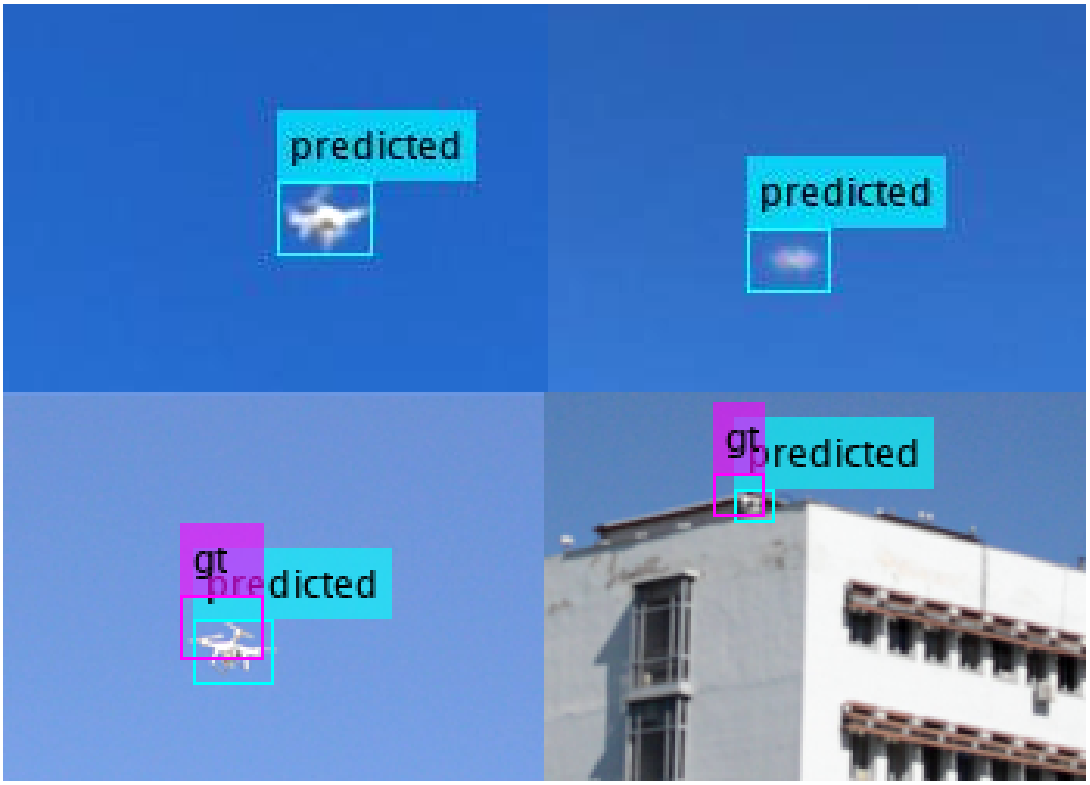


Figure 4.4: At the top, examples of inferred boxes (predicted) which are tabulated as false alarms in Table 4.2 and 4.3; at the bottom examples for the annotations which should be evaluated with $\theta_{IoU} = 0.2$ instead of $\theta_{IoU} = 0.5$ due to the shown incorrect original ground-truth (gt). Displayed frames are (a) 8445 of 00001, (b) 7912 of 00002, (c) 3682 of DSC_6299, (d) 1222 of DSC_6304.

it is noticed that the original annotations might be incorrect, especially might contain box shift errors. As the original annotations are slightly shifted, the proposed annotations have low IoU with them, which results in more clicks than needed. To demonstrate the effectiveness of the proposed method in a more fair case, AUTH `uav_detection_2` subset is also annotated. For this experiment, `uav_detection` subset is used for initial training. As shown in Table 4.4, in 4 iterations 22262 frames can be annotated with total number of 810 clicks, which corresponds to 96.25% workload reduction with respect to unaided case. This performance enhancement might be due to box shift error in original annotations as mentioned above, but it might be due to the larger initial training set as well. However, as seen in Table 4.4 at the first iteration only 43% of the frames can be annotated, while in Tables 4.2 and 4.3 at the first

iteration semi-automatic annotation rates are 70% and 90% respectively, supporting the performance enhancement due to getting rid of box shift errors.

Table 4.4: Annotation Effort for **UAV_Detection_2 Subset**

Iter	Click	Annotation #	Annotation %
1	280	9765	43.00
2	231	18771	82.66
3	195	21828	96.12
4	104	22262	98.03
Total Clicks: 810		Annotation (%): 98.03	

Comparison with Alternative Methods: RoyChowdhury et al. [87] increased their face detection performance from a baseline of 15.66 AP to 20.65 on WIDER-Face unlabeled CS6 dataset [90] by automatic annotation. For pedestrian detection, they use a part of BDD-100K dataset [91] as initial training set and automatically annotated the rest with a fully automatic method which increased their AP from 15.21 to 28.43 in automatically annotated set. AP for drone detection is increased from 69.73 to 80.73 for enlarging training set scenario.

Workload reduction can be calculated as the decrease of the number of clicks needed to annotate the dataset completely. Comparison with different approaches on literature is presented in Table 4.5. As it can be observed in the Table, the proposed approach is able to reduce workload up to 15% more by exploiting temporal data and using tracklet level user interaction.

4.5 Conclusions

The proposed method differs from the previous literature by the following points: a) a low confidence threshold is applied for detection to increase recall, but a high confidence threshold for track initiation, and MHT to eliminate false-positives, b) User verification is performed at tracklet-level rather than frame-by-frame, c) The whole set is tried to be annotated on each iteration rather than annotating batch-by-

Table 4.5: Comparison of proposed annotation method with alternative approaches in terms of workload reduction

Method	Reduction (%)
Adhikari et al. [80] (Best)	80.56
Adhikari et al. [81] (Best)	81.26
Enlarging Training Set (Ours)	82.84
Initiating Training Set (Ours)	94.14
Annotating New Set (Ours)	96.25

batch, which helps to converge more rapidly.

The proposed semi-automatic method for bounding box annotation for single object tracking scenario is shown to reduce the human workload by 82-96% for two different enlarging training set scenarios. For initiating training set scenario, if some videos are fully annotated and then the proposed method is applied, workload reduction is 82%. However, if the initial labeling effort is spent on temporally uniformly sampled frames, workload reduction increases up to 94%. It is obvious that selecting more temporally spaced frames for initial manual annotation helps to increase the variation of the initial set and having an initial set with more variation helps to converge faster with less workload. The proposed method reduces the workload more than the other semi-automatic methods in literature.

The proposed method has some limitations. First of all, even if the incremental learning approach is also applicable for bounding box annotation on independent still images as demonstrated in [81], the proposed method is applicable only for videos. Moreover, human interaction and incremental learning approaches utilized by the proposed method might fail for the videos containing multiple objects. For such scenarios, a more suitable human interaction step should be designed.

CHAPTER 5

MOVING OBJECT DETECTION VIA TEMPORAL INFORMATION AT DECISION LEVEL

5.1 Introduction

Still image object detectors, which are examined in Chapter 2, process each frame independently. In other words, they do not consider the consecutive detection results during inference. Therefore, any image object detector might detect an object in one frame with high objectness score, then miss the same object in another frame. Similarly, due to the background clutter, in some frames, some background images may look like an object and it causes the detector to give false alarms.

Due to the above-mentioned reasons, exploitation of temporal information might increase the accuracy of object detectors. Temporal information can either be exploited by post processing algorithm at decision level, or during feature extraction. In this chapter, utilization of temporal information at decision level will be considered. The methods which utilize the temporal information at feature extraction step will be investigated in Chapter 6.

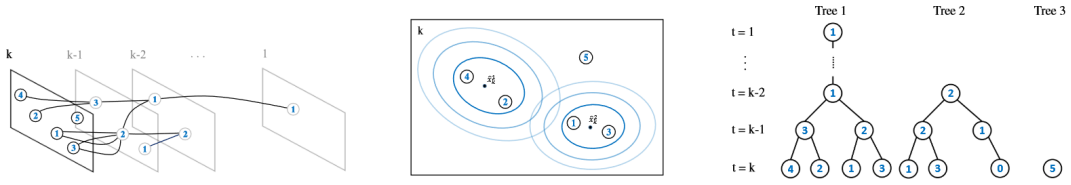
5.2 Related Work

A widely applicable filter which is used on different areas of signal processing, such as denoising or estimation of partially observed variables, is proposed more than 70 years ago by Kalman [12]. With the basis of a hidden Markov model, the variables are first observed fully or partially with a measurement noise. Using these measurements, variable of interest is estimated in an optimal manner. This estimation is expected to

have a higher accuracy than the one which is based on a single measurement.

Using the alarms generated by an object detector to create tracklets, or namely tracking-by-detection, is one of the main approaches for using temporal information at the decision level. Reid [13] defines multiple hypothesis tracking (MHT) for the first time with a trajectory model. The algorithm simply merges similar target candidates to one hypothesis and eliminate the candidates that are not temporally consistent.

In an improved MHT-based approach, Kim et. al. [14] added an appearance model to the classical MHT, which increases the performance even further. In [14], constant velocity Kalman Filter is used as the trajectory model. Revisited MHT algorithm gets competitive results on MOT challenge. Main ideas of MHT is illustrated in Figure 5.1.



(a) Tracks in Video Frames (b) Gating (c) Track Trees

Figure 5.1: MHT. (a) (A subset of) tracklets (hypotheses) at time k. (b) Sample gating areas for two tracklets with different thresholds. (c) Respective track trees whose nodes are related with an examination in (a). Taken from [14].

Bergmann et. al. [15] introduce Tracktor, as another tracking-by-detection based approach, which utilizes the underlying Markov process. The algorithm uses Faster R-CNN outputs to create a motion model and re-identification. The approach is useful for MOT challenge which is focused on human detection and tracking. Tracking part of the algorithm works independent of the dataset. Hence, Tracktor does not need any additional data for training.

5.3 Proposed Method

5.3.1 M-out-of-N Algorithm

If small false alarm rate is desired, a simple, but efficient, approach, M-out-of-N algorithm (abbreviated as M/N algorithm in the rest of the chapter) can be employed. The algorithm simply checks the last N frames, and if there are M detections out of these N frames, then the detection output is accepted as valid. Otherwise, the detection output is discarded. As M/N algorithm focuses on a single object, on each frame only the detection output with highest objectness score is considered. The flowchart of M/N is given in Figure 5.2.

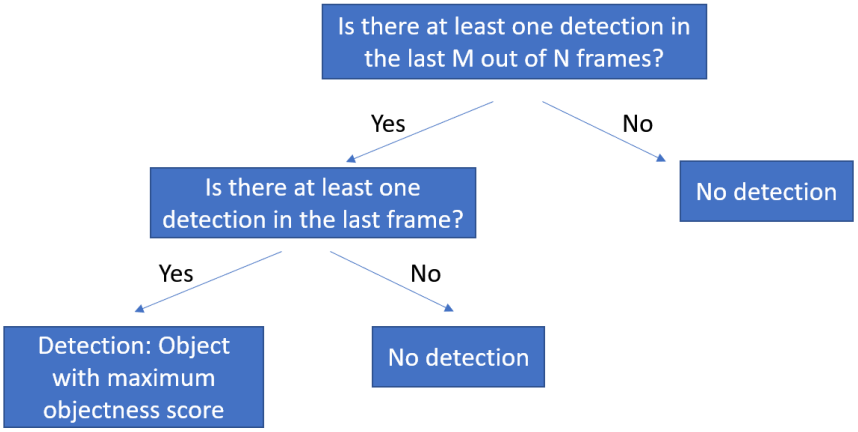


Figure 5.2: M/N (M-out-of-N) Flowchart.

5.3.2 M-out-of-N with Visual Similarity

Since M/N algorithm considers a single detection output on each frame, the output with maximum objectness score in our case, the algorithm would lost the track even though the target object is detected, if the target object does not have the highest objectness score. In order to overcome this problem, M/N algorithm is modified to take visual similarity into account.

After first detection, the modified M/N algorithm checks every detection output, whether they are visually similar to the object of interest or not. This similarity check

can be achieved by template matching, using normalized cross correlation (NCC) algorithm.

The last detection output accepted by M/N algorithm is flattened, its mean is subtracted and normalized to achieve an 1D image vector, X, whose norm is one. Every detection output in the current frame, Y_i, is also flattened, its mean is subtracted and the vector is normalized. Then the NCC is obtained by getting the inner product of these two vectors:

$$CC_i = \langle X, Y_i \rangle = X^T Y_i \tag{5.1}$$

where $\langle \cdot, \cdot \rangle$ indicates the dot product. After finding all the cross correlation scores, the rest of the algorithm works as in Figure 5.3.

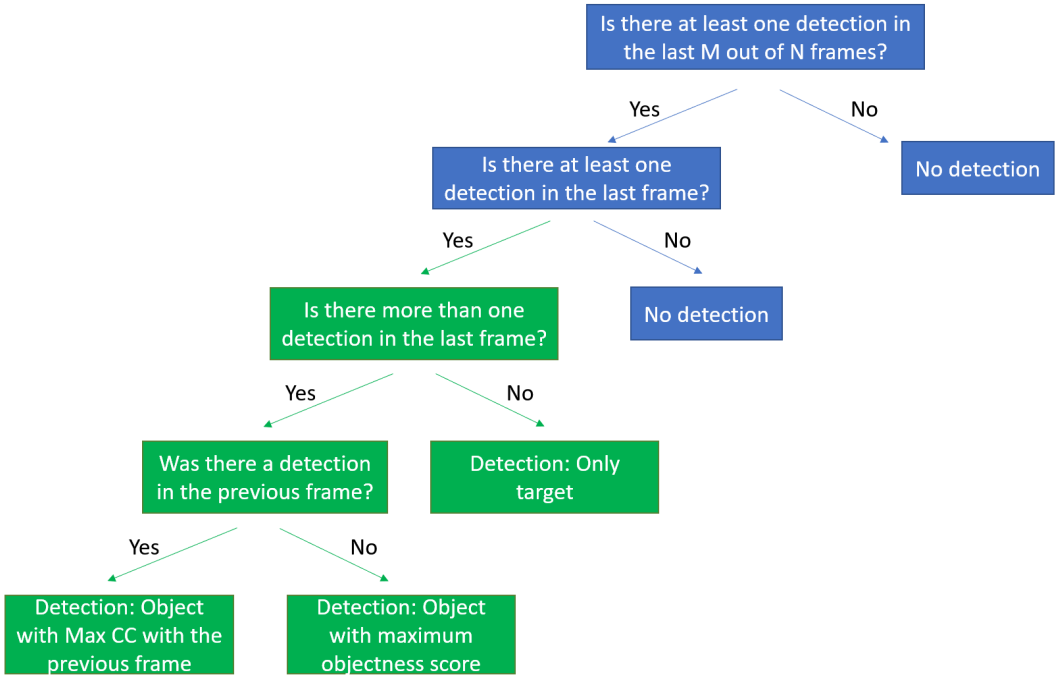


Figure 5.3: M/N with Cross Correlation Flowchart.

5.4 Experiments

5.4.1 Experimental Setup

5.4.1.1 Dataset

M/N algorithm is tested on test-dev folder of ICCV 2021 - 2nd Anti-UAV Challenge Workshop dataset at [58] as it is explained in Chapter 2.

Dataset consists of 140 thermal infrared videos which include UAVs of various sizes. There are different types of occurrences of drones, such as in front of a building, clear sky, cloudy sky, mountain, sea etc. This type of variety makes it possible to compare the object detection algorithms in different cases.

Throughout the experiments, the same randomly selected 30% of the videos, as in Chapter 2, are used as the validation set and the rest is used as the training set.

5.4.1.2 Metrics

For the comparison of the algorithms, the same metrics in Chapter 2, *Hit Rate* and *False Alarm Rate*, are used. Before these metrics are used, *IoU, True Positives, False Positives* should also be revisited.

$$IoU = \frac{\textit{Area of overlap between detection and ground truth}}{\textit{Area of union of detection and ground truth}} \tag{5.2}$$

$$\textit{True Positive} : \text{Detections whose } IoU > 0.5 \text{ with ground truth.} \tag{5.3}$$

$$\textit{False Positive} : \text{Detections whose } IoU = 0 \text{ with ground truth.} \tag{5.4}$$

$$\textit{Hit Rate} = \frac{\textit{Number of true positives}}{\textit{Number of samples with an object present}} \tag{5.5}$$

$$\textit{False Alarm Rate} = \frac{\textit{Number of false positives}}{\textit{Length of dataset in terms of minutes}} \tag{5.6}$$

5.4.2 Experimental Results

YOLOv5 is used as a reference object detector for the comparison. When the objectness threshold is fixed to 0.5 and $M/N = 22/25$, performance comparison on validation set in terms of *Hit Rate* and *False Alarm Rate* for M/N and M/N with Visual Similarity is presented in Table 5.1.

Table 5.1: Performance comparison of M/N and M/N with Visual Similarity using a fixed threshold (0.5) and $M/N = 22/25$

	HR	FA
YOLOv5	83.34	30.99
M/N	77.74	5.19
M/N with Visual Similarity	77.72	2.61

According to the results, both M/N algorithms are useful to decrease the false alarm rates sharply with some sacrifice of the hit rate. On the other hand, M/N with Visual Similarity is a more powerful algorithm than the original (vanilla) M/N algorithm in terms of rejecting false alarms while preserving the same hit rate. This is an expected result, since it considers both visual information in addition to the existence of the target in time.

5.5 Conclusion

In this chapter, firstly some elementary applications are examined to observe the effect of using temporal information on object detection. Then, more advanced post processing algorithms are used to exploit temporal data. It can be deduced that using temporal data at decision level increases detection performance in terms of getting consistent outputs with less false alarms.

Visual similarity is a powerful metric and when employed by M/N algorithm, it reduces the false alarm rate while maintaining the same hit rate of the original M/N algorithm. This also indicates that using visual information might be useful in spatio-temporal feature extraction.

While the decision level algorithms are powerful tools to use temporal information for decreasing false alarm rate, they do not help the object detector to increase hit rate performance with additional detections. For that reason, the temporal information should be used before the decision layer. Therefore, the research efforts in this chapter will not be carried further with more sophisticated methods. Instead, in the next chapter, neural networks will be employed to extract some spatio-temporal features.

CHAPTER 6

MOVING OBJECT DETECTION VIA TEMPORAL INFORMATION AT FEATURE LEVEL

6.1 Introduction

Based on the simulation results of Chapter 5, using temporal information at feature level might be a more promising way to exploit temporal information. In other words, some neural networks can be trained to learn spatio-temporal features. This approach might increase the precision and recall accuracy between consecutive frames. It should be remembered that the main aim is to improve object detection with temporal information; explicit tracking of moving objects is not considered in this thesis.

As discussed in Chapter 2, there are two main approaches to do that in supervised learning. The first one is 3D CNN and another one is recurrent networks (LSTM, Conv-LSTM etc.). For the extraction of the spatio-temporal features one can modify the object detector architecture with the layers that can exploit such information.

In the rest of this chapter, after brief literature review on how the temporal information is used on feature level for video object detection, two novel methods will be proposed that will be compared by experimental results.

6.2 Related Works

Deng et. al. [18] uses FPN [41] for feature extraction. Then, the algorithm registers the extracted features spatially with optical flow estimate. Finally, for a frame t, registered features between the frames $t-1$ and $t+1$ are averaged and the decision

layer uses this averaged feature vector. This idea is quite complicated to apply to a single shot object detector with the real time inference expectancy.

Kang et. al. proposes two algorithms [16, 17] to create tubelet proposals by using information from R-CNN [36] and Faster R-CNN [38], respectively. To achieve high precision and recall accuracy using temporal information, they used 1D Temporal Convolutional Network (TCN) for the first one and LSTM for the latter one, respectively. Both of these algorithms are computationally inefficient; they process one frame in 150 seconds and 0.5 seconds, respectively. Therefore, the proposed idea is not applicable to work with single shot object detectors, whose main target is to work in real time.

Feichtenhofer et. al. [19] proposes an end-to-end trainable video object detector by adding a CNN-based tracking algorithm at the end of R-FCN [39]. In contrast to the algorithms discussed in Chapter 5, the correlation feature extraction and tracklet generation processes are also obtained by Convolutional layers. The overall architecture of the algorithm is presented in Figure 6.1. The idea is a simple and an efficient approach for the exploitation of temporal information at feature level; hence, it seems to be applicable for the single shot object detectors.

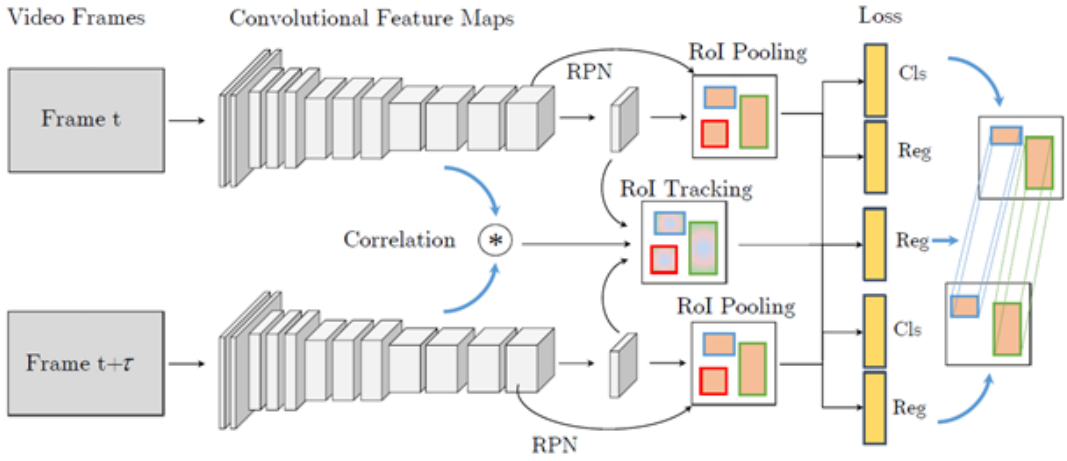


Figure 6.1: Architecture of D&T (Detection and Tracking). Taken from [19].

Bochkovskiy [92] implemented YOLO-LSTM on top of YOLOv3 [3] by just modifying the convolutional block at the detection part of the network. The first convolutional layer after the feature extraction block is changed with Conv-LSTM [35]

block. According to Bochkovskiy, this approach improves the video object detection performance by 4-9% AP, solving the blinking issue on consecutive frames.

6.3 Proposed Methods

6.3.1 YOLOv5-Temporal

YOLOv5-Temporal algorithm is proposed to exploit temporal information by using 3D Convolutional Neural Networks. In the proposed scheme, feature extraction part of the YOLO remains the same. For the regression, the last layers are modified to extract spatio-temporal features for small, medium and large targets. The detailed structure is presented in Figure 6.2 and each block is explained in following paragraphs.

Concatenate: The last layers before decision, which are responsible for the detection of small, medium and large targets, are concatenated in time, with frames of k, $k-\tau$ and $k+\tau$. The concatenated data is transferred to 3D Convolutional Block, for spatio-temporal feature extraction.

3D Conv Block: This block consists of a couple of 3D convolutional layers, which has an ability to include neighbor temporal data into feature extraction. At the end, by using stride = 3, this data will be compatible with YOLOv5 1x1 Conv Layer for detection result.

1x1 Conv and Detection: Without any difference from YOLOv5, spatio temporal features are fed into a 1x1 Convolutional Layer for regression. The output of this layer gives the detection output. For 1 class case, this output has a length of 18, given that there is 3 anchors, 4 for bounding box 1 for objectness score and 1 for class score.

6.3.2 YOLOv5-LSTM

YOLOv5-LSTM algorithm is proposed to exploit temporal information using Convolutional LSTM [35]. In the proposed scheme, the feature extraction part of the YOLO remains the same. During regression, the last layers are modified to extract

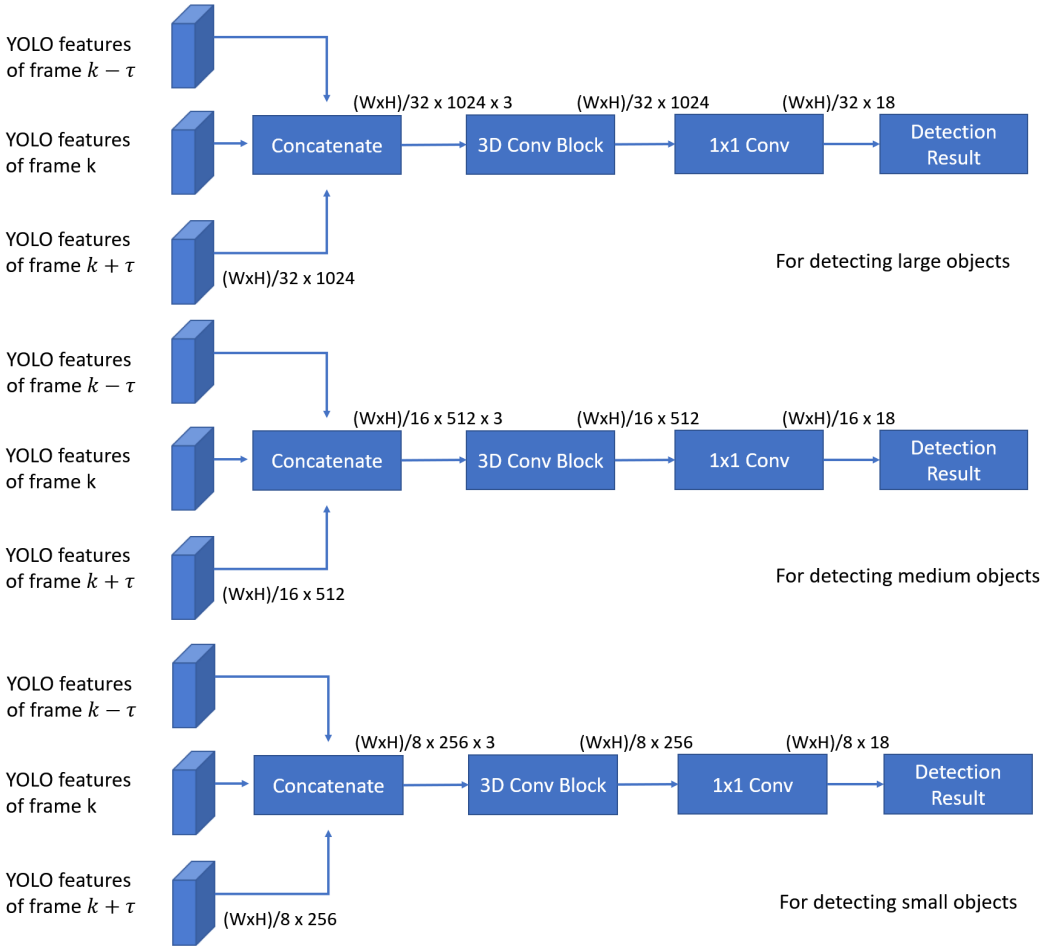


Figure 6.2: Proposed YOLOv5-Temporal Structure

spatio-temporal features for small, medium and large targets, just like in YOLOv5 Temporal. Detailed structure is presented in Figure 6.3 and each block is explained in the following paragraphs.

Concatenate: Just like YOLOv5 Temporal, the features at the last layers before decision, which are responsible for the detection of small, medium and large targets, are concatenated in time, of frames of k, $k-\tau$ and $k+\tau$. The concatenated data is transferred to Convolutional LSTM Block, for the extraction of spatio-temporal information.

Conv-LSTM Block: This block consists of a couple of Convolutional LSTM layers and 3D convolutional layers, which has an ability to include neighbor temporal data into feature extraction and create a memory in the network. At the end, with using

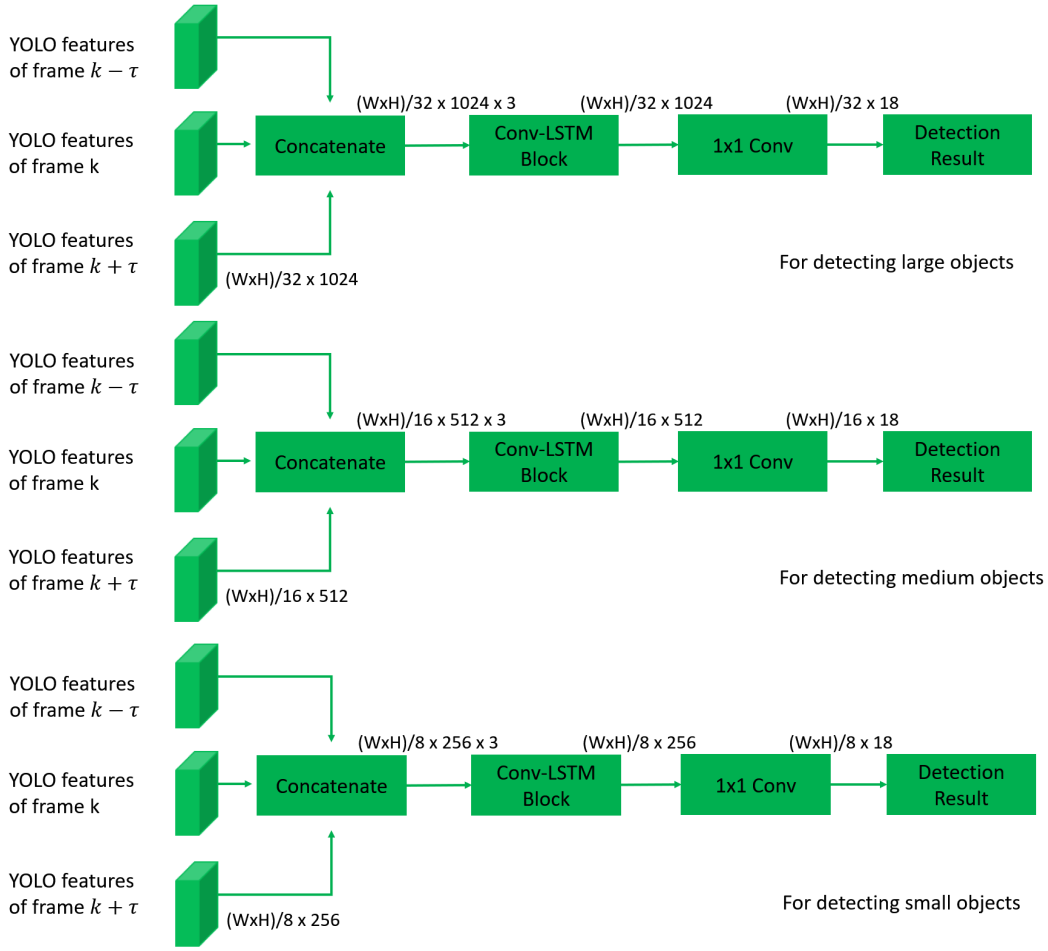


Figure 6.3: Proposed YOLOv5-LSTM Structure

stride = 3, this data will be compatible with YOLOv5 1x1 Conv Layer for detection result.

1x1 Conv and Detection: Without any difference from YOLOv5, spatio temporal features are fed into a 1x1 Convolutional Layer for regression. This layer gives the detection output. For 1 class case, this output has a length of 18, given that there is 3 anchors, 4 for bounding box 1 for objectness score and 1 for class score.

6.4 Experiments

6.4.1 Experimental Setup

6.4.1.1 Dataset and Metrics

In this chapter, like Chapters 2 and 5, test-dev folder of ICCV 2021 - 2nd Anti-UAV Challenge Workshop dataset [58] is used for a fair comparison.

Dataset consists of 140 thermal infrared videos which include UAVs of various sizes. There are different types of occurences of drones such as in front of a building, clear sky, cloudy sky, mountain, sea etc. This type of variety makes it possible to compare the object detection algorithms in different cases.

In this work, the same randomly selected 30% of the videos as in Chapters 2 and 5 are used the validation set and the rest is used as the training set.

As a small but an important note, only the frames where the object is visible in all three consecutive frames (in frame k, frame $k - \tau$ and frame $k + \tau$) are used in training set and test set for the following experiments. Therefore, the performance of YOLOv5 will differ from the Chapters 2 and 5.

For the comparison of the algorithms, the same metrics as in Chapters 2 and 5, *Hit Rate* and *False Alarm Rate*, are used. Before these metrics are used,

6.4.2 Experimental Results

YOLOv5 is used as a reference object detector, since YOLOv5-Temporal and YOLOv5-LSTM are modified versions of this algorithm. When the objectness threshold is fixed to 0.5, the performance comparison on validation set in terms of *Hit Rate* and *False Alarm Rate* for YOLOv5, YOLOv5-Temporal and YOLOv5-LSTM are presented in Table 6.1.

According to Table 6.1, algorithms cannot be compared with each other, since when the hit rate is higher, false alarm rate is also higher which only indicates that some of

Table 6.1: Performance comparison of YOLOv5, YOLOv5-Temporal and YOLOv5-LSTM with a fixed threshold (0.5)

	HR	FA
YOLOv5	83.33	28.42
YOLOv5-Temporal	83.28	26.55
YOLOv5-LSTM	83.82	71.43

these algorithms tend to give more detection outputs.

In order to have a fair comparison, false alarm rates are fixed to reference false alarm rates (when the threshold is 0.5) of each of the algorithms by changing the objectness threshold. For the fixed false alarm rates the results are presented in Table 6.2.

Table 6.2: Performance comparison of YOLOv5, YOLOv5-Temporal and YOLOv5-LSTM with fixed FAs

	FA = 28.42		FA = 26.55		FA = 71.43	
	HR	THRS	HR	THRS	HR	THRS
YOLOv5	83.33	0.50	**83.29**	**0.52**	83.76	0.21
YOLOv5-Temporal	**83.41**	**0.48**	83.28	0.50	**83.90**	**0.35**
YOLOv5-LSTM	82.72	0.70	82.89	0.69	83.82	0.50

According to the results, all three algorithms have similar detection performance in terms of *Hit Rate*. While the *False Alarm Rate* is smaller, YOLOv5 algorithm is slightly better than the algorithms which uses temporal data. On the other hand, as the *False Alarm Rate* increases, YOLOv5-Temporal gets better than the others. YOLOv5 and YOLOv5-LSTM have better detection performance than each other at their reference false alarm rates. These results can also be confirmed by a precision-recall curve in Figure 6.4.

According to Figure 6.4, YOLOv5 algorithm has higher recall value than the others at the points where precision is high. As the precision gets lower, YOLOv5-Temporal and YOLOv5-LSTM algorithms start to outperform YOLOv5.

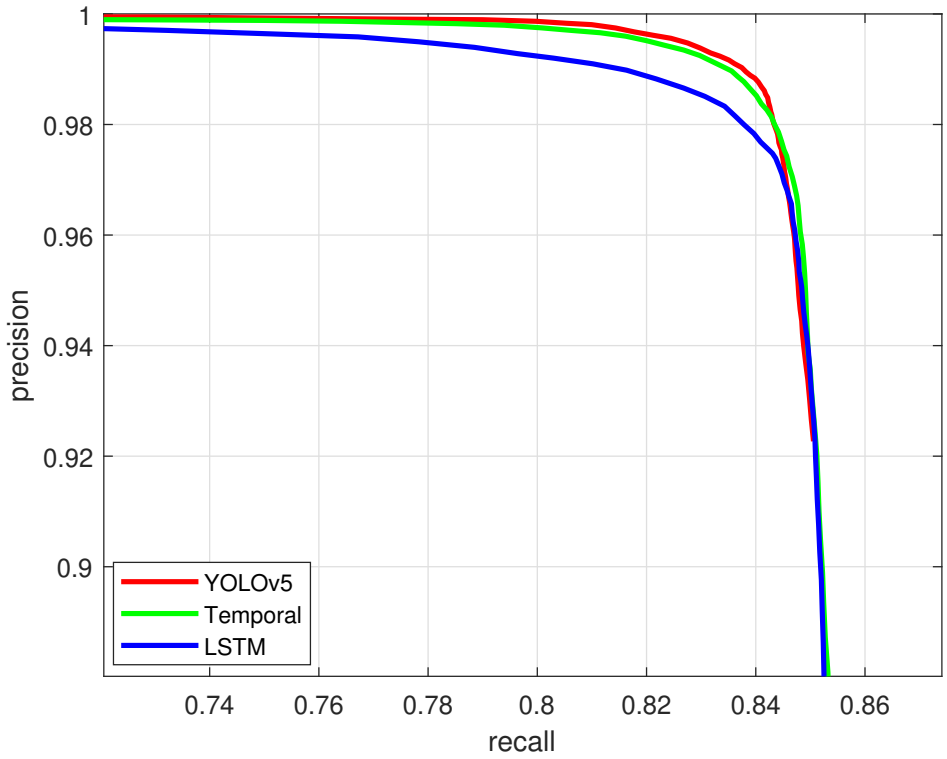


Figure 6.4: Precision-Recall Curve for a comparison of YOLOv5, YOLOv5-Temporal and YOLOv5-LSTM

6.5 Conclusion

In this chapter, the algorithms proposed are the ones using temporal data at the feature extraction step. To achieve this, the features from the neighboring frames are combined using LSTM and 3D convolution and regression is performed on these spatio-temporal feature vectors. This approach increases detection performance slightly in terms of hit rate, which is not possible with the ideas discussed in Chapter 5.

The moving object detectors proposed in this chapter are expected to have higher detection performance compared to a still image detector, specifically YOLOv5. Even though there are some improvements in terms of hit rate, there could be more.

As discussed in Chapter 2, YOLOv5 uses same idea with YOLOv3 on grid division, which is, intermediate feature levels are divided to a regular grid and each grid cell is responsible of the decision according to whether the center of an object in that cell

or not. On the other hand, YOLOv5-Temporal and YOLOv5-LSTM combines these features in temporal and spatial neighborhood. In consecutive frames, if the object location changes swiftly, especially due to the camera movements, there is a risk that the related object center falls into the neighboring grid. Hence, the grids concatenated in the moving object detectors may be irrelevant. Such situations might yield the detection accuracy not as high as expected. One solution could be removing camera movement from the dataset, but as the camera movement information is not available in the dataset, a new research should be carried to remove the camera movement.

CHAPTER 7

CONCLUSIONS

In this thesis, up-to-date object detection literature is covered thoroughly. First of all, still image object detection methods are examined. Considering the real-time processing requirement for video object detection, current YOLO variations are the most prominent algorithms in the literature. Therefore, some preliminary experiments are conducted by using YOLOv3, YOLOv4 and YOLOv5 with a fixed Anti-UAV dataset. Based on the simulation results, it is argued that YOLOv5 is the best performing variant for this family of algorithms.

Supervised learning based object detection models are data hungry. They require large amounts of annotated data in order to achieve high performance. Data annotation process is a costly work which requires lots of human labor. Therefore, some automatic or semi-automatic annotation methods are employed to decrease the human effort. These methods are generally employ video object trackers, which can drift and lead to shifted box errors. In this thesis, the effect of such annotation errors is also examined. It can be summarized as the most critical type of annotation error is the shifted box error. In this thesis, a method is also proposed to deal with shifted box errors. Throughout the experiments, it is shown that these type of errors can be fixed with the proposed approach. Utilizing the corrected dataset for training is shown to improve the detection accuracy.

A follow up idea to semi-automatic annotation correction is a proposal for a semi-automatic data annotation method. Tracklets, which are the visually similar subsets of detected objects, are introduced for the annotation. By the help of a user interface, tracklets can be annotated with much smaller human effort. In this study, it is shown that human workload can be reduced by up to 96% using temporal information.

Image object detectors work frame by frame, so they do not consider temporal information during their operation. This may easily cause to miss the object in one frame, while able to detect the same object in the previous frame; i.e. blinking issue. Similarly, some background clutter may cause the algorithm to generate false alarms.

In this work, firstly, some basic methods are investigated to use temporal information to check whether using that type of data is beneficial for video object detection problem or not. Two different types of M/N algorithm are proposed. It is shown that even using the simplest idea in temporal dimension reduces the false alarm rate significantly, with some sacrifice from the hit rate. Using visual similarity decreases the false alarms even further with the same amount of hit rate.

Observing the benefits of the temporal data, more sophisticated approaches are also proposed. YOLOv5 algorithm is modified to extract spatio-temporal features. This modification has also two different variants: YOLOv5-Temporal which is based on 3D CNN Block and YOLOv5-LSTM which is formed by Convolutional LSTM blocks. Even though these algorithms increase the detection accuracy, that improvement is not that significant. One possible reason of this is the vulnerability of YOLOv5-Temporal and YOLOv5-LSTM to the responsible grid changes on neighboring frames due to the camera movement.

As a future work, spatio-temporal registration should be studied. This registration can be performed by either estimating the camera motion or registering the decision grid utilized in YOLO variants cell by cell. This approach is expected to increase the moving object detection performance even further.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[5] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *CoRR*, vol. abs/1708.02002, 2017.

[6] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[7] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4293–4302, 2016.

[8] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*, pp. 850–865, Springer, 2016.

[9] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6638–6646, 2017.

[10] 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," 2014.

[11] 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.

[12] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 03 1960.

[13] D. Reid, "An algorithm for tracking multiple targets," *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[14] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proceedings of the IEEE international conference on computer vision*, pp. 4696–4704, 2015.

[15] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 941–951, 2019.

[16] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 817–825, 2016.

[17] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang, "Object detection in videos with tubelet proposal networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 727–735, 2017.

[18] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei, "Single shot video object detector," *IEEE Transactions on Multimedia*, vol. 23, pp. 846–858, 2020.

[19] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3038–3046, 2017.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[22] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[23] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[24] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European conference on computer vision*, pp. 391–405, Springer, 2014.

[25] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3286–3293, 2014.

[26] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping for image segmentation and object proposal generation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 128–140, 2016.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

[30] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[32] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," *Advances in neural information processing systems*, pp. 473–479, 1997.

[33] C. Olah, "Understanding lstm networks." http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[34] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[35] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, pp. 802–810, 2015.

[36] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[37] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[38] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[39] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, pp. 379–387, 2016.

[40] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[41] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[42] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014.

[44] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[45] DMP-Blog, "A closer look at yolov3." https://blog.dmprof.com/post/a-closer-look-at-yolov3, 2018.

[46] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[47] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 390–391, 2020.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[49] Z. Huang and J. Wang, "Dc-spp-yolo: Dense connection and spatial pyramid pooling based yolo for object detection," 2019.

[50] D. Misra, "Mish: A self regularized non-monotonic activation function," 2020.

[51] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018.

[52] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

[53] G. Jocher, A. Stoken, J. Borovec, NanoCode012, A. Chaurasia, TaoXie, L. Changyu, A. V, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana,

AlexWang1900, J. Hajek, L. Diaconu, Marc, Y. Kwon, oleg, wanghaoyang0106, Y. Defretin, A. Lohia, ml5ah, B. Milanko, B. Fineran, D. Khromov, D. Yiwei, Doug, Durgesh, and F. Ingham, "ultralytics/yolov5," Apr. 2021.

[54] G. Jocher, Y. Kwon, guigarfr, perry0418, J. Veitch-Michaelis, Ttayu, D. Suess, F. Baltacı, G. Bianconi, IlyaOvodov, Marc, e96031413, C. Lee, D. Kendall, Falak, F. Reveriano, FuLin, GoogleWiki, J. Nataprawira, J. Hu, LinCoce, LukeAI, NanoCode012, NirZarrabi, O. Reda, P. Skalski, SergioSanchezMontesUAM, S. Song, T. Havlik, and T. M. Shead, "ultralytics/yolov3," Apr. 2021.

[55] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, p. 217, 02 2021.

[56] T. Ridnik, H. Lawen, A. Noy, E. Ben Baruch, G. Sharir, and I. Friedman, "Tresnet: High performance gpu-dedicated architecture," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1400–1409, 2021.

[57] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *CoRR*, vol. abs/1907.09408, 2019.

[58] J. Zhao, "Iccv 2021 anti-uav challenge dataset." https://anti-uav.github.io/dataset/, 2021.

[59] J. Hui, "map (mean average precision) for object detection." https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, 2018.

[60] A. Koksal, K. G. Ince, and A. Alatan, "Effect of annotation errors on drone detection with yolov3," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1030–1031, 2020.

[61] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *CoRR*, vol. abs/1809.02165, 2018.

[62] Y. Wang, Q. Yao, J. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," 2019.

[63] F. Zhao, J. Zhao, S. Yan, and J. Feng, "Dynamic conditional networks for few-shot learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–35, 2018.

[64] B. Frenay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[65] C. Pérez, F. Girón, J. Martín, M. Ruiz, and C. Rojano, "Misclassified multinomial data: a bayesian approach," *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A, Matemáticas*, vol. 101, pp. 71–80, 01 2007.

[66] F. A. Breve, L. Zhao, and M. G. Quiles, "Semi-supervised learning from imperfect data through particle cooperation and competition," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010.

[67] J. Sun, F. Zhao, C. Wang, and S. Chen, "Identifying and correcting mislabeled training instances," in *Future Generation Communication and Networking (FGCN 2007)*, vol. 1, pp. 244–250, 2007.

[68] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," 2017.

[69] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, and X. Gao, "Learning from weak and noisy labels for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 486–500, 2017.

[70] J. Xu, A. G. Schwing, and R. Urtasun, "Learning to segment under various forms of weak supervision," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3781–3790, 2015.

[71] S. Chadwick and P. Newman, "Training object detectors with noisy data," *CoRR*, vol. abs/1905.07202, 2019.

[72] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. J. Rob. Res.*, vol. 32, p. 1231–1237, Sept. 2013.

[73] K. G. Ince, A. Koksal, A. Fazla, and A. A. Alatan, "Semi-automatic video annotation for object detection," 2021.

[74] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing annotations for visual object detection," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[75] Ericsson, "eva." https://github.com/Ericsson/eva, 2020.

[76] H. Bilen, M. Pedersoli, and T. Tuytelaars, "Weakly supervised object detection with posterior regularization," *Proceedings BMVC 2014*, pp. 1–12, 2014.

[77] H. Bilen, M. Pedersoli, and T. Tuytelaars, "Weakly supervised object detection with convex clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1081–1089, 2015.

[78] A. Yao, J. Gall, C. Leistner, and L. Van Gool, "Interactive object detection," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3242–3249, 2012.

[79] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," in *CVPR 2011*, pp. 1449–1456, 2011.

[80] B. Adhikari and H. Huttunen, "Iterative bounding box annotation for object detection," *arXiv preprint arXiv:2007.00961*, 2020.

[81] B. Adhikari, J. Peltomaki, J. Puura, and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, IEEE, 2018.

[82] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 854–863, 2016.

[83] K. Konyushkova, J. Uijlings, C. H. Lampert, and V. Ferrari, "Learning intelligent dialogs for bounding box annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9175–9184, 2018.

[84] O. Russakovsky, L.-J. Li, and L. Fei-Fei, "Best of both worlds: human-machine collaboration for object annotation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2121–2131, 2015.

[85] A. Kuznetsova, A. Talati, Y. Luo, K. Simmons, and V. Ferrari, "Efficient video annotation with visual interpolation and frame selection guidance," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3070–3079, 2021.

[86] S. Jin, A. RoyChowdhury, H. Jiang, A. Singh, A. Prasad, D. Chakraborty, and E. Learned-Miller, "Unsupervised hard example mining from videos for improved object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 307–324, 2018.

[87] A. RoyChowdhury, P. Chakrabarty, A. Singh, S. Jin, H. Jiang, L. Cao, and E. Learned-Miller, "Automatic adaptation of object detectors to new domains using self-training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 780–790, 2019.

[88] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-uav cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 627–635, 2019.

[89] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous uav cinematography: A tutorial and a formalized shot-type taxonomy," *ACM Comput. Surv.*, vol. 52, Sept. 2019.

[90] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5525–5533, 2016.

[91] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

[92] A. Bochkovskiy, "Yolo-lstm." https://github.com/AlexeyAB/darknet/issues/3114, 2019.